Sensore di Irraggiamento Predittivo

Paolo Casoretti

August 30, 2019

Abstract

La principale sfida odierna è il rispetto dell'ecosistema in cui viviamo e il sostentamento dello stesso tramite energie, sempre in maggiore percentuale rinnovabili, che vanno a sostituire le più obsolete e inquinanti fonti di energia derivanti dai combustibili fossili.

L'energia rinnovabile più interessante e più utilizzata ma anche quella su cui si pone un maggiore focus è sicuramente il fotovoltaico nonostante la sua particolarità di concentrare la maggiore produzione nelle ore centrali della giornata per poi lasciare spazio a una produzione quasi nulla nelle ore notturne.

La continua diffusione delle energie rinnovabili sulla rete elettrica pone delle sfide di gestione per via dell'aspetto stocastico delle nuove fonti di generazione.

La difficoltà di previsione e la necessità di avere una stabilità di rete sempre migliore implica l'utilizzo di nuove tecnologie.

L'avvento di una elettronica sempre più compatta ed economica,anche per basso numero di prototipi prodotti, sono la chiave di volta per lo sviluppo, il design e la prototipazione di un dispositivo atto alla cattura dei dati ambientali, gestione e previsione dell'irragiamento solare, che in futuro rappresenta con assoluta certezza la maggiore fonte di energia per il nostro pianeta.

Chapter 1

Summary and Index

Contents

List of Figures

Chapter 2

PV Scenario

Lo scenario del fotovoltaico nel mondo è importante per capire quanta importanza abbia oggi la previsione e il monitoraggio della sua produzione vista la potenza installata sempre più alta in ogni continente, e i ratio di crescita molto incisivi che fanno del fotovoltaico la più diffusa fonte di energia rinnovabile.

2.1 2000-2017

Il 2017/8 è stato un altro biennio storico per il settore dell'energia solare. Sono state installate a livello globale più impianti SolarPV rispetto a qualsiasi altra tecnologia di generazione di energia. In effetti, il solare ha visto l'impiego di molti nuovi impianti di produzione rispetto ai combustibili fossili e al nucleare combinati. Il solare ha addirittura aumentato di quasi il doppio la capacità rispetto alle altre energie rinnovabili. Infatti, il solo solare ha visto dispiegare più nuovi impianti rispetto ai combustibili fossili e al nucleare combinati. Il solare ha persino aggiunto quasi il doppio della capacità rispetto alle altre energie rinnovabili, l'energia eolica. Nonostante i notevoli tassi di crescita degli ultimi anni, c'è ancora molta strada da fare per le energie rinnovabili.

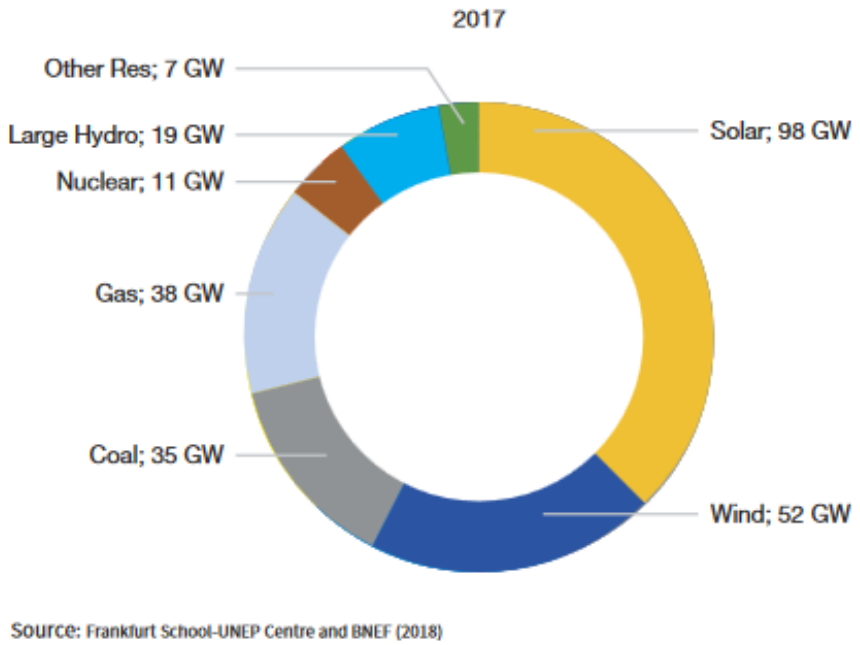


Figure 2.1: Energy Pie Chart Worldwide

Il potenziamento del solare è in gran parte il risultato del suo incredibile abbassamento dei costi. I prezzi record record raggiunti nel 2016 hanno colto di sorpresa molti esperti di energia. Quell'anno, le offerte aggiudicate dai costruttori erano al di sotto dei 3 centesimi per kWh (2,95 centesimi per un progetto da 800 MW a Dubai, 2,81 centesimi per un contratto di fornitura di energia in Cile, 2,42 centesimi per la fornitura "invernale" del 1,18 GW impianto PPA ad Abu Dhabi). Le discussioni sulla sostenibilità sono diventate rapidamente obsolete man mano che la spirale dei prezzi ha continuato a scendere. Nel febbraio 2018, una gara d'appalto di 300 MW in Arabia Saudita è stata vinta dalla società locale ACWA Power a un prezzo basso da newworld di 2,34 centesimi / kWh, mentre le prime offerte finalizzate erano tutte inferiori a 2,90 centesimi / kWh.

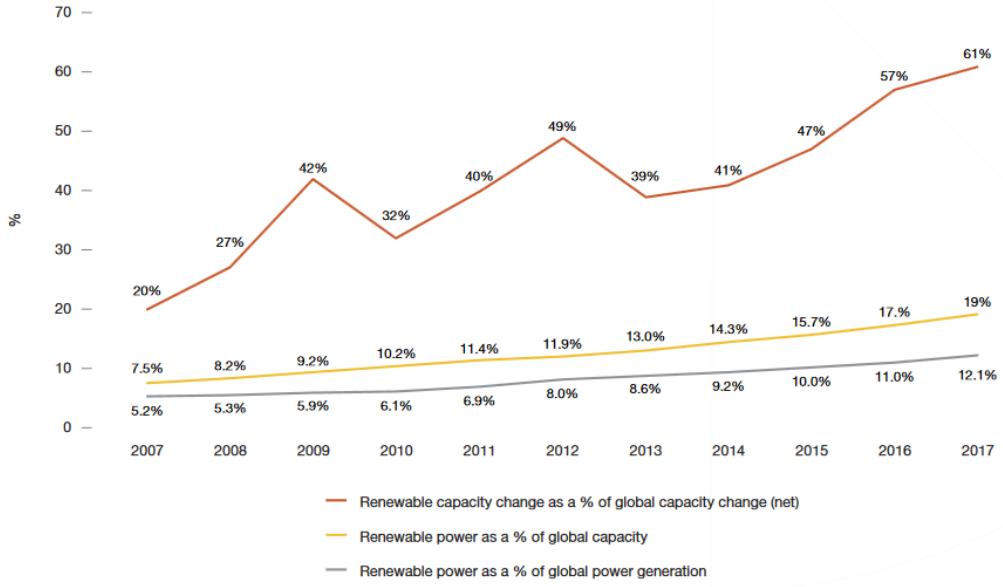


Figure 2.2: Renewable Energy Change Ratio

Nel 2017 sono stati installati un totale di 99,1 GW di energia solare collegata alla rete. Si tratta di una crescita di quasi il 30% su base annua rispetto ai 76,6 GW aggiunti nel 2016. Questo è molto inferiore al tasso di crescita del 49% registrato nel 2016, ma molto superiore alle aspettative di qualsiasi analista solare. Dopo la crescita eccezionalmente forte del 2016, la maggior parte degli analisti solari inizialmente non avevano previsto alcuna crescita. Lo scenario medio, stimando un leggero tasso di crescita del 5% a 80,5 GW, è stato tra le previsioni più ottimistiche per il 2018. Il 2017 è stato dominato più che mai dalla Cina. per la prima volta, la Cina ha installato oltre la metà della capacità solare del mondo in un anno per la precisione, il 53,3%. Questo 52,8 GW di capacità appena aggiunta significa un aumento del 53% rispetto ai 34,5 GW installati nel 2016, quando il mercato cinese è cresciuto del 128% rispetto al 15,1 GW distribuito nel 2015. Mentre l'enorme crescita nel 2017 ha stupito quasi tutti, la spiegazione è molto semplice : Il programma tariffario cinese era sostanzialmente senza limiti, i livelli tariffari per le centrali solari erano più alti che in molti altri luoghi del mondo e le aziende volevano ridurre i sussidi previsti per le sovvenzioni. Ecco perché i produttori di moduli cinesi hanno nuovamente dato la priorità al mercato pericoloso rispetto alla domanda dall'estero.

Come l'anno precedente, gli Stati Uniti erano il secondo mercato fotovoltaico più grande del mondo nel 2017. Hanno installato 10,6 GW, e il ratio annuale è sceso del 42% rispetto al livello record di 15,1 GW raggiunto nel 2016. Mentre il solare su scala industriale è rimasto il segmento più ampio, quasi l'intera recessione deriva da quella parte. Tuttavia, questo declino era ampiamente previsto, poiché nel 2016 molti progetti sono stati finalizzati a battere la scadenza inattesa del Federal Investment TaxCredit (ITC) del 30%, che non si è concretizzata alla fine. Inoltre, diversi progetti sono stati accantonati a causa dell'incertezza che circonda le tariffe d'importazione.

Il 2017 è stato un anno record per il fotovoltaico in India. La capacità installata cumulativa ha superato i 19 GW, con aggiunte annue nette di 9,6 GW, una crescita del mercato sconcertante del 127% rispetto ai 4,3 GW dell'anno scorso. La crescita avrebbe potuto essere più accentuata, se non fosse stato per gli aumenti dei prezzi dei moduli dalla Cina nel corso dell'anno, un segmento sul tetto in ritardo e un'incertezza per quanto riguarda le tasse sulle importazioni. Tuttavia, nel 2017 il solare era la principale fonte di nuova potenza installata, costituendo il 45% della nuova capacità aggiunta. L'India ha preso il posto del Giappone come il terzo mercato più grande al mondo.

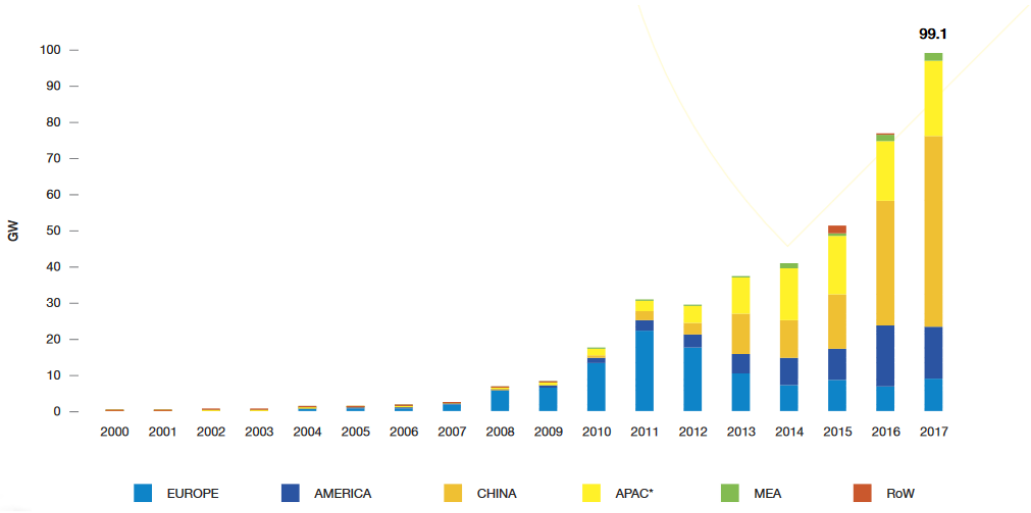


Figure 2.3: Energy Installed

2.2 2018-2022 Scenario

Ancora una volta, tutti gli scenari di Global Market Outlook 2018 mostrano una crescita più forte rispetto alla precedente previsione. Nel 2017, si ipotizzava una capacità installata cumulativa di 4271,2 GW per lo scenario medio nel 2018, quest'anno stimiamo 505,2 GW, che è superiore di circa il 7% . Nell'OGM 2018, si prevede una crescita che sta nell' intervallo compreso tra 714,6 e 1,042,1 GW, con 871,3 GW previsti per lo scenario più probabile per il 2021 ossia circa il 13% in più. In condizioni ottimali, la capacità dell'impianto di solargenerazione mondiale potrebbe raggiungere i 1.270,5 GW entro la fine del 2022, ma si ritiene più probabile una cifra intorno a 1.026,2 GW. Tuttavia, ciò significa che il solare raggiungerebbe il livello di 1TW di capacità di produzione di energia elettrica nel 2022. Raggiungendo il traguardo di 400 MW già nel 2017, ora si prevede di superare i 500 GW nel 2018, 600 GW nel 2019 700 GW nel 2020 , 800 GW nel 2021 e 1 TW nel 2022. Sebbene il solare stia diventando sempre più la fonte di generazione di energia a basso costo in molte regioni, ha bisogno di condizioni di gioco pari con il giusto design del mercato per liberare i suoi costi e vantaggi tecnici rispetto alle centrali elettriche non flessibili. Oggi, molti ostacoli sono ancora in grado per il solare di sfruttare il suo potenziale: di solito mancano schemi funzionali di emissione di combustibili fossili, come nell'UE; gli investimenti e il funzionamento della tecnologia di generazione di energia non centralizzata sono spesso ancora fortemente sovvenzionati, come la centrale nucleare di Hinkley nel Regno Unito; mentre l'energia solare autoconsumata viene tassata in modo inadeguato, come in Germania. La dipendenza del settore solare da pochi mercati è un'altra questione che deve essere affrontata: nel 2017, la Cina, era responsabile di oltre la metà della domanda globale; i primi 3 mercati solari (Cina, Stati Uniti, India) hanno addirittura coperto il 74%.

Qualunque sia lo sviluppo del mercato solare cinese nei prossimi cinque anni e se la crescita globale seguirà lo scenario basso o alto, l'Asia continuerà a dominare il settore solare in futuro . Nel 2017, si presumeva che la regione Asia-Pacifico assorba circa due terzi delle installazioni totali e copra il 60% fino al 2022.

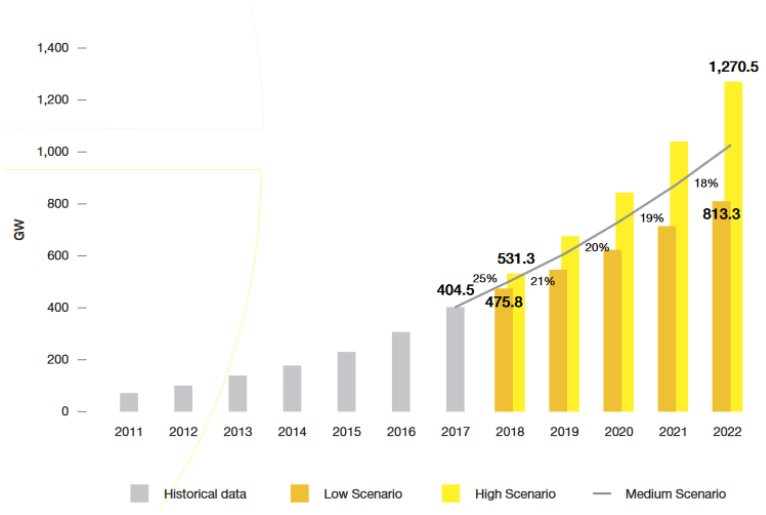


Figure 2.4: Energy Installed Prevision

Chapter 3

Irradiance Forecasting

Una delle principali caratteristiche e sfide per le reti interconnesse è il controllo in tempo reale delle discrepanze tra la produzione programmata e il consumo effettivo di energia elettrica, vale a dire il controllo della frequenza. Dalla liberalizzazione dei mercati dell'elettricità e dall'aumento della generazione intermittente decentralizzata, il sistema energetico dell'Europa continentale è stato esposto a forti deviazioni di frequenza persistenti. Questa tesi indaga sulla possibilità di ottenere un dispositivo in grado di generare un forecasting della produzione di energia solare. L'obiettivo è quello di contribuirea migliorare le possibiltà di controllo tramite un dispositivo low cost della frequenza nel sistema di alimentazione interconnesso.

La regolazione per una sistema che si basa sempre di più sull'energia rinnovabile derivante dal fotovoltaico, crea una sfida per chi fa compravendita di energia.
Questa sfida è facilmente deducibile dalla non predicibilità dell'energia prodotta dagli stessi impianti e dal frastagliamento della stessa nell'arco della giornata se vogliamo parlare di giornate parzialmente coperte e totalmente nuvolose.
Essendo inoltre una fonte di energia che si basa sul meteo, essa gode di una volatilità e imprevedibilità estrema se ci troviamo in ambienti particolarmente soggetti a cambiamenti di meteo repentini e difficilmente predicibili.

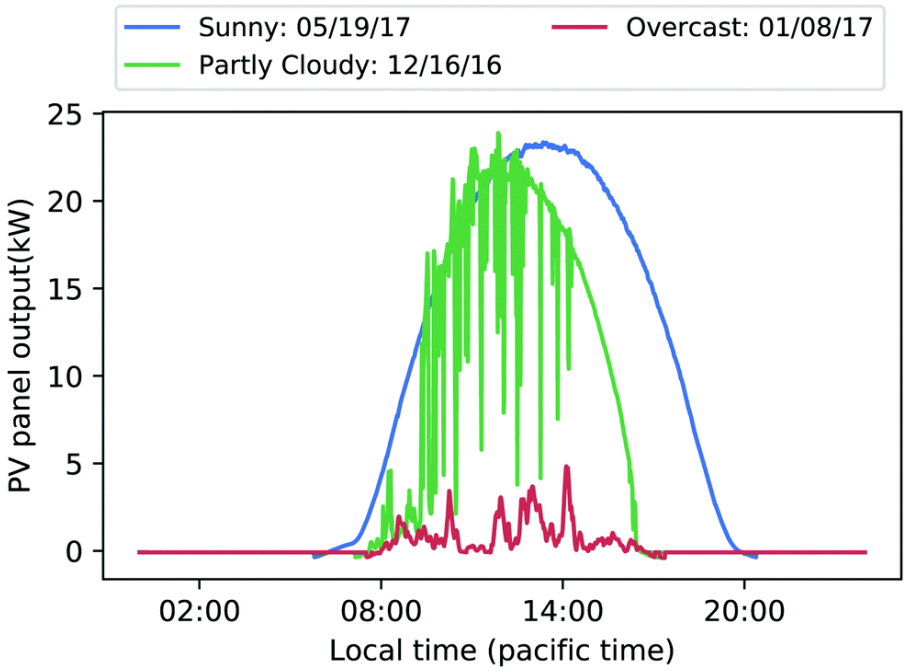


Figure 3.1: Classic Daily Production

Come si può evincere dalla figura superiore, ci troviamo davanti a una fonte di energia che se cumulata nell'ordine di MW offre delle oscillazioni percentuali altissime in un tempo ΔT ridotto.
Ne consegue una difficoltà di regolazione decisamente alta, e ad una fonte di energia presente soprattutto nelle fasce orarie che circondano lo di Zenith del sole, quindi lontane da una *Base Load* ottimale.

Il contributo della produzione di energia dei sistemi fotovoltaici per quanto riguarda la produzione di elettrica è in costante aumento. Società di servizi pubblici e trasmismissione devono fare attenzione a un sistema

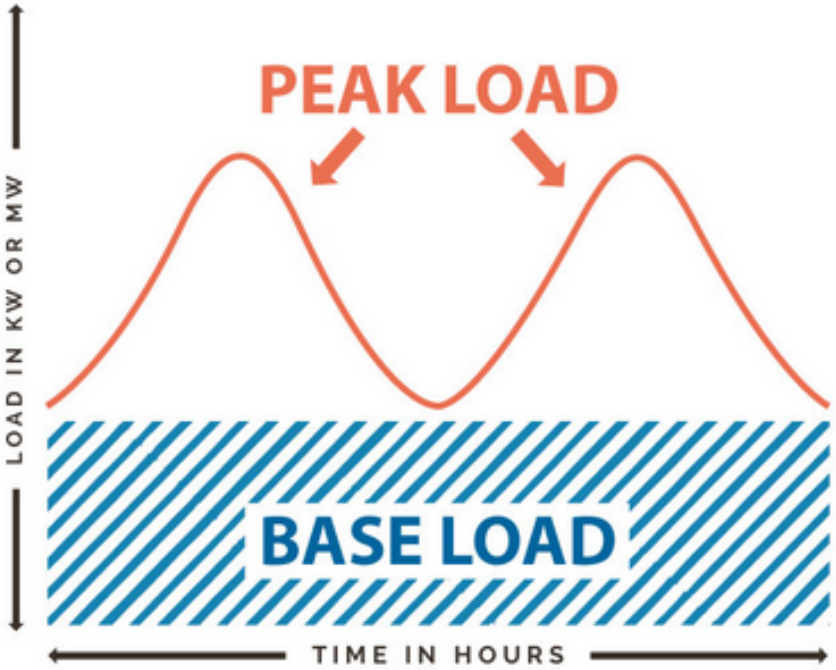


Figure 3.2: Base Load and PV Peak

MISO con una volatilità di *Input* elevata.

La previsione dell'irradianza orizzontale globale (GHI) è il primo e più importante passaggio per quanto riguarda i sistemi di previsione della potenza fotovoltaica.

Gli approcci di previsione al GHI possono essere classificati in base ai dati di input utilizzati che determinano anche l' orizzonte di previsione. Vengono applicati modelli statistici basati su misure di irradianza online e sviluppano una cronologia a brevissimo termine da 5 minuti a 6 ore Esempi di modelli di serie temporali dirette sono autoregressivi (AR) e modelli di media mobile autoregressivi (ARMA). Inoltre si possono utilizzare sistemi di intelligenza artificiale come le reti neurali (ANN) che possono essere applicate per derivare le previsioni di irraggiamento.

Per la previsione dell'irraggiamento a breve termine, il riconoscimento del path delle nuvole, che in gran parte determinano l'irraggiamento solare superficiale, possono essere utilizzati:

- Previsioni basate su vettori di movimento delle nuvole da immagini satellitari) mostrano buone prestazioni per il range temporale da 30 minuti fino a 6 ore.

- Per la gamma subhour, immagini delle nuvole scattate da terra possono essere utilizzate per derivare previsioni di irradianza con una risuluzione spaziale migliore rispetto a quella satellitare. previsioni.

Per orizzonti di previsione più lunghi, da circa 4-6 ore in poi, previsioni basato su modelli numerici di previsione del tempo (NWP) tipicamente formano le previsioni basate sui satelliti.

Esistono anche approcci combinati che integrano diversi tipi di input dati per ricavare una previsione ottimizzata in base all'orizzonte di previsione.

La situazione spaziale e temporale può essere raccolta in questa analisi .

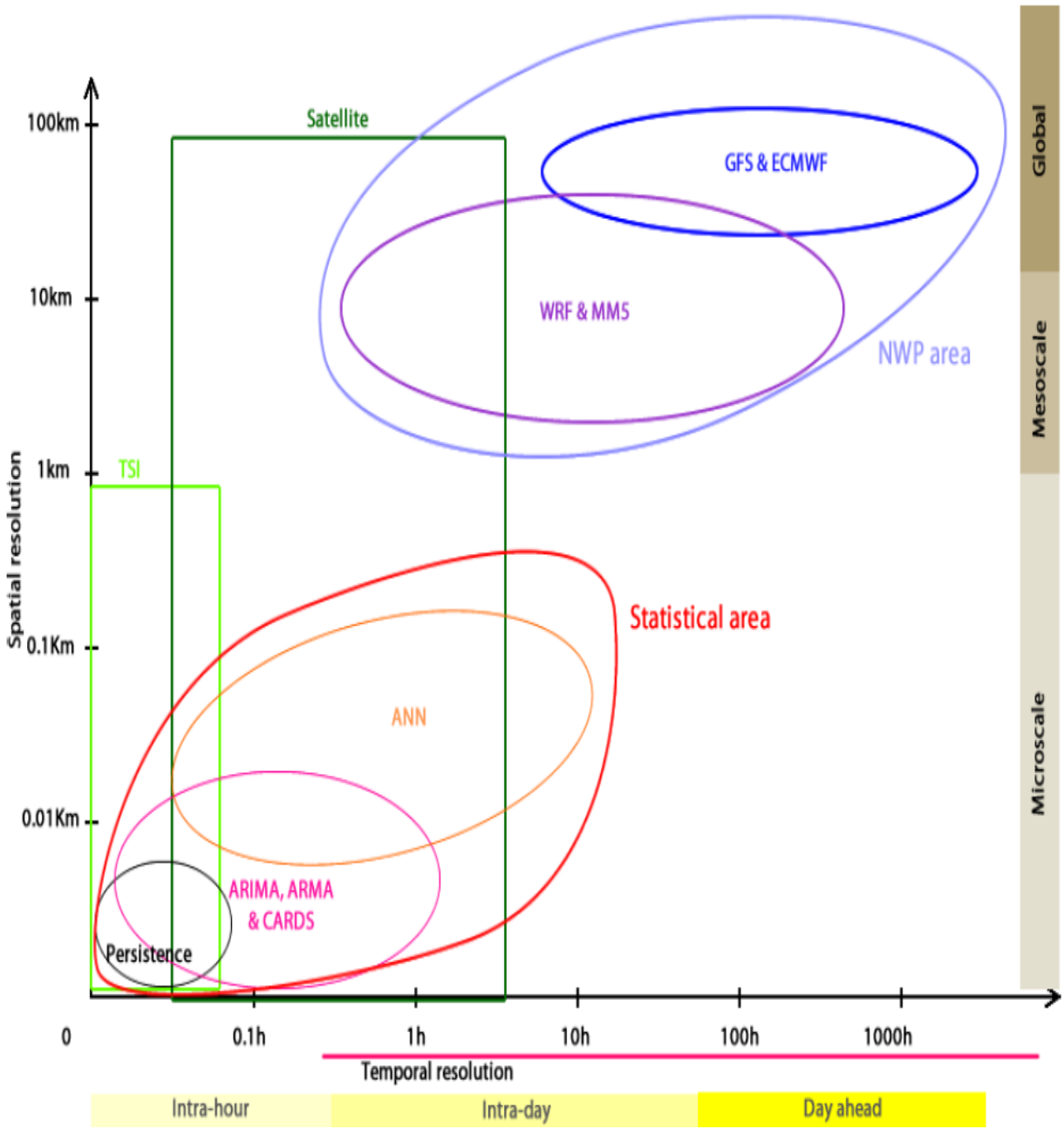


Figure 3.3: Temporal Analysis

Chapter 4

Sensors

La premessa di questa sezione è di poter valutare e identificare quali possono essere i sensori che meglio si prestano al raccoglimento dei dati che successivamente formeranno la base dati elaborata dal predittore. Oggi il mondo dell'embedded offre numerose possibilità di scelta per quanto concerne la sensoristica, tuttavia anche se non è necessario l'utilizzo di sensori di tipo digitale esclude numerosi problemi legati ai disturbi che possono creare nel *routing* del segnale così come permette un trattamento dei segnali che viene eseguito in maniera ottimale.

Nonostante il design del sistema volga verso una interfaccia in grado di essere *reliable* così come *low cost*, in via preliminare non è possibile conoscere quali segnali e quali tipi di sensori creano una ottima correlazione tra i dati ricevuti e il modello predittivo che si vuole creare.

Inizialmente si era pensato allo sviluppo di una board che contenesse solamente un sensore di irraggiamento che dati alcuni *papers* già presenti si è rivelato un ottimo metodo in relazione alla sua semplicità per valutare un *nowCasting* con orizzonti temporali relativamente brevi *t<1h*.

Tuttavia la possibilità di avere una sensoristica più complessa e moderna a un prezzo accettabile rende interessante la valutazione di numerosi sensori che poi nel caso non trovassero una correlazione con il modello possono essere inibiti e non implementati nella board definitiva.

Il meteo gioca un ruolo fondamentale nella predizione in quanto sono gli elementi climatici che determinano la presenza di nuvole e il conseguente cambiamento di irraggiamento.

Al fine di scattare una fotografia della situazione ambientale sono necessari almeno le misure di temperatua, umidità e pressione barometrica. Esiste la possibilità di avere questi sensori in una unica breakBoard sperimentale in modo che in maniera facile e veloce si possa avere una lettura di tutte e tre i fondamentali input meteo. Queste variabili esogene sono indispensabili per porre una base meteo e un solido input.

4.1 Pressure,Temperature,Humidity

Come detto in precedenza esistono diversi modulo che integrano queste tre misure in una unica breakboard, tuttavia la scelta di un componente con librerie già implementate sembra la scelta migliore da percorrere.

4.1.1 BME280

Il BME280 è un sensore combinato di umidità, pressione e temperatura basato su comprovati principi di rilevamento. Il modulo sensore è alloggiato in un package LGA con coperchio in metallo estremamente compatto con un ingombro di soli 2,5 $\times$ 2,5 mm^2 con un'altezza di 0,93 mm. Le sue dimensioni ridotte e il suo basso consumo energetico consentono l'implementazione in dispositivi alimentati a batteria quali telefoni, moduli GPS o orologi. Il BME280 è un registratore e le prestazioni compatibili con il sensore di pressione digitale Bosch Sensortec BMP280 (vedere il capitolo 5.2 per dettagli). Il BME280 raggiunge alte prestazioni in applicazioni che richiedono misure di umidità e pressione. Queste applicazioni emergenti del controllo domotico, della navigazione interna, dell'assistenza sanitaria e della raffinatezza del GPS richiedono un'elevata precisione e un basso TCO allo stesso tempo. Il sensore di umidità fornisce un tempo di risposta estremamente rapido per applicazioni di riconoscimento rapido del contesto e alta precisione in un ampio intervallo di temperature. Il

sensore di pressione è un sensore di pressione barometrica assoluto con altissima precisione e risoluzione e un rumore drasticamente basso. Il sensore di temperatura integrato è stato ottimizzato per il rumore più basso e la massima risoluzione. La sua uscita viene utilizzata per la compensazione della temperatura dei sensori di pressione e umidità e può anche essere utilizzata per la stima della temperatura ambiente. Il sensore fornisce entrambe le interfacce SPI e I²C e può essere alimentato da 1,71 a 3,6 V . Le misurazioni possono essere attivate dall'host o eseguite a intervalli regolari. Quando il sensore è disabilitato, il consumo di corrente scende a 0.1 microA. BME280 può funzionare in tre modalità di alimentazione:

- Sospensione
- Modalità normale
- Modalità forzata

Per adeguare velocità dati, rumore, tempo di risposta e consumo corrente alle esigenze dell'utente, una varietà di modalità di sovracampionamento, filtro modalità e velocità dati possono essere selezionate.

Block Diagram e Pinout

Internamente il sensore è organizzato come di seguito in questo schema a blocchi:

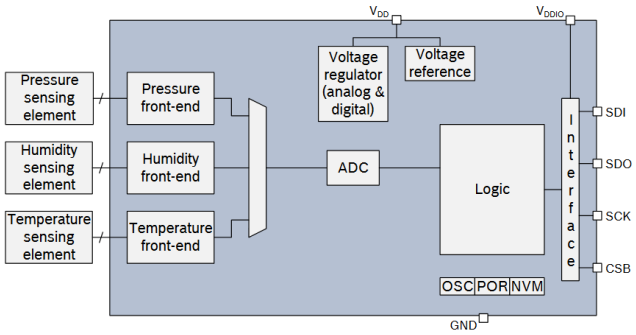


Figure 4.1: Schema Blocchi BME280

Generalmente è composto da due alimentazioni *Vdd* e *Vddio* che sono rispettivamente l'alimentazione per i componenti digitali e analogici e quella per l'interfaccia.

Il pin Gnd fa capo al ground analogico dei sensori , mentre SDI, SDO, SCK or CSB sono i pin per la gestione della comunicazione I^2C.

Operation Mode

- Sospensione

  Si verifica nel momento in cui viene alimentato e in condizioni di *Reset* e il consumo della sua corrente I_{ddsm} è posto al minimo. Tutti i registri sono accessibili e non ci sono restrizioni di nessun tipo.

- Modalità normale

  Consiste in una perpetua e automatica azione di lettura tra il ciclo di lettura e il periodo di standby. Le misure sono svolte in accordo con i filtri scelti, inoltre è possibile gestire anche il periodo di stanby che può essere impostato tra gli 0.5 e i 1000ms.

- Modalità forzata

  Solo una singola misura viene performata, in accordo con le impostazioni di misura e i filtr impostati. Quando la misura è finita il sensore ritorna in modalità di sospensione e il dato è accessibile nel registro. Per ogni successiva misura è necessario la riselezione della modalità stessa.

Measurement Spec

Temperature

Temperature sensor specification

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Operating range	T	Operational	-40	25	85	°C
		Full accuracy	0		65	°C
Supply current	$I_{DD,T}$	1 Hz forced mode, temperature measurement only		1.0		µA
Absolute accuracy temperature[9]	$A_{T,25}$	25 °C		±0.5		°C
	$A_{T,full}$	0…65 °C		±1.0		°C
Output resolution	R_T	API output resolution		0.01		°C
RMS noise	N_T	Lowest oversampling		0.005		°C

Figure 4.2: Temperature Details

Humidity

Humidity parameter specification

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Operating range[3]	R_H	For temperatures < 0 °C and > 60 °C see Figure 1	-40	25	85	°C
			0		100	%RH
Supply current	$I_{DD,H}$	1 Hz forced mode, humidity and temperature		1.8	2.8	µA
Absolute accuracy tolerance	A_H	20…80 %RH, 25 °C, including hysteresis		±3		%RH
Hysteresis[4]	H_H	10→90→10 %RH, 25 °C		±1		%RH
Nonlinearity[5]	NL_H	10→90 %RH, 25 °C		1		%RH
Response time to complete 63% of step[6]	$\tau_{63\%}$	90→0 or 0→90 %RH, 25°C		1		s
Resolution	R_H			0.008		%RH
Noise in humidity (RMS)	N_H	Highest oversampling, see chapter 3.6		0.02		%RH
Long term stability	ΔH_{stab}	10…90 %RH, 25 °C		0.5		%RH/ year

Figure 4.3: Humidity Details

Barometric Pressure

Pressure sensor specification

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Operating temperature range	T_A	operational	-40	25	+85	°C
		full accuracy	0		+65	
Operating pressure range	P	full accuracy	300		1100	hPa
Supply current	$I_{DD,LP}$	1 Hz forced mode, pressure and temperature, lowest power		2.8	4.2	µA
Temperature coefficient of offset[7]	TCO_P	25…65 °C, 900 hPa		±1.5		Pa/K
				±12.6		cm/K
Absolute accuracy pressure	$A_{P,full}$	300 . . . 1100 hPa 0 . . . 65 °C		±1.0		hPa
Relative accuracy pressure V_{DD} = 3.3V	A_{rel}	700 ... 900hPa 25 . . . 40 °C		±0.12		hPa

Figure 4.4: Pressure Details

4.1.2 BME680

Il BME680 è un sensore digitale 4 in 1 con misurazione di gas, umidità, pressione e temperatura basate su comprovati principi di rilevamento. Il modulo sensore è alloggiato in un contenitore LGA con coperchio metallico estremamente compatto con un ingombro di soli 3,0 × 3,0 mm² con un'altezza massima di 1,00 mm (0,93 ± 0,07 mm). Le sue dimensioni ridotte e il basso consumo energetico consentono l'integrazione in dispositivi alimentati a batteria o accoppiati in frequenza, quali telefoni ,dispositivi indossabili e stazioni di rilevamento forecasting.

Block Diagram e Pinout

Similmente al BME280 anche il BME680 ha una struttura a blocchi standard:

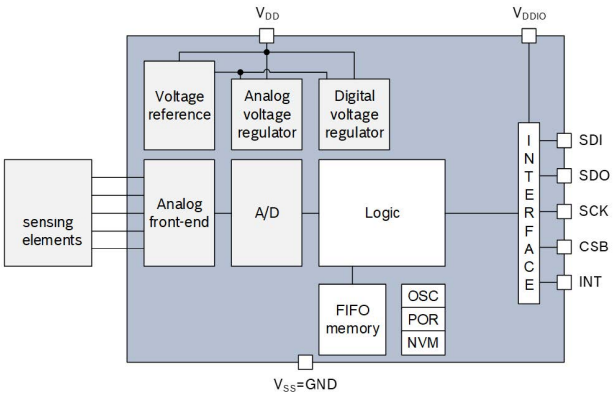


Figure 4.5: Schema Blocchi BME280

Measurament Spec

Oltre alle 3 misure che può effettuare anche il BME280 , il suo successore permette la determinazione della qualità dell'aria che in caso di ambienti molto inquinati ha sicuramente effetto sull'irraggiamento.

Air Quality Index AQI

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Operational range[1]			-40		85	°C
			10		95	% r.H.
Supply Current during heater operation	I_{DD}	Heater target temperature 320 °C, constant operation (V_{DD} ≤ 1.8 V, 25°C)	9	12	13	mA
Peak Supply Current	I_{Peak}	Occurs within first ms of switching on the hotplate	15	17	18	mA
Average Supply Current (V_{DD} ≤ 1.8 V, 25°C)	$I_{DD,IAQ}$	Ultra-low power mode		0.09		mA
		Low power mode		0.9		mA
		Continuous mode		12		mA
Response time[2] (brand-new sensors)	**$\tau_{33-63\%}$**	**Ultra-low power mode**		**92**		**s**
	$\tau_{33-63\%}$	Low power mode		1.4		s
	$\tau_{33-63\%}$	Continuous mode		0.75		s
Resolution of gas sensor resistance measurement			0.05	0.08	0.11	%
Noise in gas sensor resistance (RMS)	N_R			1.5		%

Figure 4.6: AQI Spec

La Tabella elenca le specifiche del sensore di gas. Tutti i parametri sono dedotti dalle misure di laboratorio in condizioni ambientali controllate, conformi alla norma ISO16000-29 "Metodi di prova per rivelatori di VOC". Il software è progettato con cura per funzionare perfettamente con i sensori integrati 4 in 1 all'interno del BME680. Basato su un algoritmo intelligente, il BSEC fornisce un'uscita di qualità dell'aria interna (IAQ). In linea di principio, questa uscita si trova in un indice che può avere valori compresi tra 0 e 500 con una risoluzione di 1 per indicare o quantificare la qualità dell'aria disponibile nell'ambiente circostante. Inoltre, la soluzione BSEC supporta diverse modalità operative per il sensore di gas per soddisfare il budget energetico necessario e i requisiti di velocità di aggiornamento dell'applicazione finale.

I valori di output sono riassunti in questa tabella:

IAQ Index	Air Quality
0 – 50	good[10]
51 – 100	average
101 – 150	little bad
151 – 200	bad
201 – 300	worse[2]
301 – 500	very bad

Figure 4.7: AQI Scale

4.2 Irradiance

Dato indispensabile per la creazione di una base dati che possa creare un modello affidabile è sicuramente la misura dell'irraggiamento. Le celle fotovoltaiche convertono la luce in energia elettrica tuttavia esse non ricevono la totalità dello spettro solare.

Importante è tenere in considerazione come, il sole in giornate di cielo limpido possa raggiungere intensità luminose di oltre i $100k$ lux. Molti sensori di luminosità non raggiungono valori di intensità superiori ai 70 mila lux che non coprono la totalità dell'intensità luminosa.

4.2.1 VEML7700

Figure 4.8: VEML 7700

VEML7700 è un sensore di risoluzione digitale ad alta luminosità a 16 bit ad alta precisione in un involucro in miniatura trasparente da 6,8 mm x 2,35 mm x 3,0 mm. Include un diodo foto ad alta sensibilità, un amplificatore a basso rumore, un convertitore A / D a 16 bit e supporta un'interfaccia di comunicazione bus I2C di facile utilizzo. Il risultato della luce ambientale è un valore digitale manipolabile facilmente.

Block Diagram e Pinout

Internamente il sensore è organizzato come di seguito in questo schema a blocchi:

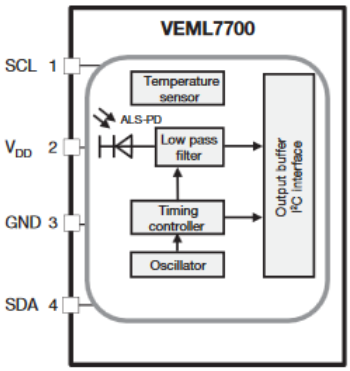


Figure 4.9: VEML7700 Block Diagram

Anche in questo caso il pinout è composto da *Vdd* che è l'alimentazione per i componenti digitali e le porte *SDA* e *SCL* volte alla diffusione dei messaggi digitali.

Measured Spectrum

Lo spettro di misura del VEML7700 è doppio, ossia il primo centrato sullo spettro di visibilità umano per cui con una grande lacuna nella regione degli *UV* e degli *IR* e il secondo denominato *WhiteChannel* in cui si ha a disposizione una maggiore larghezza spettrale.

Nonostante la maggior parte dell'energia proveniente dal sole sia in questa fascia, si ha bisogno di un criterio discriminante tra un cielo *cloudy* e un cielo *clear*.

Normal Channel

Segue con un mismatch del 3 percento la risposta spettrale dell'occhio umano.

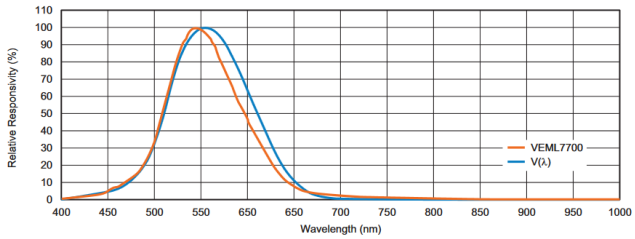


Figure 4.10: Spectral Response Normal Channel

White Channel

Oltre al canale ALS che segue molto bene la cosiddetta curva dell'occhio umano, c'è anche un secondo canale disponibile chiamato *White Channel*, che offre una reattività molto più elevata per uno spettro di lunghezze d'onda più lunghe. Questo canale bianco potrebbe essere usato visualizzare le ultime percentuali di energia che le sorgenti luminose con forte contenuto infrarosso mostrano da 750 nm a 800 nm.

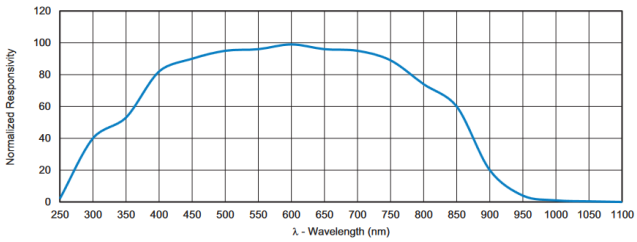


Figure 4.11: Spectral Response White Channel

4.2.2 PV Module - INA219

Analogamente al sensore digitale di irraggiamento è possibile affiancare, al fine di calibrare ed estrarre una forte correlazione tra corrente e tensione e irraggiamento, un modulo fotovoltaico in miniatura.

One Diode Model

Il modulo fotovoltaico si comporta come un generatore di corrente con delle non idealità che risiedono nelle sue resistenze Rs eRp e nella sua tensione massima erogabile.

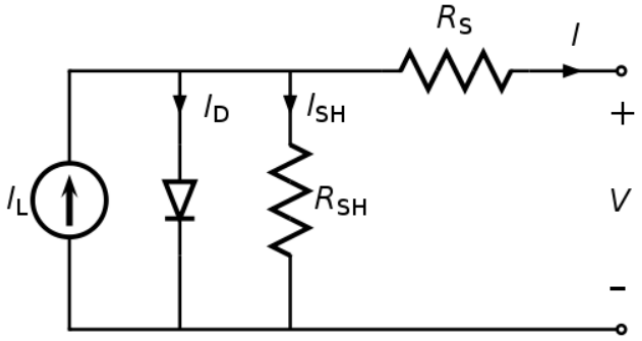


Figure 4.12: One Diode Model

- Rs: è introdotta al fine di considerare il voltage drop interno dovuto alle perdite resistive.
- Rsh: tiene conto delle perdite di corrente quando il diodo è polarizzato inversamente

Come detto in precedenza, il diodo rappresenta la massima tensione che la cella può erogare.

Il sistema tuttavia è caratterizzato dalle seguenti relazioni:

$I = I_{pv} - I_d - I_{sh}$

$U_{sh} = U + IR_s$

$I_{sh} = \frac{U_{sh}}{R_{sh}} = \frac{U+IR_s}{R_{sh}}$

$I = I_{pv} - I_0[\exp \frac{U+IR_s}{nVT} - 1] - \frac{U+IR_s}{R_{sh}}$

G-IV Relationship

Data la diretta proporzionalità tra Irraggiamento e Corrente erogata è possibile stilare dei grafici G vs I da comprendere la relazione che vige tra le due grandezze.

La corrente quindi prodotta può ritenersi proporzionale direttamente all'irraggiamento che la cella riceve.

La proporzionalità diretta tuttavia viene a mancare nel momento in cui ci avviciniamo al regime di circuito aperto dove la corrente per forza di cose diminuisce.

La tensione invece rimane praticamente costante per carichi che vanno dall'ordine dell'*ohm* fino a un valore che dipende dalla cella presa in carico.

Potrebbe essere considerato il fatto di misurare solo la corrente in quanto è essa che e' in relazione con l'irraggiamento, ma in analisi preliminare si vorrebbe tener conto anche della tensione della cella che potrebbe rilevare una correlazione in alcune situazioni .

INA219

L'INA219 calcola la corrente che fluisce attraverso il resistore di rilevamento "Shunt di corrente" misurando prima la tensione attraverso il resistore e quindi applicando la legge di Ohm:

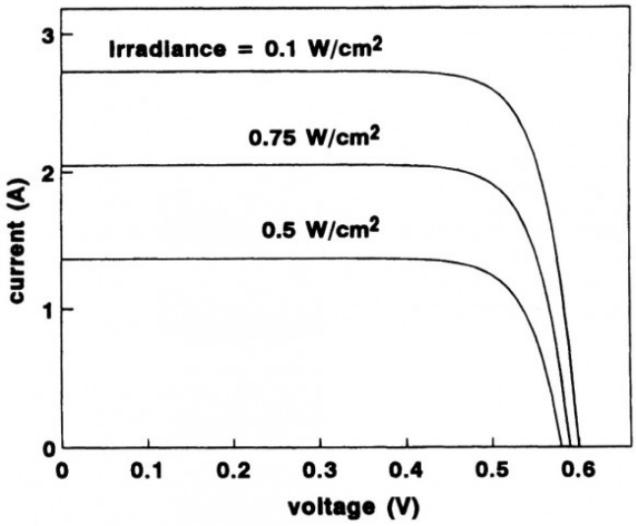


Figure 4.13: G vs I Chart

Figure 4.14: INA 219

$\frac{V_{shunt}}{R_{shunt}} = I_{shunt}$
La tensione del bus viene misurata direttamente sul lato di carico del resistore shunt (VIN-
).
La potenza viene quindi calcolata moltiplicando questi due valori campionati, $Power = I_{shunt} * Vshunt$

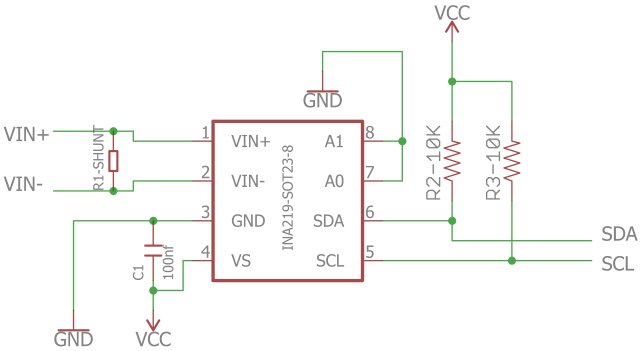


Figure 4.15: Ina 219 Schematic 1

Current Measure

Registers

L'INA21 usa banchi di registri per mantenere le configurazioni, i risultati della misurazione, i valori massimi / minimi e le informazioni generali Un ritardo di 4-micro-s è richiesto dal complemento di un documento, una data registrazione e una lettura successiva di quel registro.

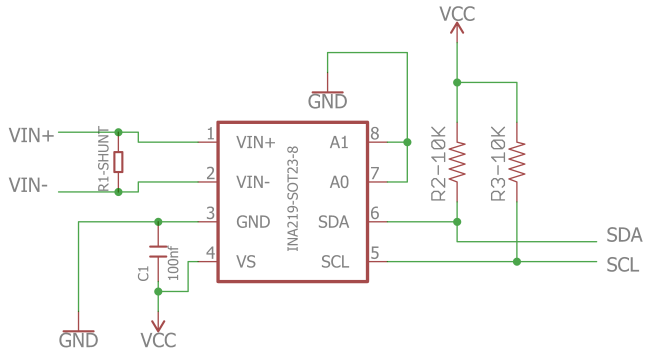


Figure 4.16: Ina 219 Schematic 2

POINTER ADDRESS	REGISTER NAME	FUNCTION	POWER-ON RESET		TYPE(1)
HEX			BINARY	HEX	
00	Configuration	All-register reset, settings for bus voltage range, PGA Gain, ADC resolution/averaging.	00111001 10011111	399F	R/$\overline{W}$
01	Shunt voltage	Shunt voltage measurement data.	Shunt voltage	—	R
02	Bus voltage	Bus voltage measurement data.	Bus voltage	—	R
03	Power(2)	Power measurement data.	00000000 00000000	0000	R
04	Current(2)	Contains the value of the current flowing through the shunt resistor.	00000000 00000000	0000	R
05	Calibration	Sets full-scale range and LSB of current and power measurements. Overall system calibration.	00000000 00000000	0000	R/$\overline{W}$

Figure 4.17: Ina 219 Register Block

4.2.3 AS7262

AS7262 è una soluzione in grado di rilevare differenze di irraggiamento in un contesto multi-spettrale. Questo dispositivo altamente integrato fornisce il rilevamento multispettrale a 6 canali nelle lunghezze d'onda visibili da circa 430 nm a 670 nm con una larghezza massima di canale (FWHM) di 40 nm. Un driver LED integrato con corrente programmabile è fornito per applicazioni di otturatore elettronico. L'AS7262 integra filtri gaussiani in silicio standard CMOS tramite tecnologia di filtro interferenziale nano-ottico depositato ed è confezionato in un pacchetto LGA che fornisce un'apertura incorporata per controllare la luce che entra nel array di sensori. L'accesso ai dati spettrali e di controllo è implementato tramite il set di registri I^2C o con un comando AT Spectral di alto livello impostato tramite un UART seriale.

Block Diagram e Pinout

Internamente il sensore è organizzato come di seguito in questo schema a blocchi:

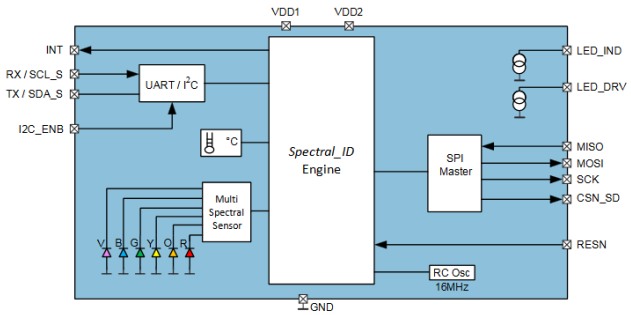


Figure 4.18: AS762 Block Diagram

Measured Spectrum

Ogni canale ha una caratteristica di filtro gaussiana con una larghezza di banda massima (FWHM) di 40 nm. Il sensore contiene convertitori analogico-digitale (risoluzione ADC a 16 bit), che integrano la corrente dal fotodiodo di ciascun canale. Al completamento del ciclo di conversione, il risultato integrato viene trasferito ai registri dati corrispondenti. I trasferimenti sono a doppio buffer per garantire che l'integrità dei dati sia mantenuta. I filtri interferenziali consentono una stabilità alle alte temperature e una deriva del ciclo di vita minima. La precisione del filtro sarà influenzata dall'angolo di incidenza che a sua volta è limitato dall'apertura integrata e dalla struttura interna della microlente. Il campo visivo limitato all'apertura è ± 20,0 ° per fornire una precisione specificata.

Symbol	Parameter	Test Conditions	Channel (nm)	Min	Typ	Max	Unit
V	Channel V	5700K White LED [(2), (4)]	450		45 [(3), (4)]		counts/ (µW/cm²)
B	Channel B	5700K White LED [(2), (4)]	500		45 [(3), (4)]		counts/ (µW/cm²)
G	Channel G	5700K White LED [(2), (4)]	550		45 [(3), (4)]		counts/ (µW/cm²)
Y	Channel Y	5700K White LED [(2), (4)]	570		45 [(3), (4)]		counts/ (µW/cm²)
O	Channel O	5700K White LED [(2), (4)]	600		45 [(3), (4)]		counts/ (µW/cm²)
R	Channel R	Incandescent [(2), (4)]	650		45 [(3), (4)]		counts/ (µW/cm²)
FWHM	Full Width Half Max		40		40		nm
Wacc	Wavelength Accuracy				±5		nm
dark	Dark Channel Counts	GAIN=64, T_{AMB}=25°C				5	counts
PFOV	Package Field of View				±20.0		deg

Figure 4.19: AS7262 Channels List

Il che si traduce in una visione spettrale determinata dal grafico *Lunghezza d'onda* rispetto all'*Intensità Luminosa*

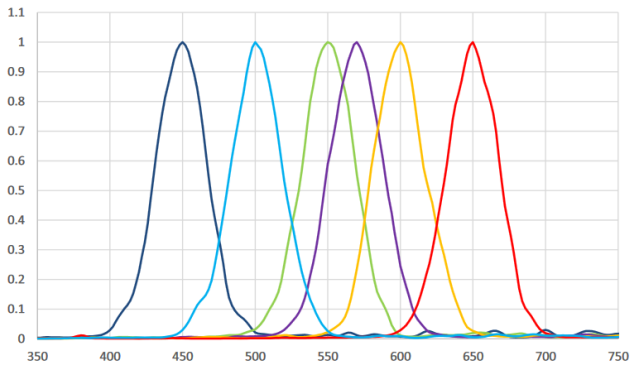


Figure 4.20: AS7262 Spectrum

4.3 UV Content

4.3.1 VEML 6070

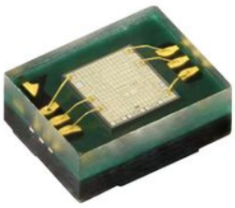


Figure 4.21: VEML 6070

VEML6070 è un sensore di luce ultravioletta (UV) avanzato con interfaccia di protocollo I2C e progettato con processo CMOS. È facilmente gestibile tramite un semplice comando I2C. La funzione di riconoscimento attivo (ACK) con l'impostazione delle finestre di soglia consente al sensore UV di inviare un messaggio di avviso UVI. In condizioni di forte UVI solare, il segnale smart ACK può essere facilmente implementato dalla programmazione del software. VEML6070 incorpora un fotodiodo, amplificatori e circuiti analogici / digitali in un unico chip. L'adozione della tecnologia UV FiltronTM da parte del VEML6070 offre la migliore sensibilità spettrale per coprire il rilevamento dello spettro UV. Ha un'eccellente compensazione della temperatura e una robusta impostazione della frequenza di aggiornamento che non utilizza un filtro passa basso RC esterno. VEML6070 ha una sensibilità lineare alla luce UV solare ed è facilmente regolabile da un resistore esterno. Viene fornita la modalità di arresto del software, che riduce il consumo di energia a meno di 1 microA. La tensione operativa di VEML6070 varia tra 2,7 V e 5,5 V.

Block Diagram e Pinout

Simile al VEML7700 aggiunge tuttavia un pin di *ACK* e uno di *RESET* per il resto è configurato esattamente come il precedente.

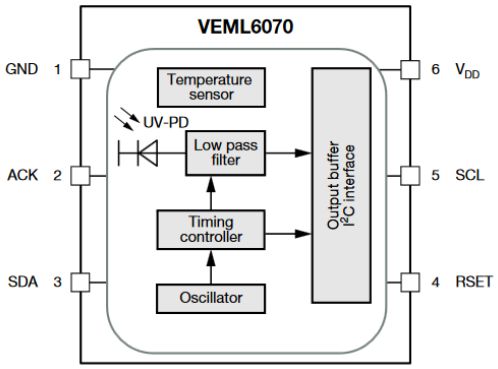


Figure 4.22: VEML 6070 Block Diagram

Measured Spectrum

Lo spettro completa la componente mancante nel sensore precedente incentrando l'attenzione sulla componente *UVA* e parzialmente la *UVB*.

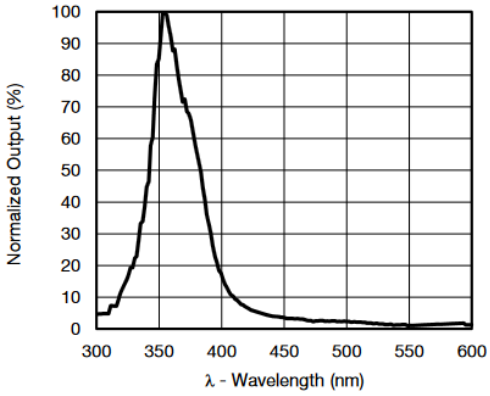


Figure 4.23: VEML 6070 Spectrum

4.3.2 VEML6075

Di più nuova ingegnerizzazione il VELM6075 offre la possibilità di scorporare *UVA* e *UVB* in due dati completamente distaccati e leggibili singolarmente.

Il VEML6075 rileva la luce UVA e UVB e incorpora un fotodiodo, amplificatori e circuiti analogici / digitali in un unico chip usando un processo CMOS. Quando si applica il sensore UV, è in grado di rilevare l'intensità UVA e UVB per fornire una misura della potenza del segnale e consentire l'acquisizione nello spettro degli ultravioletti. Il VEML6075 fornisce un'eccellente capacità di compensazione della temperatura per mantenere stabile l'uscita in caso di variazione della stessa. Le funzionalità di VEML6075 sono facilmente gestibili tramite il formato di comando semplice del protocollo di interfaccia I2C (SMBus compatibile).

Block Diagram e Pinout

Anche questo integrato è simile al 6070 e ed al 7700.

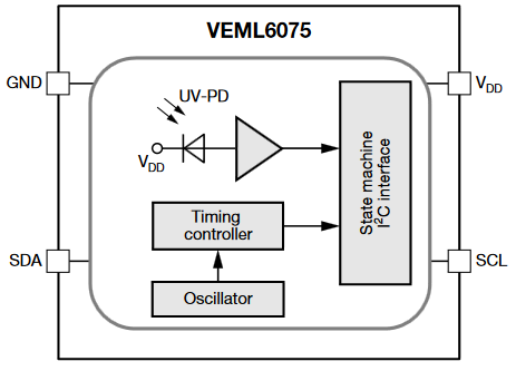


Figure 4.24: VEML 6075 Diagram

Measured Spectrum

Lo spettro del 6075 permette la differenziazione tra *UVA* e *UVB* fornendo dati migliori per un possibile input esogeno correlato ad essi.

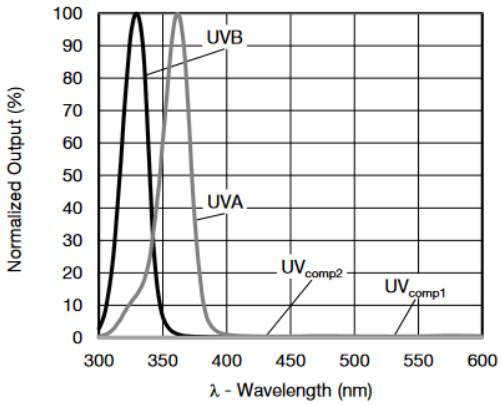


Figure 4.25: VEML 6075 Spectrum Response

Chapter 5

Input Communication

La comunicazione degli input è basata esclusivamente su un bus di comunicazione I^2C che ad oggi nei moduli*Raspberry* e *Arduino* rappresenta l'alternativa alla lettura degli ingressi per via analogica o per mezzo dell'interfaccia SPI

5.1 I^2C

Il protocollo Inter-integrated Circuit (I2C) è un protocollo destinato a consentire a più circuiti integrati digitali "slave" ("chip") di comunicare con uno o più chip "master". Come la Serial Peripheral Interface (SPI), è inteso solo per le comunicazioni a breve distanza all'interno di un singolo dispositivo. Come le interfacce seriali asincrone (come RS-232 o UART), per scambiare le informazioni sono necessari solo due cavi di segnale.

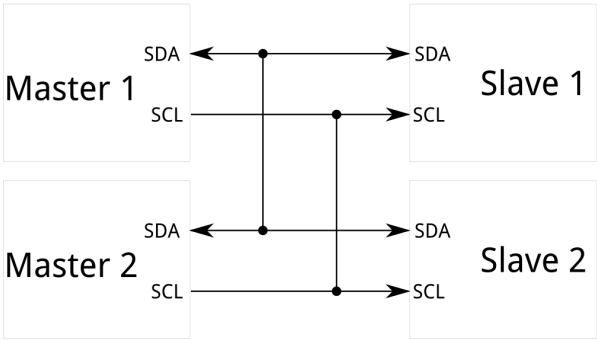


Figure 5.1: I^2C

5.1.1 Advantages vs Uart

Figure 5.2: I^2C Vs Uart

Poiché le porte seriali sono asincrone (non vengono trasmessi segnali di clock), i dispositivi che li utilizzano devono concordare in anticipo la velocità di trasmissione dei dati. I due dispositivi devono anche avere clock simili che rimarranno tali le eccessive differenze tra le frequenze di clock su entrambe le estremità causeranno dati confusi.

Le porte seriali asincrone richiedono un sovraccarico dell'hardware: l'UART su entrambe le estremità è relativamente complessa e difficile da implementare con precisione nel software. Almeno un bit di inizio e fine è una parte di ciascun frame di dati, il che significa che sono necessari 10 bit di tempo di trasmissione per ogni 8 bit di dati inviati,il che va a influire drasticamente sulla velocità dei dati.

Un altro difetto di base nelle porte seriali asincrone e' che sono intrinsecamente adatti alle comunicazioni tra due e solo due dispositivi. Sebbene sia possibile connettere più dispositivi a una singola porta seriale, il conflitto del bus è sempre un problema e deve essere maneggiato con attenzione per evitare danni ai dispositivi in questione, di solito attraverso l'hardware esterno.

Infine, la velocità dei dati è un problema. Sebbene non esista un limite teorico alle comunicazioni seriali asincrone, la maggior parte dei dispositivi UART supporta solo un certo insieme di baud rate fissi, e il più alto di questi è di solito intorno a 230400 bit al secondo.

5.1.2 Advantages vs SPI

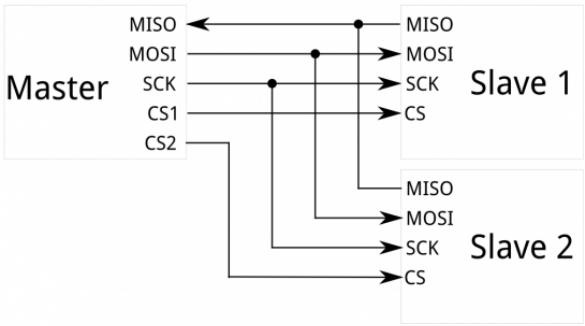


Figure 5.3: I^2C Vs SPI

Lo svantaggio più ovvio di SPI è il numero di pin richiesti. Il collegamento di un singolo master a un singolo slave con un bus SPI richiede quattro linee; ogni slave aggiuntivo richiede un pin di I / O di selezione del chip aggiuntivo sul master. La rapida proliferazione di connessioni e di pin rende indesiderabili situazioni in cui molti dispositivi devono essere asserviti a un master. Inoltre, il numero elevato di connessioni per ciascun dispositivo può rendere più difficili i segnali di routing in situazioni di layout di PCB stretti. SPI consente solo un master sul bus, ma supporta un numero arbitrario di slave soggetto solo alla capacità di guida dei dispositivi collegati al bus e al numero di pin di selezione del chip disponibili.

SPI è adatto per connessioni dati full duplex ad alta velocità (trasmissione e ricezione simultanea di dati), con frequenze di clock fino a 10 MHz (e quindi 10 milioni di bit al secondo) per alcuni dispositivi e la velocità si adatta in modo ottimale. L'hardware alle due estremità è di solito un registro a scorrimento molto semplice, che consente una facile implementazione nel software.

5.1.3 General Advantages I^2

Esso richiede solo due fili, come quelli asincroni seriali, ma questi due fili possono supportare fino a 1008 dispositivi slave. Inoltre, a differenza di SPI, I2C può supportare un sistema multi-master, consentendo a più di un master di comunicare con tutti i dispositivi sul bus,sebbene i dispositivi master non possano parlare tra loro sul bus e debbano fare a turno usando le linee del bus stesso.

Le velocità di trasmissione dei dati sono comprese tra seriale asincrona e SPI; la maggior parte dei dispositivi I2C può comunicare a 100kHz o 400kHz. C'è un sovraccarico con I2C; per ogni 8 bit di dati da inviare, deve essere trasmesso un ulteriore bit di metadati (il bit "ACK / NACK", di cui parleremo in seguito).

L'hardware necessario per implementare I2C è più complesso di SPI, ma inferiore alla seriale asincrona. Può essere abbastanza banalmente implementato nel software.

5.1.4 History I^2

I2C è stato originariamente sviluppato nel 1982 da Philips per vari chip Philips. La specifica originale consentiva solo comunicazioni a 100kHz e solo per indirizzi a 7 bit, limitando il numero di dispositivi sul bus a 112 (ci sono diversi indirizzi riservati, che non saranno mai usati per indirizzi I2C validi). Nel 1992, è stata pubblicata la prima specifica pubblica, aggiungendo una modalità veloce a 400 kHz e uno spazio di indirizzi esteso a 10 bit. Sono disponibili tre modalità aggiuntive: fast-mode plus, a 1MHz; modalità ad alta velocità, a 3,4 MHz; e modalità ultraveloce, a 5 MHz.

Oltre a I2C "vanilla", Intel ha introdotto una variante nel 1995 chiamata "System Management Bus" (SMBus). SMBus è un formato più strettamente controllato, inteso a massimizzare la prevedibilità delle comunicazioni tra gli IC di supporto sulle schede madri del PC. La differenza più significativa tra SMBus è che limita le velocità da 10kHz a 100kHz, mentre I2C può supportare dispositivi da 0kHz a 5MHz. SMBus include una modalità di timeout dell'ora che rende non consigliate le operazioni a bassa velocità, sebbene molti dispositivi SMBus lo supportino comunque per massimizzare l'interoperabilità con i sistemi I2C incorporati.

5.1.5 Funzionamento

Operation

Ogni bus I2C è costituito da due segnali: SCL e SDA. SCL è il segnale di clock e SDA è il segnale di dati. Il segnale di clock viene sempre generato dal master del bus corrente; alcuni dispositivi slave possono forzare il clock a volte a ritardare l'invio di più dati da parte del master (o richiedere più tempo per preparare i dati prima che il master tenti di eseguirne il clock). Questo è chiamato "clock stretching" ed è descritto nella pagina del protocollo.

A differenza delle connessioni UART o SPI, i driver del bus I2C sono "open drain", ovvero possono portare allo stato *logico basso* la corrispondente linea di segnale, ma non possono portarla al livello *high*. Pertanto, non può esserci alcuna contesa del bus in cui un dispositivo sta tentando di guidare la linea in alto mentre un altro tenta di abbassarlo, eliminando il potenziale di danni ai driver o l'eccessiva dissipazione di potenza nel sistema. Ogni linea di segnale ha un resistore di pull-up su di esso, per ripristinare il segnale ad alta quando nessun dispositivo lo sta affermando basso.

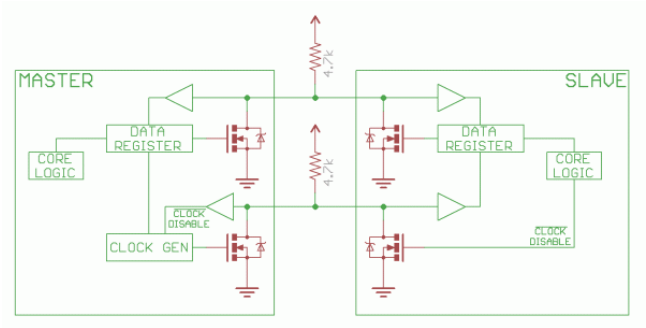


Figure 5.4: SDA and SCL Explained

Signals Levels

Poiché i dispositivi sul bus non portano i segnali a livello *high*, I2C consente una certa flessibilità nel collegamento di dispositivi con diverse tensioni I / O. In generale, in un sistema in cui un dispositivo è a una tensione superiore rispetto a un altro, potrebbe essere possibile collegare i due dispositivi tramite I2C senza alcun circuito di adeguamento di tensione. Il trucco è connettere i resistori pull-up alla minore delle due tensioni. Questo funziona solo in alcuni casi, dove la tensione più bassa tra i due sistemi supera quella ad alto livello del sistema ad alta tensione, ad esempio un Raspberry da 5 V e un sensore da 3,3 V.

Se la differenza di tensione tra i due sistemi è troppo grande (ad esempio, 5 V e 2,5 V), è possibile cambiare il livello di alimentazione.

Communication Protocol

La comunicazione tramite I2C è più complessa rispetto a una soluzione UART o SPI. I segnali deveno rispettare un determinato protocollo in modo che i dispositivi i riconoscano come istruzioni valide.

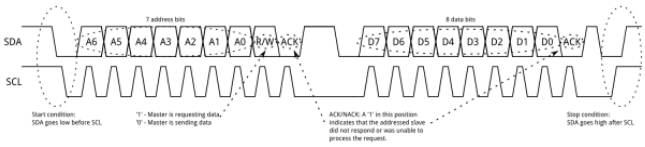


Figure 5.5: Communication

I messaggi sono suddivisi in due tipi di frame: un frame di indirizzo, in cui il master indica lo slave a cui viene inviato il messaggio e uno o più frame di dati, che sono messaggi di dati a 8 bit passati da master a slave

o viceversa . I dati vengono posizionati sulla linea SDA dopo che SCL diventa basso e viene campionato dopo che la linea SCL diventa alta. Il tempo tra il fronte del clock e la lettura / scrittura dei dati è definito dai dispositivi sul bus e varierà da chip a chip.

Start

Per iniziare il frame dell'indirizzo, il dispositivo master lascia SCL alto e impone SDA basso. Questo mette tutti i dispositivi slave in avviso che una trasmissione sta per iniziare. Se due dispositivi master desiderano assumere la proprietà del bus contemporaneamente, il primo dispositivo che imposta SDA su low, vince la *race* e ottiene il controllo del bus. È possibile emettere ripetute partenze, iniziando una nuova sequenza di comunicazione senza rinunciare al controllo del bus verso altri master.

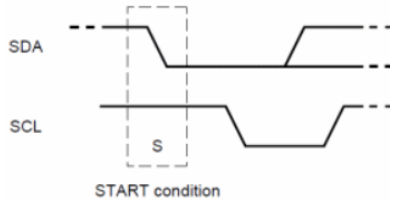


Figure 5.6: Start Frame

Frame Indirizzo

I frame dell'indirizzo è sempre il primo di ogni nuova sequenza di comunicazione. Per un indirizzo a 7 bit, l'indirizzo ha per primo il bit più significativo (MSB), seguito da un bit R / W che indica se si tratta di un'operazione di lettura (1) o scrittura (0).

Il nono bit del frame è il bit NACK / ACK. Questo è il caso di tutti i frame (dati o indirizzo). Una volta inviati i primi 8 bit del frame, il dispositivo ricevente riceve il controllo su SDA. Se il dispositivo ricevente non mette la linea SDA bassa allo stato *low* prima del 9 ° impulso di clock, si può dedurre che il dispositivo ricevente non ha ricevuto i dati o non ha saputo come analizzare il messaggio. In tal caso, la sostituzione si interrompe e spetta al comandante del sistema decidere come procedere.

Frame Data Dopo che il frame dell'indirizzo è stato inviato, i dati possono iniziare a essere trasmessi. Il master continuerà semplicemente a generare impulsi di clock ad intervalli regolari e i dati verranno posizionati su SDA dal master o dallo slave, a seconda che il bit R / W indichi un'operazione di lettura o scrittura. Il numero di frame di dati è arbitrario e la maggior parte dei dispositivi slave incrementerà automaticamente il registro interno, il che significa che le successive letture o scritture verranno dal successivo registro in linea.

Condizione Stop

Una volta che tutti i frame di dati sono stati inviati, il master genererà una condizione di arresto. Le condizioni di arresto sono definite da una transizione 0-¿ 1 (da bassa a alta) su SDA dopo una transizione 0-¿ 1 su SCL, con SCL rimanente alto. Durante la normale operazione di scrittura dei dati, il valore su SDA non dovrebbe cambiare quando SCL è alto, per evitare false condizioni di arresto.

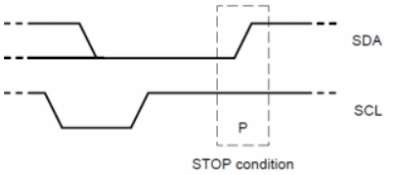


Figure 5.7: Stop Frame

5.2 I^2C Sensor Application e Modalità

Ogni tipo di sensore campiona ed elabora dati in una maniera differente. Questi tre sensori scelti preliminarmente danno luogo a un loro rispettivo *flow* di elaborazione dati.

5.2.1 BME280

Measurament Flow

Data la complessità del sensore, e la presenza di tre *data sensing elements* si rivela necessario svolgere una analisi del flow di misura.

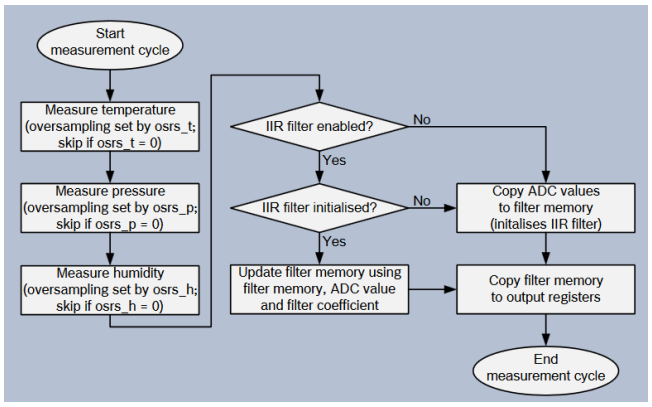


Figure 5.8: BME 280 Block Diagram

Humidity measurement

La misurazione dell'umidità può essere abilitata o saltata. Se abilitata, esistono diverse opzioni di sovracampionamento. La misurazione dell'umidità è controllata dall'impostazione osrs h [2: 0]. Per la misurazione dell'umidità, il sovracampionamento è possibile per ridurre il rumore. La risoluzione della misurazione dell'umidità è fissata all'uscita ADC a 16 bit.

Pressure measurement

La misurazione della pressione può essere abilitata o saltata. Se abilitata, esistono diverse opzioni di sovracampionamento. La misurazione della pressione è controllata dall'impostazione osrs p [2: 0]. Per la misura della pressione, il sovracampionamento è possibile per ridurre il rumore. La risoluzione dei dati di pressione dipende dal filtro IIR e dall'impostazione di sovracampionamento :

- IIR è abilitato:la risoluzione della pressione è 20 bit.

- IIR è disabilitato: la risoluzione della pressione è di è 16 + *(osrs p-1)* bit, ad es. 18 bit quando *osrs pis* è impostato su '3'.

Temperature measurement

La misurazione della temperatura può essere abilitata o saltata. Saltare la misurazione potrebbe essere utile per misurare la pressione estremamente rapidamente. Se abilitato, esistono diverse opzioni di sovracampionamento. La misurazione della temperatura è controllata dall'impostazione osrs t [2: 0]. Per la misurazione della temperatura, il sovracampionamento è possibile per ridurre il rumore. La risoluzione dei dati di temperatura dipende dal filtro IIR e dall'impostazione di sovracampionamento :

- IIR è abilitato:la risoluzione della pressione è 20 bit.

- IIR è disabilitato: la risoluzione della pressione è di è 16 + *(osrs p-1)* bit, ad es. 18 bit quando *osrs pis* è impostato su '3'.

IIR Filter

Il valore di umidità all'interno del sensore non è generalmente instabile e non richiede il filtraggio. Tuttavia, la pressione ambientale è soggetta a molti cambiamenti a breve termine, causati ad es. sbattendo una porta o una finestra o soffiando il vento nel sensore. Per sopprimere questi disturbi nei dati di uscita senza causare ulteriore traffico di interfaccia e carico di lavoro del processore, il BME280 dispone di un filtro IIR interno. Riduce efficacemente la larghezza di banda dei segnali di uscita di temperatura e pressione e aumenta la risoluzione della pressione e della temperatura di uscita a 20 bit.

L'output di un successivo passaggio di misurazione viene filtrato utilizzando la seguente formula:

$$\text{data\_filtered} = \frac{\text{data\_filtered\_old} \cdot (\text{filter\_coefficient} - 1) + \text{data\_ADC}}{\text{filter\_coefficient}}$$

Figure 5.9: Filter Phormula

$Datafilteredold$ sono i dati provenienti dalla memoria del filtro corrente e $dataADC$ sono i dati provenienti dall'acquisizione ADC corrente. $Datafiltered$ è il nuovo valore della memoria del filtro e il valore che verrà inviato ai registri di output.

Filter coefficient	Samples to reach ≥75 % of step response
Filter off	1
2	2
4	5
8	11
16	22

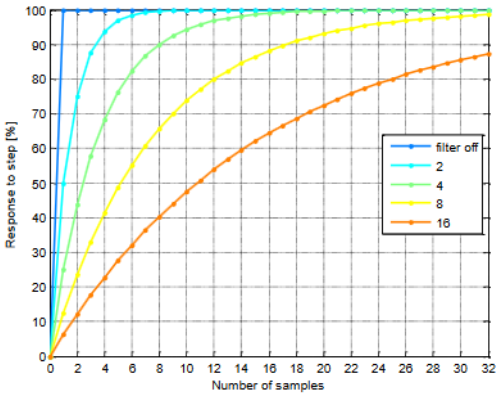


Figure 5.10: Pass Filter

Registers Structure

La struttura della memoria è composta da 16 registri a *8 bit*.

Register Name	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Reset state
hum_lsb	0xFE	hum_lsb<7:0>								0x00
hum_msb	0xFD	hum_msb<7:0>								0x80
temp_xlsb	0xFC	temp_xlsb<7:4>				0	0	0	0	0x00
temp_lsb	0xFB	temp_lsb<7:0>								0x00
temp_msb	0xFA	temp_msb<7:0>								0x80
press_xlsb	0xF9	press_xlsb<7:4>				0	0	0	0	0x00
press_lsb	0xF8	press_lsb<7:0>								0x00
press_msb	0xF7	press_msb<7:0>								0x80
config	0xF5	t_sb[2:0]			filter[2:0]				spi3w_en[0]	0x00
ctrl_meas	0xF4	osrs_t[2:0]			osrs_p[2:0]			mode[1:0]		0x00
status	0xF3					measuring[0]			im_update[0]	0x00
ctrl_hum	0xF2						osrs_h[2:0]			0x00
calib26..calib41	0xE1…0xF0	calibration data								individual
reset	0xE0	reset[7:0]								0x00
id	0xD0	chip_id[7:0]								0x60
calib00..calib25	0x88…0xA1	calibration data								individual

Registers:	Reserved registers	Calibration data	Control registers	Data registers	Status registers	Chip ID	Reset
Type:	do not change	read only	read / write	read only	read only	read only	write only

Figure 5.11: Registers Structure

- Registro 0xD0 **"id"** Il registro "id" contiene l'identificativo del chip numberchip id [7: 0], che è 0x60. Questo numero può essere letto non appena il dispositivo ha terminato il reset all'accensione.
- Registro 0xE0 **"reset"** Il registro "reset" contiene il reset della parola di reset software [7: 0]. Se il valore 0xB6 viene scritto nel registro, il dispositivo viene resettato utilizzando la procedura completa di ripristino dell'accensione. La scrittura di valori diversi da 0xB6 non ha alcun effetto. Il valore di lettura è sempre 0x00.
- Registro 0xF2 **"ctrl hum"** Il registro "ctrl hum" imposta le opzioni di acquisizione dei dati di umidità del dispositivo. Le modifiche a questo registro diventano. effettive solo dopo un'operazione di scrittura su "ctrl meas".
- Registro 0xF3 **"stato"** Il registro "stato" contiene due bit che indicano lo stato del dispositivo.
- Registro 0xF4 **"ctrl meas"** Il registro "ctrl meas" imposta le opzioni di acquisizione dei dati di pressione e temperatura del dispositivo. Il registro deve essere scritto dopo aver modificato "ctrl hum" affinché le modifiche diventino effettive.
- Registro 0xF5 **"config"** Il registro "config" imposta le opzioni therate, filter e interface del dispositivo. Le scritture sul registro "config" in modalità normale possono essere ignorate. Nelle scritture in modalità sleep non vengono ignorate.
- Registro 0xF7 ... 0xF9(msb, lsb, xlsb) Il registro "press" contiene i dati di uscita della misura della pressione grezza su [19: 0].
- Registro 0xFA ... 0xFC(msb, lsb, xlsb) Il registro "temp" contiene i dati di uscita della misurazione della temperatura grezza [19: 0].
- Registro 0xFD ... 0xFE(msb, lsb) Il registro "temp" contiene i dati di uscita della misurazione della temperatura grezza [19: 0].

5.2.2 VELM7700

Light Measurament

VEML7700 contiene sei codici di comando effettivi a 16 bit per il controllo delle operazioni, l'impostazione dei parametri e il buffering dei risultati. Tutti i registri sono accessibili tramite comunicazione I2C. La Figura 7 mostra la comunicazione I2C di base con VEML7700. L'interfaccia I2C integrata è compatibile con le modalità I2C "standard" e "veloce": da 10 kHz a 400 kHz. Intervallo di livello I2C H = da 1,3 V a 3,6 V.

Registers Structure

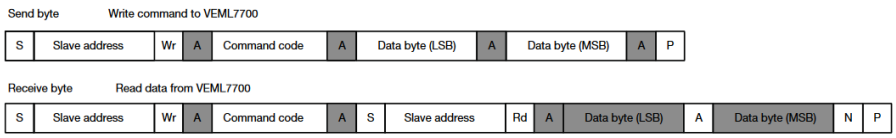


Figure 5.12: Register Structure

- Registro 0x00h: Serve per la configurazione delle misure della luce ambientale.

  Bit [12 : 11]: Bit di set per la sensibilità del sensore.

  Bit [9 : 6]: Bit per set del tempo di integrazione.

  Bit [5 : 4]: Bit Settaggio persistenza.

COMMAND CODE	REGISTER NAME	BIT	FUNCTION / DESCRIPTION	R / W
00	reserved	15 : 13	Set 000b	W
	ALS_SM	12 : 11	Sensitivity mode selection 00 = ALS sensitivity x 1 01 = ALS sensitivity x 2 10 = ALS sensitivity x (1/8) 11 = ALS sensitivity x (1/4)	W
	reserved	10	Set 0b	W
	ALS_IT	9 : 6	ALS integration time setting 1100 = 25 ms 1000 = 50 ms 0000 = 100 ms 0001 = 200 ms 0010 = 400 ms 0011 = 800 ms	W
	ALS_PERS	5 : 4	ALS persistence protect number setting 00 = 1 01 = 2 10 = 4 11 = 8	W
	reserved	3 : 2	Set 00b	W
	ALS_INT_EN	1	ALS interrupt enable setting 0 = ALS INT disable 1 = ALS INT enable	W
	ALS_SD	0	ALS shut down setting 0 = ALS power on 1 = ALS shut down	W
01	ALS_WH	15 : 8	ALS high threshold window setting (MSB)	W
		7 : 0	ALS high threshold window setting (LSB)	W
02	ALS_WL	15 : 8	ALS low threshold window setting (MSB)	W
		7 : 0	ALS low threshold window setting (LSB)	W
04	ALS	15 : 8	MSB 8 bits data of whole ALS 16 bits	R
		7 : 0	LSB 8 bits data of whole ALS 16 bits	R
05	reserved	3 : 2	Set 00b	R
06	ALS_IF_L	15	ALS crossing low threshold INT trigger level	R
	ALS_IF_H	14	ALS crossing high threshold INT trigger level	R
	reserved	13 : 0		

Figure 5.13: Register Structures 2

5.2.3 VEML6070

UV Measurament

VEML6070 contiene un registro comandi a 8 bit scritto tramite il bus I2C. Tutte le operazioni possono essere controllate dal registro di comando. La semplice struttura di comando consente agli utenti di programmare facilmente le impostazioni operative e di leggere i dati da VEML6070. Le sezioni bianche indicano l'attività dell'host e le sezioni grigie indicano il riconoscimento di VEML6070 dell'attività dell'host.

Receive byte → read data from UVS

S	Slave address	Rd	A	Light data (1 byte)	A	P

Send byte → write command to UVS

S	Slave address	Wr	A	Command (1 byte)	A	P

S = start condition
P = stop condition
A = acknowledge
Shaded area = VEML6070 acknowledge

Figure 5.14: UV Registers 1

5.2.4 VELML6075

Il VEML segue uno schema di registri leggermente diverso dal fratello minore VEML6070 in quanto ha a disposizione un altra misura, quella degli *UVB*.

COMMAND CODE	DATE BYTE LOW / HIGH	REGISTER NAME	R / W	DEFAULT VALUE	FUNCTION DESCRIPTION
00h	L	UV_CONF	R / W	0x00	UV integration time, function enable and disable
	H	Reserved	R / W	0x00	Reserved
01h	L	Reserved	R / W	0x00	Reserved
	H	Reserved	R / W	0x00	Reserved
02h	L	Reserved	R / W	0x00	Reserved
	H	Reserved	R / W	0x00	Reserved
03h	L	Reserved	R / W	0x00	Reserved
	H	Reserved	R / W	0x00	Reserved
04h	L	Reserved	R / W	0x00	Reserved
	H	Reserved	R / W	0x00	Reserved
05h	L	Reserved	R / W	0x00	Reserved
	H	Reserved	R / W	0x00	Reserved
06h	L	Reserved	R / W	0x00	Reserved
	H	Reserved	R / W	0x00	Reserved
07h	L	UVA_Data	R	0x00	UVA LSB output data
	H	UVA_Data	R	0x00	UVA MSB output data
08h	L	Dummy	R	0x00	UVD
	H	Dummy	R	0x00	UVD
09h	L	UVB_Data	R	0x00	UVB LSB output data
	H	UVB_Data	R	0x00	UVB MSB output data
0Ah	L	UVCOMP1_Data	R	0x00	UV_{comp1} LSB output data
	H	UVCOMP1_Data	R	0x00	UV_{comp1} MSB output data
0Bh	L	UVCOMP2_Data	R	0x00	UV_{comp2} LSB output data
	H	UVCOMP2_Data	R	0x00	UV_{comp2} MSB output data
0Ch	L	ID	R	0x26	Device ID LSB
	H	ID	R	0x00	Device ID MSB

Figure 5.15: UV Registers 2

5.2.5 AS7262

6 Channel Measurament

La conversione spettrale AS7262 è implementata tramite due banchi di fotodiodi per dispositivo. Il banco 1 è costituito dai dati dei fotodiodi V, G, B, Y. Bank 2 è costituito da dati dai fotodiodi G, Y, O, R. La conversione spettrale richiede il completamento del tempo di integrazione (IT in ms). Se entrambe le banche dei fotodiodi sono obbligate a completare la conversione, la seconda banca richiede un MS IT aggiuntivo. L'IT minimo per una conversione bancaria singola è 2. 8 m s. F u n z io n e d u r i d e r i m i 6 fotodiodi, quindi il dispositivo deve eseguire 2 conversioni complete (2 x Tempo di integrazione). Il processo di conversione spettrale è controllato con le impostazioni della modalità BANK come segue:

Capture Mode

- Modalità Mode 0: i dati saranno disponibili nei registri V, B, G ed Y (i registri O e R saranno zero) con conversioni che si verificano continuamente.
- Modalità BANK 1: i dati saranno essere disponibile nei registri G, Y, O ed R (i registri V e B saranno zero) con conversioni che si verificano continuamente.
- Modalità BANK 2: i dati saranno disponibili nei registri V, B, G, Y, O ed R con conversioni che si verificano continuamente . Quando l'impostazione del banco è Modalità 0, Modalità 1 o Modalità 2, il processo di conversione dei dati spettrali funziona in modo continuo, con nuovi dati disponibili dopo ogni periodo di integrazione IT ms.
- Modalità BANK 3: i dati saranno disponibili nei registri V, B, G, Y, O ed R in modalità One-Shot quando l'impostazione del banco è impostata su Modalità 3, il dispositivo avvia l'operazione One-Shot. Il bit DATA RDY è impostato su 1 una volta che i dati sono disponibili, a indicare che la conversione spettrale è completa in Modalità One-Shot è destinata all'uso quando è fondamentale per garantire che i risultati della conversione spettrale siano ottenuti contemporaneamente.

Dettaglio grafico delle modalità:

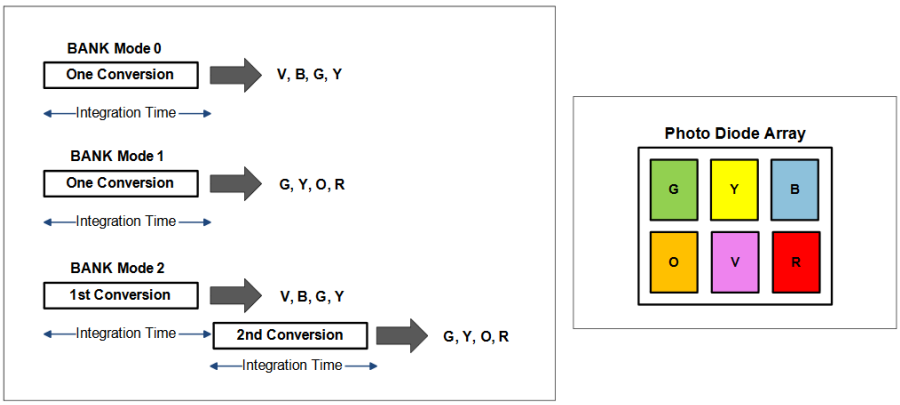


Figure 5.16: Use Mode

Registers Structure

I registri suddividono la lettura da 16 bit in due registri da 8 bit, H e L leggibili separatamente, inoltre presenta è presente anche un sensore di temperatura al fine di correggere i dati letti.

Addr	Name	<D7>	<D6>	<D5>	<D4>	<D3>	<D2>	<D1>	<D0>
Version Registers									
0x00:0x01	HW_Version	Hardware Version							
0x02:0x03	FW_Version	Firmware Version							
Control Registers									
0x04	Control_Setup	RST	INT	GAIN		Bank		DATA_RDY	RSVD
0x05	INT_T	Integration Time							
0x06	Device_Temp	Device Temperature							
0x07	LED_Control	RSVD		ICL_DRV		LED_DRV	ICL_IND		LED_IND
Sensor Raw Data Registers									
0x08	V_High	Channel V High Data Byte							
0x09	V_Low	Channel V Low Data Byte							
0x0A	B_High	Channel B High Data Byte							
0x0B	B_Low	Channel B Low Data Byte							
0x0C	G_High	Channel G High Data Byte							
0x0D	G_Low	Channel G Low Data Byte							
0x0E	Y_High	Channel Y High Data Byte							
0x0F	Y_Low	Channel Y Low Data Byte							
0x10	O_High	Channel O High Data Byte							
0x11	O_Low	Channel O Low Data Byte							
0x12	R_High	Channel R High Data Byte							
0x13	R_Low	Channel R Low Data Byte							

Sensor Calibrated Data Registers		
0x14:0x17	V_Cal	Channel V Calibrated Data (floating point)
0x18:0x1B	B_Cal	Channel B Calibrated Data (floating point)
0x1C:0x1F	G_Cal	Channel G Calibrated Data (floating point)
0x20:0x23	Y_Cal	Channel Y Calibrated Data (floating point)
0x24:0x27	O_Cal	Channel O Calibrated Data (floating point)
0x28:0x2B	R_Cal	Channel R Calibrated Data (floating point)

Figure 5.17: AS7262 Register Structure

Chapter 6

Ambiental Factor Correlation

6.1 UV

Le nuvole alterano lo spettro solare specialmente nella zona che riguarda gli UV e la regione del *royal blue*. Questo è perchè esso e' correlato con il *clearness Index* KT che misura otticamente lo spessore delle nuvole.

Lo spettro si scosta relativamente del ± 10 percento lungo la fascia che va da 400 a 1000nm, tuttavia esistono deviazioni nella zona UV e nella zona del vapore acqueo.

Se si considera lo spettro nella sua integrità sono più di una le zone in cui ci sono scostamenti nello spettro tra situazioni *cloudy* e *sunny*. Tuttavia queste differenze non sono tangibili con l'obbiettivo lowcost del progetto.

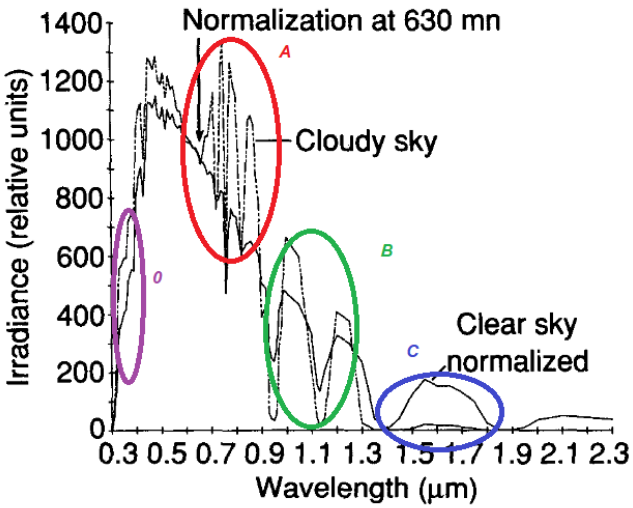


Figure 6.1: UV chart

Se andiamo ad analizzare le 4 zone possiamo constatare che:

- 0 [UVA/B region]: fornisce una importante informazione tra giornate nuvolose o limpide.
- A [Visible region]: facilmente detettabile tramite sensore di luminosità propriamente calibrato nella zona del visibile
- B and C [H_2O e CO2 reflection region]: importante scostamento che tuttavia non riesce ad essere detettato col nostro HW *lowcost*.

Nella parte iniziale appunto, ossia nella fascia UVA esiste un forte scostamento che può essere detettato tramite il sensore *UVA*. Potrebbe essere considerata la seconda zona di correlazione tra i due fenomeni che danno origini a profondi livelli diversi di irraggiamento.

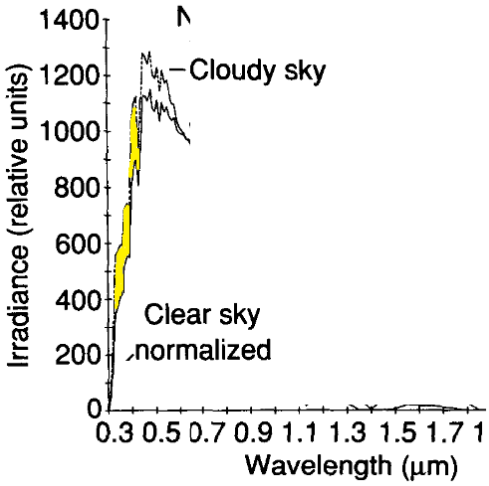


Figure 6.2: Detail 1

Per concludere quindi possiamo classificare con *1* e *2* le regioni su cui risulta più intelligente lavorare a livello di sensoristica con gli strumenti a disposizione.

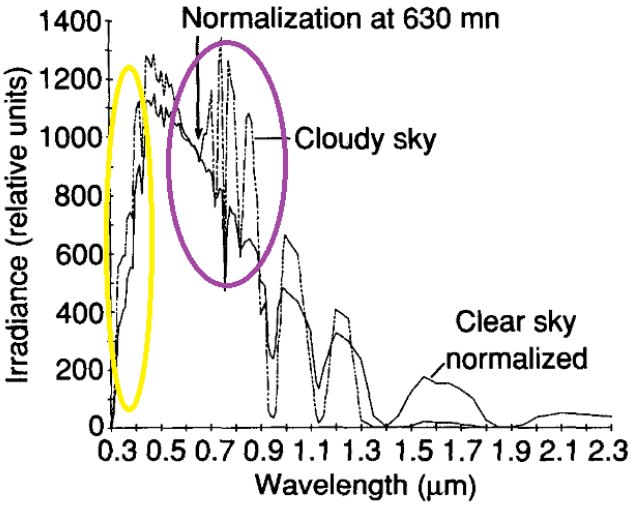


Figure 6.3: Detail 2

Da numerosi paper visti pare evidente come lo spettro ricevuto durante un cielo limpido o uno nuvoloso presenti una netta differenza nel momento in cui si va ad analizzare lo spettro nella zona *sub 400nm*.

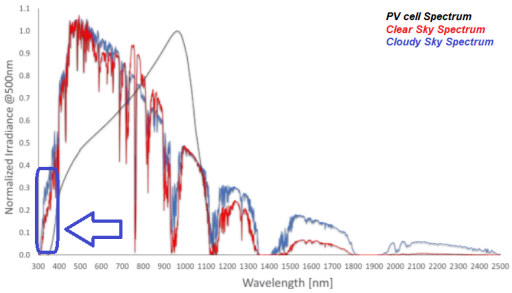


Figure 6.4: Detail 3

Esistono più marcate differenze anche nell'infrarosso ma non sono facilmente detettabili con un hardware low cost.

6.2 Temperature

La temperatura può influenzare il tipo di precipitazione che si forma. Se le condizioni atmosferiche sono tali da provocare precipitazioni e la temperatura è al di sopra del punto di congelamento, potrebbe formarsi della pioggia. Se la temperatura è inferiore al punto di congelamento, potrebbe formarsi neve. Il tipo di precipitazione è anche influenzato dalla temperatura negli strati di atmosfera attraverso cui cade la precipitazione. Ad esempio, se la precipitazione inizia a cadere dalle nuvole come neve e poi passa attraverso gli strati più caldi dell'atmosfera, si trasforma in pioggia. Se la pioggia passa attraverso strati d'aria più freddi, potrebbe formarsi la grandine. A volte, la precipitazione non cadrà affatto. Se gli strati d'aria sono abbastanza asciutti, l'umidità può evaporare prima di raggiungere il suolo.

Esiste una correlazione molto forte tra temperatura e la potenza che viene generata per mezzo di una cella solare.

Le celle solari infatti variano la loro *output power* sotto i cambiamenti di temperatura. Il cambiamento di temperatura influirà sulla potenza prodotta dalle celle. La tensione dipende molto dalla temperatura e un aumento della temperatura ridurrà la tensione massima e sposterà il punto di lavoro.

Importante quindi è misurare le due caratteristiche, ossia tensione e corrente per capire la variazione di potenza prodotta.

Non è necessario misurare solo la corrente perchè essa non presenta correlazione con la temperatura, ma è necessario esaminare anche la tensione che la cella produce.

Questo sistema non funge solamente da dato addizionale da inserire come variabile esogena nel predittore, ma fungerà anche da estimatore della variazione di potenza riportata alla temperatura.

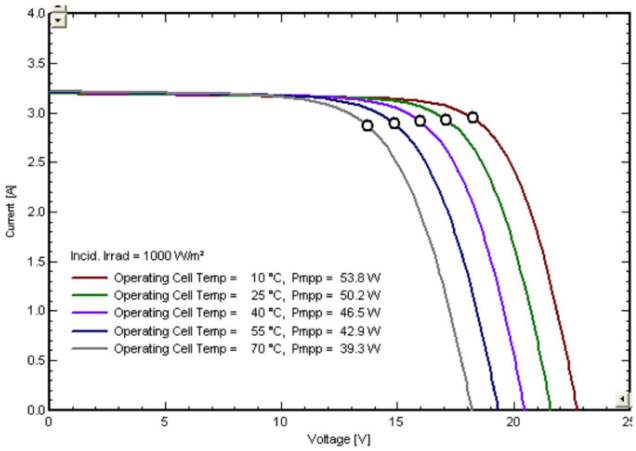


Figure 6.5: V vs I Chart

Come si può intuire dalla figura, un aumento della temperatura a carico e irraggiamento fisso produce una diminuizione tangibile della tensione della cella, ma nessun cambiamento nella corrente.

6.3 Humidity and DewPoint

6.3.1 Relative Humidity

Il calore conduce all'aria sopra la superficie, che causa un causando un sistema instabile. L'aria salirà quindi a causa di un gradiente di pressione verticale, e quando l'aria si raffredda con l'altitudine, l'umidità relativa aumenta, dato che tutti gli altri fattori sull'umidità sono costanti. Quando l'umidità relativa raggiunge il 100% l'aria diventa satura e si formeranno delle nuvole. Questo è il meccanismo per cui si formano le nuvole nella nostra atmosfera.

Citando:

Relation between Cloud Cover and Relative Humidity [George Hannon,Miker Foser, 1998]

> I conclude that there is a relationship between the relative humidity at the sur- face and the cloud cover. And because it's there throughout every season, I also conclude that it does not matter what the cloud type is.

Anche se la correlazione diretta non esiste per tutti i mesi tuttavia questo elemento alla base di un sistema possibile di reti neurali fornisce un input addizionale per il predittore.

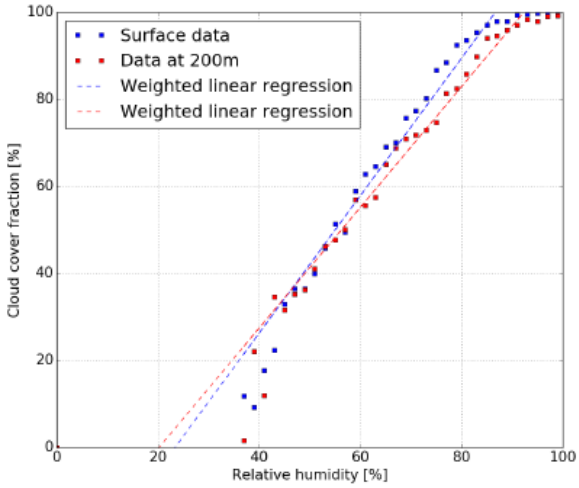


Figure 6.6: HR Correlation

6.3.2 DewPoint

Il punto di rugiada è un indicatore migliore dell'umidità rispetto all'umidità relativa poiché non è una percentuale dipendente dalla temperatura. Il punto di rugiada è la temperatura alla quale l'aria dovrebbe essere raffreddata per saturarsi. Sotto il punto di rugiada, l'acqua si condenserà dall'aria sulle superfici. Al mattino presto, le superfici in erba saranno ricoperte di acqua se la temperatura notturna è scesa sotto il punto di rugiada. Quando l'umidità è alta, la temperatura del punto di rugiada è solo di pochi gradi sotto, o uguale alla temperatura dell'aria. In luoghi asciutti, come i deserti , la temperatura dell'aria può essere di 50 o 60 gradi sopra il punto di rugiada. Generalmente il punto di rugiada è un indicatore più affidabile dell'umidità rispetto all'umidità relativa poiché il punto di rugiada non viene modificato da una variazione della temperatura dell'aria e non oscilla molto durante l'arco della giornata.

Calcoli

Per calcolare la temperatura di DewPoint è possibile approssimare la curva di DewPoint con questa linearizzazione.

- 1-Calcolo del vapore di saturazione: $E_s = 6.11 * 10^{\frac{7.5*Tc}{237.7+Tc}}$
- 2- Pressione di vapore effettiva nell'aria: $E = \frac{R_h * E_s}{100}$
- 3- DewPoint $= \frac{-430.22+237.7*\ln(E)}{-\ln(E)+19.08}$

dove Hr è la umidità relativa in percentuale mentre T_s è la temperatura in gradi *Celsius*.

Tratto da un ulteriore *paper*
Cloud Coverage vs Dew Point [George Hannon,Miker Foser, 1998]

> When the dew point is higher there is more clouds.
>
> The higher the dew point is the more like it was close to the temperature. The closer the temperature is to the dew point the higher more water vapor in the air. The water vapor condenses on dust to form clouds.

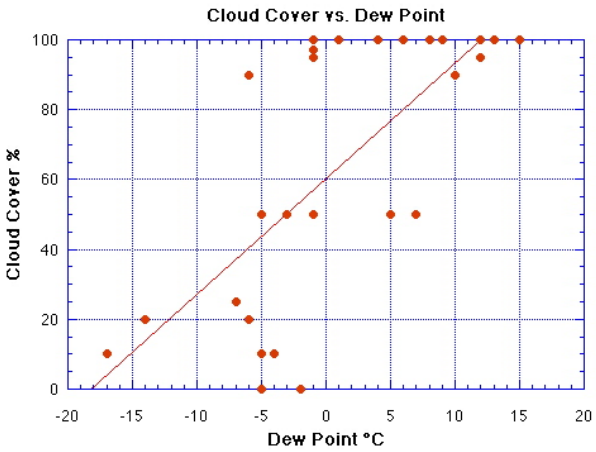


Figure 6.7: Cloud Cover vs DP

Quando il punto di rugiada è più alto ci sono più nuvole ed inoltre più alto è il punto di rugiada, più è vicino alla temperatura ambiente. Quindi il vapore acqueo si condensa sulla polvere per formare nuvole.

Chapter 7

Output Communication

L'importanza di poter scambiare pacchetti dati con l'esterno è un punto fondamentale al fine di poter divulgare la previsione effettuata.

Oltre ad essere uno strumento di *Output* la comunicazione con l'esterno può essere importante anche per recuperare dati che mancano in una soluzione piccola e low cost come la pcb che si vuole sviluppare.

Oggi potrebbe essere anche un valore aggiunto la possibilità di avere come collegamento l'alimentazione e comunicare attraverso un sistema senza fili.

Le modalità di comunicazione possono essere per quanto riguarda il *CM3*:

- Wireless
- Bluetooth
- Ethernet

Il Wifi come soluzione migliore può essere analizzata ma deve presentera la particolarità di avere un range di azione abbastanza elevato da permettere lo scambio di pacchetti tra gateway posti a distanza considerevole data la facile implementazione del sensore sulle falde del tetto.

La connettività non nativa del CM3 riguardo al Wifi ci obbliga a valutare una soluzione comoda e disponibile immediatamente visto anche i tempi relativamente brevi che affliggono il progetto.

La scelta può ricadere su un modulo confermo alla IEEE802.11 b/g/n che è basato su un chip *Ralink RT5370N* ed è supportato così dal Kernel Linux. Può essere pilotato direttamente da periferica usb e viene alimentato a 3.3v.

7.1 RT5370

Figure 7.1: RT 5370

L'RT5370 è un chip singolo altamente integrato di tipo MAC / BBP e 2,4 GHz RF / PA / LNA con supporto di frequenza PHY a 150Mbps. Soddisfa pienamente la specifica IEEE 802.11n e IEEE 802.11 b / g offre una ricca connettività wireless ad elevati standard. L'architettura RF ottimizzata e gli algoritmi sofisticati garantiscono prestazioni eccellenti e bassi consumi energetici. Il design MAC intelligente implementa un motore DMA ad alta efficienza e acceleratori di elaborazione dati hardware senza sovraccaricare il processore. L'RT5370 è progettato per supportare funzionalità basate su standard nelle aree di sicurezza, qualità del servizio e regolamentazione internazionale, offrendo agli utenti finali le massime prestazioni in qualsiasi momento e in ogni circostanza.

Disponibile in 2 versioni:

- WIFI-2: necessita di una antenna supplementare di tipo i-PEX MHF

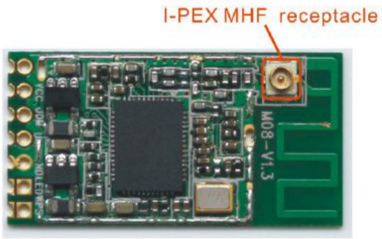


Figure 7.2: EXT Antenna

- WIFI-2-IA: non necessita di nessuna antenna esterna in quanto l'antenna è contenuta sulla board.

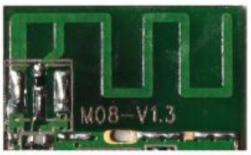


Figure 7.3: Internal Antenna

7.1.1 Block Diagram

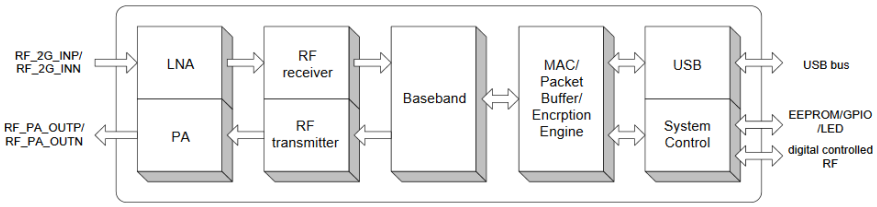


Figure 7.4: RT5370 Block Diagram

La scelta ricadrebbe nel caso nel modello con antenna, schematizzato nel seguente schema a blocchi:

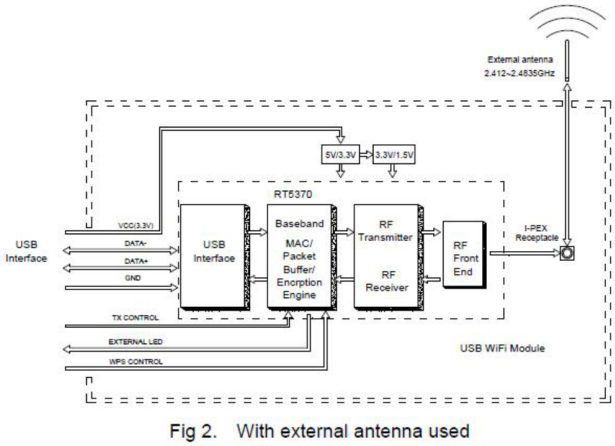


Figure 7.5: RT5370 Block Diagram with Antenna

7.1.2 Register

Il pacchetto dei registri si pone come un classico pacchetto dei registri per un adattatore Wifi:

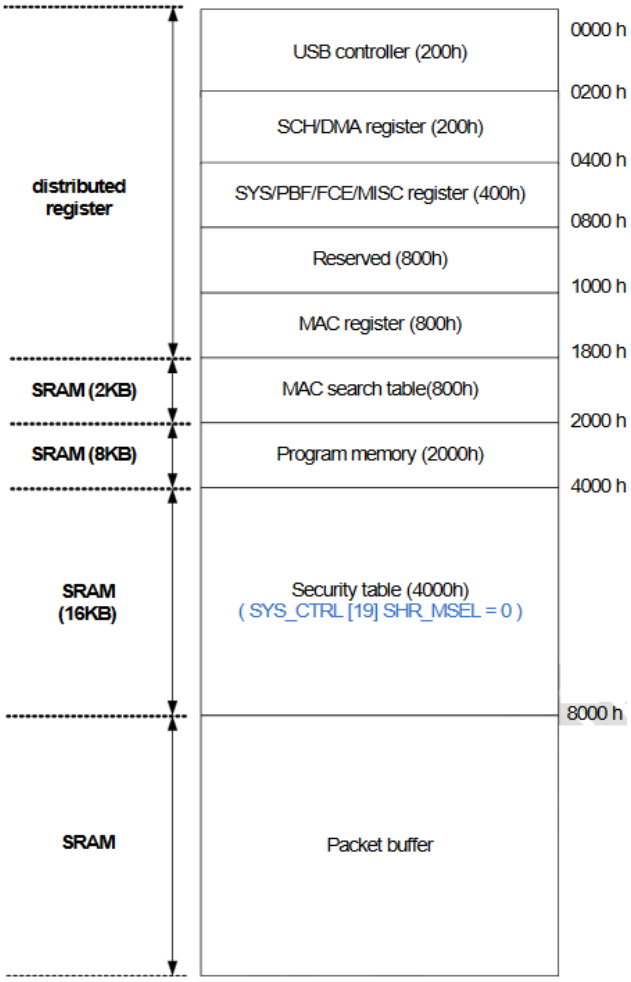


Figure 7.6: RT5370 Block Diagram

Chapter 8

CM3

Recentemente è stata presentata la versione 3 della Compute Module ispirata all'architettura della Raspberry PI 3. Questa scheda è pensata principalmente per un utilizzo industriale e per essere installata in sistemi embedeed. Questa soluzione basata sul processore Broadcom BCM2837 (1.2GHz, 64bit, quad-core, 1GB Ram) permette di avere un modulo molto più performante del precedente (presentato nel 2014 e basato sul processore BCM2835) mantenendo, in gran parte, la retro compatibilità col Compute Module 1. La versione standard del CM3 è dotata di un modulo di memoria eMMC da 4GB che permette di ospitare il sistema operativo mentre la versione Lite è sprovvista di tale memoria e quindi sarà necessaria una memoria SD esterna come avviene per la Raspberry PI 3.

Figure 8.1: Raspberry CM3

Il Compute Module 3 è molto compatto, le sue dimensioni sono quelle di una classica memoria DDR2 SoDIMM. Questo form factor permette l'installazione del modulo tramite un socket SoDIMM montato sul proprio pcb.

8.1 CM3 Regular vs Lite

Disponibile anche in versione Lite non presenta l'equipaggiamento con memoria eMMC ma gode solamente della possibilità di avere una SD card.

La versione CM3l presenta 30mm di altezza a fronte dei 31mm della regular e defice nella possibilità di assorbire alte richieste di corrente per lunghi periodi. Il Pinout rimane tuttavia invariato .

Disponibile fondamentalmente in 2 versioni

8.2 Hardware Periphery

La parte Hardware del CM3 si compone di :

- 48x GPIO
- 2x I2C
- 2x SPI
- 2x UART
- 2x SD/SDIO
- 1x HDMI 1.3a
- 1x USB2 HOST/OTG
- 1x DPI (Parallel RGB Display)
- 1x NAND interface (SMI)
- 1x 4-lane CSI Camera Interface (up to 1Gbps per lane)
- 1x 2-lane CSI Camera Interface (up to 1Gbps per lane)
- 1x 4-lane DSI Display Interface (up to 1Gbps per lane)
- 1x 2-lane DSI Display Interface (up to 1Gbps per lane)

8.3 Software Periphery

- ARMv6 (CM1) or ARMv7 (CM3, CM3L) Instruction Set
- Mature and stable Linux software stack

  Latest Linux Kernel support

  Many drivers upstreamed

  Stable and well supported userland

  Full availability of GPU functions using standard APIs

8.4 Mechanical Characteristics

I moduli di calcolo sono conformi alle specifiche meccaniche JEDEC MO-224 per moduli SODIMM a 200 pin DDR2 (1,8 V) (con l'eccezione che i moduli CM3, CM3L hanno un'altezza di 31 mm anziché 30 mm di CM1) e pertanto potrebbero **NON** funzionare con molti DDR2 SODIMM. Il SODIMM è stato scelto come un modo per fornire le connessioni a 200 pin usando uno standard, connettore facilmente disponibile e di basso costo compatibile con la fabbricazione di PCB a basso costo. L'altezza massima del componente sul lato inferiore del modulo di calcolo è 1,2 mm. L'altezza massima del componente sul lato superiore del modulo di calcolo è 1,5 mm. Lo spessore del PCB del modulo di calcolo è 1,0 mm +/- 0,1 mm. Si noti che la posizione e la disposizione dei componenti sul Modulo possono variare leggermente a causa di revisioni per costi e considerazioni di produzione; tuttavia, le massime altezze dei componenti e lo spessore del PCB saranno mantenute come specificate.

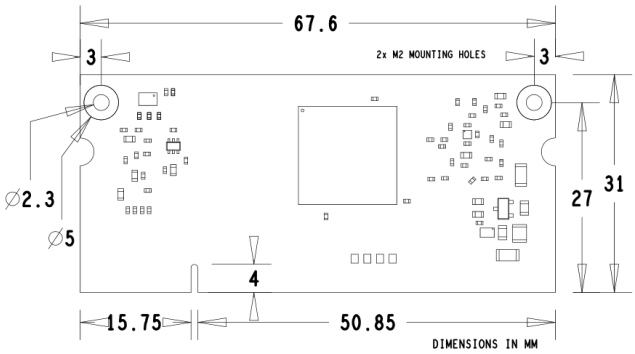


Figure 8.2: CM3 Dimensional Draw

8.5 Block Diagram

Interessante la descrizione *Block Diagram* del CM3 sui suoi 200pin.

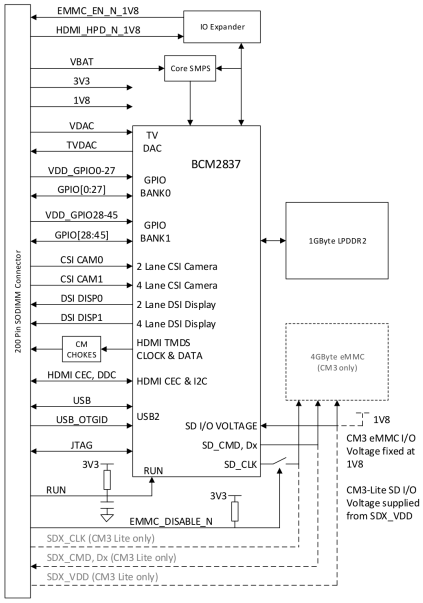


Figure 8.3: Block Diagram

Enorme il numero di GPIO pari a 54 inoltre sono possibili comunicazioni con interfaccia CSI (solo video seriale) possibilità di HDMI e uscita video composita TVDAC.

Il modulo CM3 di Raspberry non è utilizzabile in modalità *StandAlone* per cui per poter approcciare il suo utilizzo è necessario valutarlo nel contesto di una Dev Kit Board.

Queste Dev Kit Board sono prodotte da diversi brand ma ne esiste una ufficiale distribuita da *Raspberry.*

8.6 First Start Configuration

Al fine di poter rendere il modulo CM3 operativo si rivela necessario, partendo da una unità usata poter effettuare la formattazione e una nuova installazione pulita.

Purtroppo non godendo di una capacità di archiviazione elevata, ossia $4gb$ è necessaria l'installazione di una distribuzione *Linux* minimale e custom, escludendo tutti i pacchetti non necessari.

Questo perchè una installazione classica appesantirebbe il dispositivo che gestendo un linguaggio *scripting* risulta già particolarmente lento.

Le distribuzioni che possono essere installate sono molteplici, tuttavia la soluzione più efficace si rivela l'installazione della *Raspbian Lite* senza *GUI.*

Pin Name	DIR	Voltage Ref	PDN[a] State	If Unused	Description/Notes
RUN and Boot Control (see text for usage guide)					
RUN	I	3V3[b]	Pull High	Leave open	Has internal 10k pull up
EMMC_DISABLE_N	I	3V3[b]	Pull High	Leave open	Has internal 10k pull up
EMMC_EN_N_1V8	O	1V8	Pull High	Leave open	Has internal 2k2 pull up
GPIO					
GPIO[27:0]	I/O	GPIO0-27_VDD	Pull or Hi-Z[c]	Leave open	GPIO Bank 0
GPIO[45:28]	I/O	GPIO28-45_VDD	Pull or Hi-Z[c]	Leave open	GPIO Bank 1
Primary SD Interface[d,e]					
SDX_CLK	O	SDX_VDD	Pull High	Leave open	Primary SD interface CLK
SDX_CMD	I/O	SDX_VDD	Pull High	Leave open	Primary SD interface CMD
SDX_Dx	I/O	SDX_VDD	Pull High	Leave open	Primary SD interface DATA
USB Interface					
USB_Dx	I/O	-	Z	Leave open	Serial interface
USB_OTGID	I	3V3		Tie to GND	OTG pin detect
HDMI Interface					
HDMI_SCL	I/O	3V3[b]	Z[f]	Leave open	DDC Clock (5.5V tolerant)
HDMI_SDA	I/O	3V3[b]	Z[f]	Leave open	DDC Data (5.5V tolerant)
HDMI_CEC	I/O	3V3	Z	Leave open	CEC (has internal 27k pull up)
HDMI_CLKx	O	-	Z	Leave open	HDMI serial clock
HDMI_Dx	O	-	Z	Leave open	HDMI serial data
HDMI_HPD_N_1V8	I	1V8	Pull High	Leave open	HDMI hotplug detect
CAM0 (CSI0) 2-lane Interface					
CAM0_Cx	I	-	Z	Leave open	Serial clock
CAM0_Dx	I	-	Z	Leave open	Serial data
CAM1 (CSI1) 4-lane Interface					
CAM1_Cx	I	-	Z	Leave open	Serial clock
CAM1_Dx	I	-	Z	Leave open	Serial data
DSI0 (Display 0) 2-lane Interface					
DSI0_Cx	O	-	Z	Leave open	Serial clock
DSI0_Dx	O	-	Z	Leave open	Serial data
DSI1 (Display 1) 4-lane Interface					
DSI1_Cx	O	-	Z	Leave open	Serial clock
DSI1_Dx	O	-	Z	Leave open	Serial data
TV Out					
TVDAC	O	-	Z	Leave open	Composite video DAC output
JTAG Interface					
TMS	I	3V3	Z	Leave open	Has internal 50k pull up
TRST_N	I	3V3	Z	Leave open	Has internal 50k pull up
TCK	I	3V3	Z	Leave open	Has internal 50k pull up
TDI	I	3V3	Z	Leave open	Has internal 50k pull up
TDO	O	3V3	O	Leave open	Has internal 50k pull up

Figure 8.4: Pinout Detail

Con un peso di 380mb e 1080mb rispettivamente da installata è la distro che occupa meno risorse fisiche del sistema.

- Formattazione
  La prima fase riguarda la formattazione del dispositivo e avviene in ambiente windows scaricando lo specifico applicativo **RPiBoot.exe**.

  Una volta entrato in comunicazione il CM3 viene riconosciuto come una normale periferica esterna e quindi è possibile formattarla in **FAT32**.

- Flashing
  Così successivamente è possibile installare, una volta estratta, la distro di Linux che così sarà scritta sulla EMMC. Una volta installata occupa 1080mb sulla memoria flash.

- Package Installation
  E' necessario snellire il SO utilizzando e installando i pacchetti strettamenti necessari.

  In sequenza sono stati installati:

  LXQt - The Lightweight Qt Desktop Environment: Desktop minimale

  Synaptic - App Installer: Gestore e installatore di APP

  Spider - Python IDE: Ambiente di sviluppo per linguaggio Python

  Pandas - Data Analysis Library: Addon per la gestione di dati

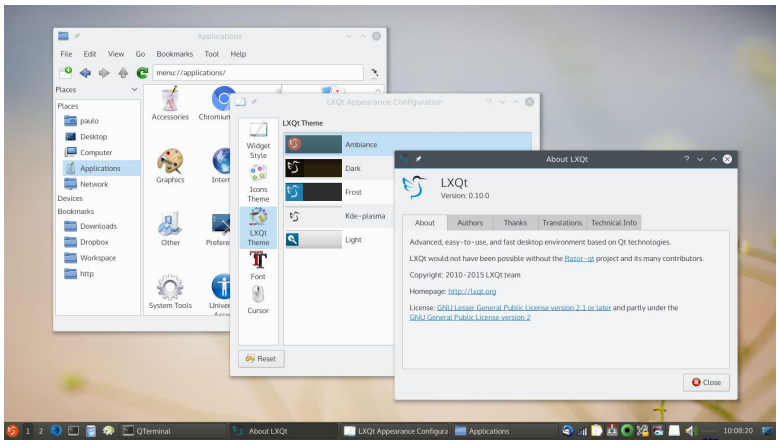


Figure 8.5: LXQT

Chapter 9

Dev Board Prototiping

La prima analisi dell'hardware non che del software è stata effettuata utilizzando la board fornita dal dipartimento DACD.

La board ospita 120 GPIO e una porta HDMI , USB e due camera host.

Figure 9.1: Dev Kit

Chapter 10

Math Models

10.1 Naive Models

10.1.1 Persistence

Persistence model uses the previous day (or the corresponding day in the previous week) as a prediction.

From: Big Data Application in Power Systems, 2018

Il modello di persistenza è spesso usato come riferimento per determinare lo skill Factor. È utile sapere se un modello di previsione fornisce risultati migliori di qualsiasi modello di riferimento banale, che è il modello di persistenza.

Il modello di persistenza considera che la radiazione solare a *t + 1* come uguale alla radiazione solare a *t* inoltre oresume che le condizioni atmosferiche siano stazionarie. È anche chiamato ***naive predictor***.

Possiamo dunque come ipotesi affermare che:
$Gt + 1 = Gt$

La sua precisione diminuisce con l'ampliarsi dell'orizzonte temporale e generalmente non è una applicazione che si utilizza per *forecast* a più di 1 ora.

Una tecnica comune più utilizzata nelle previsioni solari è Smart Persistence. Questo presuppone l'energia solare rimarrà costante nel tempo. In un dato momento t, la Persistenza predice nel futuro h minuti avanti al punto temporale t + h come l'energia fotovoltaica misurata in t. Pertanto, siccome la persistenza segue PV (t + h) = PV (t). Il modello di previsione della persistenza è preciso sugli orizzonti vicini, ma il l'accuratezza diminuisce all'aumentare dell'orizzonte temporale.

Smart Persistence è una versione migliorata di Persistenza che presuppone che le condizioni del cielo rimarranno costante (invece di irradianza stessa). L'algoritmo di previsione prevede la radiazione solare attuale come il prodotto dell'attuale rapporto di cielo sereno (Kt) e radiazione di cielo sereno (Ics) nel punto previsto. È ampiamente usato come linea di base per la convalida di modelli più complessi ed è abbastanza preciso negli orizzonti a breve termine.

$$G_{t+1} = G_t \frac{G_{t+1}^{clearsky}}{G_t^{clearsky}}$$

Numerosi studi dimostrano l'efficacia di questa previsione semplice: la persistenza. La persistenza è estremamente dettagliata e gli autori di diversi *paper* scrivono:

It has been found that for short time horizons, beating persistence models is a difficult task [...]

e hanno dimostrato che, spesso, la persistenza è il miglior metodo da usare per short-casting (¡1 ora) e ora casting (1h-6h). In diversi studi, la semplice persistenza consente di ottenere risultati molto buoni per i quali la differenza in termini di errore di predizione è ottim. Per quanto riguarda il confronto tra apprendimento automatico e persistenza smart, questa differenza è ancora minore. Per condizioni di cielo sereno stabile , i metodi non lineari migliorano leggermente la persistenza smart. Per condizioni instabili del cielo, la discrepanza tra i metodi di apprendimento della macchina e i modelli semplici è più 155 pronunciata con una differenza media del 2% nRMSE.

In conclusione, sembra che la persistenza dovrebbe fornire una previsione interessante . Tuttavia,è da tenere presente che le dinamiche atmosferiche hanno un'importanza maggiore e non possono essere eliminate dai predittori senza influire sulle loro prestazioni, specialmente quando l'orizzonte temporale di previsione è maggiore di 1 ora. Quindi, in teoria, questo tipo di predizione basata sulla persistenza del fenomeno è dedicato agli orizzonti molto brevi e non sarà mai potente quanto i modelli basati su altre dinamiche atmosferiche.

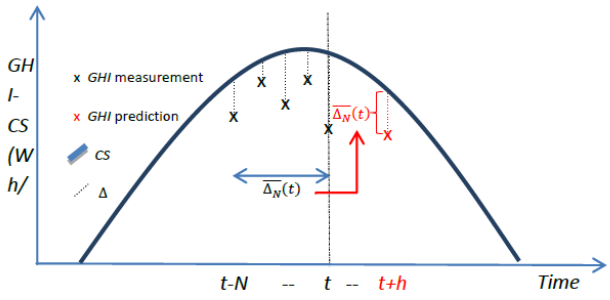


Figure 10.1: Persistence Model

10.2 Machine Learning Models

10.2.1 Arma

Auto Regressive Mobile Average (ARMA) - Il modello ARMA è un modello predittivo con due parti separate, quella auto regressiva e quella media mobile. Questo modello è in grado di prevedere i valori futuri delle serie temporali da una combinazione di valori passati delle serie temporali e un rumore bianco:

$$\widehat{CSI}(t+h) = \varepsilon_t + \sum_{i=0}^{p} \varphi_i \,.\, CSI(t-i) + \sum_{i=0}^{q} \theta_i \,.\, \varepsilon(t-i)$$

Figure 10.2: Arma Phormula

CSI (t + h) la serie temporale dell'indice di cielo sereno al tempo t + h, φ e θ ϵ i parametri della parte autoregressiva e media mobile dedotta da un metodo di minimi quadrati, p e q sono gli ordini del modello e ϵ (t) è un termine di errore distribuito come un rumore bianco gaussiano per il tempo t.

10.2.2 Multi-Layer Perceptron (MLP)

L'MLP è un tipo di reti neurale artificiale, in questa rete un MLP feed forward con due strati (uno nascosto e uno strato di uscita) viene utilizzato con, in input, le serie temporali di irradiazione solare [9]. La dimensione della matrice di input è definita in base ai risultati del metodo reciproco automatico applicato ai dati solari. La formula matematica per un MLP con uno strato nascosto di m neuroni, un neurone di uscita e le variabili di input p è una funzione descritta da:

$$\widehat{CSI}(t+h) = \sum_{j=1}^{m} \omega_{j,s} \,.\, (g.\left(\sum_{i=0}^{n-1} \omega_{i,j}.\, CSI(t-j) + b_j\right) + b_s$$

Figure 10.3: MLP Phormula

con CSI il vettore di input dell'indice di cielo sereno costituito dai valori di n variabili, *CSI* (t + h) il vettore di uscita con i valori previsti del modello, *BJ* il bias del neurone nascosto J e ω (i, j) il peso influenzato all'ingresso misurato CSI (t). g è la funzione di trasferimento dei neuroni nascosti, bs la polarizzazione del neurone di uscita e ω (j, s) il suo peso influenzato dall'output del neurone nascosto j. L'ottimizzazione dell'MLP viene effettuata con una tecnica classica: vengono testate diverse configurazioni con un numero diverso di nodi nascosti (con un numero di nodi nascosti variabile tra 3 e n + 2) e viene selezionato il più efficiente.

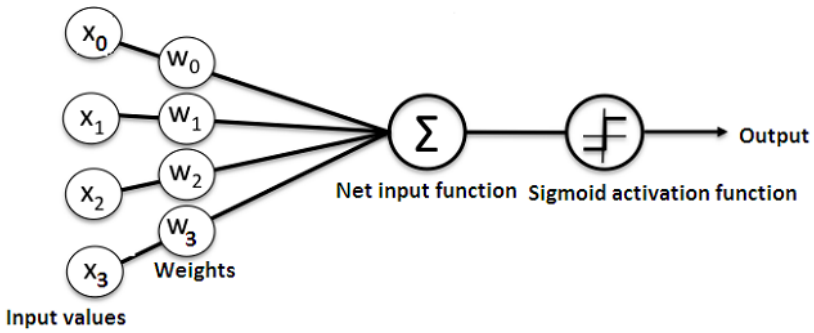


Figure 10.4: MLP Graph

10.2.3 Decision Tees

Gli alberi decisionali basati sulle regole "If-Then" sono uno dei metodi più popolari utilizzati nell'apprendimento automatico per la classificazione, poiché offrono risultati che possono essere facilmente interpretati. Pertanto, questo approccio ottiene serie ordinate di regole If-Then per la previsione che producono modelli comprensibili. Negli ultimi anni, gli algoritmi di apprendimento che generano alberi decisionali per i problemi di classificazione sono stati estesi per prevedere i valori degli attributi quando sono numerici. Queste estensioni hanno portato ad alberi di regressione. Un albero di regressione (RT) è un albero decisionale in cui i nodi foglia sono stati impostati come modelli di regressione e, pertanto, è possibile prevedere valori numerici continui.

Esistono fondamentalmente due tipologie di alberi decisionali (RT-boosted e RT-bagged).Negli RT, gli alberi successivi danno maggior peso ai punti erroneamente predetti dai predittori precedenti. Alla fine, viene presa una votazione ponderata per la predizione. Nel bagged , gli alberi successivi non dipendono da alberi precedenti, ciascuno dei quali è costruito indipendentemente utilizzando un campione bootstrap del set di dati. Alla fine, viene presa una votazione a maggioranza semplice per la predizione. Il metodo di potenziamento consiste nell'assemblare deboli classificatori RT e prendere la media delle previsioni al fine di migliorare l'efficienza. In questo caso un predittore debole è un semplice albero di regressione a divisione singola. La tecnica principale è che gli alberi successivi danno più pesi ai dati che hanno una cattiva previsione nel punto precedente, e al termine della simulazione, un voto ponderato per la previsione.

10.2.4 Random forest

Breiman ha proposto il metodo random forest , che aggiungono un ulteriore livello di casualità al bagging. Oltre a una costruzione per ogni albero che utilizza un campione di dati bootstrap differente, le foreste casuali modificano la modalità di costruzione degli alberi di regressione. Negli alberi standard, ogni nodo viene suddiviso usando la migliore divisione tra tutte le variabili. In una foresta casuale, il set di dati è suddiviso equamente in campioni ma ciascun albero di regressione cresce in modo diverso, ogni nodo viene diviso usando il migliore tra un sottogruppo di predittori scelti casualmente su quel nodo. Questo miglioramento da casualità conferisce robustezza al modello e riduce i rischi di over-training.

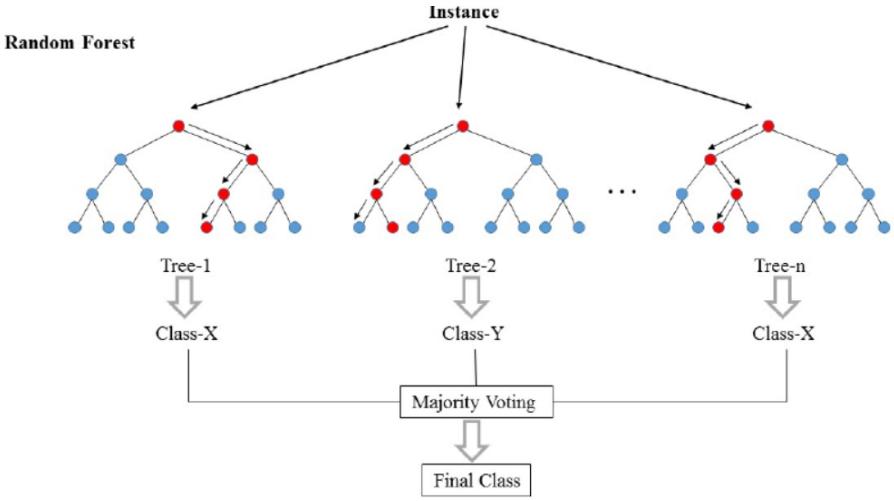


Figure 10.5: Random Forest Sketch

10.2.5 Neuronal Network

Al giorno d'oggi il deep learning è un argomento caldo.

Le reti neurali sono ad oggi la la tecnologia più matura per quanto riguarda il Deep Learning. Anche se possono sembrare delle scatole nere, stanno cercando di realizzare la stessa cosa di qualsiasi altro modello ossia per fare buone previsioni.

Le reti neurali sono reti multi-strato di neuroni (i nodi blu e magenta nella tabella seguente) che usiamo per classificare le cose, fare previsioni. Di seguito è riportato il diagramma di una semplice rete neurale con cinque ingressi, 5 uscite e due strati nascosti di neuroni.

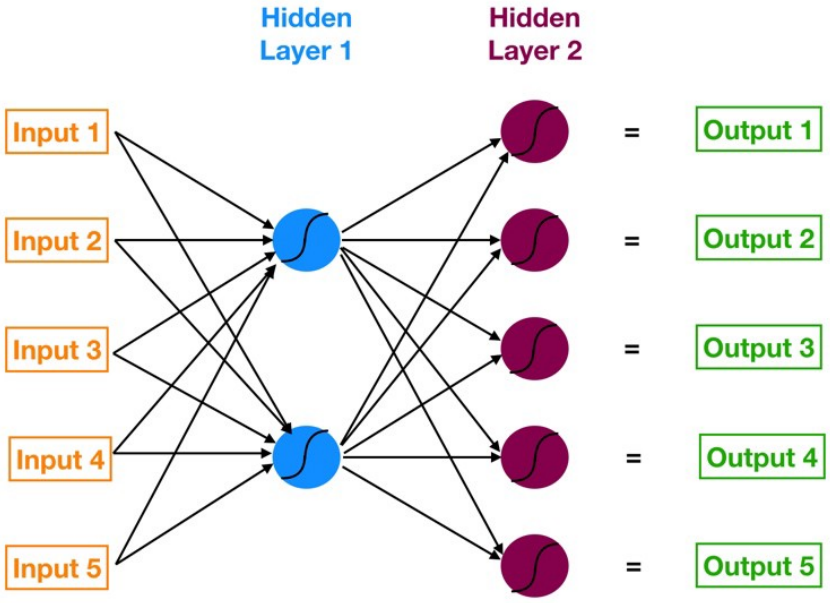


Figure 10.6: Neuronal Network

A partire da sinistra, abbiamo:

- Il livello di input del nostro modello in arancione.
- Il nostro primo strato nascosto di neuroni in blu.
- Il nostro secondo strato nascosto di neuroni in magenta.
- Il livello di output, ossia la previsione del nostro modello in verde.

Le frecce che collegano i punti mostrano come tutti i neuroni sono interconnessi e come i dati viaggiano dal livello di input fino al livello di output.

Successivamente calcoleremo passo per passo ciascun valore di output. Osserveremo anche come la rete neurale apprende dal suo errore usando un processo noto come backpropagation.

Abbiamo una serie di input e una serie di valori target e stiamo cercando di ottenere previsioni che corrispondano il più possibile a tali valori target.

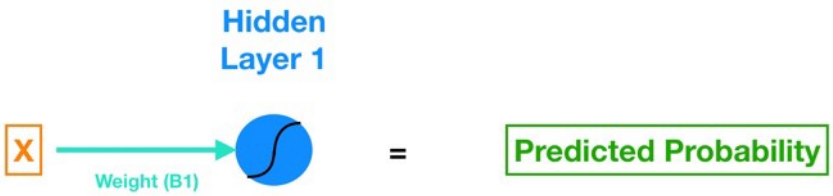


Figure 10.7: Neuronal Network 2

Questa è una regressione logistica a singola caratteristica, ossia il modello ha una sola variabile esogena X, espressa attraverso una rete neurale . Per vedere come si collegano si può l'equazione della regressione logistica usando i nostri codici colore della rete neurale.

$$\textit{Sigmoid}\,(B_1 * X + B_0) = \textit{Predicted Probability}$$

Figure 10.8: Neuronal Network 3

Esaminando ogni elemento:

- X (in arancione) è il nostro input, la caratteristica solitaria che diamo al nostro modello per calcolare una previsione.
- B1 in turchese è il parametro di pendenza stimato della nostra regressione logistica - B1 ci dice in che misura i LogOdds cambiano quando X cambia. Si noti che B1 vive sulla linea turchese, che collega l'ingresso X al neurone blu in Hidden Layer 1.
- B0 in blu è il bias - molto simile al termine di intercettazione dalla regressione. La differenza chiave è che nelle reti neurali ogni neurone ha il suo termine di polarizzazione (mentre nella regressione, il modello ha un termine di intercettazione singolare).
- Il neurone blu include anche una funzione di attivazione sigmoidea (indicata dalla linea curva all'interno del cerchio blu). Ricorda che la funzione sigmoid è ciò che usiamo per passare dalle probabilità del log alla probabilità (fai una ricerca control-f per "sigmoid" nel mio post precedente).
- infine otteniamo la nostra probabilità prevista applicando la funzione sigmoide alla quantità (B1 * X + B0).

Ora è possibile complicare la rete e renderla più estesa:

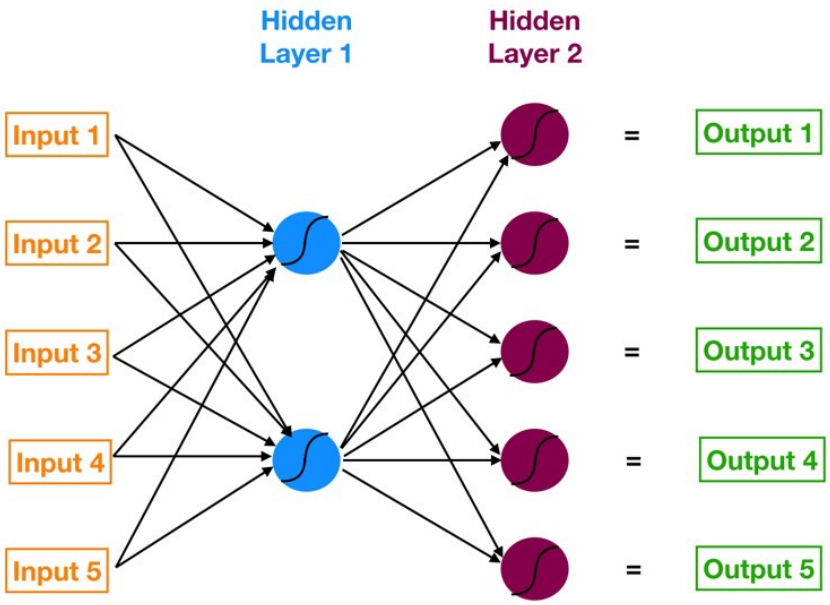


Figure 10.9: Neuronal Network 4

Il primo strato nascosto è costituito da due neuroni. Quindi per collegare tutti e cinque gli input ai neuroni in Hidden Layer 1, abbiamo bisogno di dieci connessioni. L'immagine successiva mostra solo le connessioni tra Input 1 e Hidden Layer.

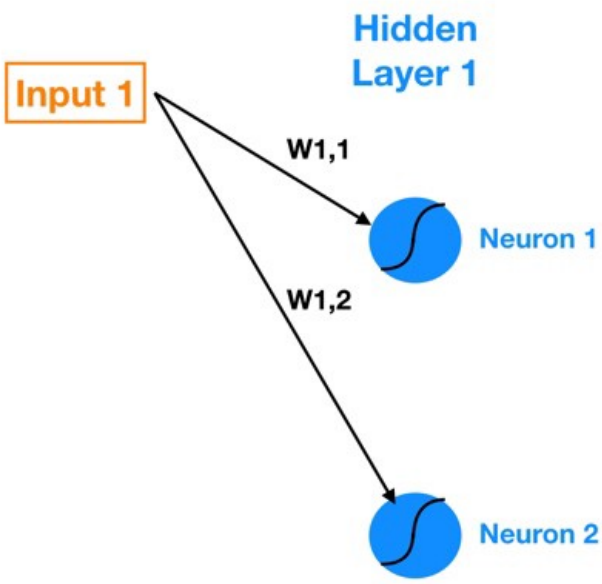


Figure 10.10: Neuronal Network 5

W1,1 indica il peso che vive nella connessione tra Input 1 e Neurone 1 e W1,2 indica il peso nella connessione tra Input 1 e Neurone 2. Quindi la notazione generale è Wa, B indica il peso sulla connessione tra Input a (o Neurone A) e Neurone B.

Ora si possono calcolare le uscite di ciascun neurone in Hidden Layer 1 (noto come attivazioni) utilizzando le seguenti formule (W indica peso, In indica input).

$W1 * In1 + W2 * In2 + W3 * In3 + W4 * In4 + W5 * In5 + Bias_N euron1$

Possiamo usare la matematica matriciale per riassumere questo calcolo:

$$\begin{bmatrix} W1,1 & W2,1 & W3,1 & W4,1 & W5,1 \\ W1,2 & W2,2 & W3,2 & W4,2 & W5,2 \end{bmatrix} \times \begin{bmatrix} X1 \\ X2 \\ X3 \\ X4 \\ X5 \end{bmatrix} + \begin{bmatrix} Bias1 \\ Bias2 \end{bmatrix} = \begin{bmatrix} Z1 \\ Z2 \end{bmatrix}$$

Figure 10.11: Neuronal Network 6

Dove [W] è la matrice n per m di pesi (le connessioni tra il livello precedente e il livello corrente), [X] è m per 1 matrice di input o attivazioni di partenza dal livello precedente, [Bias] è n per 1 matrice di polarizzazione dei neuroni e [Z] è n per 1 matrice di output intermedi. Nell'equazione precedente, seguendo la notazione Python si usa @ per indicare la moltiplicazione della matrice. Una volta che abbiamo [Z], possiamo applicare la funzione di attivazione (sigmoid nel nostro caso) a ciascun elemento di [Z] e che ci fornisce le nostre uscite neuronali (attivazioni) per il layer corrente.

Infine, mappiamo visivamente ciascuno di questi elementi sul grafico della rete neurale per legare tutto ([Bias] è incorporato nei neuroni blu).

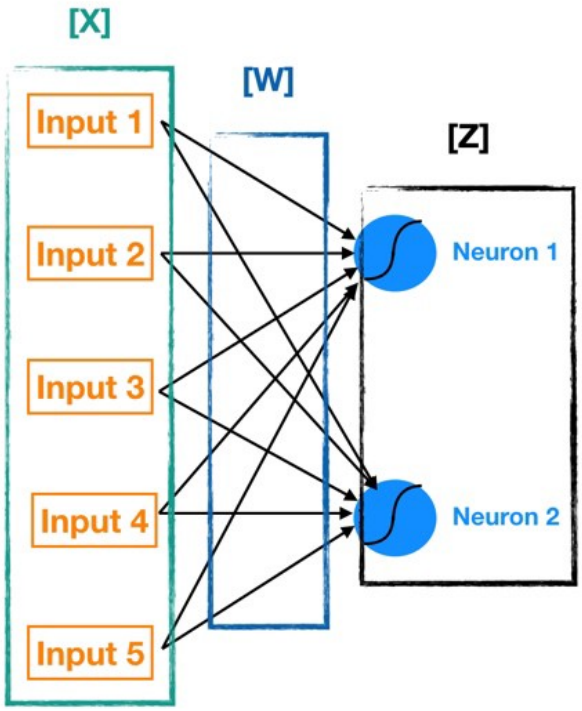


Figure 10.12: Neuronal Network 7

Calcolando ripetutamente [Z] e applicando ad essa la funzione di attivazione per ogni livello successivo, possiamo passare da input a output. Questo processo è noto come propagazione diretta. Il passo successivo è quello di allenare la rete neurale.

Il processo di allenamento di una rete neurale, ad alto livello, è come quello di molti altri modelli di data science: definire una funzione di costo e utilizzare l'ottimizzazione della diminuzione del gradiente per minimizzarla.

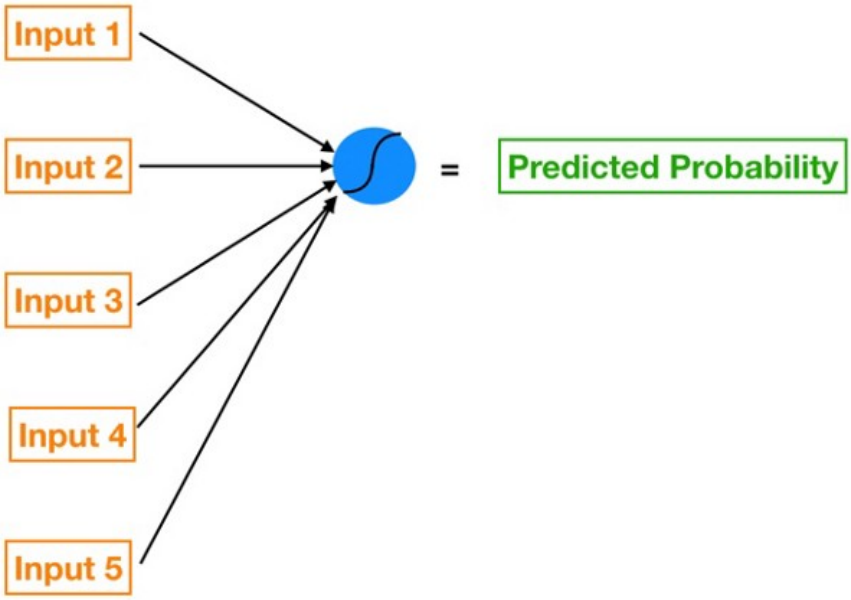


Figure 10.13: Neuronal Network 8

In regressione tradizionale, possiamo cambiare qualsiasi beta particolare in isolamento senza influire sugli altri coefficienti beta. Quindi, applicando piccoli shock isolati a ciascun coefficiente beta e misurando il suo impatto sulla funzione di costo, è relativamente semplice capire in quale direzione dobbiamo spostarci per ridurre e infine minimizzare la funzione di costo.

In una rete neurale, cambiare il peso di una qualsiasi connessione (o il pregiudizio di un neurone) ha un effetto riverberante su tutti gli altri neuroni e le loro attivazioni negli strati successivi.

Questo perché ogni neurone in una rete neurale è come il suo piccolo modello.

Quindi ogni strato nascosto di una rete neurale è fondamentalmente una pila di modelli (ogni singolo neurone nello strato agisce come il proprio modello) le cui uscite si alimentano in un numero ancora maggiore di modelli più a valle.

Dato un insieme di input esogeni e di risultati è necessario quindi allenare la nostra rete neurale.

Per addestrare la nostra rete neurale, si usa il Mean Squared Error (MSE) come funzione di costo:

$MSE = Sum[(Prediction - Actual)^2] * \frac{(1}{num_o bservations})$

L'MSE eleva al quadrato gli errori delle nostre previsioni prima di farne una media, puniamo le previsioni che sono molto più gravi di quelle che sono solo leggermente fuori. Le funzioni di costo della regressione lineare e della regressione logistica operano in modo molto simile.

Il modello migliore viene successivamente testato quindi sui dati ritenuti validi per il test.

Ogni neurone è il suo modello in miniatura con la sua propensione e il suo insieme di caratteristiche e pesi. Ogni singolo modello / neurone si nutre di numerosi altri singoli neuroni attraverso tutti gli strati nascosti del modello.

Ciò consente alle reti neurali di adattarsi ai nostri dati, comprese le parti non lineari ponendo un occhio all'overfitting.

La versatilità dell'approccio di molti modelli interconnessi e la capacità del processo di backpropagation di impostare in modo efficiente e ottimale pesi e distorsioni di ciascun modello consente alla rete neurale di "apprendere" in modo robusto dai dati in modi impossibili da molti altri algoritmi.

Chapter 11

Historical Data

11.1 Raw Data

I dati *raw* raccolti dell'istituto *ISAAC* sono stati elaborati e suddivisi per tipologia in file *.csv* con un timeframe di 10minuti su base annuale.

Il set di dati presenta 144 valori giornalieri distribuiti in 365 giorni.

La stazione di Trevano fornisce dati appunto con cadenza di 10 minuti, e può godere della possibilità di avere un dataset di dati riguardante:

- G_{Hor}
- P
- P_{DC}
- T_{Ext}
- $HR\%$
- $WindSpeed$

In ordine troviamo appunto, irraggiamento sul piano orizzontale, pressione atmosferica, potenza inverter in DC, temperatura esterna, umidità relativa, velocità del vento.

11.1.1 G

L'irraggiamento sicuramente è un dato che risente della stagionalità e della copertura del cielo, tuttavia esso a sua volta influenza altri dati.

E' possibile creare un grafico attraverso il quale si può valutare l'andamento annuale dell'irraggiamento su piano orizzontale.

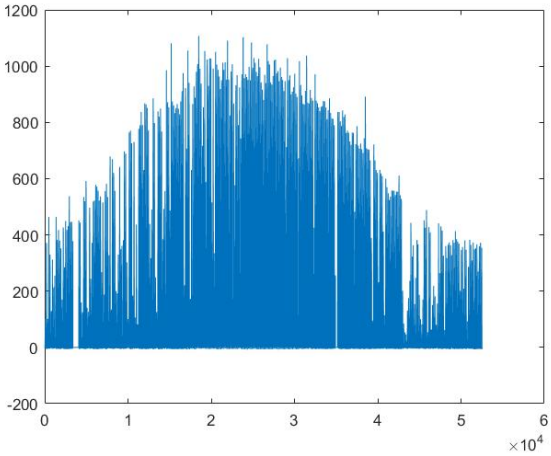


Figure 11.1: Irradiance on plane

Dividendo il dataset in giornate è possibile poi avere un subset per stabilire una correlazione.

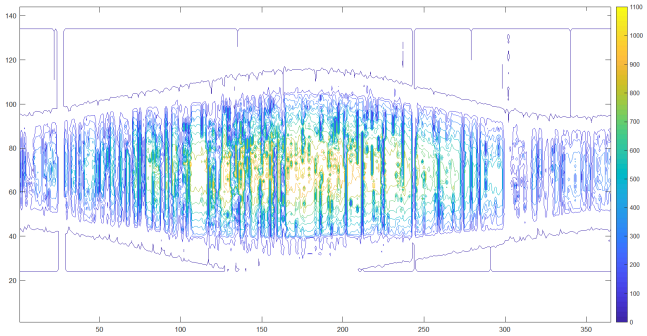


Figure 11.2: Irradiance Contour

Come si evince dal grafico, nel periodo centrale dell'anno l'irraggiamento è molto più intenso con picchi di $1100\frac{W}{m^2}$

L'andamento annuale della potenza irradiata è facilmente distinguibile anche attraverso un tipo di dato

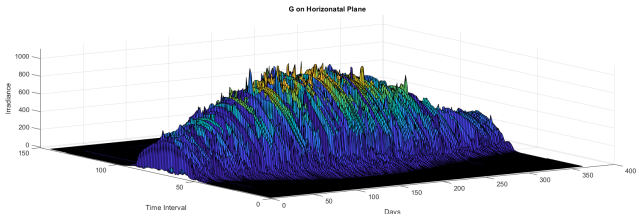


Figure 11.3: Irradiance Surf

11.1.2 Pressure

Come già valutato nei capitoli precedenti, la variazione repentina di pressione ha effetto sul clima, sulla copertura nuvolosa e quindi poi direttamente sull'irraggiamento.

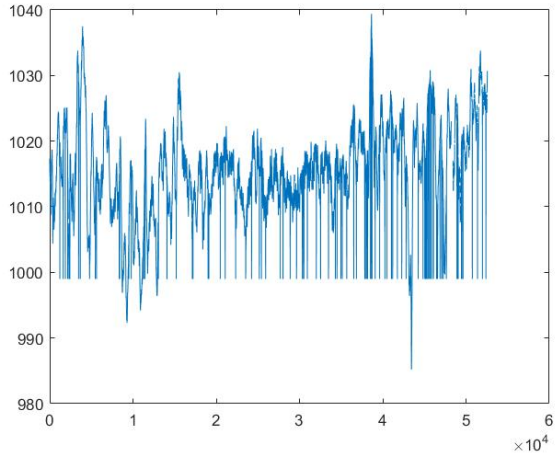


Figure 11.4: Pressure

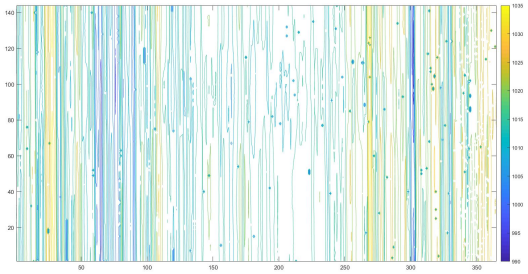


Figure 11.5: Pressure Contour

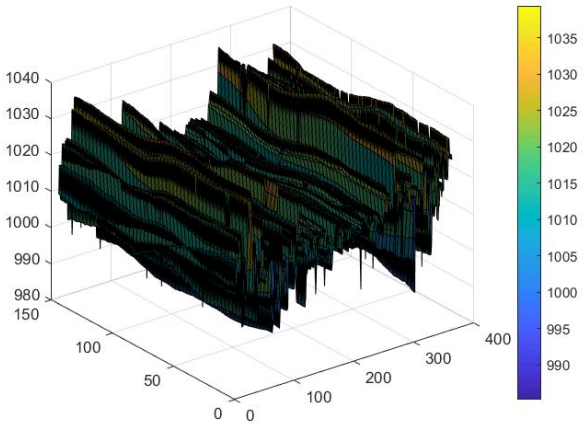


Figure 11.6: Pressure Surf

11.1.3 P-DC

La potenza in dc convertita dall'inverter è sicuramente un dato interessante e fortemente correlato con l'irraggiamento e la temepratura,per cui è importante includerlo nel subset di dati da analizzare.

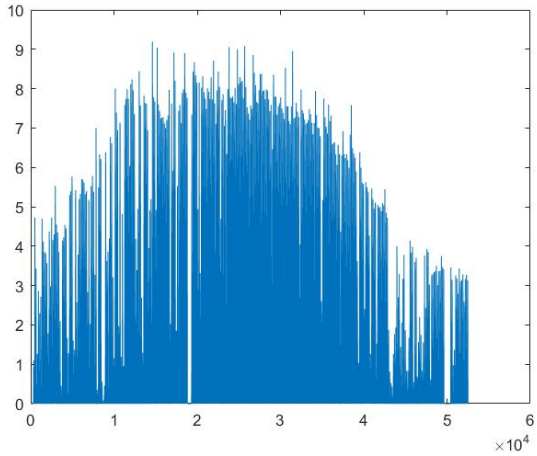


Figure 11.7: P-DC

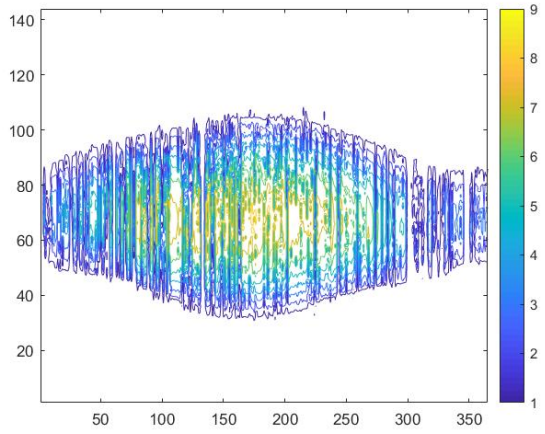


Figure 11.8: P-DC Contour

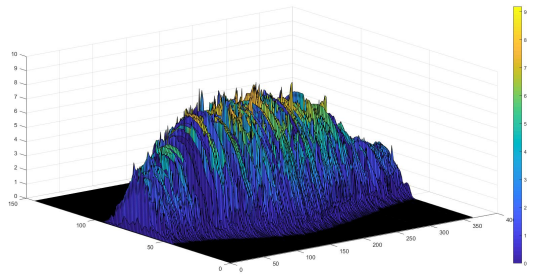


Figure 11.9: P-DC Surf

11.1.4 T Ext

La temperatura può essere usata sia come predittore per l'irraggiamento sia come fattore correttivo per la produzione solare, in questo caso ci si aspetta una correlazione forte tra irraggiamento e temperatua in quanto le giornate più serene avranno sicuramente una temperatura maggiore.

I dati di temperatura presentavano dei rumori a tratti con valori saturati a fondoscale che sono stati prontamente rimossi.

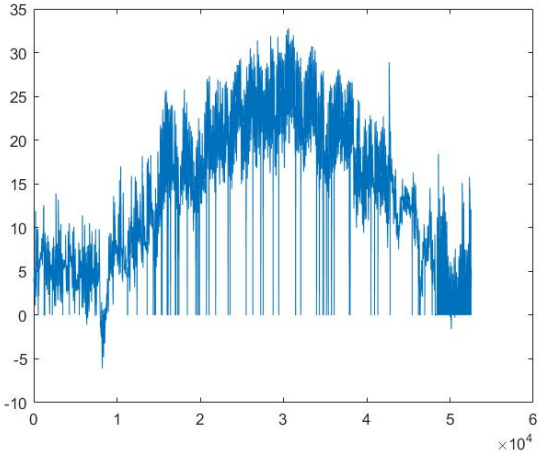


Figure 11.10: Temperature

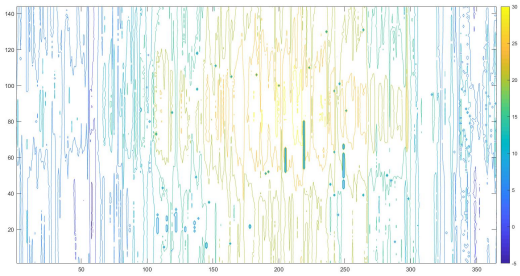


Figure 11.11: Temperature Contour

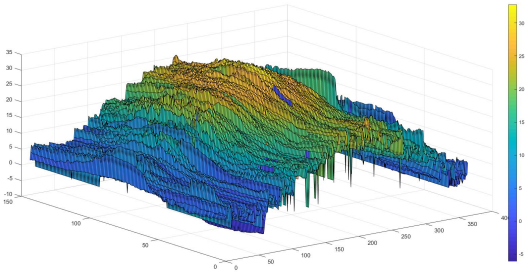


Figure 11.12: Temperature Surf

11.1.5 HR

Espressa in percentuale può essere utile strumento di correlazione e può generare un dato matematico come il DewPoint.

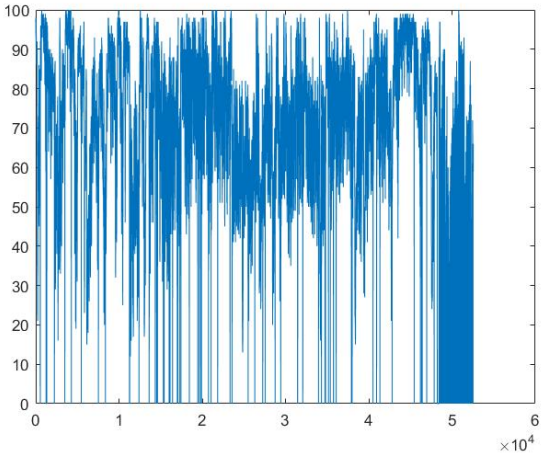


Figure 11.13: Relative Humidity

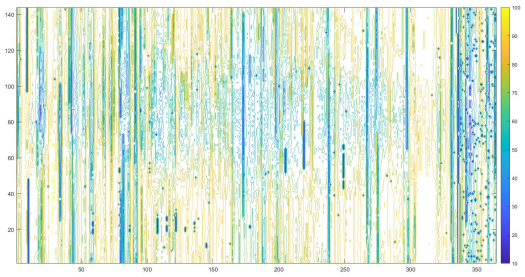


Figure 11.14: Relative Humidity Contour

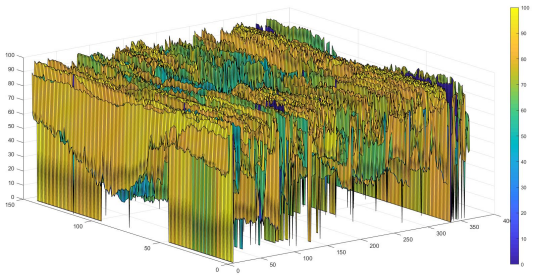


Figure 11.15: Relative Humidity Surf

11.1.6 WS

Ultimo ma non meno importante elemento loggato, è la velocità del vento espressa in $\frac{m}{s}$

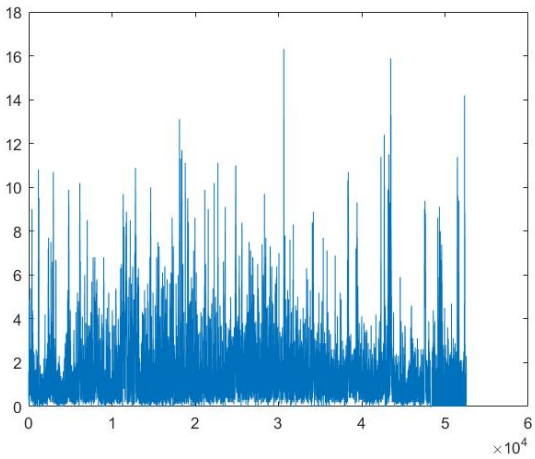


Figure 11.16: Wind Speed

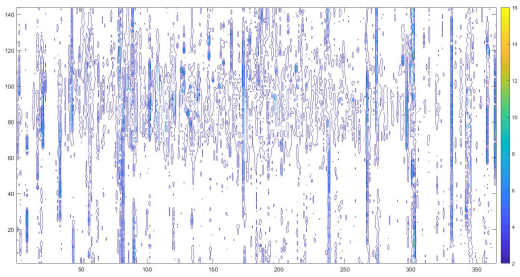


Figure 11.17: Wind Speed Contour

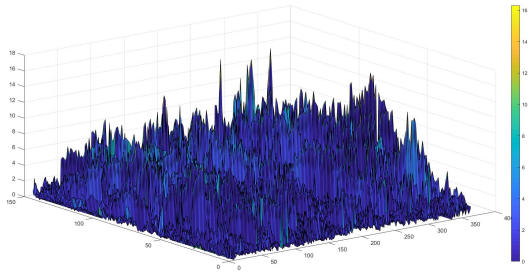


Figure 11.18: Wind Speed Surf

11.1.7 DewPoint

Calcolato matematicamente e spiegato nella sezione dedicata [10.3.2] può essere un buon indice di correlazione.

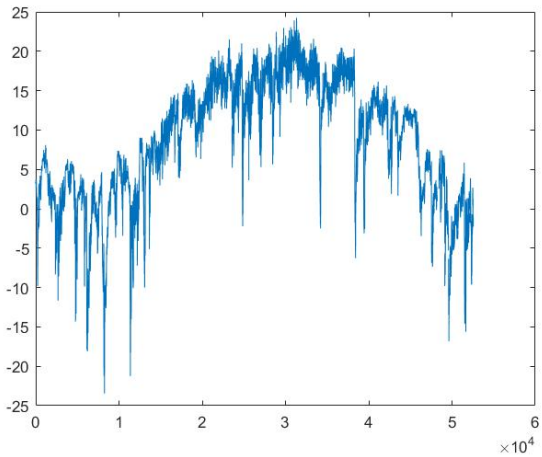


Figure 11.19: DP

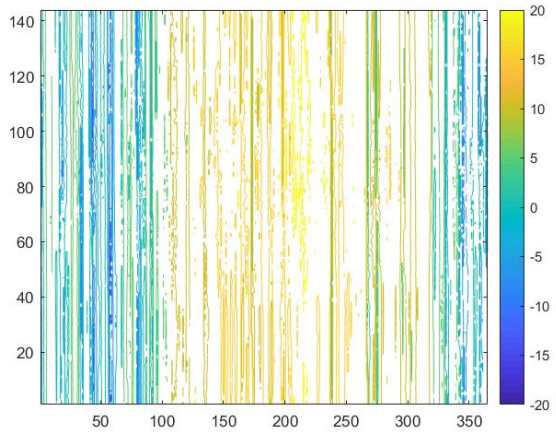


Figure 11.20: Dp Contour

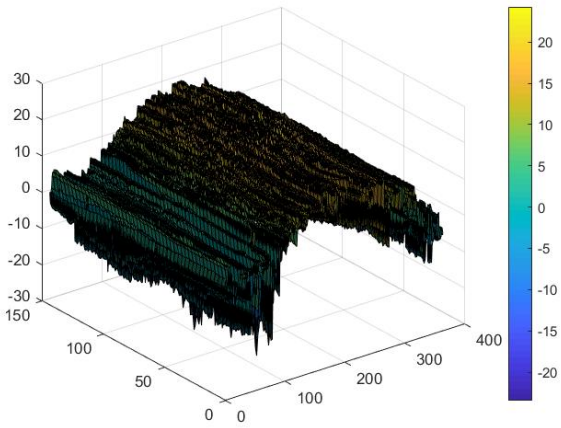


Figure 11.21: DP Surf

11.1.8 Filtering

La totalità dei dati che sono stati presentati e raccolti dalla stazione di Trevano presenta tuttavia un numero molto elevato di elementi incorretti che sono entrati a far parte del dataset per una questione di rumore o errata acquisizione.

Come ogni dataset che comprenda numerosi campioni è necessaria una opera di filtraggio e pulizia.

Sono state riscontrate le seguenti anomalie:

- G=999 Saturazione
- P=0 Dato Nullo
- RH=999 Saturazione
- T= 999 Saturazione
- Pdc=999 Saturazione
- WS=277.5 Saturazione

I dati saturati e nulli sono stati eliminati e, data la dinamica lenta del meteo, sostituiti con dati linearmente approssimati.

11.1.9 Correlation and Autocorrelation Analysis

Not filtered- ALL G

Eliminando solo gli spike privi di significato e realizzando un algoritmo di correlazione attraverso il software *Matlab* è possibile valutare la correlazione lineare tra l'output G e gli input esogeni.

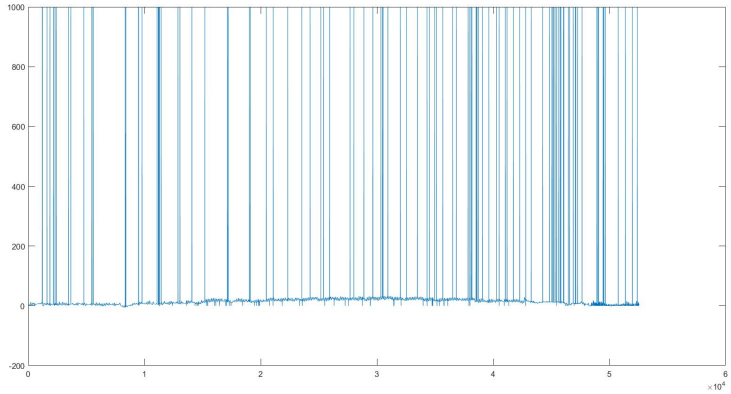


Figure 11.22: G not Filtered

La situazione iniziale si mostrava come sopra, con dati che presentano acquisizioni non elaborabili.

Successivamente elaborati invece trasformando gli spike in *nan* e conseguentemente in dati che corrispondono alla media tra i due dati esterni:

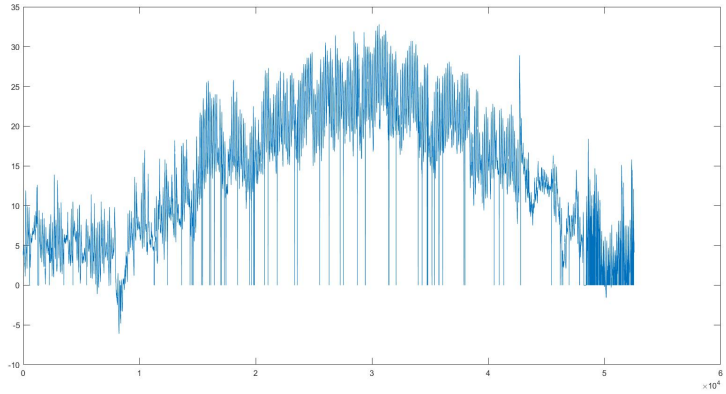


Figure 11.23: G Filtered Linearly

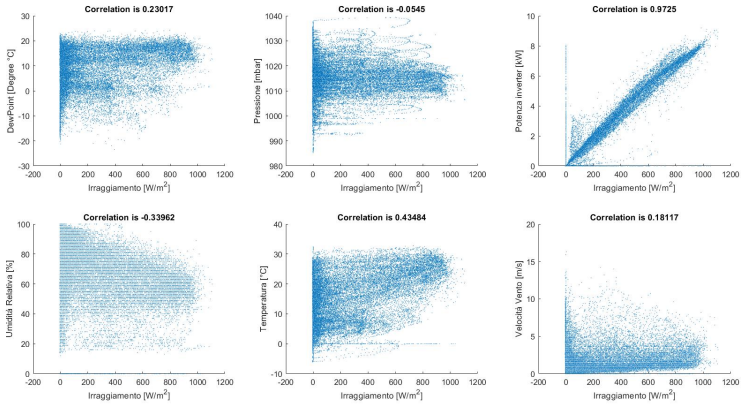


Figure 11.24: Correlation Exogenous not Filtered

E' stata valutata anche l'autocorrelazione su G che analogamente a Pdc è molto forte a delay di 144,288 e 1440 intervalli, ossia 1,2,10 giorni.

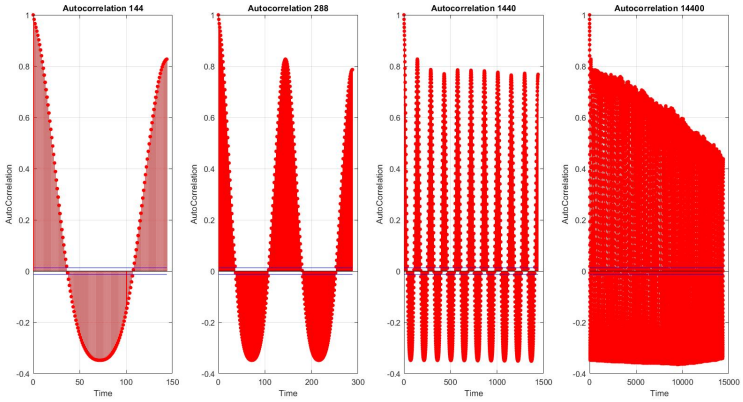


Figure 11.25: AutoCorrelation not filtered

Filtered > 5 $\frac{W}{M^2}$

Sono stati eliminati tutti i dati relativi agli irraggiamenti inferiori ai 5 watt al metro quadro per evidenziare o valutare diverse correlazioni. La serie di dati comprensiva di notte potrebbe non essere di aiuto.

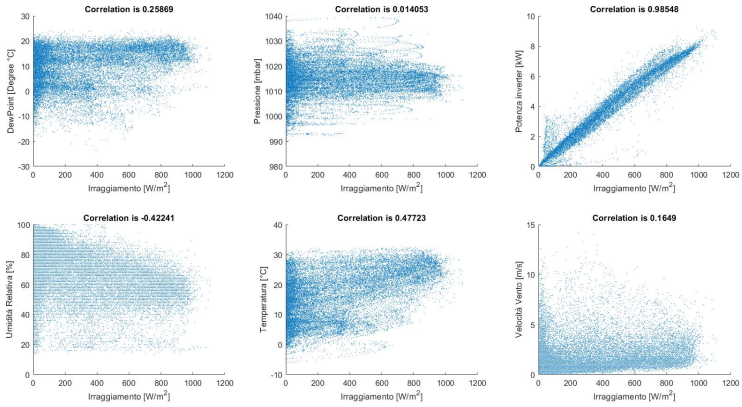


Figure 11.26: Filter 5$\frac{W}{M^2}$ Correlation

Cosi come è stata rivalutata la nuova autocorrelazione

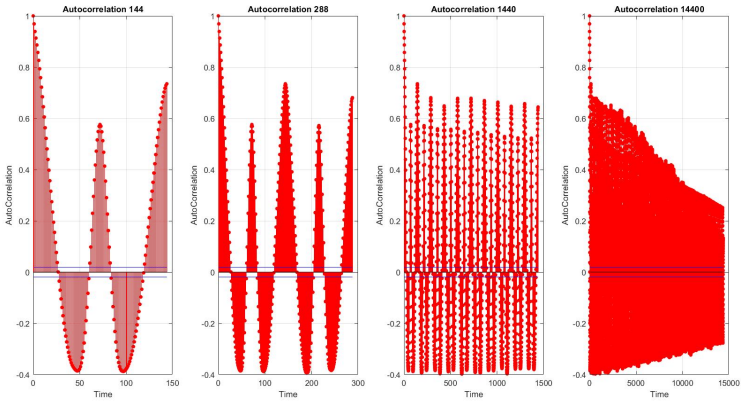


Figure 11.27: Filter 5$\frac{W}{M^2}$ Autocorrelation

Filtered > 50 $\frac{W}{M^2}$

Sono stati , in questa nuovo scenario,eliminati tutti i dati relativi agli irraggiamenti inferiori ai 50 watt al metro quadro per evidenziare o valutare diverse correlazioni. La serie di dati comprensiva di bassi irraggiamenti è stata eliminata.

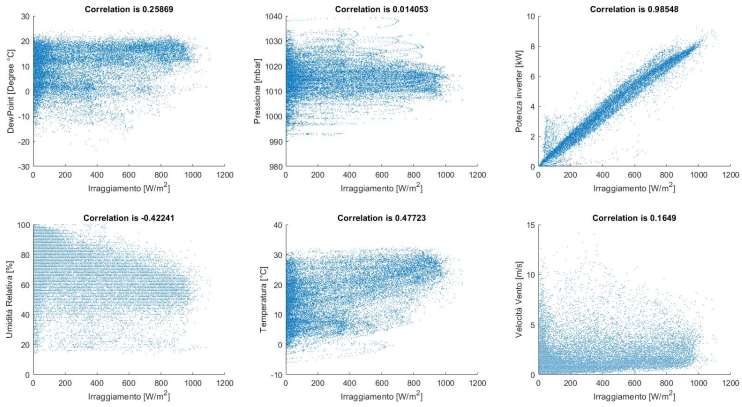


Figure 11.28: Filter 50$\frac{W}{M^2}$ Correlation

Di nuovo è stata rivalutata la nuova autocorrelazione:

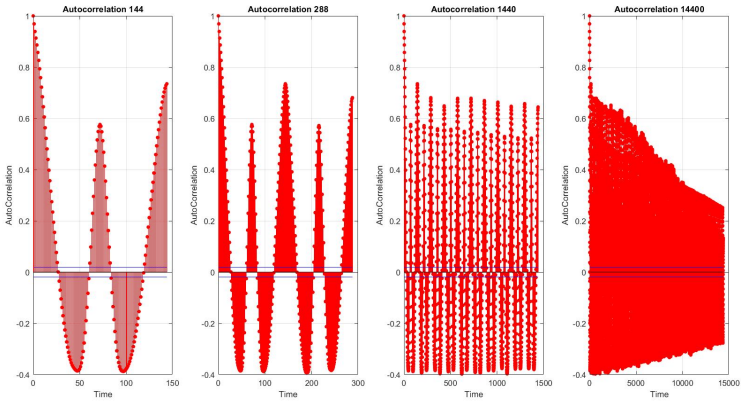


Figure 11.29: Filter 50$\frac{W}{M^2}$ Autocorrelation

Oltre a questa analisi è possibile capire la correlazione che ha una serie di dati con se stessa.

Riferendosi alla matrice *NXN* formata da 144 intervalli temporali in 365 giorni la funzione ***xcorr(X)*** restituisce una serie di cross correlazioni per tutte le combinazioni delle colonne della matrice X.

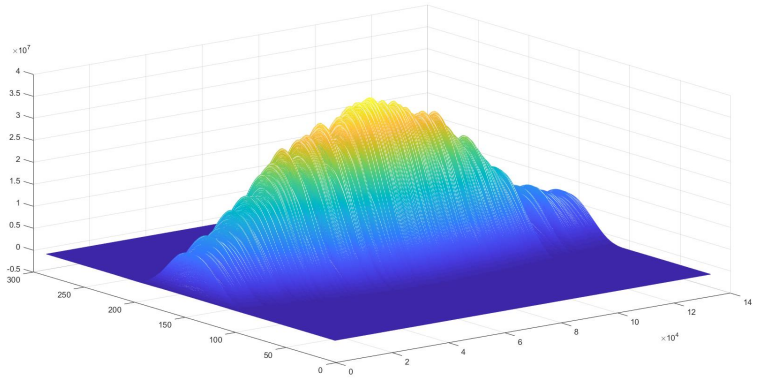


Figure 11.30: Autocorrelation Mesh

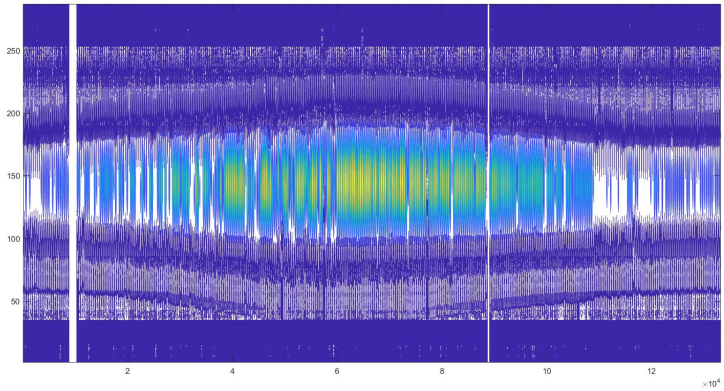


Figure 11.31: Surf Autocorrelation Mesh

Includendo le variabili calcolate matematicamente come la *Pressione di vapore effettiva* e il *vapore di saturazione* si calcola quindi il box plot delle correlazioni finali.

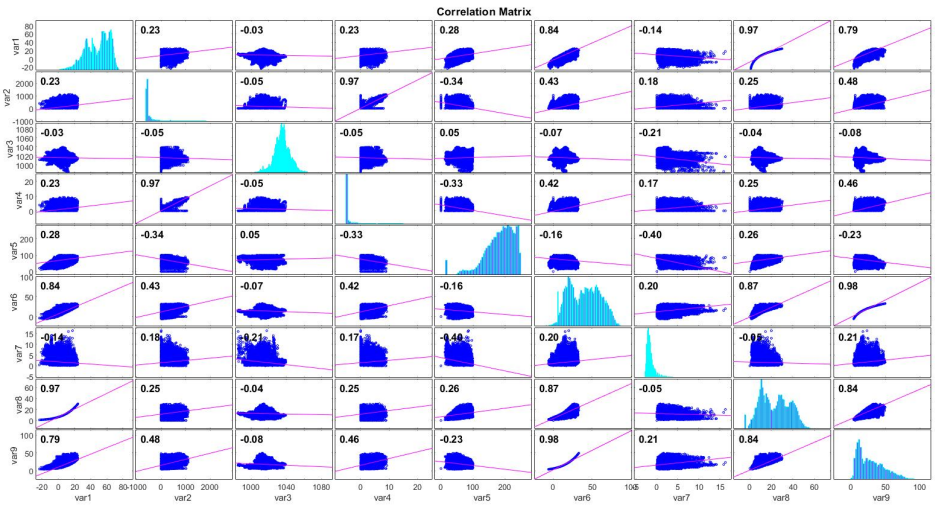


Figure 11.32: Correlation Box

11.1.10 Results

E' cosi' possibile notare all'interno di questa tabella riassuntiva il cambiamento delle correlazione in base ai filtri applicati e la loro variazione percentuale.

Correlation Index

	Filtered	**Filtered >5 W/mˆ2**	**Filtered >50 W/mˆ2**
DP	0.23017	0.25895	0.27354
PRESSURE	0.0545	0.012764	0.016655
DC POWER	0.9725	0.98537	0.98032
HR %	0.3396	0.42224	0.41636
TEMPERATURE	0.43484	0.47735	0.49063
WINDSPEED	0.18117	0.1653	0.18436

Percentage Change

	Filtered	**Filtered >5 W/mˆ2**	**Filtered >50 W/mˆ2**
DP	-	+12.23%	+18.8%
PRESSURE	-	-23.12%	-30.32%
DC POWER	-	+1.3%	+1.1%
HR %	-	+25.6%	+23.4%
TEMPERATURE	-	+9.7%	+12.4%
WINDSPEED	-	-8.1%	+2.1%

Sarà possibile quindi nell'acquisizione dati applicare dei filtri ove necessario per migliorare la correlazione tra i dati.

Threshold Influence

Analizzando successivamente per un intervallo di Treshold maggiori è possibile vedere come cambia la correlazione con varie soglie di irraggiamento.

Vettore con valori Threshold:

1	5	10	20	50	100	150	200	250	300	350	400
450	500	550	600	650	700	750	800	850	900	950	1000

A questo punto viene valutato ogni singolo fattore di correlazione:

RGDP	*Irradiance*	*DewPoint*
RGP	*Irradiance*	*Pressure*
RGPdc	*Irradiance*	*Power DC*
RGRH	*Irradiance*	*Relative Humidity*
RGT	*Irradiance*	*Termperature*
RGWS	*Irradiance*	*Wind Speed*

E come seguentemente mostrato ecco i pair migliori.

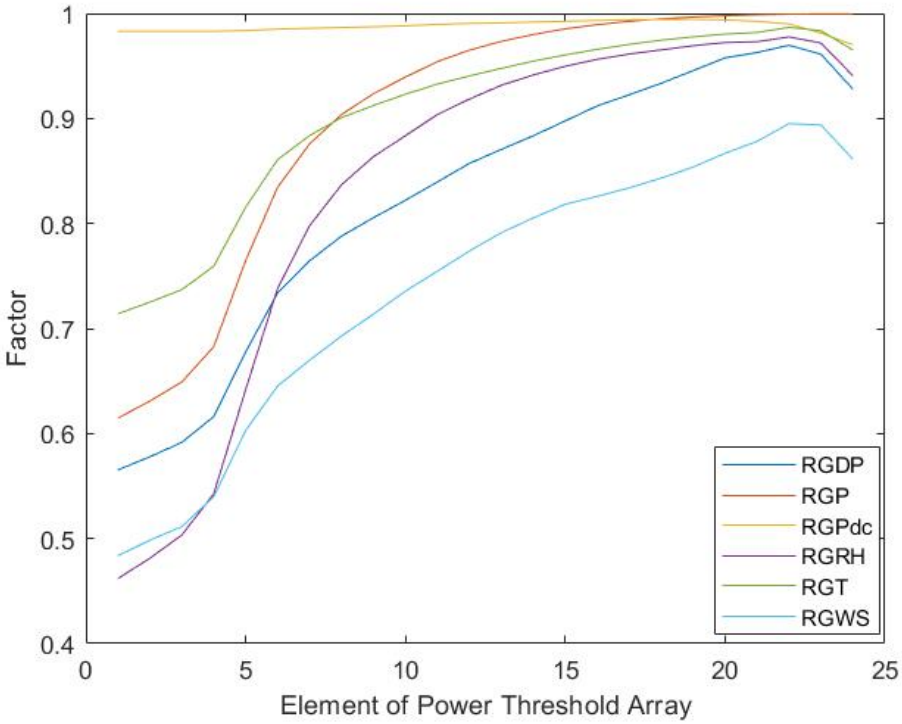


Figure 11.33: Correlation gain vs Treshold

Tuttavia un filtro troppo forte sull'irraggiamento abbatte completamente le casistiche e il numero di campioni che verrebbero poi elaborati dal modello.

E' necessario valutare poi le performance del modello in base a questa considerazione.

11.1.11 Mathlab Code

datareader.m

```
labels= {'DewPoint','G','P','Pdc','RH','T','WS'};


%Import GHOR Data
P=csvread('P.csv',1,2);
G=csvread('Ghor.csv',1,2);
RH=csvread('RH.csv',1,2);
T=csvread('T.csv',1,2);
Pdc=csvread('pdc INV1.csv',1,2);
WS=csvread('WS.csv',1,2);

% %filtraggio
G(G==999)=nan;
P(P==0)=nan;
RH(RH==999)=nan;
T(T==999)=nan;
Pdc(Pdc==999)=nan;
WS(WS==277.5)=nan;

%Math predictor
%Pressione di Vapore di saturazione
ES=6.11.*10.^((T.*7.5)./(237.7+T));

%PressioneDiVaporeInAria
E=RH.*ES/100;

DewPoint=(-430.22+237.7.*log(E))./(-log(E)+19.08);


%Divide in Day
GDaily= reshape(G,144,365);
PDaily= reshape(P,144,365);
RHDaily= reshape(RH,144,365);
TDaily= reshape(T,144,365);
PdcDaily=reshape(Pdc,144,365);
WSDaily=reshape(WS,144,365);
DPDaily=reshape(DewPoint,144,365);

%% Correlazioni intero spettro
    %Generazione correlazioni e scatter
        %1
        figure;
        subplot(2,3,1)
        scatter(G,DewPoint,0.2);
        R1= corr(G,DewPoint,'rows','complete');

        title(['Correlation is ',num2str(R1(1,1)),''])
        xlabel('Irraggiamento [W/m^2]');
        ylabel('DewPoint [Degree °C]');
        hold on;

        %2
        subplot(2,3,2)
        scatter(G,P,0.2);
        R1= corr(G,P,'rows','complete');

        title(['Correlation is ',num2str(R1(1,1)),''])
        xlabel('Irraggiamento [W/m^2]');
        ylabel('Pressione [mbar]');
        hold on;

```



```
        %3
        subplot(2,3,3)
        scatter(G,Pdc,0.2);
        R1= corr(G,Pdc,'rows','complete');

        title(['Correlation is ',num2str(R1(1,1)),''])
        xlabel('Irraggiamento [W/m^2]');
        ylabel('Potenza inverter [kW]');
        hold on;

        %4
        subplot(2,3,4)
        scatter(G,RH,0.2);
        R1= corr(G,RH,'rows','complete');

        title(['Correlation is ',num2str(R1(1,1)),''])
        xlabel('Irraggiamento [W/m^2]');
        ylabel('Umidità Relativa [%]');
        hold on;

        %5
        subplot(2,3,5);
        scatter(G,T,0.2);
        R1= corr(G,T,'rows','complete');

        title(['Correlation is ',num2str(R1(1,1)),''])
        xlabel('Irraggiamento [W/m^2]');
        ylabel('Temperatura [°C]');
        hold on;

        %6
        subplot(2,3,6)
        scatter(G,WS,0.2);
        R1= corr(G,WS,'rows','complete');

        title(['Correlation is ',num2str(R1(1,1)),''])
        xlabel('Irraggiamento [W/m^2]');
        ylabel('Velocità Vento [m/s]');
        hold on;

        %Autocorrelazione

        %1
        acf = autocorr(G);
        figure;
        subplot(1,4,1)
        autocorr(G,'NumLags',144,'NumSTD',3)

        title(['Autocorrelation 144'])
        xlabel('Time');
        ylabel('AutoCorrelation');
        hold on;

        %2
        acf = autocorr(G);
        subplot(1,4,2)
        autocorr(G,'NumLags',288,'NumSTD',3)

        title(['Autocorrelation 288'])
        xlabel('Time');
        ylabel('AutoCorrelation');
```



```
        hold on;

        %3
        acf = autocorr(G);
        subplot(1,4,3)
        autocorr(G,'NumLags',1440,'NumSTD',3)

        title(['Autocorrelation 1440'])
        xlabel('Time');
        ylabel('AutoCorrelation');
        hold on;

        %4
        acf = autocorr(G);
        subplot(1,4,4)
        autocorr(G,'NumLags',14400,'NumSTD',3)

        title(['Autocorrelation 14400'])
        xlabel('Time');
        ylabel('AutoCorrelation');
        hold on;

        %% Correlazioni solo in produzione , tolgo irraggiamenti sotto 5w/M2

        GpreFilter=G;
        %Creo Maschera
        GpreFilter = G > 5;

        GFiltered=GpreFilter.*G;
        %Moltiplico maschera per vettore
        GFiltered  = G.*GpreFilter;
        PFiltered  = P.*GpreFilter;
        RHFiltered = RH.*GpreFilter;
        TFiltered  = T.*GpreFilter;
        PdcFiltered=Pdc.*GpreFilter;
        WSFiltered =WS.*GpreFilter;
        DPFiltered =DewPoint.*GpreFilter;

        %Ricollocazione in matrici giornaliere
        GDailyFiltered  = reshape(GFiltered,144,365);
        PDailyFiltered  = reshape(PFiltered.*GpreFilter,144,365);
        RHDailyFiltered = reshape(RHFiltered.*GpreFilter,144,365);
        TDailyFiltered  = reshape(TFiltered.*GpreFilter,144,365);
        PdcDailyFiltered=reshape(PdcFiltered.*GpreFilter,144,365);
        WSDailyFiltered =reshape(WSFiltered.*GpreFilter,144,365);
        DPDailyFiltered =reshape(DPFiltered.*GpreFilter,144,365);

        %Tutti 0 diventano NaN

        GFiltered(find(GFiltered==0)) = NaN;
        PFiltered(find(PFiltered==0)) = NaN;
        RHFiltered(find(RHFiltered==0)) = NaN;
        TFiltered(find(TFiltered==0)) = NaN;
        PdcFiltered(find(PdcFiltered==0)) = NaN;
        WSFiltered(find(WSFiltered==0)) = NaN;
        DPFiltered(find(DPFiltered==0)) = NaN;



     %% Correlazioni spettro filtrato
    %Generazione correlazioni e scatter
```



```
        %1
        figure;
        subplot(2,3,1)
        scatter(GFiltered,DPFiltered,0.2);
        R2= corr(GFiltered,DPFiltered,'rows','complete');

        title(['Correlation is ',num2str(R2(1,1)),''])
        xlabel('Irraggiamento [W/m^2]');
        ylabel('DewPoint [Degree °C]');
        hold on;

        %2
        subplot(2,3,2)
        scatter(GFiltered,PFiltered,0.2);
        R2= corr(GFiltered,PFiltered,'rows','complete');

        title(['Correlation is ',num2str(R2(1,1)),''])
        xlabel('Irraggiamento [W/m^2]');
        ylabel('Pressione [mbar]');
        hold on;

        %3
        subplot(2,3,3)
        scatter(GFiltered,PdcFiltered,0.2);
        R2= corr(GFiltered,PdcFiltered,'rows','complete');

        title(['Correlation is ',num2str(R2(1,1)),''])
        xlabel('Irraggiamento [W/m^2]');
        ylabel('Potenza inverter [kW]');
        hold on;

        %4
        subplot(2,3,4)
        scatter(GFiltered,RHFiltered,0.2);
        R2= corr(GFiltered,RHFiltered,'rows','complete');

        title(['Correlation is ',num2str(R2(1,1)),''])
        xlabel('Irraggiamento [W/m^2]');
        ylabel('Umidità Relativa [%]');
        hold on;

        %5
        subplot(2,3,5);
        scatter(GFiltered,TFiltered,0.2);
        R2= corr(GFiltered,TFiltered,'rows','complete');

        title(['Correlation is ',num2str(R2(1,1)),''])
        xlabel('Irraggiamento [W/m^2]');
        ylabel('Temperatura [°C]');
        hold on;

        %6
        subplot(2,3,6)
        scatter(GFiltered,WSFiltered,0.2);
        R2= corr(GFiltered,WSFiltered,'rows','complete');

        title(['Correlation is ',num2str(R2(1,1)),''])
        xlabel('Irraggiamento [W/m^2]');
        ylabel('Velocità Vento [m/s]');
        hold on;

```

```
        %Autocorrelazione

        %1
        acf5 = autocorr(GFiltered);
        figure;
        subplot(1,4,1)
        autocorr(GFiltered,'NumLags',144,'NumSTD',3)

        title(['Autocorrelation 144'])
        xlabel('Time');
        ylabel('AutoCorrelation');
        hold on;

        %2
        acf5 = autocorr(GFiltered);
        subplot(1,4,2)
        autocorr(GFiltered,'NumLags',288,'NumSTD',3)

        title(['Autocorrelation 288'])
        xlabel('Time');
        ylabel('AutoCorrelation');
        hold on;

        %3
        acf5 = autocorr(GFiltered);
        subplot(1,4,3)
        autocorr(GFiltered,'NumLags',1440,'NumSTD',3)

        title(['Autocorrelation 1440'])
        xlabel('Time');
        ylabel('AutoCorrelation');
        hold on;

        %4
        acf5 = autocorr(GFiltered);
        subplot(1,4,4)
        autocorr(GFiltered,'NumLags',14400,'NumSTD',3)

        title(['Autocorrelation 14400'])
        xlabel('Time');
        ylabel('AutoCorrelation');
        hold on;

                %% Correlazioni solo in produzione , tolgo irraggiamenti sotto 50w/M2

        GpreFilter=G;
        %Creo Maschera
        GpreFilter = G > 50;

        Gfiltered50=GpreFilter.*G;
        %Moltiplico maschera per vettore
        GFiltered50  = G.*GpreFilter;
        PFiltered50  = P.*GpreFilter;
        RHFiltered50 = RH.*GpreFilter;
        TFiltered50  = T.*GpreFilter;
        PdcFiltered50=Pdc.*GpreFilter;
        WSFiltered50 =WS.*GpreFilter;
        DPFiltered50 =DewPoint.*GpreFilter;

        %Ricollocazione in matrici giornaliere
        GDailyFiltered50  = reshape(GFiltered50,144,365);
```



```
        PDailyFiltered50  = reshape(PFiltered50.*GpreFilter,144,365);
        RHDailyFiltered50 = reshape(RHFiltered50.*GpreFilter,144,365);
        TDailyFiltered50  = reshape(TFiltered50.*GpreFilter,144,365);
        PdcDailyFiltered50=reshape(PdcFiltered50.*GpreFilter,144,365);
        WSDailyFiltered50 =reshape(WSFiltered50.*GpreFilter,144,365);
        DPDailyFiltered50 =reshape(DPFiltered50.*GpreFilter,144,365);

        %Tutti 0 diventano NaN

        GFiltered50(find(GFiltered50==0)) = NaN;
        PFiltered50(find(PFiltered50==0)) = NaN;
        RHFiltered50(find(RHFiltered50==0)) = NaN;
        TFiltered50(find(TFiltered50==0)) = NaN;
        PdcFiltered50(find(PdcFiltered50==0)) = NaN;
        WSFiltered50(find(WSFiltered50==0)) = NaN;
        DPFiltered50(find(DPFiltered50==0)) = NaN;



     %% Correlazioni spettro filtrato
    %Generazione correlazioni e scatter
        %1
        figure;
        subplot(2,3,1)
        scatter(GFiltered50,DPFiltered50,0.2);
        R= corr(GFiltered50,DPFiltered50,'rows','complete');

        title(['Correlation is ',num2str(R(1,1)),''])
        xlabel('Irraggiamento [W/m^2]');
        ylabel('DewPoint [Degree °C]');
        hold on;

        %2
        subplot(2,3,2)
        scatter(GFiltered50,PFiltered50,0.2);
        R= corr(GFiltered50,PFiltered50,'rows','complete');

        title(['Correlation is ',num2str(R(1,1)),''])
        xlabel('Irraggiamento [W/m^2]');
        ylabel('Pressione [mbar]');
        hold on;

        %3
        subplot(2,3,3)
        scatter(GFiltered50,PdcFiltered50,0.2);
        R= corr(GFiltered50,PdcFiltered50,'rows','complete');

        title(['Correlation is ',num2str(R(1,1)),''])
        xlabel('Irraggiamento [W/m^2]');
        ylabel('Potenza inverter [kW]');
        hold on;

        %4
        subplot(2,3,4)
        scatter(GFiltered50,RHFiltered50,0.2);
        R= corr(GFiltered50,RHFiltered50,'rows','complete');

        title(['Correlation is ',num2str(R(1,1)),''])
        xlabel('Irraggiamento [W/m^2]');
        ylabel('Umidità Relativa [%]');
        hold on;
```



```

        %5
        subplot(2,3,5);
        scatter(GFiltered50,TFiltered50,0.2);
        R= corr(GFiltered50,TFiltered50,'rows','complete');

        title(['Correlation is ',num2str(R(1,1)),''])
        xlabel('Irraggiamento [W/m^2]');
        ylabel('Temperatura [°C]');
        hold on;

        %6
        subplot(2,3,6)
        scatter(GFiltered50,WSFiltered50,0.2);
        R= corr(GFiltered50,WSFiltered50,'rows','complete');

        title(['Correlation is ',num2str(R(1,1)),''])
        xlabel('Irraggiamento [W/m^2]');
        ylabel('Velocità Vento [m/s]');
        hold on;

        %Autocorrelazione

        %1
        acf50 = autocorr(GFiltered50);
        figure;
        subplot(1,4,1)
        autocorr(GFiltered50,'NumLags',144,'NumSTD',3)

        title(['Autocorrelation 144'])
        xlabel('Time');
        ylabel('AutoCorrelation');
        hold on;

        %2
        acf50 = autocorr(GFiltered50);
        subplot(1,4,2)
        autocorr(GFiltered50,'NumLags',288,'NumSTD',3)

        title(['Autocorrelation 288'])
        xlabel('Time');
        ylabel('AutoCorrelation');
        hold on;

        %3
        acf50 = autocorr(GFiltered50);
        subplot(1,4,3)
        autocorr(GFiltered50,'NumLags',1440,'NumSTD',3)

        title(['Autocorrelation 1440'])
        xlabel('Time');
        ylabel('AutoCorrelation');
        hold on;

        %4
        acf50 = autocorr(GFiltered50);
        subplot(1,4,4)
        autocorr(GFiltered50,'NumLags',14400,'NumSTD',3)

        title(['Autocorrelation 14400'])
        xlabel('Time');
```



```
        ylabel('AutoCorrelation');
        hold on;

        %% Prova ciclica migliori abbinamenti

        powerTreshold=[1,5,10,20,50,100,150,200,250,300,350,400,450,500,550,600,650,700,750,800,850,900,950,1000];

        powerIndex=size(powerTreshold);
        powerIndexTrue=powerIndex(2);

        for i=1:powerIndexTrue



                GpreFilter(:,i) = G > powerTreshold(i);


         %Moltiplico maschera per vettore
        GFiltered(:,i)  = G.*GpreFilter(:,i);
        PFiltered(:,i)  = P.*GpreFilter(:,i);
        RHFiltered(:,i) = RH.*GpreFilter(:,i);
        TFiltered(:,i)  = T.*GpreFilter(:,i);
        PdcFiltered(:,i)=Pdc.*GpreFilter(:,i);
        WSFiltered(:,i) =WS.*GpreFilter(:,i);
        DPFiltered(:,i) =DewPoint.*GpreFilter(:,i);

               %calcolo correlazioni

                RGP(i)= corr(GFiltered(:,i),PFiltered(:,i),'rows','complete');
                RGRH(i)= corr(GFiltered(:,i),RHFiltered(:,i),'rows','complete');
                RGT(i)= corr(GFiltered(:,i),TFiltered(:,i),'rows','complete');
                RGPdc(i)= corr(GFiltered(:,i),PdcFiltered(:,i),'rows','complete');
                RGWS(i)= corr(GFiltered(:,i),WSFiltered(:,i),'rows','complete');
                RGDP(i)= corr(GFiltered(:,i),DPFiltered(:,i),'rows','complete');

        end


        %Plot dei risultati



%% Valutazione rispetto a Pdc

 powerTreshold=[1,5,10,20,50,100,150,200,250,300,350,400,450,500,550,600,650,700,750,800,850,900,950,1000];

        powerIndex=size(powerTreshold);
        powerIndexTrue=powerIndex(2);

        for i=1:powerIndexTrue



                GpreFilter(:,i) = G > powerTreshold(i);


         %Moltiplico maschera per vettore
```

```
        GFiltered(:,i)  = G.*GpreFilter(:,i);
        PFiltered(:,i)  = P.*GpreFilter(:,i);
        RHFiltered(:,i) = RH.*GpreFilter(:,i);
        TFiltered(:,i)  = T.*GpreFilter(:,i);
        PdcFiltered(:,i)=Pdc.*GpreFilter(:,i);
        WSFiltered(:,i) =WS.*GpreFilter(:,i);
        DPFiltered(:,i) =DewPoint.*GpreFilter(:,i);

                %calcolo correlazioni

                 RPdcP(i)= corr(PdcFiltered(:,i),PFiltered(:,i),'rows','complete');
                 RPdcRH(i)= corr(PdcFiltered(:,i),RHFiltered(:,i),'rows','complete');
                 RPdcT(i)= corr(PdcFiltered(:,i),TFiltered(:,i),'rows','complete');
                 RPdcG(i)= corr(PdcFiltered(:,i),GFiltered(:,i),'rows','complete');
                 RPdcWS(i)= corr(PdcFiltered(:,i),WSFiltered(:,i),'rows','complete');
                 RPdcDP(i)= corr(PdcFiltered(:,i),DPFiltered(:,i),'rows','complete');

        end


%         %% funzione plot
%           function createfigure(YMatrix1)
% %CREATEFIGURE(YMatrix1)
% %  YMATRIX1:  matrix of y data
%
% %  Auto-generated by MATLAB on 13-Jun-2019 22:02:28
%
% % Create figure
% figure1 = figure;
%
% % Create axes
% axes1 = axes('Parent',figure1);
% hold(axes1,'on');
%
% % Create multiple lines using matrix input to plot
% plot1 = plot(YMatrix1,'Parent',axes1);
% set(plot1(1),'DisplayName','RGDP');
% set(plot1(2),'DisplayName','RGP');
% set(plot1(3),'DisplayName','RGPdc');
% set(plot1(4),'DisplayName','RGRH');
% set(plot1(5),'DisplayName','RGT');
% set(plot1(6),'DisplayName','RGWS');
%
% % Create ylabel
% ylabel({'Factor'});
%
% % Create xlabel
% xlabel({'Element of Power Threshold Array'});
%
% box(axes1,'on');
% % Create legend
% legend(axes1,'show');
%
%
%           end
  %% All infos are exogenous

  X = horzcat(G,DewPoint,P,Pdc,RH,T,WS,E,ES);
  XF = fillmissing(X,'linear')
```

predittori.m

```
%% Valutazione modelli di predizione


    % Creazione di un dataset di Training.


    trainX=XF(1:26280,2:9);

    trainY=XF(1:26280,1);

    testX=XF(26353:end,2:9);

    testY=XF(26353:end,1);

 %% Simple Persistence


 forecastG = cat(1,zeros(144,1),testY(1:end-144))

 [MAE,RMSE]= analyze_errors(testY,forecastG)

 %% Predittore Lineare
 %mdl = fitlm(X,y) returns a linear regression model of the responses y, fit to the data matrix X.
 linear_model = fitlm(trainX,trainY);

 [forecastG,yci] = predict(linear_model,testX);

%  label = predict( Mdl , X ) returns a vector of predicted class labels for the predictor...
%      data in the table or matrix X , based on the trained,...
%      full or compact classification tree Mdl . label = predict( Mdl , X , Name,Value ) ...
%      uses additional options specified by one or more Name,Value pair arguments.

 [MAE,RMSE]= analyze_errors(testY,forecastG)



 %% Predittore AR
 %1
 ARModel = ar(trainY,14);

[forecastG,yci] = predict(ARModel,testY);

[MAE,RMSE]= analyze_errors(testY,forecastG)


%144
 ARModel = ar(trainY,144);

[forecastG,yci] = predict(ARModel,testY);

[MAE,RMSE]= analyze_errors(testY,forecastG)

%% Predittore ARX%% MODELLI ARMAX:
% Possiamo provare a migliorare il modello aggiungendo informazioni esogene come le temperature

exogenous = trainX; % selezioniamo come input esogeno le informazioni storiche sul carico
```



```
data armax = iddata(trainY,exogenous,1); % trasforma i dati in un oggetto iddata, specificando la frequenza di campionamento
opt = armaxOptions('Focus','prediction');
% ordine della parte autoregressiva
na = ;
% ordine della parte esogena. Per semplicità assumiamo uguale influenza per tutti i predittori
nb = repmat(2,1,size(exogenous,2));
% ordine della parte a media mobile
nc = 2;
% numero di steps necessari alla parte esogena per influenzare il carico. Per semplicità assumiamolo sempre uguale a 1
nk = repmat(1,1,size(exogenous,2));

armax model = armax(data armax,[na nb nc nk],opt);

% prediciamo il segnale testY fra 24 ore
forecastG = predict(iddata(testY,testX,1),armax model,24);
analyze errors(testY,forecastG.OutputData)


%% Random Forest

model = TreeBagger(200, trainX, trainY, 'method', 'regression', 'oobvarimp', 'on', 'minleaf', 10);

figure();
barh(model.OOBPermutedVarDeltaError);
ylabel('Feature');
xlabel('Out-of-bag feature importance');
title('Feature importance results');
set(gca, 'YTickLabel', labels)

% visualizziamo un albero della random forest
view(model.Trees{1},'Mode','graph');

% rimuoviamo gli input e salviamo la random forest
model = compact(model);
save TreeModel model

% prediciamo il carico usando l'albero finale (media degli alberi nella random forest)
[forecastG,stdevs] = predict(model, testX);
analyze_errors(testY,forecastG);



[Y,Xf,Af] = myNeuralNetworkFunction(testX)
```

—

Chapter 12

Models Evaluation

Il set di modelli disponibili al fine di creare il *match* migliore tra la serie di dati ricevuta , sono vari.

Importante è valutare la fattibilità e le prestazioni del modello in modo da avere un confronto tra di essi.

Per mezzo di *Matlab* sono quindi stati valutati residui e MAE - (Mean Avarage Error), ed RMSE.
Possibile successivamente restituire i risultati delle distribuzioni degli errori e dei residui.
—

12.1 SubSet Fragmentation

Il dataset di dati purtroppo copre solamente l'anno solare scorso, e non è possibile avere un set di dati ridondante per vari anni.

A fronte di questo il dataset viene suddiviso in due dataset della durata di 6 mesi così da ottenere due dataset:

- Training
- Test

Il dataset di training in tutti i casi di studio viene uttilizzato per il learning del modello stesso, che viene successivamente testato grazie al *subset* di Test.

Nel caso della ***Persistence*** invece viene utilizzato lo stesso dataset in quanto occorre effettuare uno shift limitato nel tempo di 144 intervalli e il secondo dataset è un set troppo lontano temporalmente.

Questa operazione di subset è effettuata sia per le variabili esogene (ove necessario) sia per il valore di riferimento.

12.2 Persistence

Come descritta in precedenza il modello di persistenza rappresenta il modello base ossia di riferimento sul quale valutare la prestazione degli altri modelli in analisi.

Prestazioni $\frac{W}{m^2}$

- MAE: 51.8048
- RMSE: 134.1483

La persistenza offre ottime performance se le giornate si dimostrano costanti ma senza l'integrazione delle variabili esogene , diminuisce l'accuratezza nei giorni più variabili.

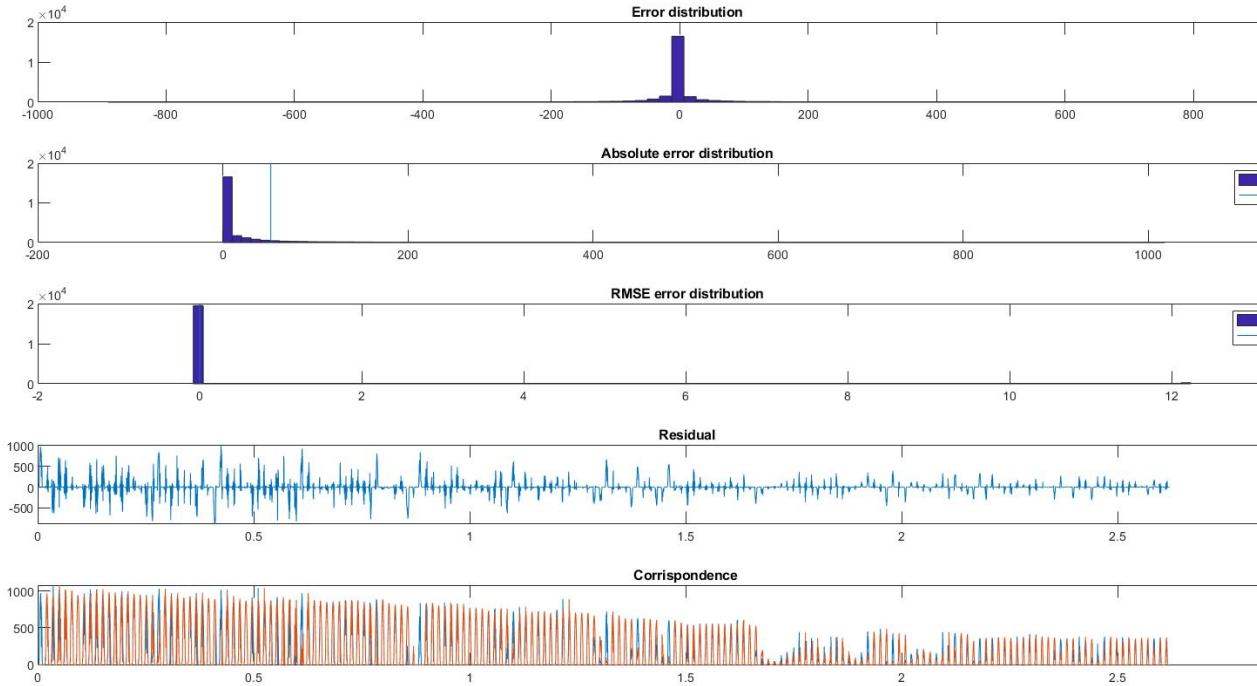


Figure 12.1: Persistence

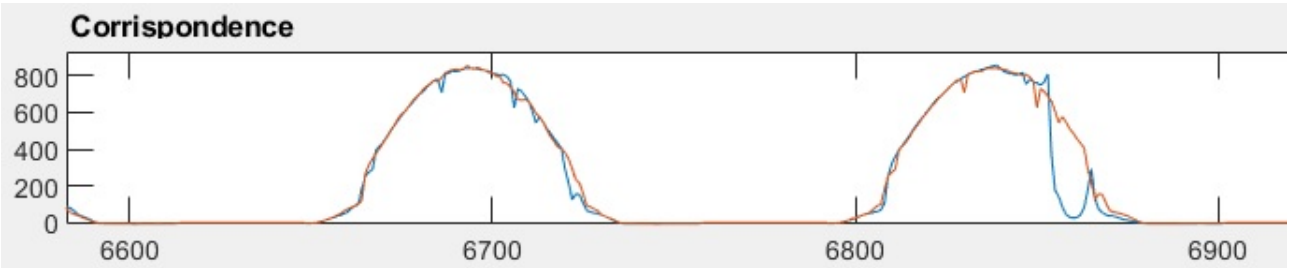


Figure 12.2: Persistence Detail

12.3 Linear Model

Il modello lineare come definito dalla funzione di Matlab "returns a linear regression model of the responses y, fit to the data matrix X".

Per questo motivo viene inserita anche la tabella relativa alle variabili esogene purificate dai valori NaN che son stati sostituiti per mezzo di una interpolazione lineare da nuovi valori.

Prestazioni $\frac{W}{m^2}$

- MAE: 23.2424
- RMSE: 53.7019

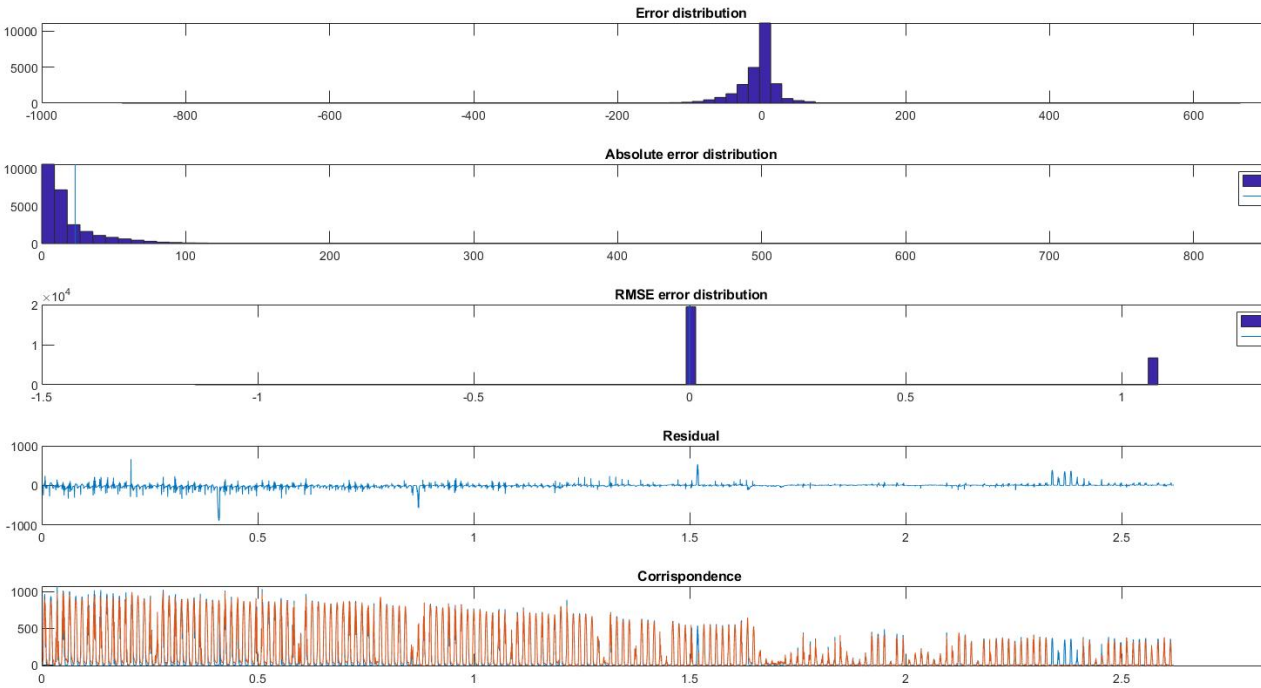


Figure 12.3: Linear Model

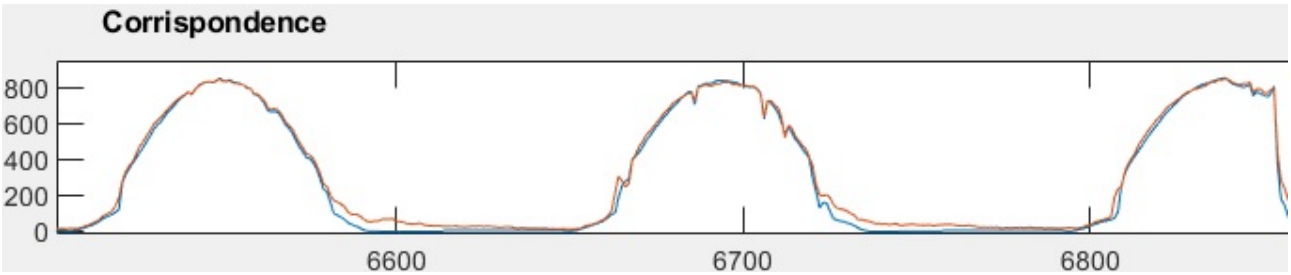


Figure 12.4: Linear Model Detail

12.4 AR

Il modello AR è disponibile con diversi gradi relativi al polinomio associato. Nella descrizione del modello si comprende che viene generato nella seguente forma:

$$y(t) + a1 * y(t-1) + a2 * y(t-2) + ... + aN * y(t-N) = e(t)$$

Inputs: Y: The time series to be modeled, an IDDATA object or a column vector of double values. The data object should contain data for one output signal and no input signals. The data must be uniformly sampled. Type "help iddata" for more information. When double values are specified, Y is assumed to be uniformly sampled using a sample time of 1 sec. N: The order of the ar model (positive integer) Output: MODEL: ar model delivered as an IDPOLY object. It has only one active polynomial - "A"; MODEL.a = [1 a1 a2 ... aN]. The estimated variance of the white noise source e(t) is stored in the "NoiseVariance" property of the Model. The model is estimated using "forward-backward" approach with no windowing.

Prestazioni $\frac{W}{m^2}$

- MAE: 14.9857
- RMSE: 41.2735

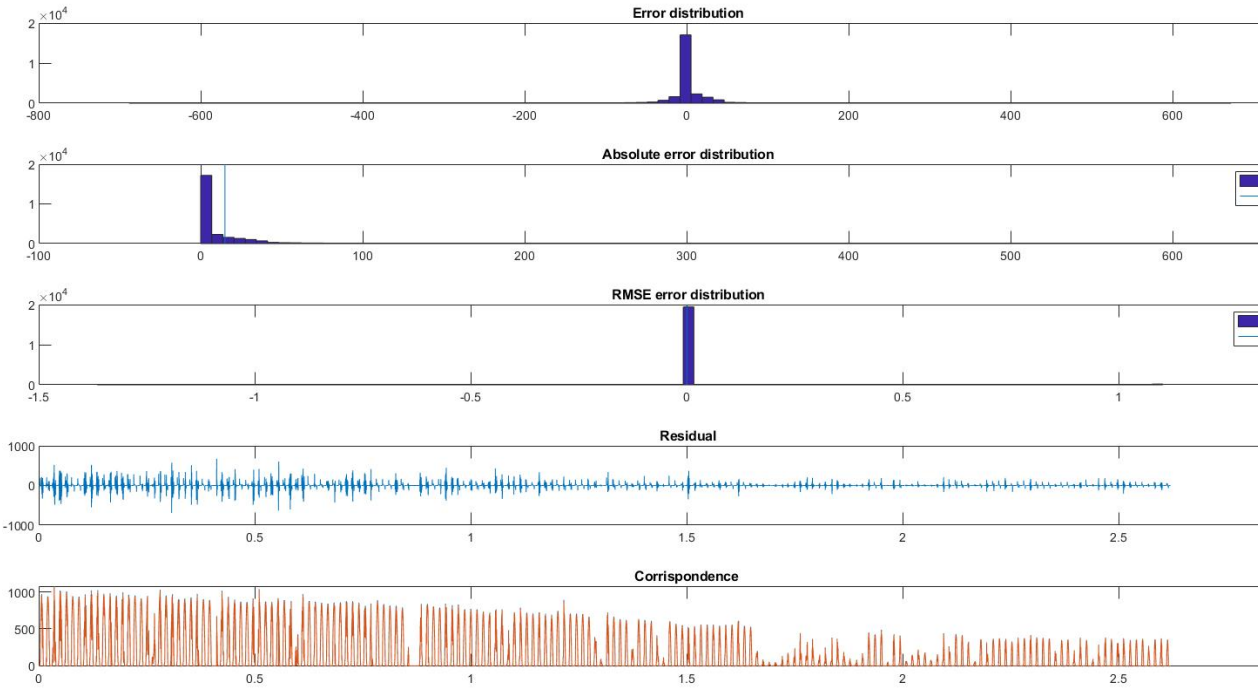


Figure 12.5: AR

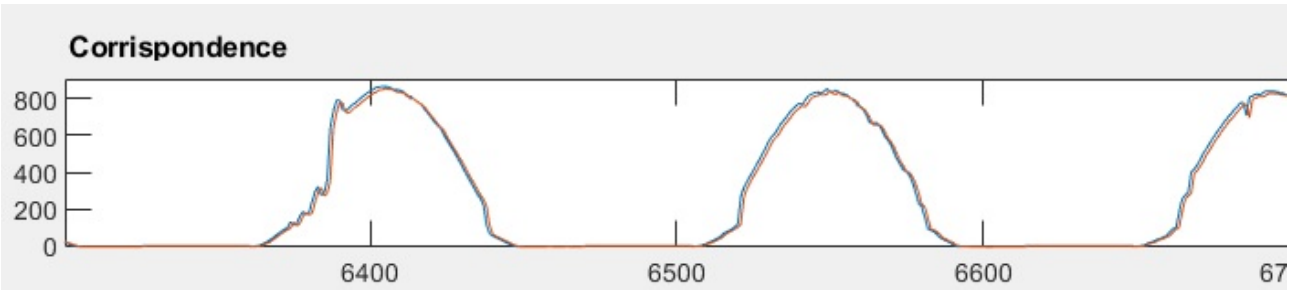


Figure 12.6: AR Detail

Iterando sul grado del polinomio come variabile, il miglior fitting è ottenuto con il grado del polinomio 14.
Prestazioni $\frac{W}{m^2}$

- MAE: 14.0703
- RMSE: 39.0968

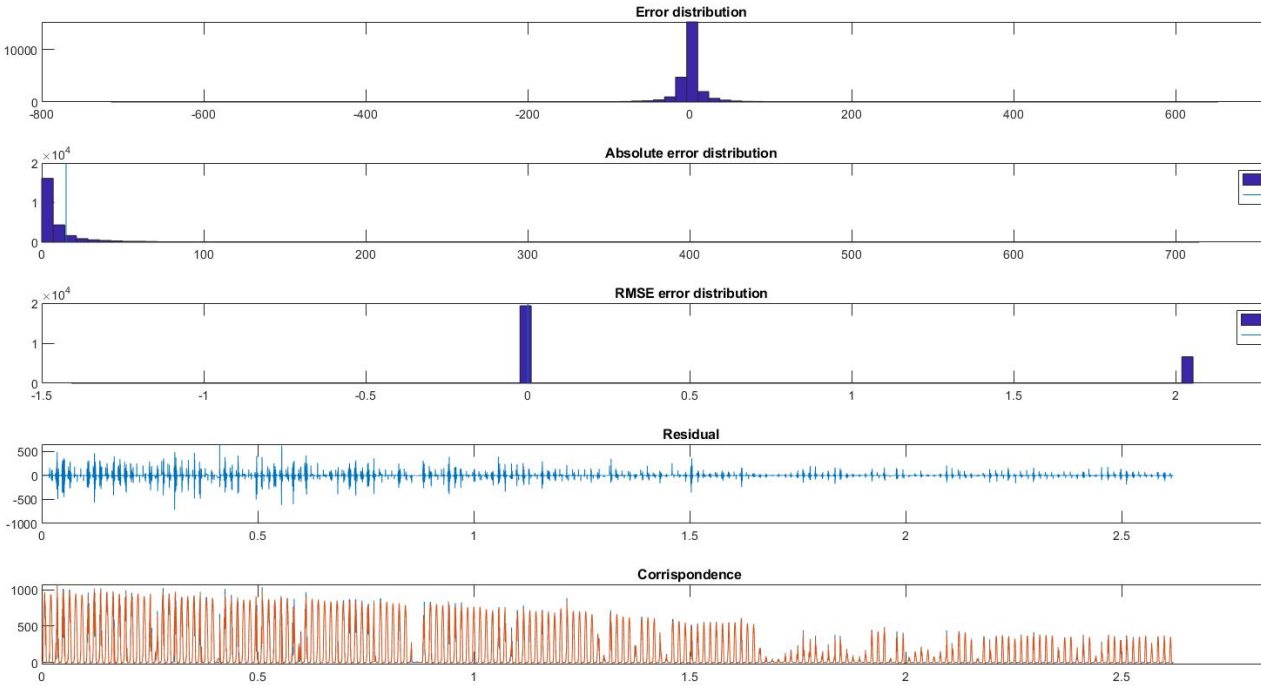


Figure 12.7: Ar Model 14

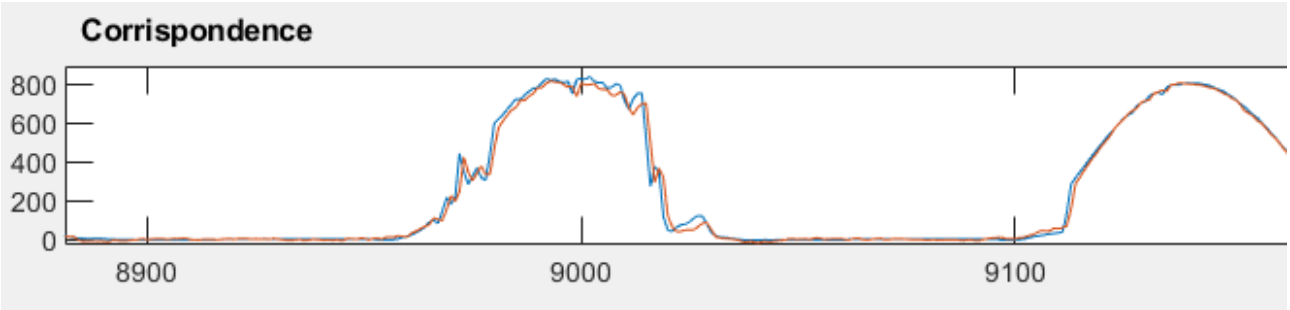


Figure 12.8: Ar 14 Detail

12.5 ARMAX Model

Con l'utilizzo della modellizzazione *Armax* è possibile definire un predittore che utilizza anche variabili esogene esterne.

> armax Estimate armax polynomial model using time domain data.
>
> SYS = armax(Z, [na nb nc nk]) estimates an armax model, M, represented by: A(q) y(t) = B(q) u(t-nk) + C(q) e(t) where: na = order of A polynomial (Ny-by-Ny matrix) nb = order of B polynomial + 1 (Ny-by-Nu matrix) nc = order of C polynomial (Ny-by-1 matrix) nk = input delay (in number of samples, Ny-by-Nu entries) (Nu = number of inputs; Ny = number of outputs)
>
> The estimated model, SYS, is delivered as an @idpoly object. SYS contains the estimated values for A, B, and C polynomials along with their covariances and structure information.

Iterando sulle varianili esogene e scegliendo la combinazione migliore otteniamo che :
Prestazioni $\frac{W}{m^2}$

- MAE: 106.14
- RMSE: 77.2954

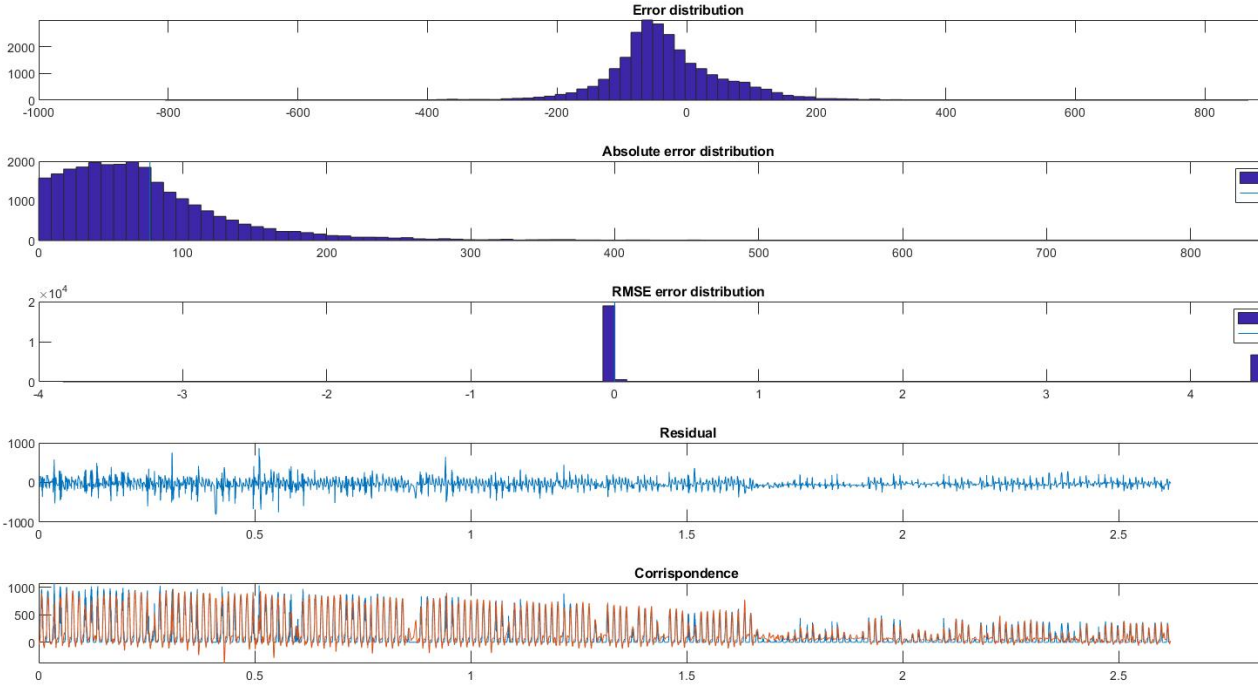


Figure 12.9: Armax Model

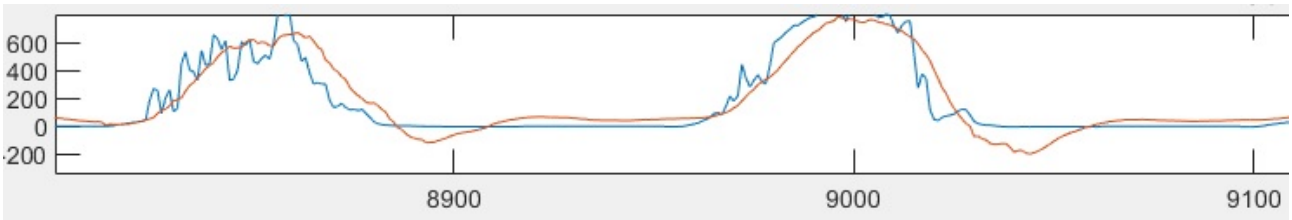


Figure 12.10: Armax Detail

12.6 Bagger Tree Model

Il penultimo metodo analizzato fa capo al metodo Bagger Tree spiegato nei capitoli precedenti.

> TreeBagger Bootstrap aggregation for an ensemble of decision trees. TreeBagger bags an ensemble of decision trees for either classification or regression. 'Bagging' stands for 'bootstrap aggregation'. Every tree in the ensemble is grown on an independently-drawn bootstrap replica of input data. Observations not included in this replica are "out of bag" for this tree. To compute prediction of an ensemble of trees for unseen data, TreeBagger takes an average of predictions from individual trees.
>
> To estimate the prediction error of the bagged ensemble, you can compute predictions for each tree on its out-of-bag observations, average these predictions over the entire ensemble for each observation and then compare the predicted out-of-bag response with the true value at this observation.

Iterando sulle variabili esogene e scegliendo la combinazione migliore otteniamo :
Prestazioni $\frac{W}{m^2}$

- MAE: 20.54
- RMSE: 54.85

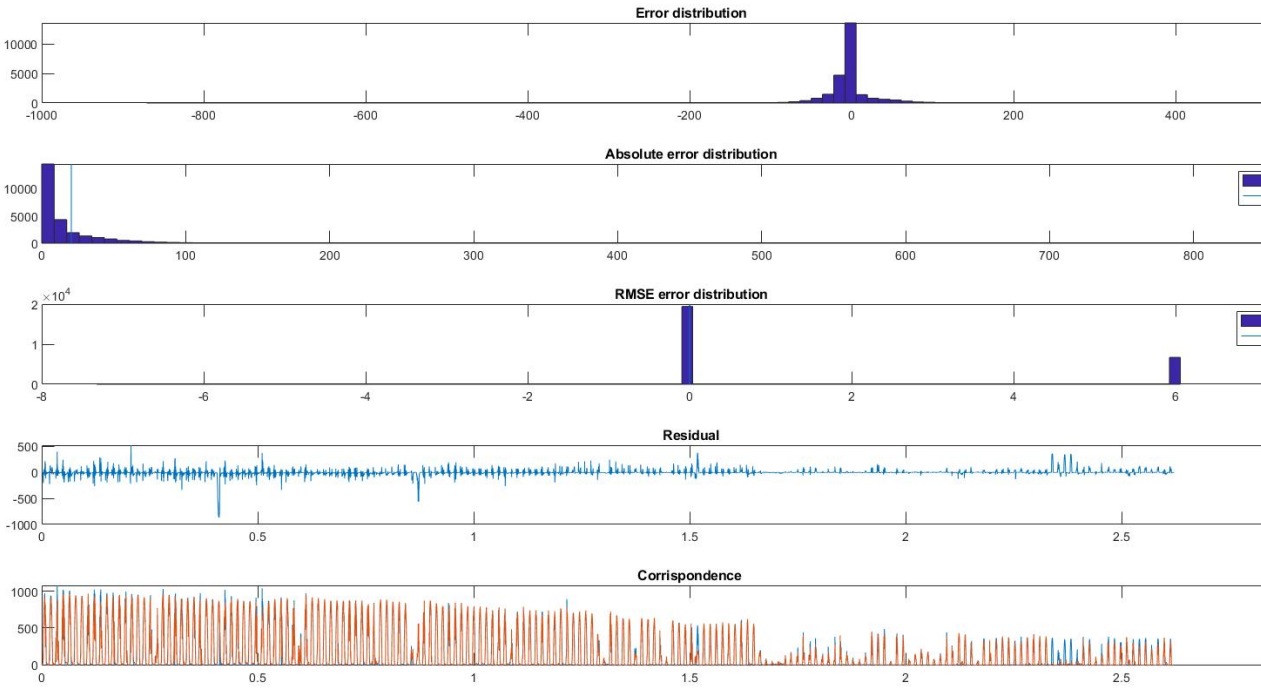


Figure 12.11: Bagger Tree

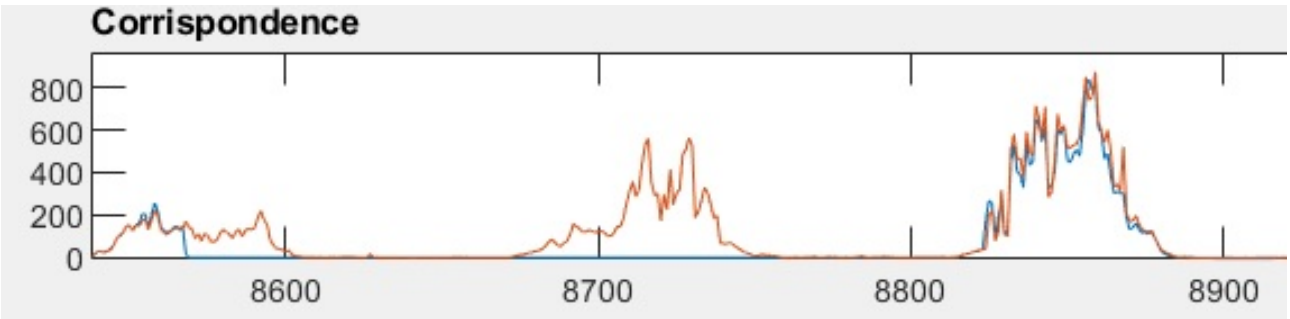


Figure 12.12: Bagger Tree Detail

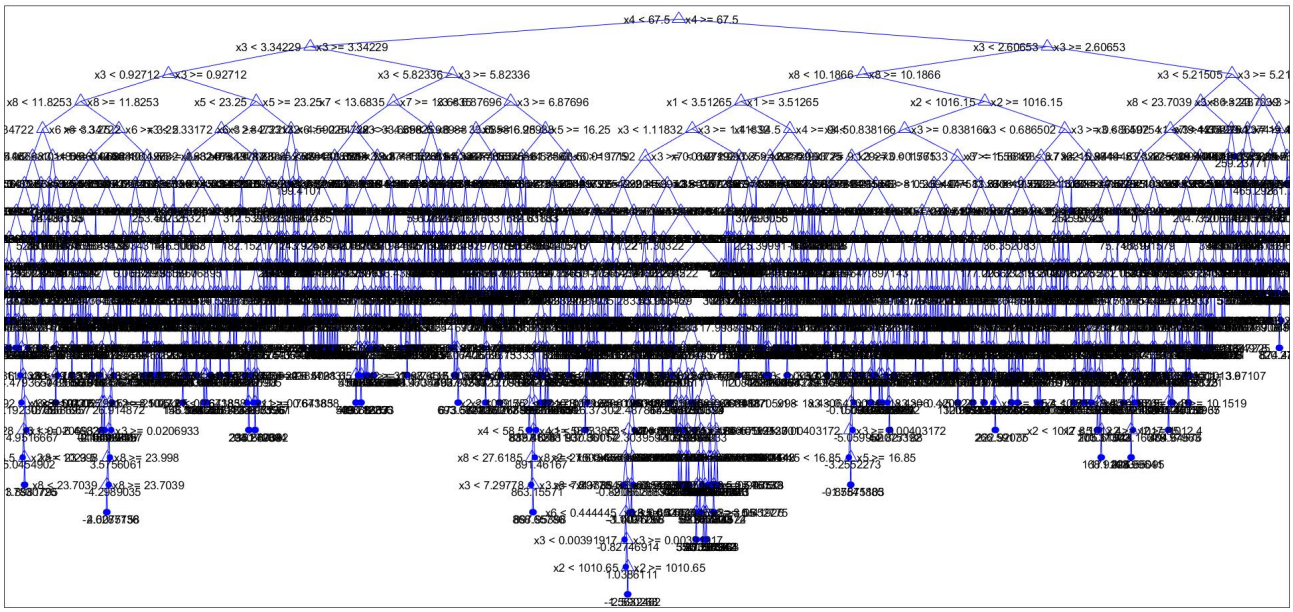


Figure 12.13: Bagger Tree Structure

12.7 NN Model

L'ultimo metodo analizzato è quello della rete neurale di cui si è parlato anche in questo caso nel capitolo precedente.

Scegliendo 10 come numero di neuroni nascosti otteniamo:

Prestazioni $\frac{W}{m^2}$

- MAE: 19.78
- RMSE: 51.34

Scegliendo invece 20 come numero di neuroni nascosti otteniamo:

Prestazioni $\frac{W}{m^2}$

- MAE: 18.18
- RMSE: 51.20

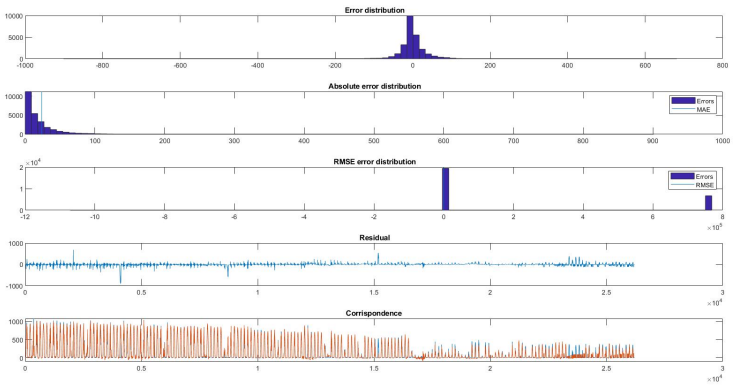


Figure 12.14: NN

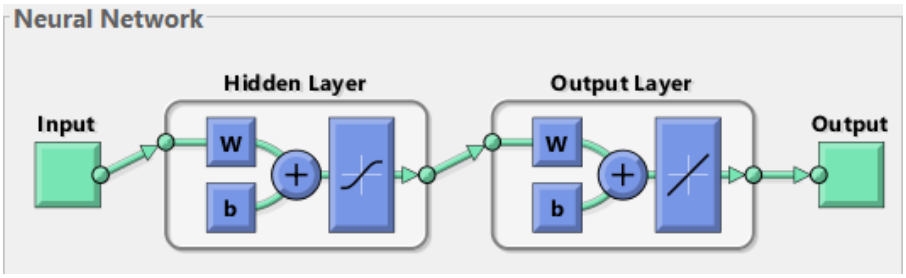


Figure 12.15: NN detail

—

Chapter 13

Circuit Development

Il circuito che deve essere realizzato funge da unità stand alone che, collegata col mondo esterno, permette di raccogliere la serie di dati, elaborarla e inviarla in remoto a un cloud esterno.
L'unità principale e cuore del progetto è il dispositivo CM3 di cui abbiamo ampiamente parlato in precedenza da cui ne deriva il circuito e lo schema in analisi in questo capitolo.
Per poter ottenere una maggiore flessibilità e modularità è stato deciso di suddividere in progetto e la conseguente *PCB* in due circuiti separati.
Questa scelta implica un leggero costo aggiuntivo a fronte di un migliore debug anche in caso di imprevisti.

13.1 Official CM3 Schematic

Il dispositivo CM3 è largamente impiegato in numerosi progetti , tanto che sono disponibili gli schemi ufficiali per poter creare una board ad hoc in base all'utilizzo custom che se ne vuole fare.

Questa sezione si pone il compito di descrivere le funzionalità del circuito disegnato.

13.1.1 Pag.1

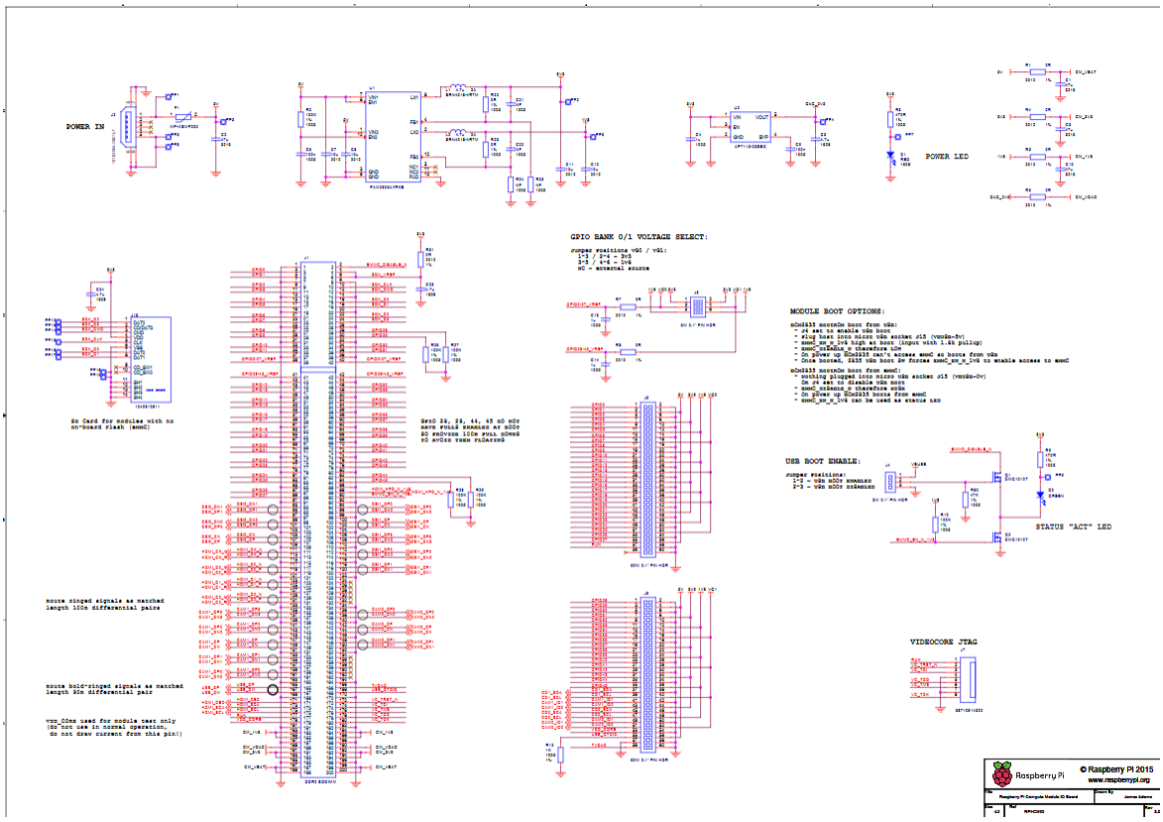


Figure 13.1: CM3 Schematic Pag1

13.1.2 Pag.2

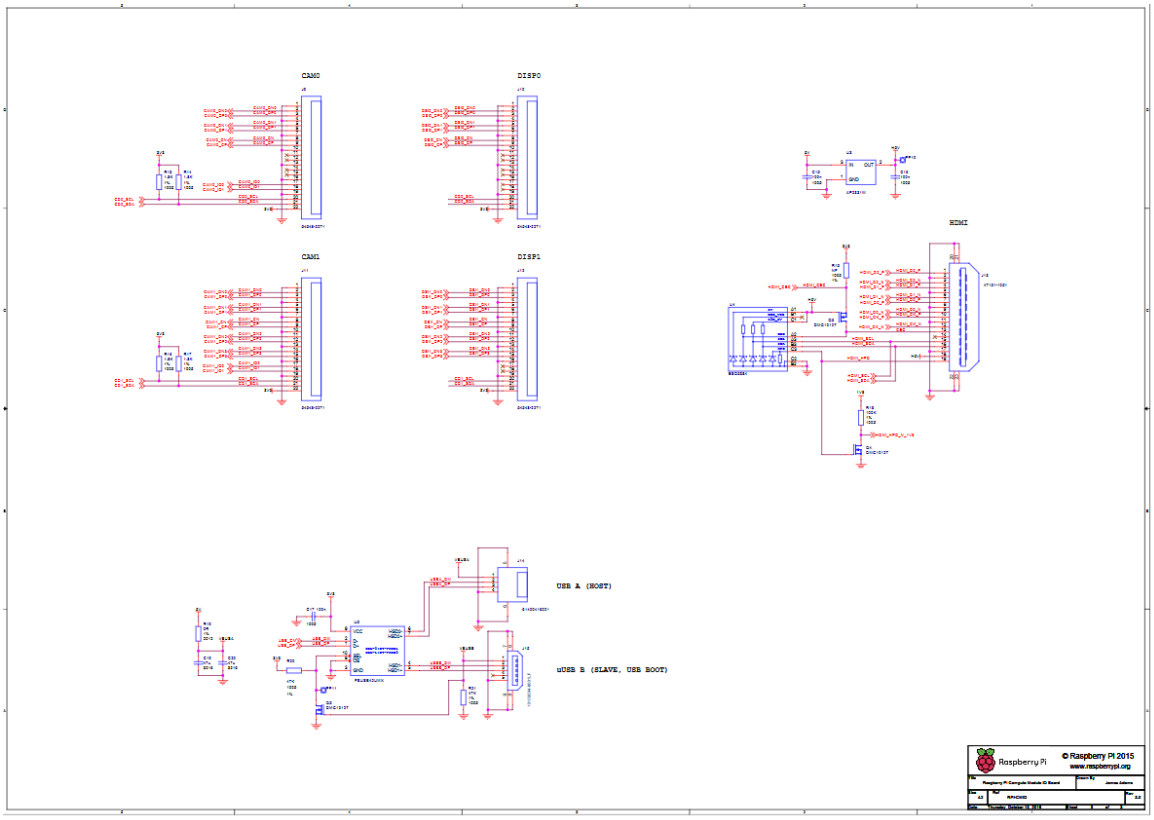


Figure 13.2: CM3 Schematic Pag2

13.1.3 Description - Avaiable BLOCKS

- POWER SUPPLY
- FILTER
- MODULE PIN 0-100
- MODULE PIN 100-200

Power Supply

L'alimentazione del Raspberry CM3 avviene per mezzo di 3 potenziali diversi.

- 5.0V
- 3.3V
- 2.5V
- 1.8V

Il potenziale da 5.0V è quello di origine e viene prelevato senza nessun intervento dal connettore di alimentazione del circuito, di tipo *USB-MINI.*

Viene posto in serie un fusibile a reinserimento automatico da 4A di corrente di intervento e 2A di corrente di mantenimento.

Il tutto è visualizzabile nel blocco seguente.

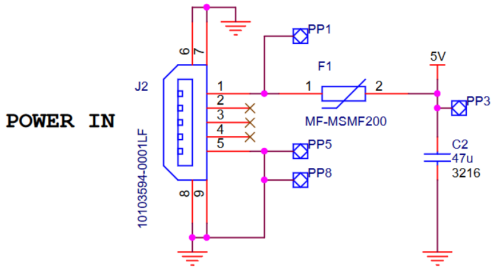


Figure 13.3: CM3 Power

L'alimentazione derivata è da 3.3V e 1.8V e sono tali da alimentare il CM3 nei pin dedicati.

Queste alimentazioni sono ottenute per mezzo di un convertitore stepdown *PAM2306 - DUAL HIGH-EFFICIENCY PWM STEP-DOWN DC-DC CONVERTER* che opera a 1.5MHz ed è in grado di fornire le due correnti di output fino a 1.0A per canale.

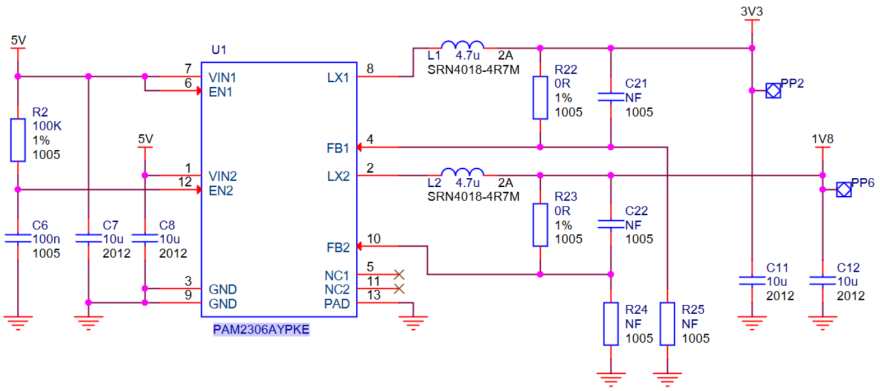


Figure 13.4: Power Supply

Ulteriore alimentazione da 2.5V è obbligatoria per il pin TV_DAC che non sarà utilizzato ma necessita comunque di potenziale presente su di esso .

Per poter ottenere questa alimentazione è importante utilizzare uno stepdown lineare *AP7115 - 150mA LOW DROPOUT LINEAR REGULATOR WITH SHUTDOWN.*

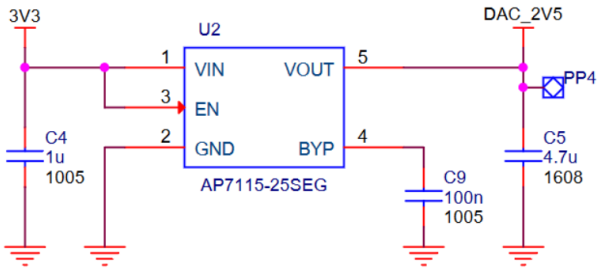


Figure 13.5: Power Supply 2

Esso come descritto nel relativo datasheet fornisce 2.5V con una uscita massima di 150mA.

Filter

Una volta stabilizzate, le tensioni, subiscono un filtraggio composto solamente da condensatori da 47uF,in parallelo che vengono posizionati nel design originale a ridosso del CM3.

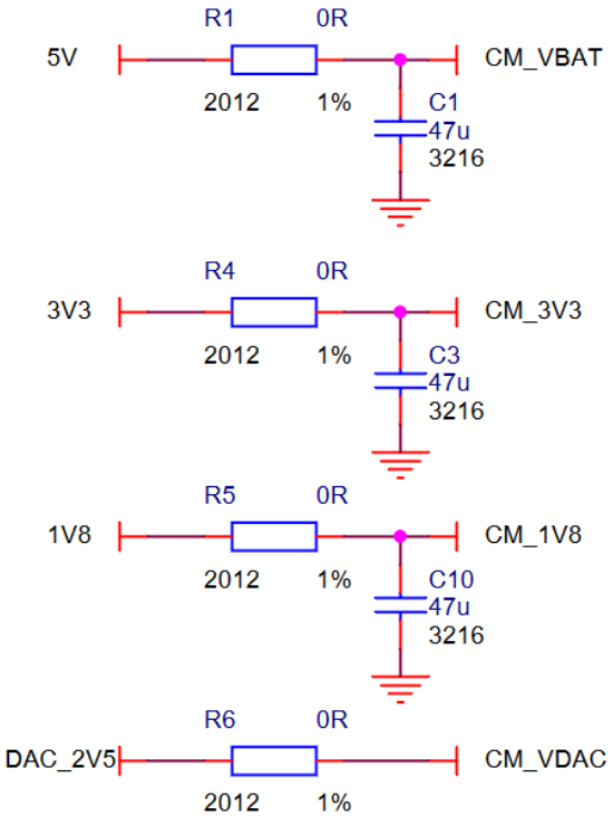


Figure 13.6: Filter 1

Per riassumere, il CM3 ha bisogno sempre delle seguenti alimentazioni anche se i GPIO e i TVDAC non vengono utilizzati.

Supply	Descripion	Minimum	Typical	Maximum	Unit
VBAT	Core SMPS Supply	2.5	-	5.0 + 5%	V
3V3	3V3 Supply Voltage	3.3 - 5%	3.3	3.3 + 5%	V
1V8	1V8 Supply Voltage	1.8 - 5%	1.8	1.8 + 5%	V
VDAC	TV DAC Supply[a]	2.5 - 5%	2.8	3.3 + 5%	V
GPIO0-27_VDD	GPIO0-27 I/O Supply Voltage	1.8 - 5%	-	3.3 + 5%	V
GPIO28-45_VDD	GPIO28-27 I/O Supply Voltage	1.8 - 5%	-	3.3 + 5%	V
SDX_VDD	Primary SD/eMMC Supply Voltage	1.8 - 5%	-	3.3 + 5%	V

Figure 13.7: Power Detail

Module Pin 0-100

Dal pin 1 al pin 100 il modulo di per sè non ha bisogno di molti accorgimenti se non alcune alimentazioni , come indicate nello schema, e l'obbligo di mettere a *GND* tramite delle resistenze da 100K 4 pin che altrimenti sarebbero floating.

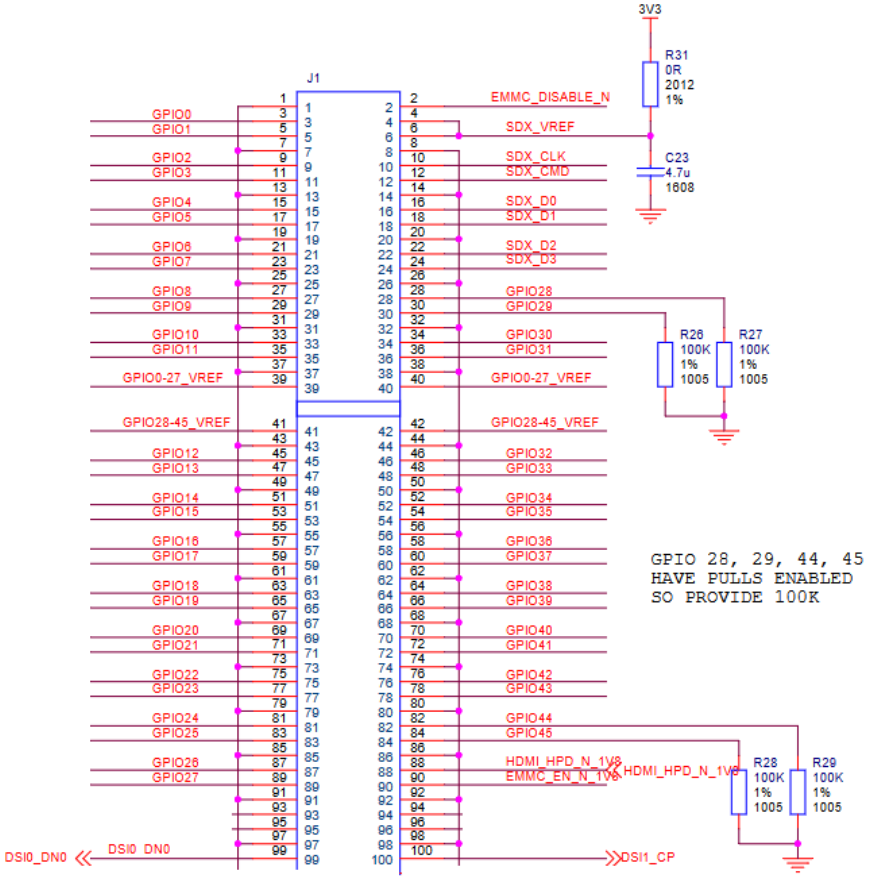


Figure 13.8: PIN 0-100

Sono presenti i GPIO 2 e 3 che contengono la funzionalità di pilotaggio del bus I^2C tramite resistenza di pullup da 1.8K a 3.3V.

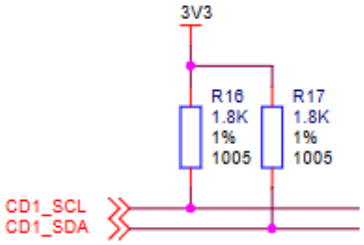


Figure 13.9: Pullup resistor

Module Pin 100-200

Anche nella seconda parte del modulo sono necessarie solo alcune alimentazioni, tuttavia risiedono anche le funzionalità di porta USB, che potranno gestire WI-FI o altri dispositivi.

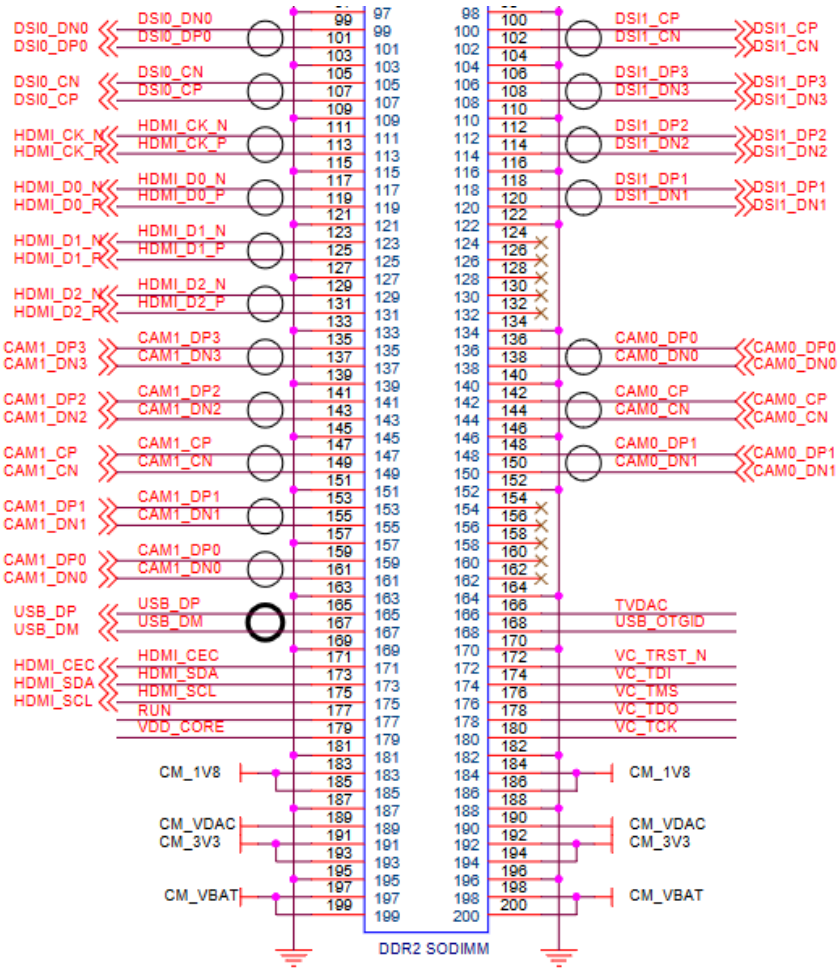


Figure 13.10: PIN 100-200

La funzionalità USB è nativa nei pin 165 e 167.

13.1.4 Description - Unavaiable BLOCKS

Alcuni blocchi sono stati omessi per molteplici ragioni che sono descritti nella relativa sezione.

- BANK VOLTAGE SELECT
- BOOT ENABLE
- VIDEOCORE JTAG
- CAM & DISP OUT
- HDMI
- USB HOST/SLAVE

Bank Voltage Select

A fronte del fatto che nessuno dei GPIO viene utilizzato con diverse tensioni se non 3.3V è stato omesso tutto il circuito di selezione delle tensioni di alimentazione degli stessi.

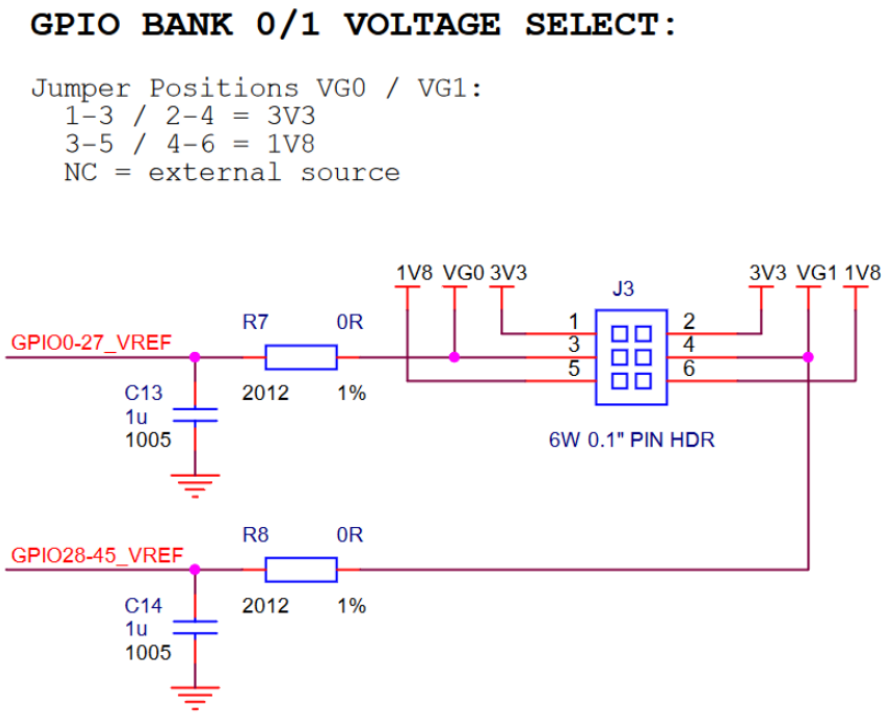


Figure 13.11: Bank Voltage Select

I pin GPIO_REF sono così connessi a una alimentazione 3.3V fissa.

Boot Enable

Per quanto riguarda la gestione della memoria eEMC e del suo contenuto si ritiene necessario che il BOOT e il caricamento del firmware avvenga dall'utente sviluppatore in modo che l'utente finale non possa avere accesso e possibilità di modifica dello stesso.

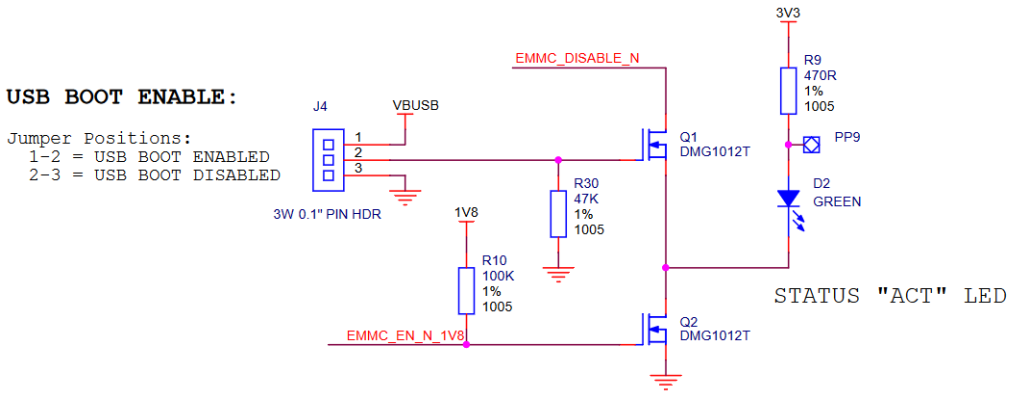


Figure 13.12: Boot Enable

VideoCore JTAG

Non essendo presente nessuna interfaccia Video è stata eliminata anche la relativa presa di diagnostica JTAG nata per il suo*Debug*.

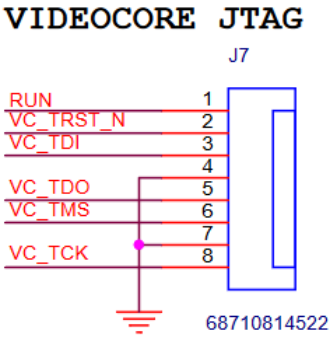


Figure 13.13: VideoCore

CAM & DISP OUT

Come da progetto non è presente nessuna camera ne nessun display associato.

Per questo motivo sono state eliminate anche le uscite trattate in questa sezione.

E' stato necessario recuperare la funzione di pullup per i pin relativi all' I^2C.

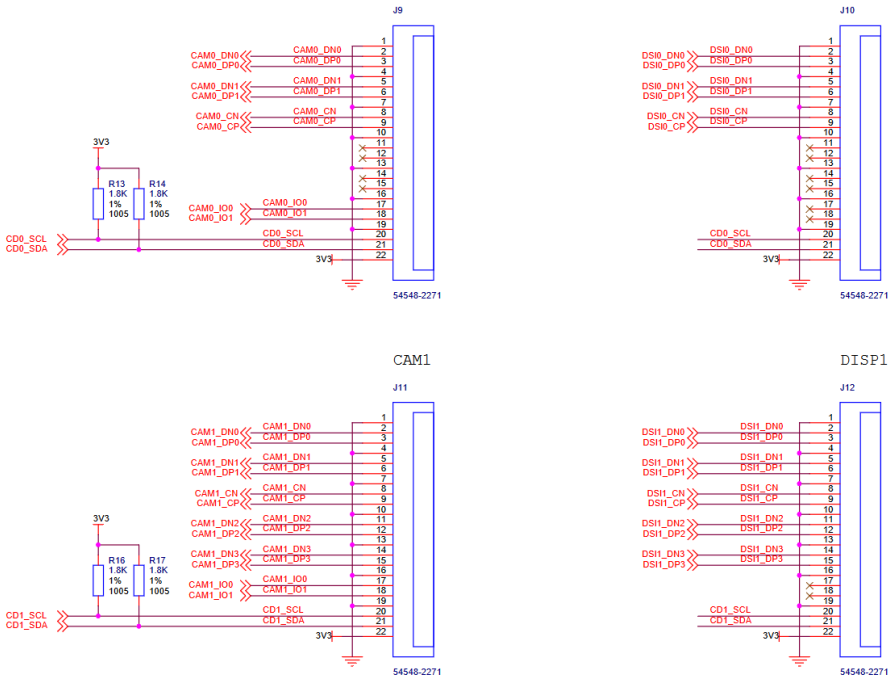


Figure 13.14: Cam and Disp Out

HDMI

Come per la sezione precedente la funzionalità HDMI è stata eliminata in quanto non sarà la modalità di diffusione dei risultati.

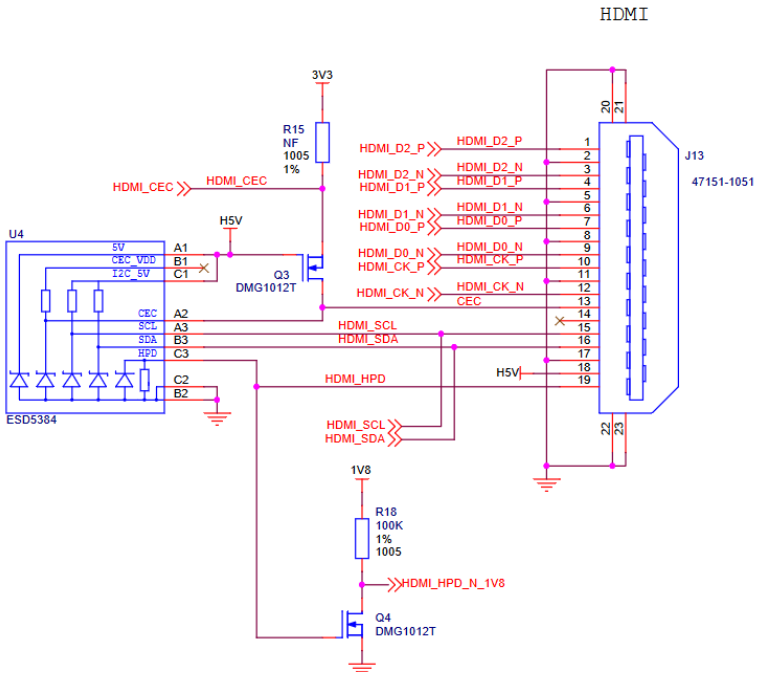


Figure 13.15: HDMI

USB Host & Slave

Non essendo necessario utilizzare la doppia funzionalità USB, si è reso possibile eliminare il MUX usb a favore di una sola porta.

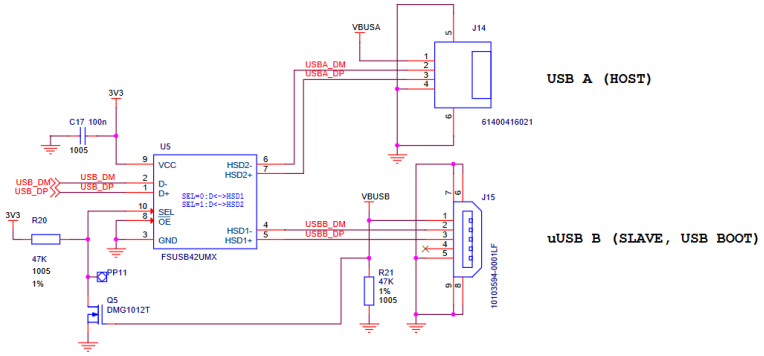


Figure 13.16: USB Host

13.2 Official Sensors Schematic

I sensori sono disponibili con breakout board presso alcuni shop come , ***sparkfun.com*** e ***adafruit.com***, per questo motivo esistono schemi relativi ad essi e alle breakout board associate.

Essi sono sostanzialmente sempre divisi in questi blocchi comuni a tutti i sensori:

- Power Supply
- Ic
- Level Shifter

Power Supply

I sensori vengono alimentati tutti a 3.3V ma per isolare l'alimentazione e fornirne una di maggiore qualità è stato scelto un buck converter lineare *MIC5225 3.3* in grado di alimentare tutti i sensori necessari privandoli del noise che un alimentatore switching fornirebbe.

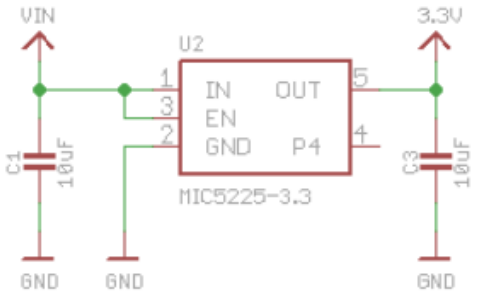


Figure 13.17: Regulator

IC

L'IC è dedicato per ogni applicazione ed è già stato descritto in precedenza in questo testo.

Level Shifter

A causa della doppia compatibilità con la piattaforma Arduino le breakout board utilizzano un level shifter da 5V a 3.3V

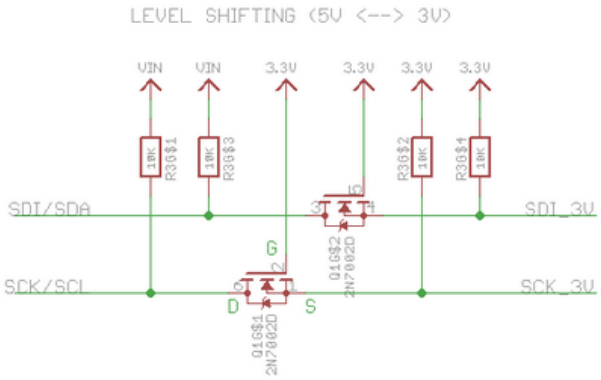


Figure 13.18: Level Shifter

13.2.1 BME680

Object

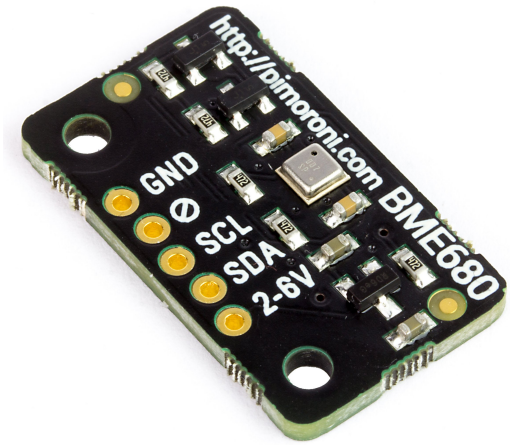


Figure 13.19: BME 680 Board

Schematic

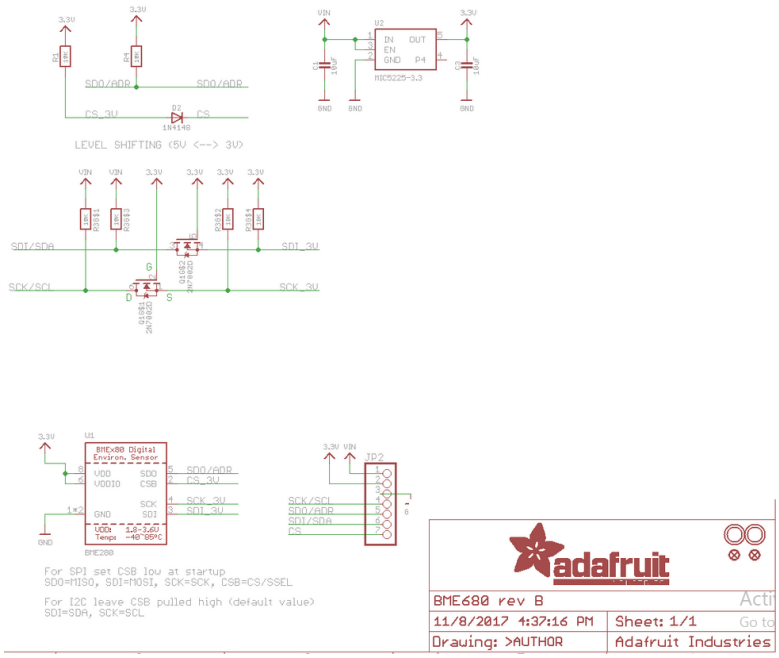


Figure 13.20: BME 680 Schematic

13.2.2 VEML7700

Object

Figure 13.21: VEML 7700 Board

Schematic

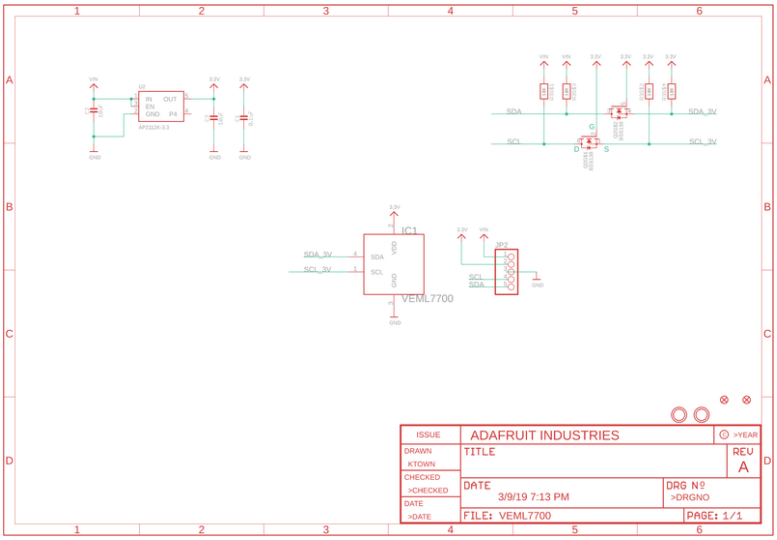


Figure 13.22: VEML 7700 Schematic

13.2.3 AS7262

Object

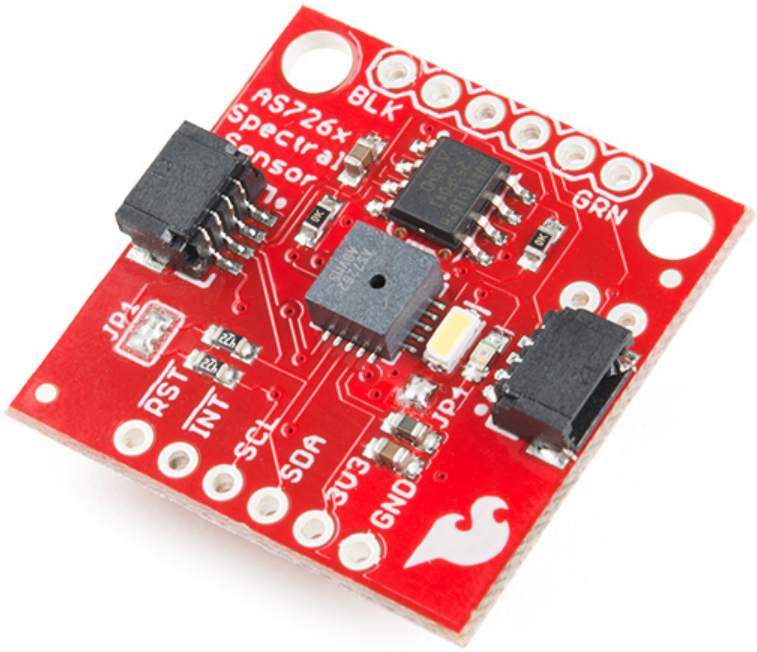


Figure 13.23: AS7262 Board

Schematic

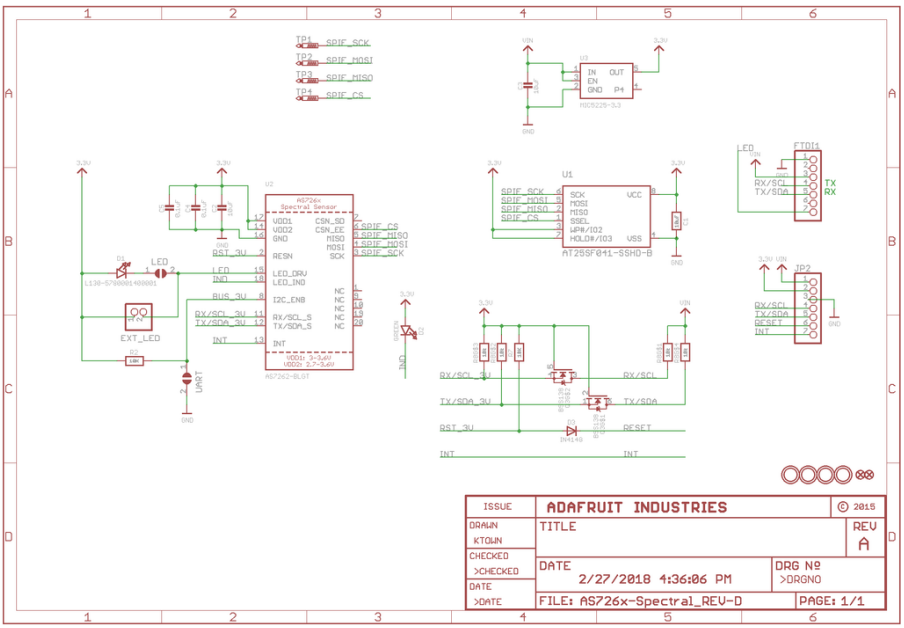


Figure 13.24: AS7262 Schematic

13.3 Development CM3 Schematic

In base a quanto detto nel capitolo precedente si è scelto di snellire al massimo la piattaforma e di eliminare tutte le perifiche non atte al nostro utilizzo.

Per questo motivo attraverso il software di modellazione pcb *Altium* si è reso possibile approcciare lo schema e il layout definitivo per il sensore.

Seguendo le generalità dello schema ufficiale è stata aggiunta successivamente la possibilità di gestire la periferica usb aggiuntiva e integrare il modulo WIFI.

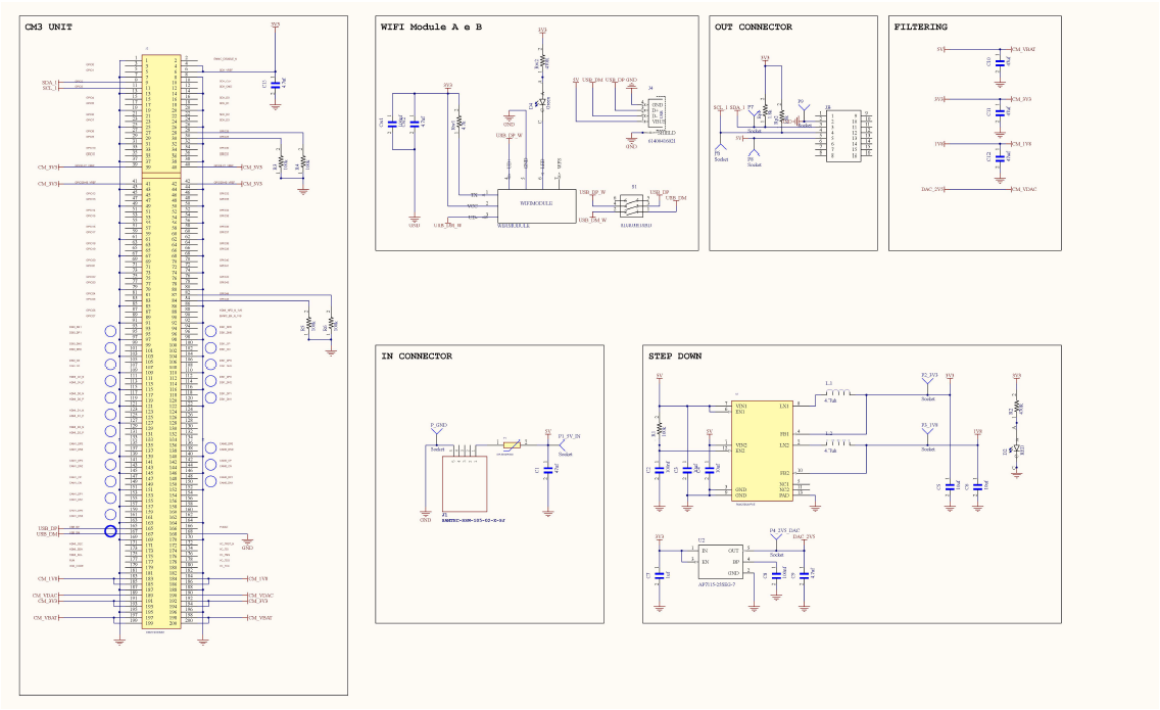


Figure 13.25: CM3 Schematic 1

In aggiunta allo schema *Official* troviamo quindi il dispositivo WIFI e la possibilità di usare un pcb aggiuntivo per i sensori.

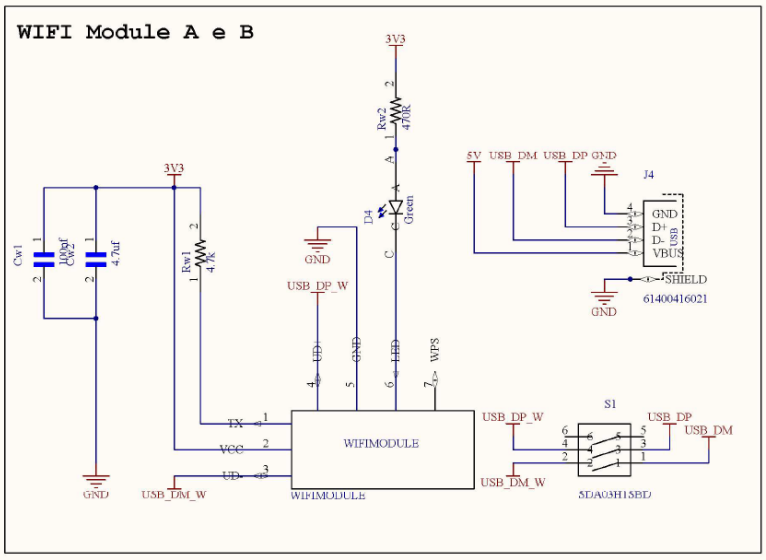


Figure 13.26: Wifi Detail

13.4 Development Sensors Schematic

La sezione sensori non comprende il BME680 che risulterà esterno ma unisce invece tutti i sensori di luce ossia, AS7262, VEML7700 E VEML 6075.

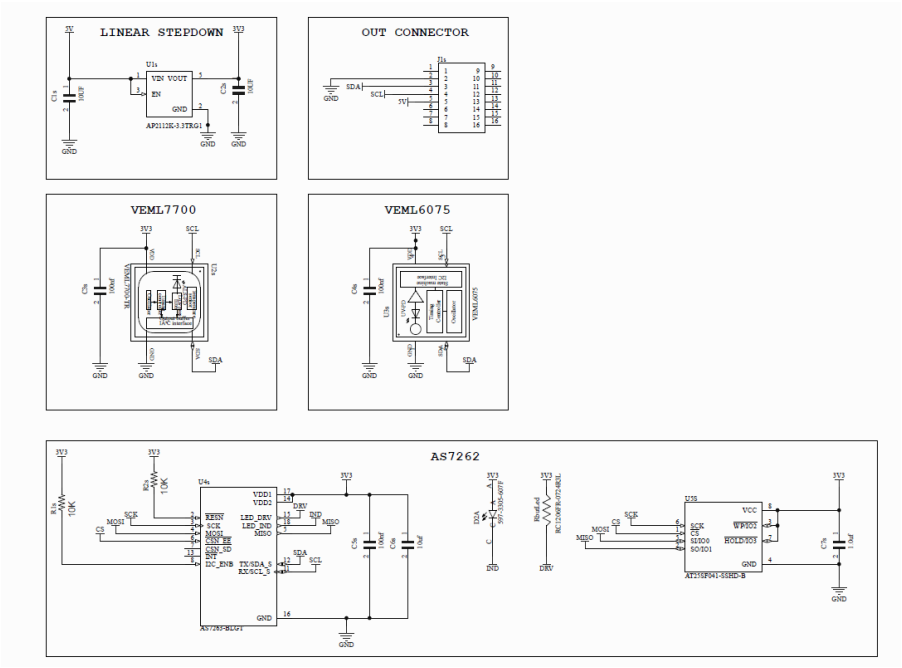


Figure 13.27: Sensor Schematic

13.5 Development CM3 Layout

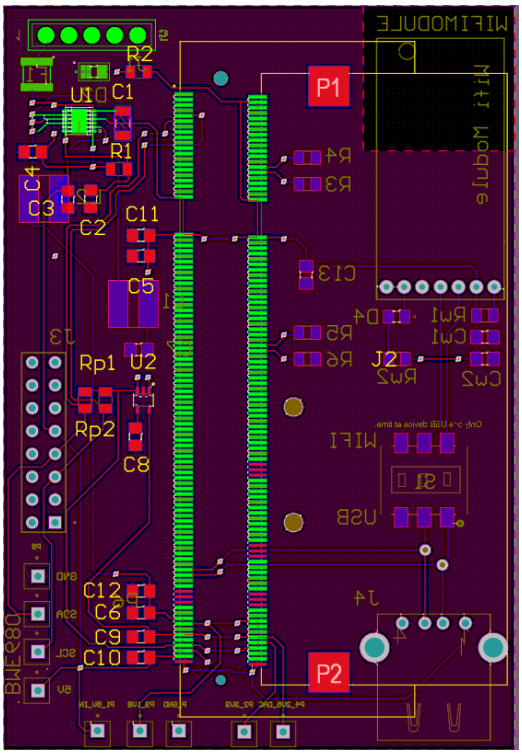


Figure 13.28: Front

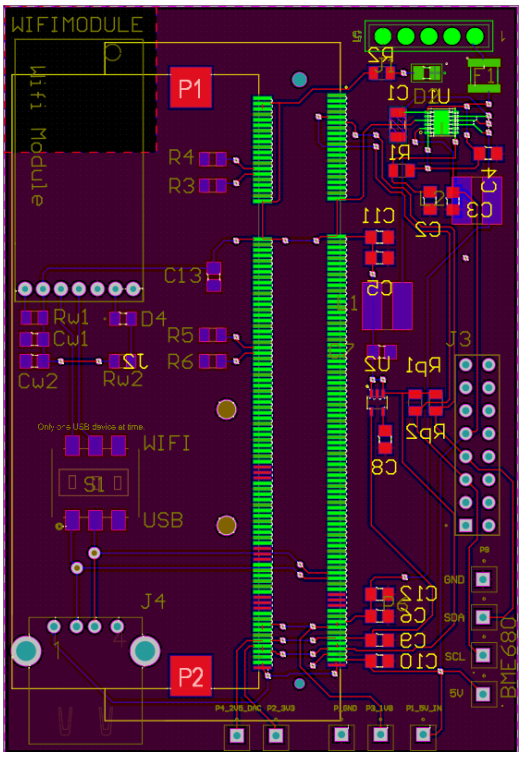


Figure 13.29: Bottom

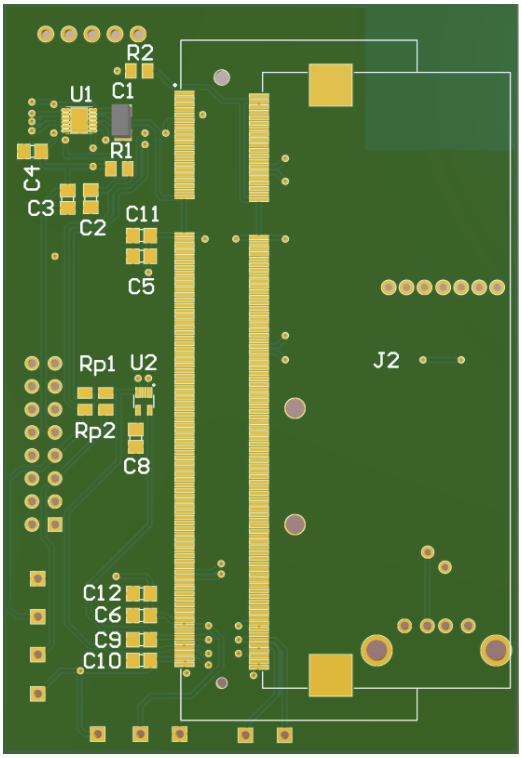


Figure 13.30: Front

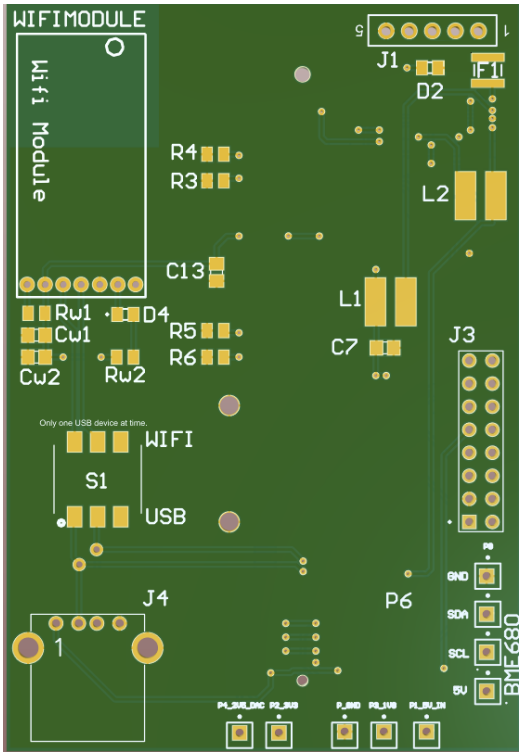


Figure 13.31: Bottom

13.6 Development Sensors Layout

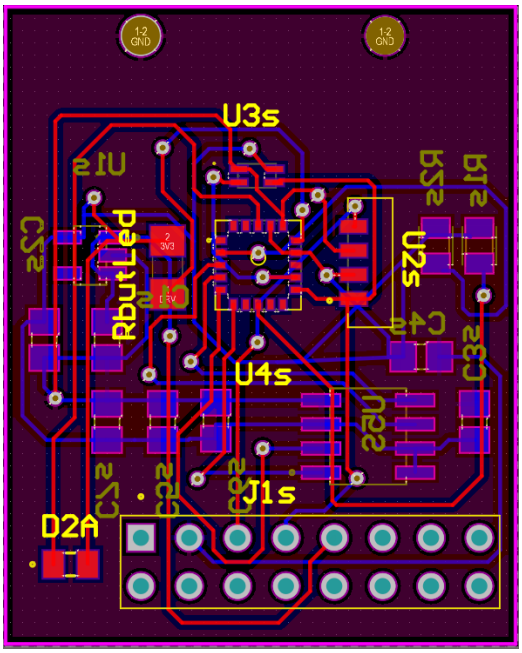


Figure 13.32: Front

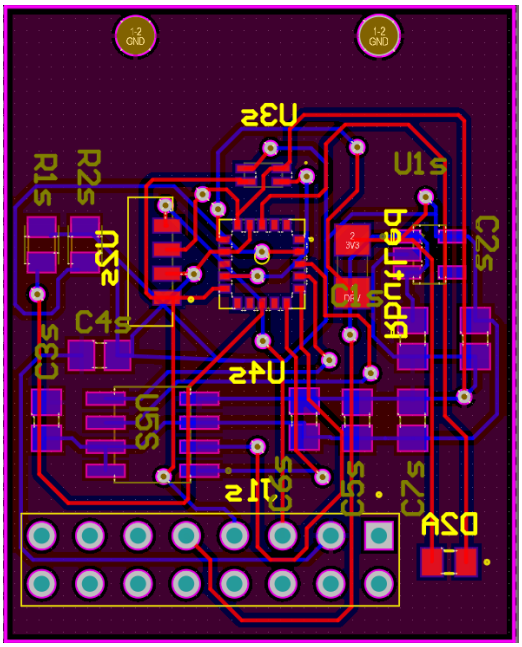


Figure 13.33: Bottom

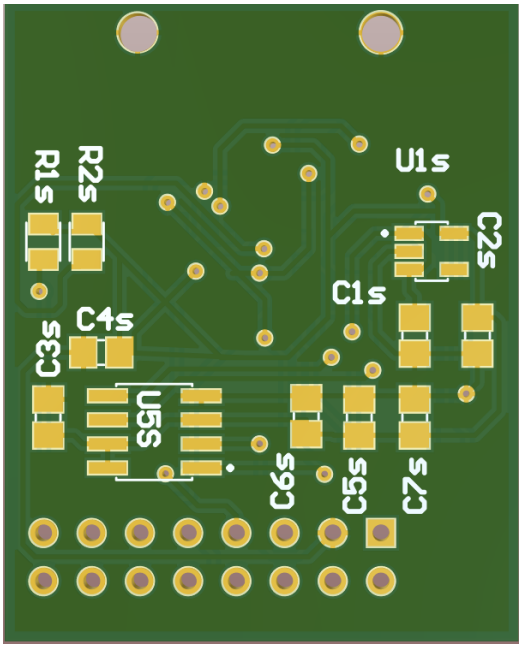


Figure 13.34: Front

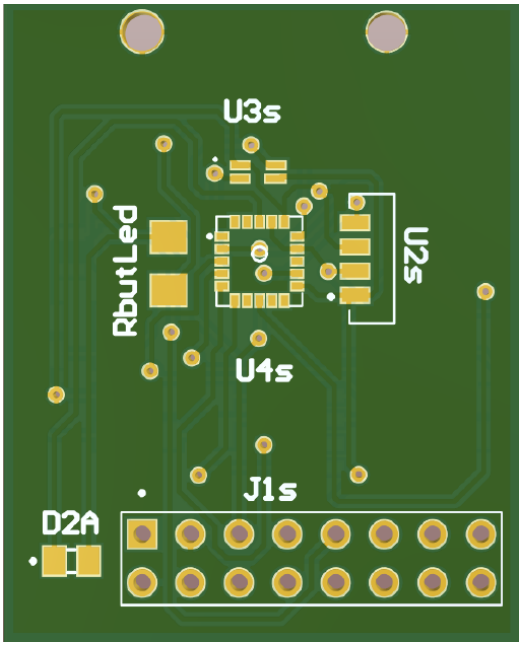


Figure 13.35: Bottom

13.7 Prototype

La realizzazione delle pcb prototipali è stata affidata al produttore cinese *PCBway* che le ha realizzate secondo le specifiche richieste.

13.7.1 CM3 Board

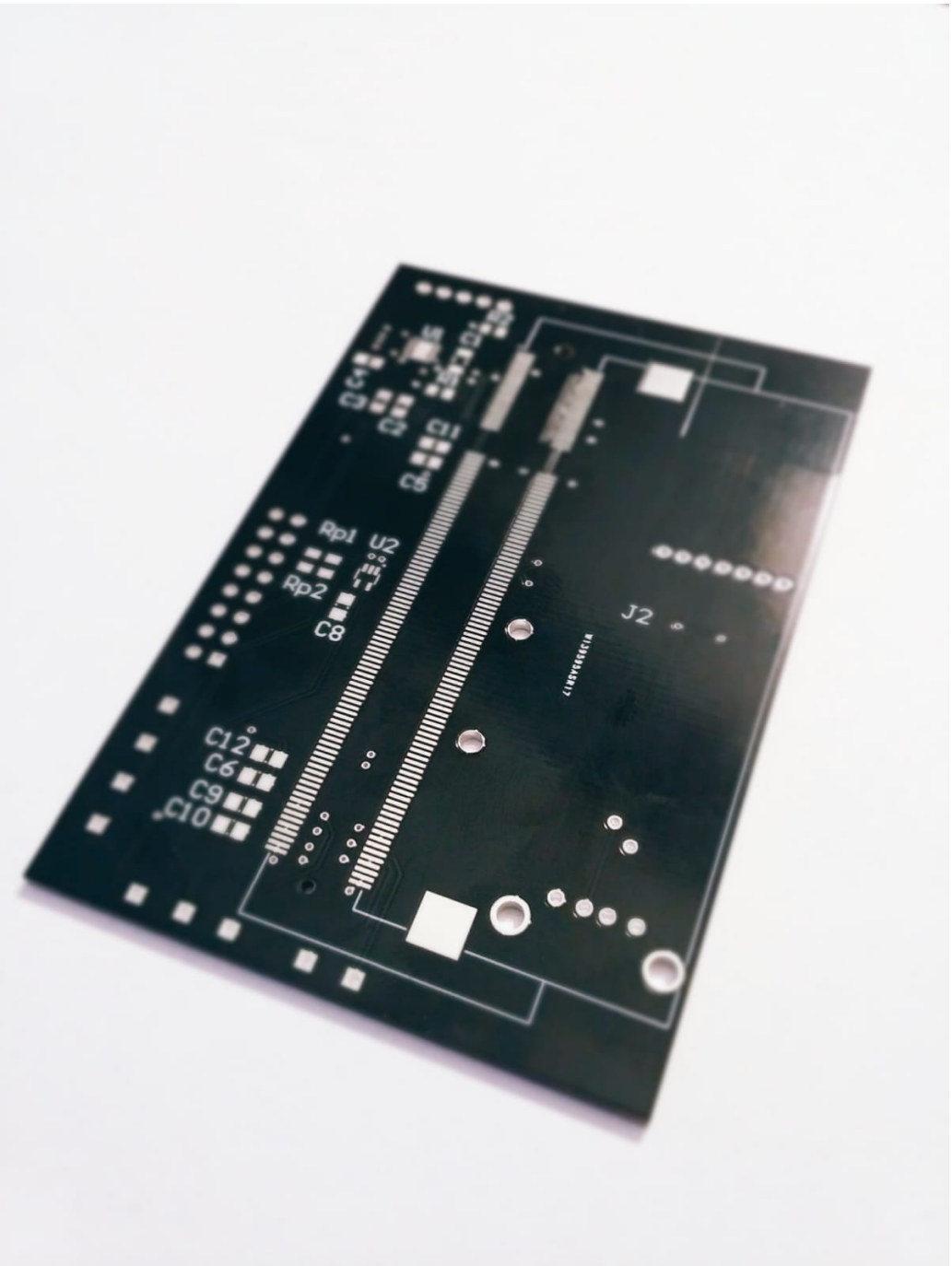


Figure 13.36: Front

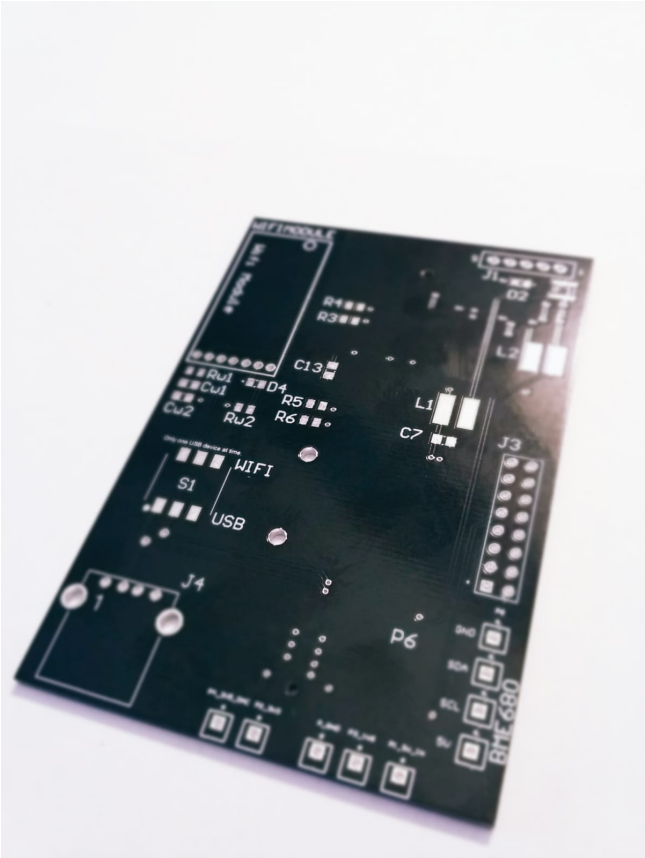


Figure 13.37: Bottom

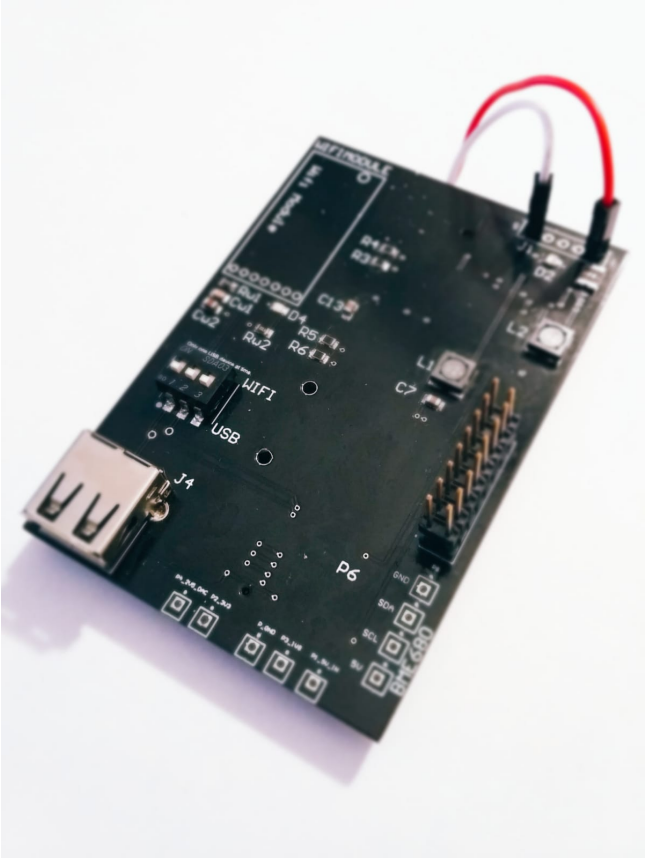


Figure 13.38: Front

Figure 13.39: Bottom

13.7.2 Sensor Board

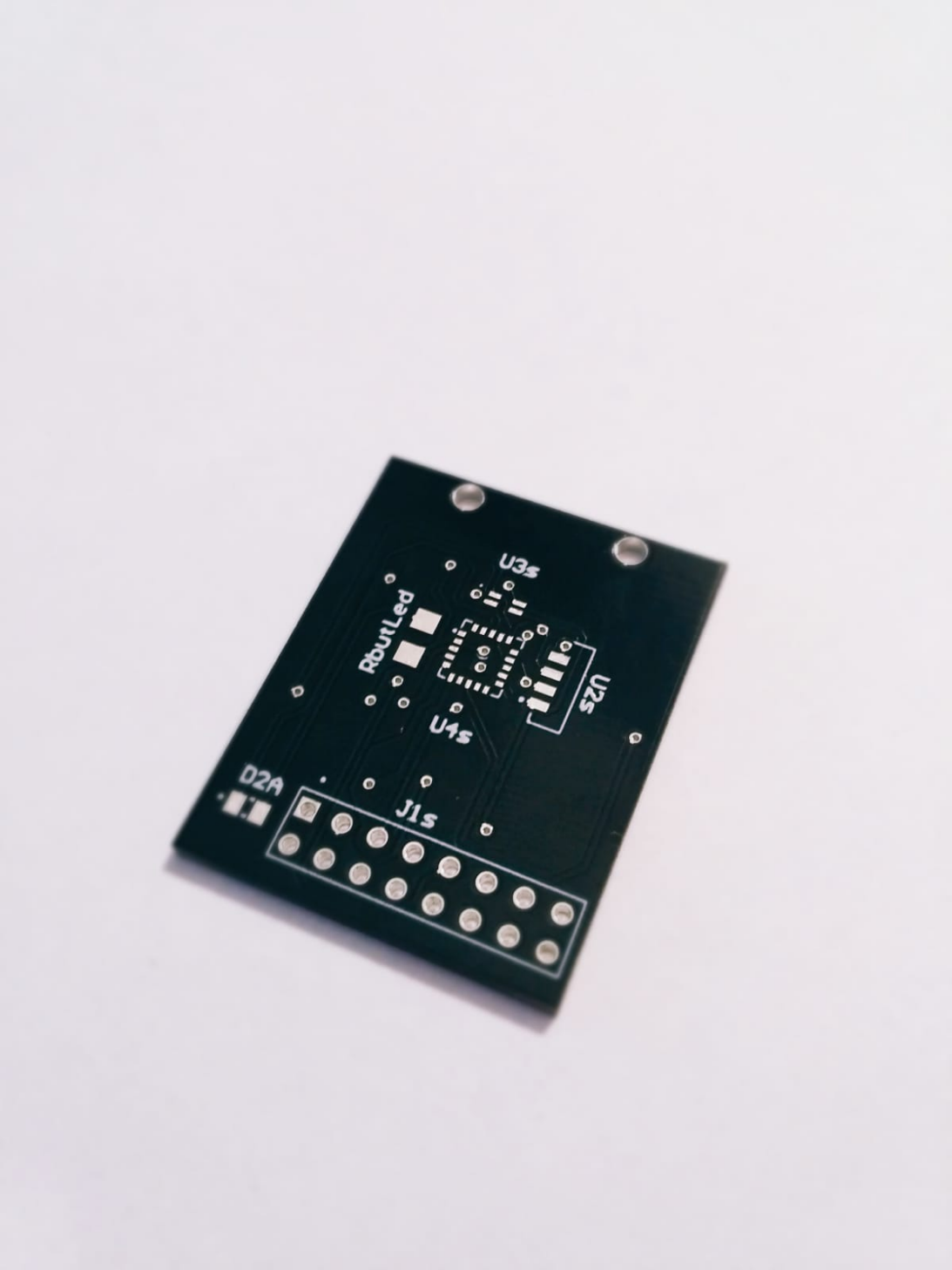


Figure 13.40: Front

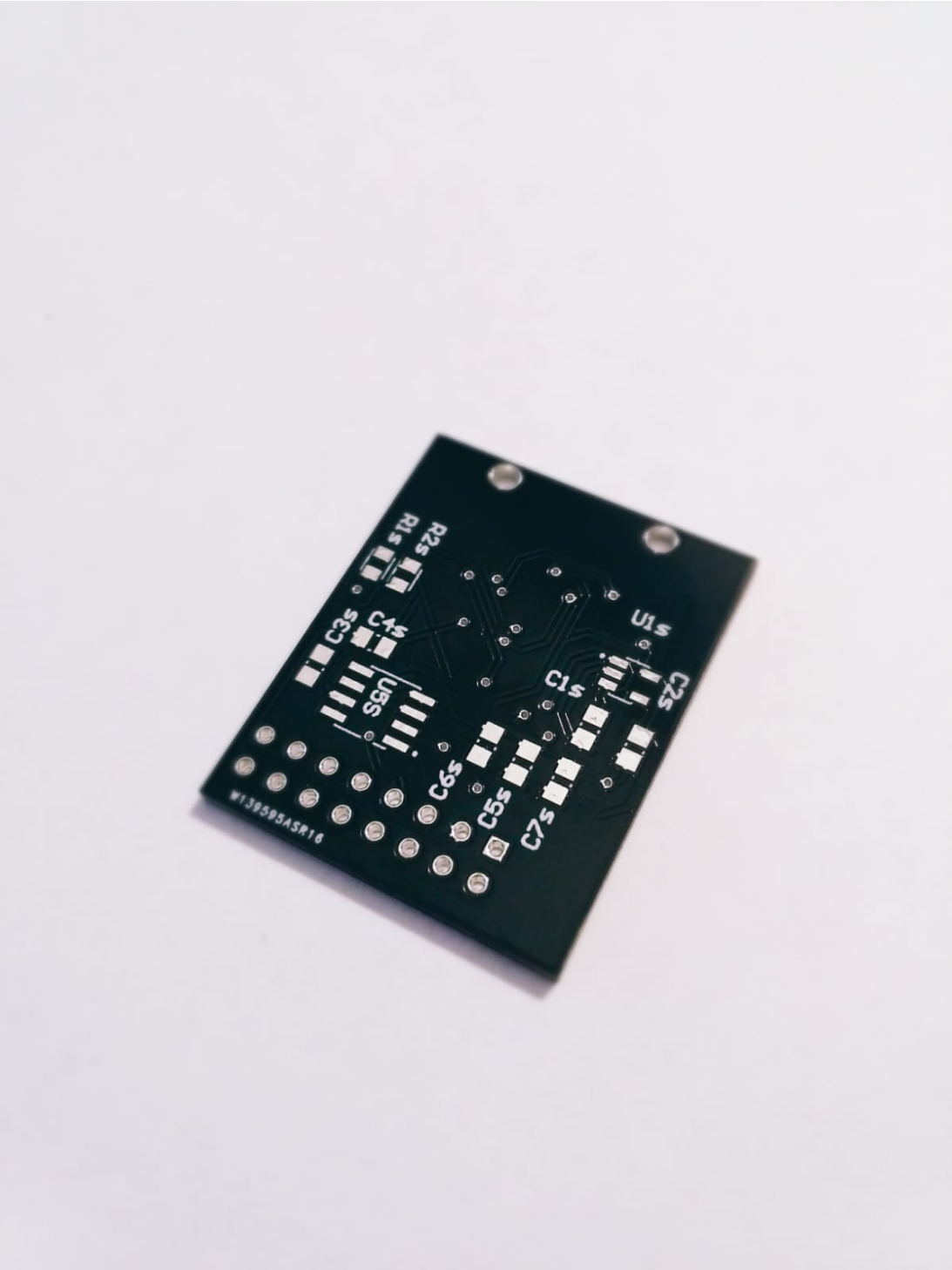


Figure 13.41: Bottom

13.7.3 Sensor Revision

Il bus di comunicazione predilige un indirizzo di oggetto unico, per cui nell'architettura I^2C non risulta possibile senza multiplexer esterni la comunicazione con device che abbiano lo stesso address.
Per questo motivo la PCB dei sensori è stata modello VEML6075V che ha un altro address e presenta però due pin supplementari, *RESET e ACK* opportunamente settati.

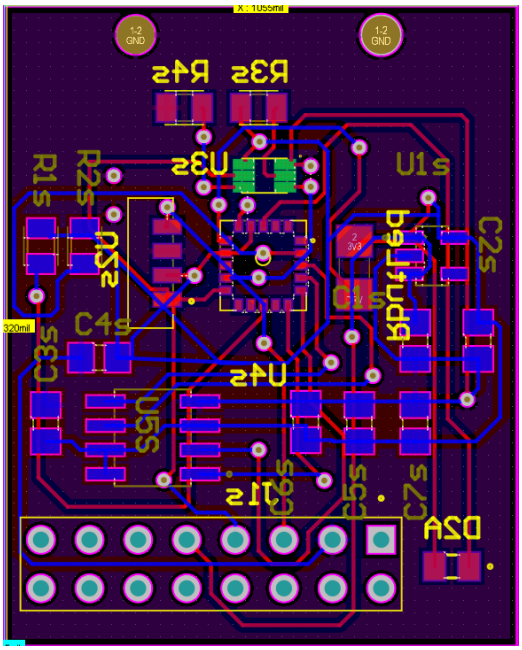


Figure 13.42: Revision 1 Front

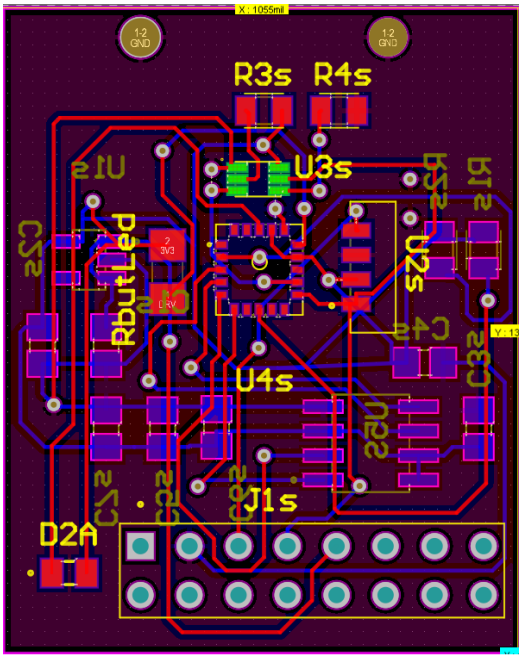


Figure 13.43: Revision 1 Bottom

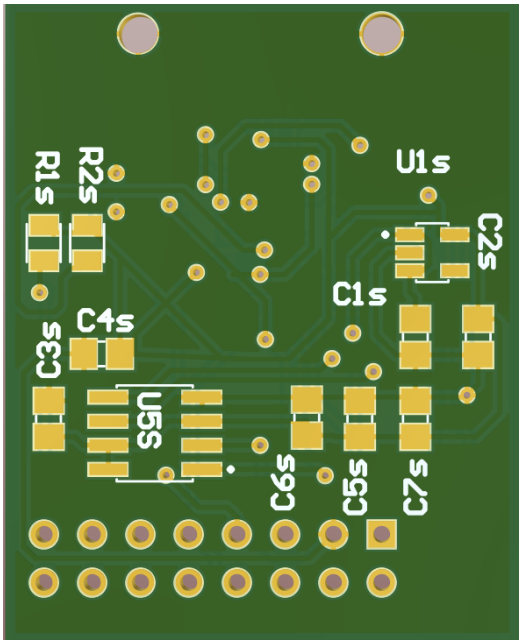


Figure 13.44: Front

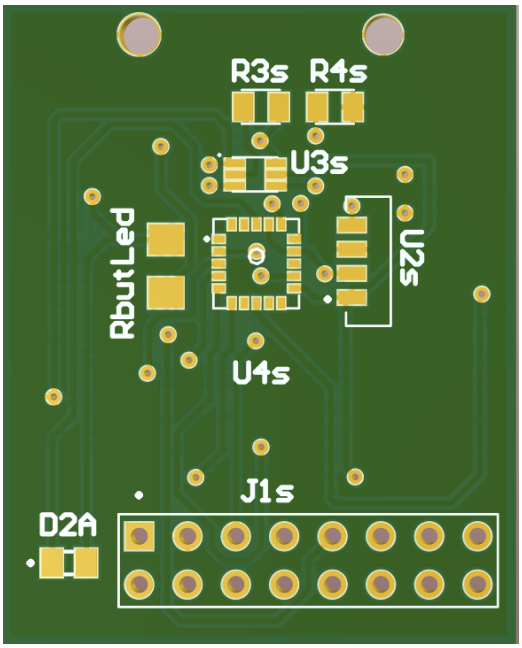


Figure 13.45: Bottom

Una volta assemblata la board si presenta come di conseguenza. La saldatura dei componenti con package QFN è stata eseguita tramite forno.

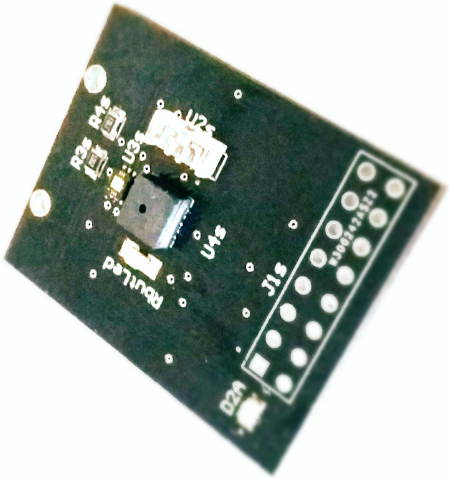


Figure 13.46: Sensor Board Front

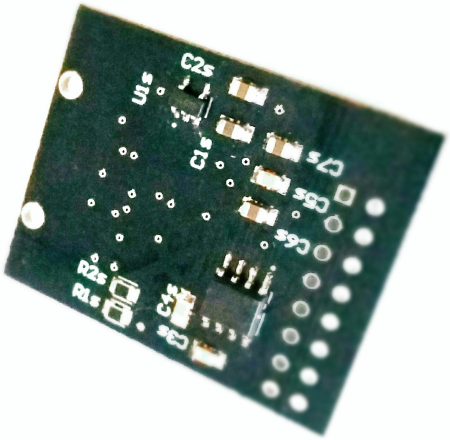


Figure 13.47: Sensor Board Rear

Chapter 14

Python Code

L'implementazione della sezione software avviene per mezzo del linguaggio *Pyhton* largamente usato in ambiente Unix e con HW Raspberry.

14.1 Story

La storia del linguaggio di programmazione Python risale alla fine degli anni '80. Fu concepito alla fine degli anni '80 e la sua attuazione iniziò nel dicembre 1989 da Guido van Rossum.Esso nasce nei Paesi Bassi come successore del linguaggio di programmazione ABC in grado di gestire eccezioni e interfacciamento con il sistema operativo Amoeba. Van Rossum è il principale autore di Python e continua ad occupare un ruolo centrale nel decidere la politica di Python.

Python è un linguaggio di programmazione di alto livello per tutti gli usi ed ad oggi è ampiamente utilizzato. La filosofia enfatizza la leggibilità del codice e la sua sintassi consente ai programmatori di esprimere concetti in meno righe di codice rispetto a linguaggi come C ++ o Java. Il linguaggio fornisce costrutti destinati a creare programmi chiari su piccola e grande scala. Python supporta molteplici paradigmi di programmazione, orientati agli oggetti, programmazione funzionale e stili procedurali. È dotato di un sistema di tipo dinamico e di una memoria automatica di gestione e ha una libreria standard ampia e completa. Gli interpreti Python sono disponibili per molti sistemi operativi, consentendo l'esecuzione di codice Python su una vasta gamma di sistemi. Python è un linguaggio di programmazione multi-paradigma.

Molti altri paradigmi sono supportati utilizzando estensioni, compresa la scrittura per contratto e la programmazione logica. Python utilizza la digitazione dinamica e una combinazione di conteggio dei riferimenti e un garbage collector a rilevazione ciclica per la gestione della memoria. Una caratteristica importante di Python è la risoluzione dinamica dei nomi (late binding), che lega i nomi dei metodi e delle variabili durante il programma esecuzione. Il design di Python offre un supporto per la programmazione funzionale secondo la tradizione Lisp. La libreria standard ha due moduli itertools e functools. Inoltre, Python ha vantaggi significativi rispetto ai vari linguaggi di programmazione, ad esempio:

- sintassi pulita (per l'assegnazione di blocchi per utilizzare le deroghe);
- programmi di tolleranza (caratteristici della maggior parte delle lingue interpretate);
- la distribuzione normale ha molti moduli utili (incluso il modulo per lo sviluppo della GUI);
- l'uso di Python in modalità interattiva (molto utile per la sperimentazione e la risoluzione di problemi semplici);
- la distribuzione normale è un ambiente di sviluppo semplice ma allo stesso tempo piuttosto potente, che lo è chiamato IDLE e ciò che è scritto in Python;
- adatto a risolvere problemi matematici (un mezzo per lavorare con numeri complessi, può operare con numeri interi di dimensioni arbitrarie, una finestra di dialogo può essere utilizzata come un potente calcolatore).

Tuttavia, Python presenta ancora alcuni svantaggi. Python come molti altri linguaggi interpretati non si applicano, ad esempio, i compilatori JIT. Inoltre, la mancanza di tipizzazione statica e alcuni altri motivi non consentono di implementare un meccanismo di verifica della funzione Python al momento della compilazione.

Python ha una grande libreria standard, comunemente citata come uno dei maggiori punti di forza del linguaggio stesso .

Per le applicazioni rivolte a Internet, un gran numero di formati e protocolli standard (come come MIME e HTTP) sono supportati. Moduli per la creazione di interfacce utente grafiche, connessione a banche dati, generatori di numeri pseudocasuali, aritmetica con decimali di precisione arbitraria, manipolazione sono inoltre incluse espressioni regolari e test unitari. Tuttavia, poiché la maggior parte della libreria standard è multipiattaforma, ci sono solo alcuni moduli che devono essere modificati o riscritti completamente in alternative implementazioni. Una caratteristica principale di Python è proporre nuove importanti funzionalità,raccogliere input da parte della comunità su un problema e documentare le decisioni di progettazione. Il miglioramento del linguaggio si accompagna allo sviluppo di CPython. La mailing list python-dev è il forum principale di discussione sulla lingua sviluppo; problemi specifici sono discussi su python.org. Lo sviluppo avviene su un repository di codice sorgente auto-ospitato.Python garantisce un numero di alfa, beta e release candidateche sono rilasciate come anteprime e per i test prima che venga effettuata la versione finale.

Sebbene ci sia una versione sperimentale per ogni versione, questa viene spesso respintoa se il codice non è pronto. Il il team di sviluppo monitora lo stato del codice eseguendo la suite di test di unità di grandi dimensioni durante lo sviluppo e utilizzando il sistema di integrazione continua "BuildBot". Anche la community di sviluppatori Python ha contribuito oltre 72.000 moduli software (da gennaio 2016). Il nome di Python deriva dalla serie televisiva Monty Python's Flying Circus ed a oggi è il terzo linguaggio più popolare in cui la sintassi grammaticale non si basa principalmente su C.
Python è ampiamente usato in molti grandi aziende.

14.2 Prototyping - PyCharm

Per consentire l'utlizzo dell'hardware effettivo composto da CM3 e dalla filiera di sensori scelti, ed essere sicuri che il tutto venga svolta in piena compatibilità col prodotto finale è stato necessario utilizzare l'applicativo PyCharm.

PyCharm è un ambiente di sviluppo integrato (IDE) utilizzato nella programmazione , in particolare per il linguaggio Python. È sviluppato dalla società ceca JetBrains. Fornisce analisi del codice, un debugger grafico, un tester di unità integrato, integrazione con i sistemi di controllo versione (VCSes) e supporta lo sviluppo Web con Django e Data Science con Anaconda. [7]

PyCharm è multipiattaforma, con versioni Windows, macOS e Linux. La Community Edition è rilasciata con licenza Apache, è disponibile anche la Professional Edition con funzionalità extra, rilasciata con licenza proprietaria di cui possiamo usufruire tramite la licenza Supsi.

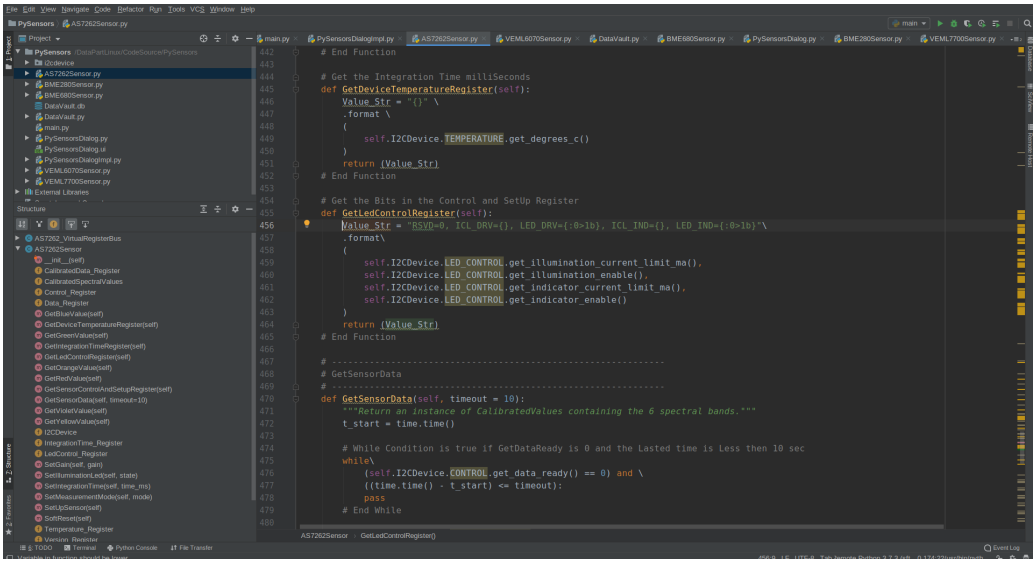


Figure 14.1: PyCharm 1

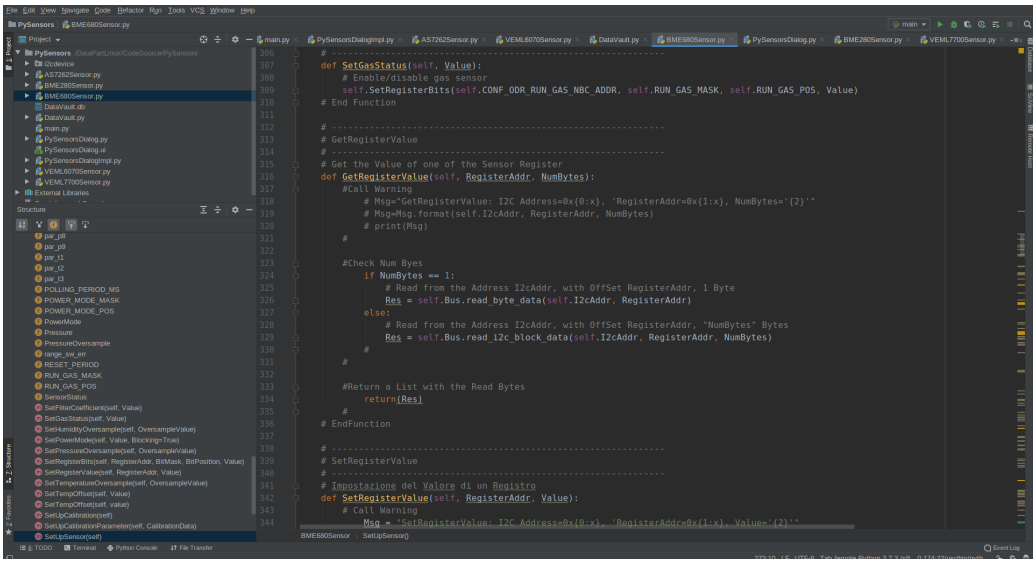


Figure 14.2: PyCharm 2

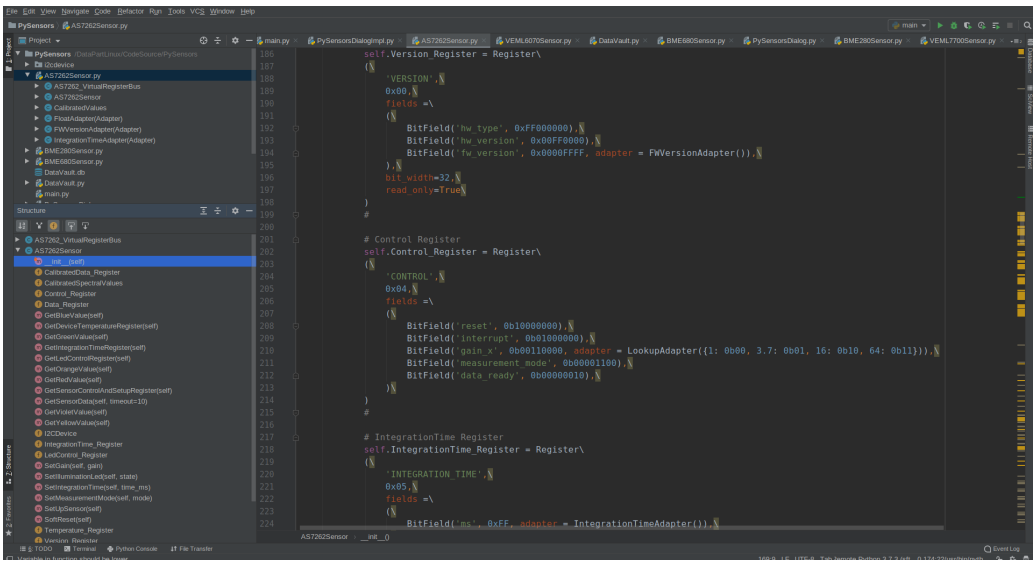


Figure 14.3: Py Charm 3

14.3 Design Software

Il paradigma di programmazione è stato intrapreso volendo implementare da prima le letture dei sensori collegati al bus e successivamente inserirli all'interno di un database *SQL* in modo che sia possibile effettuare una query degli stessi dati raccolti.

Per l'appunto quindi volta verificata la bontà della filiera è necessario essere in grado di visualizzare i dati a schermo tramite una interfaccia di visualizzazione remota.

Per poter rendere eseguibile e funzionante il codice per il sensore AS7262 è necessario importare le due dipendenze *adapter.py* e *_init_.py*

14.4 UML Diagram

Il software prima di essere sviluppato e creato attraverso l'analisi dei diagrammi UML stilati per mezzo del software *Visio*.

I due diagrammi rappresentano,in ordine, l'analisi strutturale *"Object Diagram"* con le istanze di tutte le classi che realizzano l'architettura della soluzione:

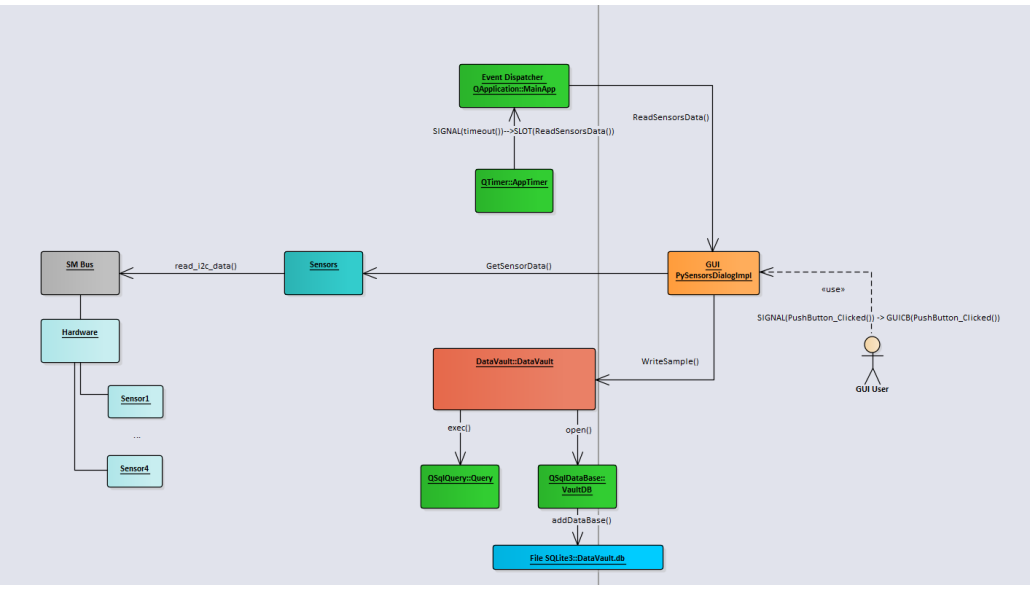


Figure 14.4: UML STRUCTURAL

Conseguentemente è presente sempre tramite diagramma (Sequence Diagram) che descrive la gestione degli eventi allo scattare del TimeOut che richiede una nuova lettura dei dati:

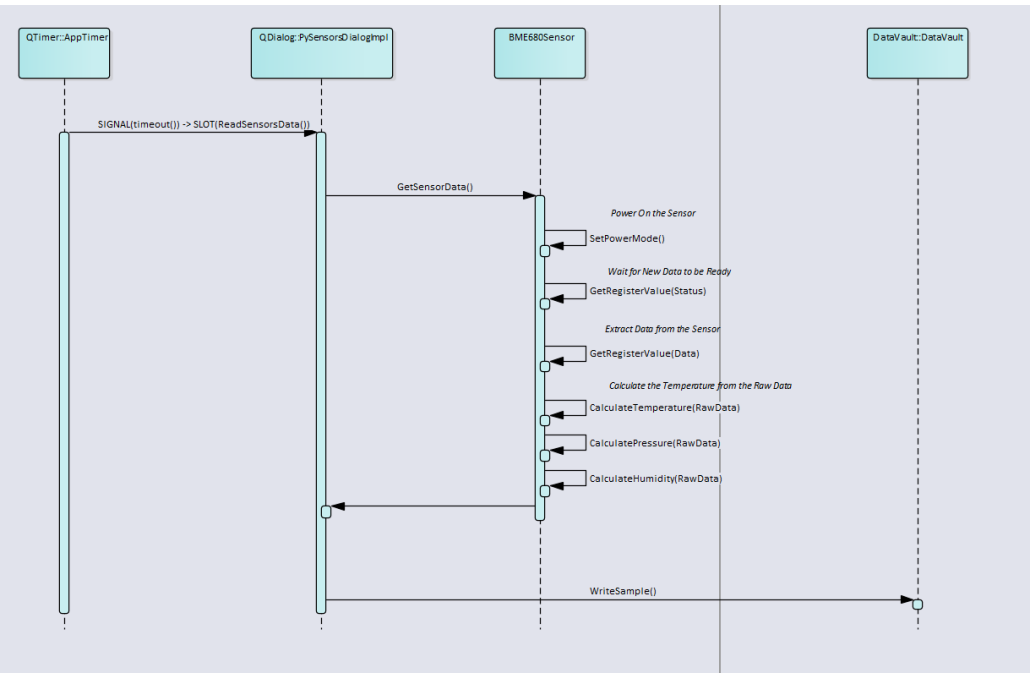


Figure 14.5: UML SEQUENCE

Per concludere , un esempio di UML con la rappresentazione delle Interfacce fornite dalle Classi e delle relazioni di "Call" fra le Funzioni:

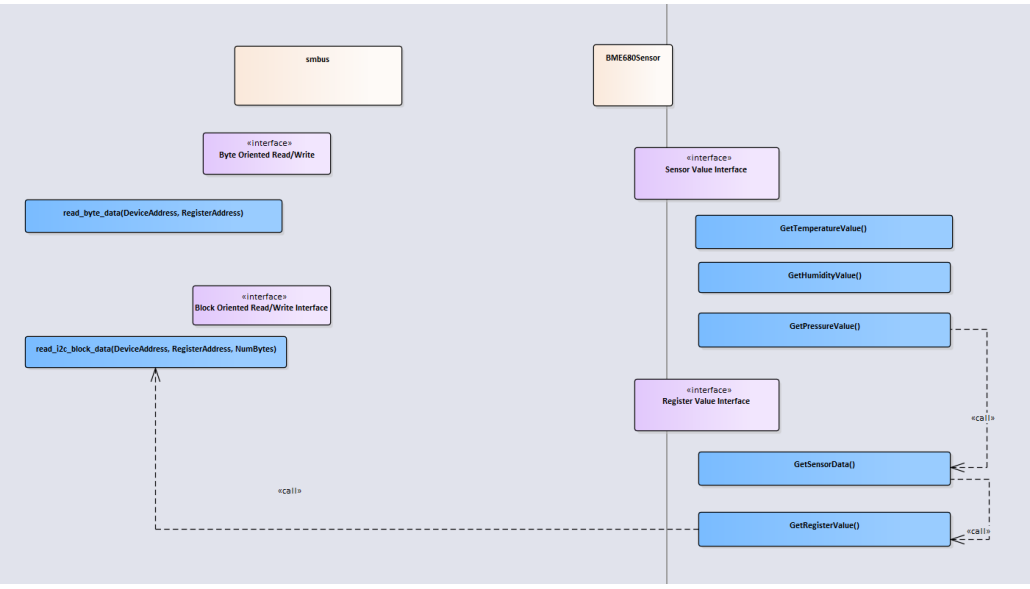


Figure 14.6: UML CALL

14.5 Main.py

```python
# coding=utf-8
import sys

# PyQt5 è un "Package" da cui si importano i "Moduli" QtCore e QtWidget
from PyQt5 import QtCore
from PyQt5 import QtWidgets

# Importazione dei Nome "QApplication" e "QWidget" dal Modulo QtWidgets
from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QWidget

# Importazione di Nomi di Classi "QMainWindow,..." dal Modulo QtWidgets contenuto nel Package PyQt5
# dopo aver importato i Nome dei Moduli, questi possono essere usati senza fare riferimento al Modulo in cui compaiono
# es, QMainWindow e non QtWidgets.QMainWindow
from PyQt5.QtWidgets import QMainWindow, QLabel, QGridLayout, QWidget

from PyQt5.QtCore import QSize

# Importazione del Modulo PySensorsDialogImpl con l'implementazione dell'Interfaccia Grafica
# il Modulo "PySensorsDialogImpl è definito nel File dal Nome "PySensorsDialogImpl.py" che viene cercato nei path
# elencati dalla Variabile Sys.Path
# fra i Path possibili esiste la cartella corrente
import PySensorsDialogImpl

# Importazione del Nome della Classe "PySensorsDialogImpl"
from PySensorsDialogImpl import PySensorsDialogImpl

# Uso della Variabile "__name__" per distinguere se il Modulo è stato eseguito od è stato invocato
# il valore della Variabile "__name__" è "__main__" nel caso di invocazioni del Modulo
if __name__ == "__main__":

    # Creazione dell'Istanza "app" della Classe "QApplication" definita nel Modulo "QtWidget" contenuto nel Package "PyQt5"
    # la "QApplication" "app" è l'EventDispatcher
        MainApp = QApplication(sys.argv)
    #

    # Creazione del Dialog PySensors
        NewDlg = PySensorsDialogImpl()
        NewDlg.show()
    #

    # avvio del Ciclo pricipale dell'Event Manager
        Res = MainApp.exec_()
    #

    # Uscita dall'Applicazione
        sys.exit(Res)
    #
#

# ToDoList:
    # aggiungere i CallBack sui pulsanti
#

```

14.6 BME680Sensor.py

```python
# Class Definition
# [INFO]: The Class has "object" as a Parent Class

import smbus
import time
import math


class BME680Sensor:

    # ------------------------------------------------------------
    # Init
    # ------------------------------------------------------------
    # Funzione di Inizializzazione della Classe
    def __init__(self):
        # [INFO]: the "Init" function is called each time a new istance of the Class is created

        #Call Warning
            print("BME680 Sensor Init: Called")
        #

        # Constant Variable Initialization
            # BME680 I2C addresses
            self.I2C_ADDR = 0x76
            #

            # Chip identifier Address
            self.CHIP_ID_ADDR = 0xD0
            #

            # Power mode settings
            self.SLEEP_POWER_MODE = 0
            self.FORCED_POWER_MODE = 1
            #

            # Reset Register Data
            # Soft reset register
            self.SOFT_RESET_ADDR = 0xe0
            # Reset command
            self.SOFT_RESET_CMD = 0xb6

            # Delay related macro declaration
            self.RESET_PERIOD = 10
            #

            # Polling Period
            self.POLLING_PERIOD_MS = 10

            # BME680 coefficients related defines
            self.COEFF_SIZE = 41
            self.COEFF_ADDR1_LEN = 25
            self.COEFF_ADDR2_LEN = 16

            # Coefficient's address
            self.COEFF_ADDR1 = 0x89
            self.COEFF_ADDR2 = 0xE1
            #

            # Field settings
            self.FIELD0_ADDR = 0x1D
```



```python
            self.FIELD_LENGTH = 15
            #

            # Sensor configuration registers
            self.CONF_ODR_RUN_GAS_NBC_ADDR = 0x71
            self.CONF_OVERSAMPLE_HUMIDITY_ADDR = 0x72
            self.CONF_TEMPERATURE_PRESSURE_POWER_MODE_ADDR = 0x74
            self.CONF_ODR_FILT_ADDR = 0x75

            # self.CONF_HEAT_CTRL_ADDR = 0x70

            # self.MEM_PAGE_ADDR = 0xf3
            #

            # Bit Mask Definitions
            self.OVERSAMPLE_TEMPERATURE_MASK = 0XE0
            self.OVERSAMPLE_PRESSURE_MASK = 0X1C
            self.OVERSAMPLE_HUMIDITY_MASK = 0X07

            self.FILTER_MASK = 0X1C

            self.POWER_MODE_MASK = 0x03
            self.RUN_GAS_MASK = 0x10

            self.NEW_DATA_MASK = 0x80

            self.BIT_H1_DATA_MASK = 0x0F

            # self.GAS_MEAS_MSK = 0x30
            # self.NBCONV_MSK = 0X0F

            # self.HCTRL_MSK = 0x08


            # self.RHRANGE_MSK = 0x30
            # self.RSERROR_MSK = 0xf0

            # self.GAS_INDEX_MSK = 0x0f
            # self.GAS_RANGE_MSK = 0x0f
            # self.GASM_VALID_MSK = 0x20
            # self.HEAT_STAB_MSK = 0x10
            # self.MEM_PAGE_MSK = 0x10
            # self.SPI_RD_MSK = 0x80
            # self.SPI_WR_MSK = 0x7f

            #

            # Bit position definitions for sensor settings
            self.OVERSAMPLE_TEMPERATURE_BIT_POSITION = 5
            self.OVERSAMPLE_PRESSURE_BIT_POSITION = 2
            self.OVERSAMPLE_HUMIDITY_BIT_POSITION = 0

            self.FILTER_POSITION = 2

            self.POWER_MODE_POS = 0

            self.RUN_GAS_POS = 4

            # self.GAS_MEAS_POS = 4
            # self.NBCONV_POS = 0
            #
```



```python

            # OverSampling possible Value
            self.OS_NONE = 0
            self.OS_1X = 1
            self.OS_2X = 2
            self.OS_4X = 3
            self.OS_8X = 4
            self.OS_16X = 5
            #

            #Filter Size
            self.FILTER_SIZE_0 = 0
            self.FILTER_SIZE_1 = 1
            self.FILTER_SIZE_3 = 2
            self.FILTER_SIZE_7 = 3
            self.FILTER_SIZE_15 = 4
            self.FILTER_SIZE_31 = 5
            self.FILTER_SIZE_63 = 6
            self.FILTER_SIZE_127 = 7
            #

            self.DISABLE_GAS_MEASURE = 0x00

            self.HUM_REG_SHIFT_VAL = 4

            # Array Index to Field data mapping for Calibration Data
            self.T2_LSB_REG = 1
            self.T2_MSB_REG = 2
            self.T3_REG = 3

            self.P1_LSB_REG = 5
            self.P1_MSB_REG = 6
            self.P2_LSB_REG = 7
            self.P2_MSB_REG = 8
            self.P3_REG = 9
            self.P4_LSB_REG = 11
            self.P4_MSB_REG = 12
            self.P5_LSB_REG = 13
            self.P5_MSB_REG = 14
            self.P7_REG = 15
            self.P6_REG = 16
            self.P8_LSB_REG = 19
            self.P8_MSB_REG = 20
            self.P9_LSB_REG = 21
            self.P9_MSB_REG = 22
            self.P10_REG = 23

            self.H2_MSB_REG = 25
            self.H2_LSB_REG = 26
            self.H1_LSB_REG = 26
            self.H1_MSB_REG = 27
            self.H3_REG = 28
            self.H4_REG = 29
            self.H5_REG = 30
            self.H6_REG = 31
            self.H7_REG = 32

            self.T1_LSB_REG = 33
            self.T1_MSB_REG = 34

            self.GH2_LSB_REG = 35
```



```python
            self.GH2_MSB_REG = 36
            self.GH1_REG = 37
            self.GH3_REG = 38
            #
        #

        # Instance Variable Definition
            # IC2 Bus Sensor Address
            self.I2cAddr = None

            # I2C Bus Instance
            self.Bus = None

            # Temperature oversampling
            self.TemperatureOversample = None

            # Pressure Oversampling
            self.PressureOversample = None

            # Humidity Oversampling
            self.HumidityOversample = None

            # Filter Size
            self.FilterCoefficient = None

            #PowerMode
            self.PowerMode = None

            # Data Read from the Sensor
            self.SensorStatus = None
            self.Temperature = None
            self.Pressure = None
            self.Humidity = None
            #

            # Calibration Variables

            self.par_t1 = None
            self.par_t2 = None
            self.par_t3 = None

            self.par_h1 = None
            self.par_h2 = None
            self.par_h3 = None
            self.par_h4 = None
            self.par_h5 = None
            self.par_h6 = None
            self.par_h7 = None

            self.par_gh1 = None
            self.par_gh2 = None
            self.par_gh3 = None

            self.par_p1 = None
            self.par_p2 = None
            self.par_p3 = None
            self.par_p4 = None
            self.par_p5 = None
            self.par_p6 = None
            self.par_p7 = None
            self.par_p8 = None
```

```python
            self.par p9 = None
            self.par p10 = None

            # Variable to store t fine size
            self.t fine = None

            # Variable to store heater resistance range
            # self.res heat range = None

            # Variable to store heater resistance value
            # self.res heat val = None

            # Variable to store error range
            self.range_sw_err = None

            # Temperature OffSet
            self.offset temp in t fine = None
        #
    # EndFunction

    # ------------------------------------------------------------
    # SetUpSensor
    # ------------------------------------------------------------
    # Write OverSampling Setting to Sensor Registry
    def SetUpSensor(self):
        # Instance Variable Initialization

        # I2C Bus Sensor Address
            self.I2cAddr = self.I2C ADDR
        #

        # I2C Bus
            self.Bus = smbus.SMBus(1)
        #

        # Reset and Put the Sensor in Sleep Mode
            self.SoftReset()
            self.SetPowerMode(self.SLEEP_POWER_MODE)
        #

        # Sensor Calibration
            self.SetUpCalibration()
        #

        # Sensor Oversampling and Filtering SetUp
            self.SetHumidityOversample(self.OS_2X)
            self.SetPressureOversample(self.OS_4X)
            self.SetTemperatureOversample(self.OS_8X)
            self.SetFilterCoefficient(self.FILTER_SIZE_3)
        #

        #Impostazione dell'OffSet della Temperatura
            self.SetTempOffset(0)
        #

        # Disable Gas Measure
            self.SetGasStatus(self.DISABLE_GAS_MEASURE)
        #
    # EndFunction

    # ------------------------------------------------------------
```



```python
    # SetGasStatus
    # ------------------------------------------------------------
    def SetGasStatus(self, Value):
        # Enable/disable gas sensor
        self.SetRegisterBits(self.CONF ODR RUN GAS NBC ADDR, self.RUN GAS MASK,
        self.RUN GAS POS, Value)
    # End Function

    # ------------------------------------------------------------
    # GetRegisterValue
    # ------------------------------------------------------------
    # Get the Value of one of the Sensor Register
    def GetRegisterValue(self, RegisterAddr, NumBytes):
        #Call Warning
            # Msg="GetRegisterValue: I2C Address=0x{0:x}, 'RegisterAddr=0x{1:x},
            NumBytes='{2}'"
            # Msg=Msg.format(self.I2cAddr, RegisterAddr, NumBytes)
            # print(Msg)
        #

        #Check Num Byes
            if NumBytes == 1:
                # Read from the Address I2cAddr, with OffSet RegisterAddr, 1 Byte
                Res = self.Bus.read byte data(self.I2cAddr, RegisterAddr)
            else:
                # Read from the Address I2cAddr, with OffSet RegisterAddr,
                "NumBytes" Bytes
                Res = self.Bus.read i2c block data(self.I2cAddr, RegisterAddr,
                NumBytes)
            #
        #

        #Return a List with the Read Bytes
            return(Res)
        #
    # EndFunction

    # ------------------------------------------------------------
    # SetRegisterValue
    # ------------------------------------------------------------
    # Impostazione del Valore di un Registro
    def SetRegisterValue(self, RegisterAddr, Value):
        # Call Warning
            Msg = "SetRegisterValue: I2C Address=0x{0:x}, 'RegisterAddr=0x{1:x},
            Value='{2}'"
            Msg = Msg.format(self.I2cAddr, RegisterAddr, Value)
            print(Msg)
        #

        # Set one or more registers
            if isinstance(Value, int):
                self.Bus.write_byte_data(self.I2cAddr, RegisterAddr, Value)
            else:
                self.Bus.write_i2c_block_data(self.I2cAddr, RegisterAddr, Value)
            #
        #
    # EndFunction

    # ------------------------------------------------------------
    # SetRegisterBits
    # ------------------------------------------------------------
```



```python
    # Impostazione del Valore di un Bit di un Registro
    def SetRegisterBits(self, RegisterAddr, BitMask, BitPosition, Value):
        Temp = self.GetRegisterValue(RegisterAddr, 1)
        Temp &= ~BitMask
        Temp |= Value << BitPosition
        self.SetRegisterValue(RegisterAddr, Temp)
    # EndFunction

    # ------------------------------------------------------------
    # GetChipID
    # ------------------------------------------------------------
    # Read the ChipID
    def GetChipID(self):

        Res = self.GetRegisterValue(self.CHIP ID ADDR, 1)

        return(Res)
    # EndFunction

    # ------------------------------------------------------------
    # SetTemperatureOversample
    # ------------------------------------------------------------
    # Set the Temperature Oversample Value
    def SetTemperatureOversample(self, OversampleValue):
        """
        Set Temperature Oversampling.

        A higher oversampling value means more stable sensor readings,
        with less noise and jitter.

        However each step of oversampling adds about 2ms to the latency,
        causing a slower response time to fast transients.

        :param value: Oversampling value, one of: OS_NONE, OS_1X, OS_2X, OS_4X,
        OS_8X, OS_16X

        """
        # Impostazione del Valore di OverSampling
        self.TemperatureOversample = OversampleValue

        # Impostazione del Valore nel Registro di Configurazione del Sensore
        self.SetRegisterBits(self.CONF_TEMPERATURE_PRESSURE_POWER_MODE_ADDR,
        self.OVERSAMPLE_TEMPERATURE_MASK,
        self.OVERSAMPLE_TEMPERATURE_BIT_POSITION, OversampleValue)
    # EndFunction

    # ------------------------------------------------------------
    # SetPressureOversample
    # ------------------------------------------------------------
    # Set the Pressure Oversample Value
    def SetPressureOversample(self, OversampleValue):

        self.PressureOversample = OversampleValue

        # Impostazione del Valore nel Registro di Configurazione del Sensore
        self.SetRegisterBits(self.CONF_TEMPERATURE_PRESSURE_POWER_MODE_ADDR,
        self.OVERSAMPLE_PRESSURE_MASK, self.OVERSAMPLE_PRESSURE_BIT_POSITION,
        OversampleValue)
    # EndFunction

    # ------------------------------------------------------------
```



```python
    # SetHumidityOversample
    # ------------------------------------------------------------
    # Set the Humifity Oversample Value
    def SetHumidityOversample(self, OversampleValue):
        self.HumidityOversample = OversampleValue

        # Impostazione del Valore nel Registro di Configurazione del Sensore
        self.SetRegisterBits\
        (\
            self.CONF OVERSAMPLE HUMIDITY ADDR,\
            self.OVERSAMPLE HUMIDITY MASK,\
            self.OVERSAMPLE HUMIDITY BIT POSITION, OversampleValue\
        )
    # EndFunction

    # ------------------------------------------------------------
    # SetFilter
    # ------------------------------------------------------------
    # Set the Filter Size
    def SetFilterCoefficient(self, Value):
        """
        Set IIR filter size.

        Optionally remove short term fluctuations from the temperature and
        pressure readings,
        increasing their resolution but reducing their bandwidth.

        Enabling the IIR filter does not slow down the time a reading takes, but
        will slow
        down the BME680s response to changes in temperature and pressure.

        When the IIR filter is enabled, the temperature and pressure resolution is
        effectively 20bit.
        When it is disabled, it is 16bit + oversampling-1 bits.

        """

        self.FilterCoefficient = Value
        self.SetRegisterBits(self.CONF_ODR_FILT_ADDR, self.FILTER_MASK,
        self.FILTER_POSITION, Value)
    # EndFunction

    # ------------------------------------------------------------
    # GetPowerMode
    # ------------------------------------------------------------
    # Get the Sensor Power Mode
    def GetPowerMode(self):
        #Get power mode
        self.PowerMode =
        self.GetRegisterValue(self.CONF_TEMPERATURE_PRESSURE_POWER_MODE_ADDR, 1)
        return(self.PowerMode)
    #

    # ------------------------------------------------------------
    # SetPowerMode
    # ------------------------------------------------------------
    # Set the Sensor Power Mode
    def SetPowerMode(self, Value, Blocking=True):
        # Set power mode

        # self.PowerMode = Value
```

```python

        self.SetRegisterBits\
        ( \
            self.CONF TEMPERATURE PRESSURE POWER MODE ADDR,\
            self.POWER MODE MASK,\
            self.POWER MODE POS,\
            Value\
        )

        # Wait for PowerMode reached
        StatusReached = False

        while Blocking == True and StatusReached == False:
            CurrPowerMode = self.GetPowerMode()

            Msg="SetPowerMode: Current Power Mode=0x{0:x}, Needed PowerMode=0x{1:x}".format(CurrPowerMode,self.PowerMode)
            print(Msg)

            # if  CurrPowerMode == Value:
            if CurrPowerMode != self.PowerMode:
                StatusReached = False
                Msg = "SetPowerMode: Status Reached False"
                print(Msg)
            else:
                StatusReached = True
                Msg = "SetPowerMode: Status Reached True"
                print(Msg)
            #

            # Wait for next Polling operation
            time.sleep(self.POLLING PERIOD MS / 1000.0)
        #
    #

    # ------------------------------------------------------------
    # GetSensorData
    # ------------------------------------------------------------
    # Read the Data from the sensor
    def GetSensorData(self):
        """
        Get sensor data.
        Stores data in .data and returns True upon success.
        """

        # Put the Sensor in the Forced PowerMode
        self.SetPowerMode(self.FORCED_POWER_MODE)

        for attempt in range(10):
            #1 Byte all'Indirizzo "FIELD0_ADDR" (il registro contiene il Byte "NewData")
            status = self.GetRegisterValue(self.FIELD0_ADDR, 1)

            # Wait for new Data to be ready
            if (status & self.NEW_DATA_MASK) == 0:
                time.sleep(self.POLLING_PERIOD_MS / 1000.0)
                continue
            # EndIf

            #Lettura di 15 Byte a partire dall'indirizzo "FIELD0_ADDR"
            #Regs contiene una Lista di 15 Byte
```



```python
            regs = self.GetRegisterValue(self.FIELD0 ADDR, self.FIELD LENGTH)

            self.SensorStatus = regs[0] & self.NEW_DATA_MASK

            # Contains the nb profile used to obtain the current measurement
            # self.data.gas index = regs[0] & constants.GAS INDEX MSK
            # self.data.meas_index = regs[1]

            # Raw Temperature Value (Analog Digital Conversion)
            ADC_Temp = (regs[5] << 12) | (regs[6] << 4) | (regs[7] >> 4)

            ADC Pres = (regs[2] << 12) | (regs[3] << 4) | (regs[4] >> 4)
            ADC Hum = (regs[8] << 8) | regs[9]

            # adc gas res = (regs[13] << 2) | (regs[14] >> 6)
            # gas range = regs[14] & constants.GAS RANGE MSK

            # self.data.status |= regs[14] & constants.GASM VALID MSK
            # self.data.status |= regs[14] & constants.HEAT STAB MSK

            # self.data.heat stable = (self.data.status & constants.HEAT STAB MSK) > 0

            # Calcolo della Temperatura dai Valori RAW
            temperature = self.CalculateTemperature(ADC_Temp)

            # Normalizzazione del Valore Letto
            self.Temperature = temperature / 100.0

            # self.ambient temperature = temperature  # Saved for heater calc

            self.Pressure = self.CalculatePressure(ADC Pres) / 100.0
            self.Humidity = self.CalculateHumidity(ADC_Hum) / 1000.0
            # self.data.gas_resistance = self._calc_gas_resistance(adc_gas_res, gas_range)

            return True
        # EndFor

        # after 10 failed retry give up
        return False
    # End Function

    # ------------------------------------------------------------
    # Calculate Temperature
    # ------------------------------------------------------------
    def CalculateTemperature(self, temperature_adc):
        # Convert the raw temperature to degrees C using calibration_data

        var1 = (temperature_adc >> 3) - (self.par_t1 << 1)
        var2 = (var1 * self.par_t2) >> 11
        var3 = ((var1 >> 1) * (var1 >> 1)) >> 12
        var3 = ((var3) * (self.par_t3 << 4)) >> 14

        # Save temperature data for pressure calculations
        self.t_fine = (var2 + var3) + self.offset_temp_in_t_fine
        calc_temp = (((self.t_fine * 5) + 128) >> 8)

        return calc_temp
    # End Function

```



```python
    def CalculatePressure(self, pressure adc):
        """Convert the raw pressure using calibration data."""
        var1 = ((self.t fine) >> 1) - 64000
        var2 = ((((var1 >> 2) * (var1 >> 2)) >> 11) * self.par p6) >> 2
        var2 = var2 + ((var1 * self.par p5) << 1)
        var2 = (var2 >> 2) + (self.par p4 << 16)
        var1 = (((((var1 >> 2) * (var1 >> 2)) >> 13) * ((self.par p3 << 5)) >> 3) + ((self.par p2 * var1) >> 1))
        var1 = var1 >> 18
        var1 = ((32768 + var1) * self.par p1) >> 15
        calc pressure = 1048576 - pressure adc
        calc pressure = ((calc pressure - (var2 >> 12)) * (3125))

        if calc pressure >= (1 << 31):
            calc pressure = ((calc pressure // var1) << 1)
        else:
            calc pressure = ((calc pressure << 1) // var1)
        #

        var1 = (self.par p9 * (((calc pressure >> 3) * (calc pressure >> 3)) >> 13)) >> 12
        var2 = ((calc pressure >> 2) * self.par p8) >> 13
        var3 = ((calc_pressure >> 8) * (calc_pressure >> 8) * (calc_pressure >> 8) * self.par_p10) >> 17

        calc_pressure = (calc_pressure) + ((var1 + var2 + var3 + (self.par_p7 << 7)) >> 4)

        return calc_pressure
    # End Function

    def CalculateHumidity(self, humidity_adc):
        """Convert the raw humidity using calibration data."""
        temp_scaled = ((self.t_fine * 5) + 128) >> 8
        var1 = (humidity_adc - ((self.par_h1 * 16))) - (((temp_scaled * self.par_h3) // (100)) >> 1)
        var2 = (self.par_h2 * (((temp_scaled * self.par_h4) // (100)) + (((temp_scaled * ((temp_scaled * self.par_h5) // (100))) >> 6) // (100)) + (1 * 16384))) >> 10
        var3 = var1 * var2
        var4 = self.par_h6 << 7
        var4 = ((var4) + ((temp_scaled * self.par_h7) // (100))) >> 4
        var5 = ((var3 >> 14) * (var3 >> 14)) >> 10
        var6 = (var4 * var5) >> 1
        calc_hum = (((var3 + var6) >> 10) * (1000)) >> 12

        return min(max(calc_hum, 0), 100000)
    # End Function

    # ------------------------------------------------------------
    # Set Temperature OffSet
    # ------------------------------------------------------------
    def SetTempOffset(self, Value):
        """
        Set temperature offset in celsius.

        If set, the temperature t_fine will be increased by given value in celsius.
        :param value: Temperature offset in Celsius, eg. 4, -8, 1.25

        """
```



```python
        if Value == 0:
            self.offset temp in t fine = 0
        else:
            self.offset temp in t fine = int(math.copysign((((int(abs(Value) * 100)) << 8) - 128) / 5, Value))
        #
    #

    # ------------------------------------------------------------
    # Get Temperature Value
    # ------------------------------------------------------------
    def GetTemperatureValue(self):
        return(self.Temperature)
    # EndFunction

    # ------------------------------------------------------------
    # Get Pressure Value
    # ------------------------------------------------------------
    def GetPressureValue(self):
        return (self.Pressure)
    # EndFunction

    # ------------------------------------------------------------
    # Get Humidity Value
    # ------------------------------------------------------------
    def GetHumidityValue(self):
        return (self.Humidity)
    # EndFunction

    # ------------------------------------------------------------
    # Set Up Calibration Data
    # ------------------------------------------------------------
    def SetUpCalibration(self):
        # Retrieve the sensor calibration data and store it in Local Calibration Data
        CalibrationData = self.GetRegisterValue(self.COEFF_ADDR1, self.COEFF_ADDR1_LEN)

        CalibrationData += self.GetRegisterValue(self.COEFF_ADDR2, self.COEFF_ADDR2_LEN)

        # heat_range = self._get_regs(constants.ADDR_RES_HEAT_RANGE_ADDR, 1)
        # heat_value = constants.twos_comp(self._get_regs(constants.ADDR_RES_HEAT_VAL_ADDR, 1), bits=8)
        # sw_error = constants.twos_comp(self._get_regs(constants.ADDR_RANGE_SW_ERR_ADDR, 1), bits=8)

        self.SetUpCalibrationParameter(CalibrationData)
        # self.calibration_data.set_other(heat_range, heat_value, sw_error)
    # End Function

    # ------------------------------------------------------------
    # Set Up Calibration Data
    # ------------------------------------------------------------
    def SetUpCalibrationParameter(self, CalibrationData):
        # Set paramaters from an array of bytes."""
        # Temperature related coefficients
        self.par_t1 = self.BytesToWord(CalibrationData[self.T1_MSB_REG], CalibrationData[self.T1_LSB_REG])
        self.par_t2 = self.BytesToWord(CalibrationData[self.T2_MSB_REG],
```

```python
            CalibrationData[self.T2 LSB REG], bits=16, signed=True)
            self.par t3 = self.TwosComp(CalibrationData[self.T3 REG], bits=8)

            # Pressure related coefficients
            self.par p1 = self.BytesToWord(CalibrationData[self.P1 MSB REG],
            CalibrationData[self.P1 LSB REG])
            self.par p2 = self.BytesToWord(CalibrationData[self.P2 MSB REG],
            CalibrationData[self.P2 LSB REG], bits=16, signed=True)
            self.par p3 = self.TwosComp(CalibrationData[self.P3 REG], bits=8)
            self.par p4 = self.BytesToWord(CalibrationData[self.P4 MSB REG],
            CalibrationData[self.P4 LSB REG], bits=16, signed=True)
            self.par p5 = self.BytesToWord(CalibrationData[self.P5 MSB REG],
            CalibrationData[self.P5 LSB REG], bits=16, signed=True)
            self.par p6 = self.TwosComp(CalibrationData[self.P6 REG], bits=8)
            self.par p7 = self.TwosComp(CalibrationData[self.P7 REG], bits=8)
            self.par p8 = self.BytesToWord(CalibrationData[self.P8 MSB REG],
            CalibrationData[self.P8 LSB REG], bits=16, signed=True)
            self.par p9 = self.BytesToWord(CalibrationData[self.P9 MSB REG],
            CalibrationData[self.P9 LSB REG], bits=16, signed=True)
            self.par p10 = CalibrationData[self.P10 REG]

            # Humidity related coefficients
            self.par h1 = (CalibrationData[self.H1 MSB REG] << self.HUM REG SHIFT VAL)
            | (CalibrationData[self.H1 LSB REG] & self.BIT H1 DATA MASK)
            self.par h2 = (CalibrationData[self.H2 MSB REG] << self.HUM REG SHIFT VAL)
            | (CalibrationData[self.H2 LSB REG] >> self.HUM REG SHIFT VAL)
            self.par h3 = self.TwosComp(CalibrationData[self.H3 REG], bits=8)
            self.par h4 = self.TwosComp(CalibrationData[self.H4 REG], bits=8)
            self.par h5 = self.TwosComp(CalibrationData[self.H5 REG], bits=8)
            self.par h6 = self.TwosComp(CalibrationData[self.H6 REG], bits=8)
            self.par h7 = self.TwosComp(CalibrationData[self.H7 REG], bits=8)

            # Gas heater related coefficients
            #self.par_gh1 = twos_comp(calibration[GH1_REG], bits=8)
            #self.par_gh2 = bytes_to_word(calibration[GH2_MSB_REG],
            calibration[GH2_LSB_REG], bits=16, signed=True)
            #self.par_gh3 = twos_comp(calibration[GH3_REG], bits=8)
        # End Function

        # ------------------------------------------------------------
        # Definizione dell'OffSet della Temperatura
        # ------------------------------------------------------------
        def SetTempOffset(self, value):
            """Set temperature offset in celsius.

            If set, the temperature t_fine will be increased by given value in celsius.
            :param value: Temperature offset in Celsius, eg. 4, -8, 1.25

            """
            if value == 0:
                self.offset_temp_in_t_fine = 0
            else:
                self.offset_temp_in_t_fine = int(math.copysign((((int(abs(value) *
                100)) << 8) - 128) / 5, value))
            #
        #

        # ------------------------------------------------------------
        # Valore del Registro "Operation Mode"
        # ------------------------------------------------------------
        def GetOperationModeRegisterValue(self):
```



```python
            HxValue =
            self.GetRegisterValue(self.CONF TEMPERATURE PRESSURE POWER MODE ADDR, 1)
            if HxValue == self.SLEEP POWER MODE:
                HrValue = "SLEEP MODE"
            elif HxValue == self.FORCED POWER MODE:
                HrValue = "FORCE MODE"
            else:
                HrValue = "NOT AVAILABLE"
            #

            Res = (HxValue, HrValue)
            return(Res)
        # End

        # ------------------------------------------------------------
        # GetFilterCoefficient
        # ------------------------------------------------------------
        # estrazione del Valore del Coefficiente del Filtro
        def GetFilterCoefficient(self):
            # Get Filter Coefficent (1 Byte)
            ResByte = self.GetRegisterValue(self.CONF ODR FILT ADDR, 1)

            # Extract the Coefficient Bit
            ResBits = ResByte & self.FILTER MASK

            # Shift the Result
            Res = ResBits >> self.FILTER POSITION
            return(Res)
        #

        # ------------------------------------------------------------
        # SoftReset
        # ------------------------------------------------------------
        # Do a Sensor Soft Reset
        def SoftReset(self):
            """Trigger a soft reset."""
            self.SetRegisterValue(self.SOFT_RESET_ADDR, self.SOFT_RESET_CMD)
            time.sleep(self.RESET_PERIOD / 1000.0)
        #

        # ------------------------------------------------------------
        # BytesToWord
        # ------------------------------------------------------------
        def BytesToWord(self, msb, lsb, bits=16, signed=False):
            # Convert a most and least significant byte into a word."""
            word = (msb << 8) | lsb
            if signed:
                word = self.TwosComp(word, bits)
            return word
        # End Function

        # ------------------------------------------------------------
        # TwosComp
        # ------------------------------------------------------------
        def TwosComp(self, val, bits=16):
            """Convert two bytes into a two's compliment signed word."""
            # TODO: Reimpliment with struct
            if val & (1 << (bits - 1)) != 0:
                val = val - (1 << bits)
            return val
        # End Function
```



```python
    # EndClass
```

14.7 VEML6075Sensor.py

```python
# Class Definition
# [INFO]: The Class has "object" as a Parent Class

import smbus
import time
import math

class VEML6070Sensor:

    # ------------------------------------------------------------
    # Init
    # ------------------------------------------------------------
    # Funzione di Inizializzazione della Classe
    def   init  (self):

        # Constant Variable Initialization
            self.I2C BUS = 1

            # 7bit address of the VEML6070 (write, read)
            self.ADDR L = 0x38

            # 7bit address of the VEML6070 (read)
            self.ADDR H = 0x39

            self.RSET 240K = 240000
            self.RSET 270K = 270000
            self.RSET 300K = 300000
            self.RSET 600K = 600000

            self.SHUTDOWN DISABLE = 0x00
            self.SHUTDOWN ENABLE = 0x01

            self.INTEGRATIONTIME_1_2T = 0x00
            self.INTEGRATIONTIME_1T = 0x01
            self.INTEGRATIONTIME_2T = 0x02
            self.INTEGRATIONTIME_4T = 0x03
        #

        # Instance Variable Definition
            self.I2CBus = None
            self.SensorAddress = None
            self.RSet = None
            self.ShutDown = None
            self.IntegrationTime = None
        #

        # Instance Variable Initialization
            self.I2CBus = smbus.SMBus(self.I2C_BUS)
            self.SensorAddress = self.ADDR_L

            # Value for the RSet = 270K
            self.RSet = self.RSET_270K
            # before set_integration_time()
            self.ShutDown = self.SHUTDOWN_DISABLE
        #
    # End Function

    # ------------------------------------------------------------
    # SetUpSensor
    # ------------------------------------------------------------
    # Write OverSampling Setting to Sensor Registry
```



```python
    def SetUpSensor(self):
        # Value for the Integration Time: 1T
        self.SetIntegrationTime(self.INTEGRATIONTIME 1T)

        # ShutDown Sensor
        self.Disable()
    # End Function

    # ------------------------------------------------------------
    # Set Integration Time
    # ------------------------------------------------------------
    # Set the Value of the Integration Time in the Command Register
    def SetIntegrationTime(self, IntTime):
        # Set the Value of the Integration Time
        self.IntegrationTime = IntTime

        # Write the Value of the resulting command register
        self.I2CBus.write byte(self.SensorAddress, self.GetCommandRegisterByte())

        # constant offset determined experimentally to allow sensor to readjust
        time.sleep(0.2)
    # End Function

    # ------------------------------------------------------------
    # Get Integration Time
    # ------------------------------------------------------------
    # Return the Value of the Integration Time
    def GetIntegrationTime(self):
        return(self.IntegrationTime)
    # End Function

    # ------------------------------------------------------------
    # Enable
    # ------------------------------------------------------------
    # Enable the Sensor
    def Enable(self):
        self.Shutdown = self.SHUTDOWN_DISABLE
        self.I2CBus.write_byte(self.SensorAddress, self.GetCommandRegisterByte())
    # End Function

    # ------------------------------------------------------------
    # Disable
    # ------------------------------------------------------------
    # Disable the Sensor
    def Disable(self):
        self.Shutdown = self.SHUTDOWN_ENABLE
        self.I2CBus.write_byte(self.SensorAddress, self.GetCommandRegisterByte())
    # End Function

    # ------------------------------------------------------------
    # Get UVA Light Intensity Raw Value
    # ------------------------------------------------------------
    # Disable the Sensor
    def GetUvaLightIntensityRawValue(self):
        # Enable the Sensor
        self.Enable()

        # wait two times the refresh time to allow completion of a previous cycle
        with old settings (worst case)
        time.sleep(self.get_refresh_time() * 2)
```



```python
        # Read the Value from the Bus
        Msb = self.I2CBus.read byte(self.SensorAddress + (self.ADDR H - self.ADDR L))
        Lsb = self.I2CBus.read byte(self.SensorAddress)
        #

        # ShutDown the Sensor
        self.Disable()

        # Form the resulting byte
        return (Msb << 8) | Lsb
    # End Function

    # ------------------------------------------------------------
    # Get UVA Light Intensity Value
    # ------------------------------------------------------------
    # Get the Uva Light Intensity Value
    def GetUvaLightIntensityValue(self):
        UvValue = self.GetUvaLightIntensityRawValue()

        # Determine the Result by Sensitivity Product
        Result = UvValue * self.GetUvaLightSensitivity()

        return (Result)
    # End Function

    # ------------------------------------------------------------
    # GetCommandByte
    # ------------------------------------------------------------
    # Calculate the Bit of the Command Byte based on the Current Configuration
    def GetCommandRegisterByte(self):
        """
        assembles the command byte for the current state
        """
        # ShutDown
        CommandRegisterByte = (self.Shutdown & 0x01) << 0

        # Integration Time
        CommandRegisterByte = (self.IntegrationTime & 0x03) << 2  # IT

        # Reserved Bits
        CommandRegisterByte = ((CommandRegisterByte | 0x02) & 0x3F)
        return(CommandRegisterByte)
    # End Function

    # ------------------------------------------------------------
    # Get Refresh Time
    # ------------------------------------------------------------
    # Determine the Refresh time
    def GetRefreshTime(self):
        """
        returns time needed to perform a complete measurement using current
        settings (in s)
        """

        case_refresh_rset = {
            self.RSET_240K: 0.1,
            self.RSET_270K: 0.1125,
            self.RSET_300K: 0.125,
            self.RSET_600K: 0.25
        }
```



```python
        case refresh it = {
            self.INTEGRATIONTIME 1 2T: 0.5,
            self.INTEGRATIONTIME 1T: 1,
            self.INTEGRATIONTIME 2T: 2,
            self.INTEGRATIONTIME 4T: 4
        }

        # Calculate the Refresh Time
        Result = case refresh rset[self.RSet] * case refresh it[self.IntegrationTime]

        return(Result)

    # End Function

    # ------------------------------------------------------------
    # Get Uva Light Sensitivity
    # ------------------------------------------------------------
    def GetUvaLightSensitivity(self):
        """
        returns UVA light sensitivity in W/(m*m)/step
        """
        case sens rset = {
            self.RSET_240K: 0.05,
            self.RSET_270K: 0.05625,
            self.RSET_300K: 0.0625,
            self.RSET_600K: 0.125
        }

        case_sens_it = {
            self.INTEGRATIONTIME_1_2T: 0.5,
            self.INTEGRATIONTIME_1T: 1,
            self.INTEGRATIONTIME_2T: 2,
            self.INTEGRATIONTIME_4T: 4
        }

        Result = case_sens_rset[self.RSet] / case_sens_it[self.IntegrationTime]

        return(Result)
    # End Function
# End Class
```

14.8 VEML7700Sensor.py

```python
# Class Definition
# [INFO]: The Class has "object" as a Parent Class

import smbus
import time
import math


class VEML7700Sensor:

    # ------------------------------------------------------------
    # Init
    # ------------------------------------------------------------
    # Funzione di Inizializzazione della Classe
    def __init__(self):
        # Call Warning
            print("VEML 7700 Sensor Init: Called")
        #

        # Sensor Address
            self.I2C_ADDR = 0x10
        #

        # Write registers
            #ALS: Ambient Light Sensor
            self.Als_Conf_0 = 0x00
            self.Als_WH = 0x01
            self.Als_WL = 0x02
            self.Power_Save = 0x03
        #

        # Read registers
            self.Als = 0x04
            self.White = 0x05
            self.Interrupt = 0x06
        #

        # Instance Variable Definition
            # IC2 Bus Sensor Address
            self.I2cAddr = None

            # I2C Bus Instance
            self.Bus = None

            self.ConfValues = None

            self.Interrupt_High = None

            self.Interrupt_Low = None

            self.Power_Save_Mode = None

            # Illuminance Value
            self.IlluminanceValue = None
        #
    # End Function

    # ------------------------------------------------------------
    # Bytes To String
    # ------------------------------------------------------------
    # Convert a List of Value [LSB, MSB] to a String
```



```python
    def BytesToString(self, Value):
        Value_Str = ""
        if isinstance(Value, list):
            iNum = len(Value)
            print("SetRegisterValue: Num. Elements in the List:{0}".format(iNum))
            for i in range(iNum):
                CurrElement = "{0:0>8b}".format(Value[iNum - 1 - i])
                print("SetRegisterValue: Element {0} in the List is :{1}".format(iNum - 1 - i, CurrElement))
                Value_Str = Value_Str + " " + CurrElement
        # End For
        else:
            Value_Str = "{0:0>8b}".format(Value)
        # End If

        return(Value_Str)
    #

    # ------------------------------------------------------------
    # SetRegisterValue
    # ------------------------------------------------------------
    # Set the Value of of of the 6 Register
    def SetRegisterValue(self, RegisterAddr, Value):
        # Call Warning
            Value_Str = self.BytesToString(Value)
            Msg = "SetRegisterValue: I2C Address=0x{0:x}, RegisterAddr=0x{1:x}, Value={2}"
            Msg = Msg.format(self.I2cAddr, RegisterAddr, Value_Str)
            print(Msg)
        #

        # Set one or more registers
            if isinstance(Value, int):
                self.Bus.write_byte_data(self.I2cAddr, RegisterAddr, Value)
            else:
                self.Bus.write_i2c_block_data(self.I2cAddr, RegisterAddr, Value)
            # End If
        #
    # EndFunction

    # ------------------------------------------------------------
    # GetRegisterValue
    # ------------------------------------------------------------
    # Get the Value of one of the Sensor Register
    def GetRegisterValue(self, RegisterAddr, NumBytes):
        # Call Warning
        # Msg="GetRegisterValue: I2C Address=0x{0:x}, 'RegisterAddr=0x{1:x}, NumBytes='{2}'"
        # Msg=Msg.format(self.I2cAddr, RegisterAddr, NumBytes)
        # print(Msg)
        #

        # Check Num Byes
            if NumBytes == 1:
                # Read from the Address I2cAddr, with OffSet RegisterAddr, 1 Byte
                Res = self.Bus.read_byte_data(self.I2cAddr, RegisterAddr)
            else:
                # Read from the Address I2cAddr, with OffSet RegisterAddr, "NumBytes" Bytes
                Res = self.Bus.read_i2c_block_data(self.I2cAddr, RegisterAddr, NumBytes)
```



```python
            #
        #

        # Return a List with the Read Bytes
            return (Res)
        #
    # EndFunction

    # ------------------------------------------------------------
    # GetAlsConf0RegisterValue
    # ------------------------------------------------------------
    # Extract 2 Byte from the "ALS CONF 0" Register. Return a List of 2 Byte
    def GetAlsConf0RegisterValue(self):
        Res = self.GetRegisterValue(self.Als_Conf_0, 2)
        return(Res)
    # End Function

    # ------------------------------------------------------------
    # GetHighThresholdWindowsRegisterValue
    # ------------------------------------------------------------
    # Extract 2 Byte from the "ALS CONF 0" Register. Return a List of 2 Byte
    def GetHighThresholdWindowsRegisterValue(self):
        Res = self.GetRegisterValue(self.Als_WH, 2)
        return (Res)
    # End Function

    # ------------------------------------------------------------
    # GetLowThresholdWindowsRegisterValue
    # ------------------------------------------------------------
    # Extract 2 Byte from the "" Register. Return a List of 2 Byte
    def GetLowThresholdWindowsRegisterValue(self):
        Res = self.GetRegisterValue(self.Als_WL, 2)
        return (Res)
    # End Function

    # ------------------------------------------------------------
    # GetLowThresholdWindowsRegisterValue
    # ------------------------------------------------------------
    # Extract 2 Byte from the "" Register. Return a List of 2 Byte
    def GetPowerSavingModesRegisterValue(self):
        Res = self.GetRegisterValue(self.Power_Save, 2)
        return (Res)
    # End Function

    # ------------------------------------------------------------
    # GetAlsHighResolutionOutputRegisterValue
    # ------------------------------------------------------------
    # Extract 2 Byte from the "ALS" Register. Return a List of 2 Byte
    def GetAlsHighResolutionOutputRegisterValue(self):
        Res = self.GetRegisterValue(self.Als, 2)
        return (Res)
    # End Function

    # ------------------------------------------------------------
    # GetWhiteRegisterValue
    # ------------------------------------------------------------
    # Extract 2 Byte from the "WHITE" Register. Return a List of 2 Byte
    def GetWhiteChannelOutputRegisterValue(self):
        Res = self.GetRegisterValue(self.White, 2)
        return (Res)
    # End Function
```



```python
    # ------------------------------------------------------------
    # SetUpSensor
    # ------------------------------------------------------------
    # SetUp VEML Sensor
    def SetUpSensor(self):
        # Instance Variable Initialization

        # I2C Bus Sensor Address
            self.I2cAddr = self.I2C_ADDR
        #

        # I2C Bus
            self.Bus = smbus.SMBus(1)
        #

        # Value for the Configuration Register
            # These settings will provide the max range for the sensor (0-120 K Lux)
            # but at the lowest precision:
            # LSB   MSB

            # 1/8 gain, 25ms IT (Integration Time)
            # Reference data sheet Table 1 for configuration settings
            # Configuration List: LSB, MSB
            # Configuration Bits put in the Register:

            # 000 10 0 1100 00 00 0 0
            # [Res.]=000
            # [ALS_GAIN]=10 : Gain=1/8
            # [Res.]=0
            # [ALS_IT]=1100 : Integration Time=25 ms
            # [ALS_PERS]=00 : Persistance=1
            # [Res.]=00
            # [ALS_INT_EN]=0
            # [ALS_SD]=0 : Power On

            self.ConfValues = [0x00, 0x13]
        #

        # Reference data sheet Table 2 for High Threshold
            # Clear values
            self.Interrupt_High = [0x00, 0x00]
        #

        # Reference data sheet Table 3 for Low Threshold
            # Clear values
            self.Interrupt_Low = [0x00, 0x00]
        #

        # Reference data sheet Table 4 for Power Saving Modes
            # Clear values
            # Power Save Disabled
            self.Power_Save_Mode = [0x00, 0x00]
        #

        # Write configuration
            # Write Configuration Value into the Configuration Register
            self.SetRegisterValue(self.Als_Conf_0, self.ConfValues)
            # self.Bus.write_i2c_block_data(self.I2cAddr, self.Als_Conf_0, self.ConfValues)
            #
```

```python

            # Clear High Threshold Window Setting
            self.SetRegisterValue(self.Als WH, self.Interrupt High)
            # self.Bus.write i2c block data(self.I2cAddr, self.Als WH, self.Interrupt High)
            #

            # Clear Low Threshold Window Setting
            self.SetRegisterValue(self.Als WL, self.Interrupt Low)
            # self.Bus.write i2c block data(self.I2cAddr, self.Als WL, self.Interrupt Low)
            #

            # SetUp Power Save Mode
            self.SetRegisterValue(self.Power Save, self.Power Save Mode)
            # self.Bus.write i2c block data(self.I2cAddr, self.Power Save, self.Power Save Mode)
            #
        #
    # End Function

    # ------------------------------------------------------------
    # GetSensorData
    # ------------------------------------------------------------
    # Read the Data from the Sensor
    def GetSensorData(self):
        # Extract the Value for the Luminance

        # Sleep for 40 msec
            #time.sleep(0.04)
        #

        # Read the Value from the Sensor
            # Read 16 bits from the Bus and convert it to an integer
            Word = self.Bus.read_word_data(self.I2cAddr, self.Als)
        #

        # Gain for 1/8 gain & 25ms Integration Time
            Gain = 1.8432
        #

        # Calculate Lux Value
            Value = Word * Gain
        #

        # Round value for presentation
            # Value = round(Value, 1)
            self.IlluminanceValue = Value
        #

        # Return Value
            return(True)
        #
    # End Function

    # ------------------------------------------------------------
    # GetIlluminanceValue
    # ------------------------------------------------------------
    # Read the Data from the sensor
    def GetIlluminanceValue(self):
        return(self.IlluminanceValue)
```



```python
        #
# End Class
```

14.9 AS7262Sensor.py

```python
# Definition of the Class that Model the AS7262 Sensor
import time
import struct

import smbus
from i2cdevice import Device, Register, BitField,  int_to_bytes
from i2cdevice.adapter import Adapter, LookupAdapter

# ##############################################################
# Class Definition
# ##############################################################
class CalibratedValues:
    """Store the 6 band spectral values."""

    # --------------------------------------------------------------
    # Init
    # --------------------------------------------------------------
    # Class Init Function
    def __init__(self, red, orange, yellow, green, blue, violet):
        # noqa D107
        self.red = red
        self.orange = orange
        self.yellow = yellow
        self.green = green
        self.blue = blue
        self.violet = violet
    # End Function

    # --------------------------------------------------------------
    # Iterator
    # --------------------------------------------------------------
    # Class Iterator Function
    def __iter__(self):
        # noqa D107
        for colour in ['red', 'orange', 'yellow', 'green', 'blue', 'violet']:
            yield getattr(self, colour)
    # End Function
# End Class

# ##############################################################
# Class Definition
# ##############################################################
class AS7262_VirtualRegisterBus():
    """AS7262 Virtual Register.

    This class implements the wacky virtual register setup
    of the AS7262 annd allows i2cdevice.Device to "just work"
    without having to worry about how registers are actually
    read or written under the hood.

    """

    #
    def __init__(self, bus=1):
        """Initialise virtual register class.

        :param bus: SMBus bus ID

        """
        self._i2c_bus = smbus.SMBus(bus)
    # End Function
```



```python

    #
    def get_status(self, address):
        """Return the AS7262 status register."""
        return self. i2c_bus.read_byte_data(address, 0x00)
    # End Function

    #
    def write_i2c_block_data(self, address, register, values):
        """Right one or more values to AS7262 virtual registers."""
        for offset in range(len(values)):
            while True:
                if (self.get_status(address) & 0b10) == 0:
                    break
                #
            #

            self. i2c_bus.write_byte_data(address, 0x01, register | 0x80)

            while True:
                if (self.get_status(address) & 0b10) == 0:
                    break
                #
            #

            self. i2c_bus.write_byte_data(address, 0x01, values[offset])
        # End For
    # End Function

    #
    def read_i2c_block_data(self, address, register, length):
        """Read one or more values from AS7262 virtual registers."""
        result = []
        for offset in range(length):
            while True:
                if (self.get_status(address) & 0b10) == 0:
                    break
                # End If
            # End While

            self._i2c_bus.write_byte_data(address, 0x01, register + offset)

            while True:
                if (self.get_status(address) & 0b01) == 1:
                    break
                # End If
            # End While

            result.append(self._i2c_bus.read_byte_data(address, 0x02))

        return result
    # End Function
#End Class

# ##############################################################
# Class Definition
# ##############################################################
class FWVersionAdapter(Adapter):
    """Convert the AS7262 firmware version number to a human-readable string."""

    #
```



```python
    def  decode(self, value):
        major_version = (value & 0x00F0) >> 4
        minor_version = ((value & 0x000F) << 2) | ((value & 0b1100000000000000) >>  ↵
        14)
        sub_version = (value & 0b0011111100000000) >> 8
        return '{}.{}.{}'.format(major_version, minor_version, sub_version)
    # End Function
# End Class

# ##############################################################
# Class Definition
# ##############################################################
class FloatAdapter(Adapter):
    """Convert a 4 byte register set into a float."""

    #
    def  decode(self, value):
        b =  int_to_bytes(value, 4)
        return struct.unpack('>f', bytearray(b))[0]
    # End Function
# End Class

# ##############################################################
# Class Definition
# ##############################################################
class IntegrationTimeAdapter(Adapter):
    """Scale integration time in ms to LSBs."""

    #
    def _decode(self, value):
        return value / 2.8

    # End Function

    #
    def _encode(self, value):
        return int(value * 2.8) & 0xff
    # End Function
# End Class

# Class Definition
class AS7262Sensor:

    # --------------------------------------------------------------
    # Init
    # --------------------------------------------------------------
    # Class Init Function
    def __init__(self):
        # Instance Variable Definition
            self.Version_Register = None
            self.Control_Register = None
            self.IntegrationTime_Register = None
            self.Temperature_Register = None
            self.LedControl_Register = None
            self.Data_Register = None
            self.CalibratedData_Register = None

            self.I2CDevice = None

            self.CalibratedSpectralValues = None
        #
```



```python

        # Instance Variable Initialization
            self.Version_Register = Register\
            (\
                'VERSION',\
                0x00,\
                fields =\
                (\
                    BitField('hw_type', 0xFF000000),\
                    BitField('hw_version', 0x00FF0000),\
                    BitField('fw_version', 0x0000FFFF, adapter =                ↵
                    FWVersionAdapter()),\
                ),\
                bit_width=32,\
                read_only=True\
            )

            self.Control_Register = Register\
            (\
                'CONTROL',\
                0x04,\
                fields =\
                (\
                    BitField('reset', 0b10000000),\
                    BitField('interrupt', 0b01000000),\
                    BitField('gain_x', 0b00110000, adapter = LookupAdapter({1:      ↵
                    0b00, 3.7: 0b01, 16: 0b10, 64: 0b11})),\
                    BitField('measurement_mode', 0b00001100),\
                    BitField('data_ready', 0b00000010),\
                )\
            )

            self.IntegrationTime_Register = Register\
            (\
                'INTEGRATION_TIME',\
                0x05,\
                fields =\
                (\
                    BitField('ms', 0xFF, adapter = IntegrationTimeAdapter()),\
                )\
            )

            self.Temperature_Register = Register\
            (
                'TEMPERATURE',\
                0x06,\
                fields =\
                (\
                    BitField('degrees_c', 0xFF),\
                )\
            )

            self.LedControl_Register = Register\
            (\
                'LED_CONTROL',\
                0x07,\
                fields =\
                (\
                    BitField('illumination_current_limit_ma', 0b00110000, adapter   ↵
                    = LookupAdapter({12.5: 0b00, 25: 0b01, 50: 0b10, 100: 0b11})),\
                    BitField('illumination_enable', 0b00001000),\
```

```python
                    BitField('indicator current limit ma', 0b00000110, adapter =
                    LookupAdapter({1: 0b00, 2: 0b01, 4: 0b10, 8: 0b11})),\
                    BitField('indicator enable', 0b00000001),\
                )\
            )

            self.Data Register = Register\
            (
                'DATA',\
                0x08,\
                fields=\
                (
                    BitField('v', 0xFFFF0000000000000000000000),\
                    BitField('b', 0x0000FFFF0000000000000000),\
                    BitField('g', 0x00000000FFFF000000000000),\
                    BitField('y', 0x000000000000FFFF00000000),\
                    BitField('o', 0x0000000000000000FFFF0000),\
                    BitField('r', 0x000000000000000000000FFFF),\
                ),\
                bit width=96\
            )

            self.CalibratedData Register = Register\
            (
                'CALIBRATED DATA',\
                0x14,\
                fields =\
                (\
                    BitField('v', 0xFFFFFFFF << (32 * 5), adapter=FloatAdapter()),\
                    BitField('b', 0xFFFFFFFF << (32 * 4), adapter=FloatAdapter()),\
                    BitField('g', 0xFFFFFFFF << (32 * 3), adapter=FloatAdapter()),\
                    BitField('y', 0xFFFFFFFF << (32 * 2), adapter=FloatAdapter()),\
                    BitField('o', 0xFFFFFFFF << (32 * 1), adapter=FloatAdapter()),\
                    BitField('r', 0xFFFFFFFF << (32 * 0), adapter=FloatAdapter()),\
                ),\
                bit_width=192\
            )

            self.I2CDevice = Device\
            (\
                0x49,\
                i2c_dev = AS7262_VirtualRegisterBus(1),\
                bit_width = 8,\
                registers = \
                (\
                    self.Version_Register,\
                    self.Control_Register,\
                    self.IntegrationTime_Register,\
                    self.Temperature_Register,\
                    self.LedControl_Register,\
                    self.Data_Register,\
                    self.CalibratedData_Register,\
                )\
            )
        #

        # Adapter Look Up Table Export to Constant
            for register in self.I2CDevice.registers:
                register = self.I2CDevice.registers[register]

                for field in register.fields:
```



```python
                    field = register.fields[field]

                    # Export the Adapter
                    if isinstance(field.adapter, LookupAdapter):
                        for key in field.adapter.lookup table:
                            value = field.adapter.lookup table[key]
                            name = 'AS7262 {register} {field} {key}'.format\
                            (
                                register=register.name,
                                field=field.name,
                                key=key
                            ).upper()
                            locals()[name] = key
                        # End For
                    # End For
                #End For
            # End For
        #
    # End Function

    # ------------------------------------------------------------
    # SetUpSensor
    # ------------------------------------------------------------
    # Set Up Sensor
    def SetUpSensor(self):
        self.SetGain(64)

        self.SetIntegrationTime(17.857)

        self.SetMeasurementMode(2)

        # as7262.set illumination led current(12.5)
        self.SetIlluminationLed(0)
    #

    # ------------------------------------------------------------
    # SoftReset
    # ------------------------------------------------------------
    # Soft Reset the Sensor
    def SoftReset(self):
        """Set the soft reset register bit of the AS7262."""
        self.I2CDevice.CONTROL.set_reset(1)

        # Polling for the state of the reset flag does not work here
        # since the fragile virtual register state machine cannot
        # respond while in a soft reset condition
        # So, just wait long enough for it to reset fully...
        time.sleep(2.0)
    # End Function

    # ------------------------------------------------------------
    # SetGain
    # ------------------------------------------------------------
    # Set the Gain Value
    def SetGain(self, gain):
        """Set the gain amount of the AS7262.

        :param gain: gain multiplier, one of 1, 3.7, 16 or 64

        """
        self.I2CDevice.CONTROL.set_gain_x(gain)
```



```python
    # End Function

    # ------------------------------------------------------------
    # SetIntegrationTime
    # ------------------------------------------------------------
    def SetIntegrationTime(self, time ms):
        """Set the AS7262 sensor integration time in milliseconds.

        :param time ms: Time in milliseconds from 0 to ~91

        """
        self.I2CDevice.INTEGRATION TIME.set ms(time ms)
    # End Function

    # ------------------------------------------------------------
    # SetIntegrationTime
    # ------------------------------------------------------------
    def SetMeasurementMode(self, mode):
        """Set the AS7262 measurement mode.

        :param mode: 0-3

        """
        self.I2CDevice.CONTROL.set_measurement_mode(mode)
    # End Function

    # ------------------------------------------------------------
    # SetIlluminationLed
    # ------------------------------------------------------------
    def SetIlluminationLed(self, state):
        """Set the AS7262 illumination LED state.

        :param state: True = On, False = Off

        """
        self.I2CDevice.LED_CONTROL.set_illumination_enable(state)
    # End Function

    # ------------------------------------------------------------
    # GetSensorData
    # ------------------------------------------------------------
    def GetSensorData(self, timeout = 10):
        """Return an instance of CalibratedValues containing the 6 spectral bands."""
        t_start = time.time()

        while self.I2CDevice.CONTROL.get_data_ready() == 0 and (time.time() -
        t_start) <= timeout:
            pass
        # End While

        with self.I2CDevice.CALIBRATED_DATA as DATA:
            self.CalibratedSpectralValues = CalibratedValues\
            (
                DATA.get_r(),
                DATA.get_o(),
                DATA.get_y(),
                DATA.get_g(),
                DATA.get_b(),
                DATA.get_v()
            )
        # End With
```



```python

        return(True)
    # End Function

    # ------------------------------------------------------------
    # GetRedValue
    # ------------------------------------------------------------
    def GetRedValue(self):
        # 'red', 'orange', 'yellow', 'green', 'blue', 'violet'
        Value = self.CalibratedSpectralValues.red
        return(Value)
    # End Function

    # ------------------------------------------------------------
    # GetOrangeValue
    # ------------------------------------------------------------
    def GetOrangeValue(self):
        # 'red', 'orange', 'yellow', 'green', 'blue', 'violet'
        Value = self.CalibratedSpectralValues.orange
        return(Value)
    # End Function

    # ------------------------------------------------------------
    # GetYellowValue
    # ------------------------------------------------------------
    def GetYellowValue(self):
        # 'red', 'orange', 'yellow', 'green', 'blue', 'violet'
        Value = self.CalibratedSpectralValues.yellow
        return(Value)
    # End Function

    # ------------------------------------------------------------
    # GetGreenValue
    # ------------------------------------------------------------
    def GetGreenValue(self):
        # 'red', 'orange', 'yellow', 'green', 'blue', 'violet'
        Value = self.CalibratedSpectralValues.green
        return(Value)
    # End Function

    # ------------------------------------------------------------
    # GetBlueValue
    # ------------------------------------------------------------
    def GetBlueValue(self):
        # 'red', 'orange', 'yellow', 'green', 'blue', 'violet'
        Value = self.CalibratedSpectralValues.blue
        return(Value)
    # End Function

    # ------------------------------------------------------------
    # GetVioletValue
    # ------------------------------------------------------------
    def GetVioletValue(self):
        # 'red', 'orange', 'yellow', 'green', 'blue', 'violet'
        Value = self.CalibratedSpectralValues.violet
        return(Value)
    # End Function

# EndClass
```

14.10 DataVault.py

```python
# Class Definition
# [INFO]: The Class has "object" as a Parent Class

import smbus
import time
import math

from PyQt5 import QtSql
from PyQt5 import QtGui
from PyQt5.QtCore import QDate
from PyQt5.QtCore import QTime
from PyQt5.QtCore import QDateTime
from PyQt5.QtSql import *
from PyQt5.QtGui import *

class DataVault:
    # ---------------------------------------------------------------
    # Init
    # ---------------------------------------------------------------
    # Funzione di Inizializzazione della Classe
    def   init  (self):
        # Call Warning
            print("DataVault Init: Called")
        #

        # Inizializzazione della Classe
            # DataBase associato al Data Vault
            self.VaultDB = None
        #
    # End Function

    # ---------------------------------------------------------------
    # Init
    # ---------------------------------------------------------------
    # Connessione al DataBase "SQLITE3"
    def ConnectToDB(self):
        # Inizializzazione della Classe
        self.VaultDB = QSqlDatabase.addDatabase('QSQLITE')
        self.VaultDB.setDatabaseName('DataVault.db')

        Res = self.VaultDB.open()
        if Res == False:
            print("ConnectToDB: Cannot Open the DataBase")
            ConnectRes=False
        else:
            ConnectRes=True
            print("ConnectToDB: Connection to DB Open")
        # End If

        return(ConnectRes)
    # End Function

    # ---------------------------------------------------------------
    # WriteSample
    # ---------------------------------------------------------------
    # Add a Data Sample to the DB
    def WriteSample(self, SampleDate_Str, SampleTimeStamp_Str, Temperature,
    Pressure, Humidity, Illuminance):
        # Write Sample to Data Vault
            print("WriteSample: Called")
        #
```



```python

        # Date Part Extraction
            SampleDate Date = QDate()
            SampleDate Date = QDate.fromString(SampleDate Str,"yyyy-MM-dd")

            SampleTimeStamp Time = QTime()
            SampleTimeStamp Time = QTime.fromString(SampleTimeStamp Str,"hh:mm:ss")

            Year = SampleDate Date.year()
            Month = SampleDate Date.month()
            Day = SampleDate Date.day()
        #

        # TimStamp Part extraction
            Hour = SampleTimeStamp Time.hour()
            Minute = SampleTimeStamp Time.minute()
            Second = SampleTimeStamp Time.second()
        #

        # Query Text Formatting
            QueryText = \
            "\
                INSERT INTO Sample\
                (\
                    Date, Year, Month, Day,\
                    TimeStamp, Hour, Minute, Second,\
                    Temperature, Pressure, Humidity, Illuminance\
                )\
                VALUES\
                (\
                    '{}', {}, {}, {},\
                    '{}', {}, {}, {},\
                    {}, {}, {}, {}\
                )\
            ".format\
            (\
                SampleDate_Str, Year, Month, Day,\
                SampleTimeStamp_Str, Hour, Minute, Second,\
                Temperature, Pressure, Humidity, Illuminance\
            )
            # print("WriteSample: Execute Query")
            # print(QueryText)
        #

        # Query Execute
            Query = QSqlQuery(self.VaultDB)
            Res = Query.exec(QueryText)
            if (Res == True):
                print("Query executed")
                print(QueryText)
            else:
                print("Query Failed")
                Msg = Query.lastError().text()
                print(Msg)
            #
        #
    #

    # ---------------------------------------------------------------
    # ReadSample
    # ---------------------------------------------------------------
```



```python
    # Return a Cursor with the Samples in the Interval
    def ReadSamples(self, InfTimeStamp = 0, SupTimeStamp = 0):
        # Read Sample to Data Vault
            print("ReadSamples: Called")
            QueryText = "SELECT IdSample, Temperature, Pressure, Humidity,
            Illuminance FROM Sample WHERE Date='{}'"
            QueryText = QueryText.format(InfTimeStamp)
            Query = QSqlQuery(self.VaultDB)
            Res = Query.exec(QueryText)

            if(Res == True):
                print("Query executed")
                print(QueryText)
                while(Query.next()):
                    Value = Query.value(1)
                    Msg = "Temperature Value: {0:.2f} C".format(Value)
                    print(Msg)
                #End While
            else:
                print("Query Failed")
                Msg = Query.lastError().text()
                print(Msg)
            # End If

            return(Query)
        #
    #
# End Class
```

14.11 PySensorsDialog.py

```python
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'PySensorsDialog.ui'
#
# Created by: PyQt5 UI code generator 5.13.0
#
# WARNING! All changes made in this file will be lost!


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_PySensorsDialog(object):
    def setupUi(self, PySensorsDialog):
        PySensorsDialog.setObjectName("PySensorsDialog")
        PySensorsDialog.resize(661, 454)
        self.verticalLayout = QtWidgets.QVBoxLayout(PySensorsDialog)
        self.verticalLayout.setObjectName("verticalLayout")
        self.tabWidget = QtWidgets.QTabWidget(PySensorsDialog)
        self.tabWidget.setTabPosition(QtWidgets.QTabWidget.South)
        self.tabWidget.setObjectName("tabWidget")
        self.tab = QtWidgets.QWidget()
        self.tab.setObjectName("tab")
        self.verticalLayout_2 = QtWidgets.QVBoxLayout(self.tab)
        self.verticalLayout_2.setObjectName("verticalLayout_2")
        self.groupBox_2 = QtWidgets.QGroupBox(self.tab)
        self.groupBox_2.setObjectName("groupBox_2")
        self.gridLayout_2 = QtWidgets.QGridLayout(self.groupBox_2)
        self.gridLayout_2.setObjectName("gridLayout_2")
        self.label_7 = QtWidgets.QLabel(self.groupBox_2)
        self.label_7.setObjectName("label_7")
        self.gridLayout_2.addWidget(self.label_7, 1, 4, 1, 1)
        self.label_9 = QtWidgets.QLabel(self.groupBox_2)
        self.label_9.setObjectName("label_9")
        self.gridLayout_2.addWidget(self.label_9, 1, 0, 1, 1)
        self.label_12 = QtWidgets.QLabel(self.groupBox_2)
        self.label_12.setObjectName("label_12")
        self.gridLayout_2.addWidget(self.label_12, 2, 4, 1, 1)
        self.OversampligHumidityHexadecimalLEdit_3 = QtWidgets.QLineEdit(self.groupBox_2)

        self.OversampligHumidityHexadecimalLEdit_3.setObjectName("OversampligHumidityHexadecimalLEdit_3")
        self.gridLayout_2.addWidget(self.OversampligHumidityHexadecimalLEdit_3, 3, 3, 1, 1)
        self.label_11 = QtWidgets.QLabel(self.groupBox_2)
        self.label_11.setObjectName("label_11")
        self.gridLayout_2.addWidget(self.label_11, 2, 2, 1, 1)
        self.OperationModeHumanReadableLEdit = QtWidgets.QLineEdit(self.groupBox_2)

        self.OperationModeHumanReadableLEdit.setObjectName("OperationModeHumanReadableLEdit")
        self.gridLayout_2.addWidget(self.OperationModeHumanReadableLEdit, 0, 5, 1, 1)
        self.label_10 = QtWidgets.QLabel(self.groupBox_2)
        self.label_10.setObjectName("label_10")
        self.gridLayout_2.addWidget(self.label_10, 2, 0, 1, 1)
        self.label_6 = QtWidgets.QLabel(self.groupBox_2)
        self.label_6.setObjectName("label_6")
        self.gridLayout_2.addWidget(self.label_6, 0, 4, 1, 1)
        self.label_14 = QtWidgets.QLabel(self.groupBox_2)
        self.label_14.setObjectName("label_14")
```



```python
        self.gridLayout_2.addWidget(self.label_14, 3, 2, 1, 1)
        self.label_5 = QtWidgets.QLabel(self.groupBox_2)
        self.label_5.setObjectName("label_5")
        self.gridLayout_2.addWidget(self.label_5, 0, 2, 1, 1)
        self.label_8 = QtWidgets.QLabel(self.groupBox_2)
        self.label_8.setObjectName("label_8")
        self.gridLayout_2.addWidget(self.label_8, 1, 2, 1, 1)
        self.label_13 = QtWidgets.QLabel(self.groupBox_2)
        self.label_13.setObjectName("label_13")
        self.gridLayout_2.addWidget(self.label_13, 3, 0, 1, 1)
        self.OversampligTemperatureHumanReadableLEdit = QtWidgets.QLineEdit(self.groupBox_2)

        self.OversampligTemperatureHumanReadableLEdit.setObjectName("OversampligTemperatureHumanReadableLEdit")
        self.gridLayout_2.addWidget(self.OversampligTemperatureHumanReadableLEdit, 1, 5, 1, 1)
        self.OversampligPressureHumanReadableLEdit_2 = QtWidgets.QLineEdit(self.groupBox_2)

        self.OversampligPressureHumanReadableLEdit_2.setObjectName("OversampligPressureHumanReadableLEdit_2")
        self.gridLayout_2.addWidget(self.OversampligPressureHumanReadableLEdit_2, 2, 5, 1, 1)
        self.label_15 = QtWidgets.QLabel(self.groupBox_2)
        self.label_15.setObjectName("label_15")
        self.gridLayout_2.addWidget(self.label_15, 3, 4, 1, 1)
        self.ReadSensorRegisterPB = QtWidgets.QPushButton(self.groupBox_2)
        self.ReadSensorRegisterPB.setObjectName("ReadSensorRegisterPB")
        self.gridLayout_2.addWidget(self.ReadSensorRegisterPB, 5, 0, 1, 6)
        self.OversampligTemperatureHexadecimalLEdit = QtWidgets.QLineEdit(self.groupBox_2)

        self.OversampligTemperatureHexadecimalLEdit.setObjectName("OversampligTemperatureHexadecimalLEdit")
        self.gridLayout_2.addWidget(self.OversampligTemperatureHexadecimalLEdit, 1, 3, 1, 1)
        self.OversampligPressureHexadecimalLEdit_2 = QtWidgets.QLineEdit(self.groupBox_2)

        self.OversampligPressureHexadecimalLEdit_2.setObjectName("OversampligPressureHexadecimalLEdit_2")
        self.gridLayout_2.addWidget(self.OversampligPressureHexadecimalLEdit_2, 2, 3, 1, 1)
        self.OversampligHumidityHumanReadableLEdit_3 = QtWidgets.QLineEdit(self.groupBox_2)

        self.OversampligHumidityHumanReadableLEdit_3.setObjectName("OversampligHumidityHumanReadableLEdit_3")
        self.gridLayout_2.addWidget(self.OversampligHumidityHumanReadableLEdit_3, 3, 5, 1, 1)
        self.OperationModeHexadecimalLEdit = QtWidgets.QLineEdit(self.groupBox_2)

        self.OperationModeHexadecimalLEdit.setObjectName("OperationModeHexadecimalLEdit")
        self.gridLayout_2.addWidget(self.OperationModeHexadecimalLEdit, 0, 3, 1, 1)
        self.label_4 = QtWidgets.QLabel(self.groupBox_2)
        self.label_4.setObjectName("label_4")
        self.gridLayout_2.addWidget(self.label_4, 0, 0, 1, 1)
        self.label_16 = QtWidgets.QLabel(self.groupBox_2)
        self.label_16.setObjectName("label_16")
```



```python
        self.gridLayout_2.addWidget(self.label_16, 4, 0, 1, 1)
        self.label_17 = QtWidgets.QLabel(self.groupBox_2)
        self.label_17.setObjectName("label_17")
        self.gridLayout_2.addWidget(self.label_17, 4, 2, 1, 1)
        self.FilterCoefficentHexadecimalLEdit_4 = QtWidgets.QLineEdit(self.groupBox_2)

        self.FilterCoefficentHexadecimalLEdit_4.setObjectName("FilterCoefficentHexadecimalLEdit_4")
        self.gridLayout_2.addWidget(self.FilterCoefficentHexadecimalLEdit_4, 4, 3, 1, 1)
        self.label_18 = QtWidgets.QLabel(self.groupBox_2)
        self.label_18.setObjectName("label_18")
        self.gridLayout_2.addWidget(self.label_18, 4, 4, 1, 1)
        self.FilterCoefficentHumanReadableLEdit_4 = QtWidgets.QLineEdit(self.groupBox_2)

        self.FilterCoefficentHumanReadableLEdit_4.setObjectName("FilterCoefficentHumanReadableLEdit_4")
        self.gridLayout_2.addWidget(self.FilterCoefficentHumanReadableLEdit_4, 4, 5, 1, 1)
        self.verticalLayout_2.addWidget(self.groupBox_2)
        self.groupBox = QtWidgets.QGroupBox(self.tab)
        self.groupBox.setObjectName("groupBox")
        self.gridLayout = QtWidgets.QGridLayout(self.groupBox)
        self.gridLayout.setObjectName("gridLayout")
        self.label_2 = QtWidgets.QLabel(self.groupBox)
        self.label_2.setObjectName("label_2")
        self.gridLayout.addWidget(self.label_2, 2, 0, 1, 1)
        self.label_3 = QtWidgets.QLabel(self.groupBox)
        self.label_3.setObjectName("label_3")
        self.gridLayout.addWidget(self.label_3, 3, 0, 1, 1)
        self.label = QtWidgets.QLabel(self.groupBox)
        self.label.setObjectName("label")
        self.gridLayout.addWidget(self.label, 1, 0, 1, 1)
        self.PressureLEdit = QtWidgets.QLineEdit(self.groupBox)
        self.PressureLEdit.setObjectName("PressureLEdit")
        self.gridLayout.addWidget(self.PressureLEdit, 2, 1, 1, 1)
        self.TemperatureLEdit = QtWidgets.QLineEdit(self.groupBox)
        self.TemperatureLEdit.setObjectName("TemperatureLEdit")
        self.gridLayout.addWidget(self.TemperatureLEdit, 1, 1, 1, 1)
        self.HumidityLEdit = QtWidgets.QLineEdit(self.groupBox)
        self.HumidityLEdit.setObjectName("HumidityLEdit")
        self.gridLayout.addWidget(self.HumidityLEdit, 3, 1, 1, 1)
        self.verticalLayout_2.addWidget(self.groupBox)
        self.horizontalLayout = QtWidgets.QHBoxLayout()
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.SetUpSensorPB = QtWidgets.QPushButton(self.tab)
        self.SetUpSensorPB.setObjectName("SetUpSensorPB")
        self.horizontalLayout.addWidget(self.SetUpSensorPB)
        self.pushButton_2 = QtWidgets.QPushButton(self.tab)
        self.pushButton_2.setObjectName("pushButton_2")
        self.horizontalLayout.addWidget(self.pushButton_2)
        self.ReadSensorValuesPB = QtWidgets.QPushButton(self.tab)
        self.ReadSensorValuesPB.setObjectName("ReadSensorValuesPB")
        self.horizontalLayout.addWidget(self.ReadSensorValuesPB)
        self.verticalLayout_2.addLayout(self.horizontalLayout)
        self.tabWidget.addTab(self.tab, "")
        self.tab_4 = QtWidgets.QWidget()
        self.tab_4.setObjectName("tab_4")
        self.verticalLayout_4 = QtWidgets.QVBoxLayout(self.tab_4)
```



```python
        self.verticalLayout_4.setObjectName("verticalLayout_4")
        self.groupBox_3 = QtWidgets.QGroupBox(self.tab_4)
        self.groupBox_3.setObjectName("groupBox_3")
        self.gridLayout_5 = QtWidgets.QGridLayout(self.groupBox_3)
        self.gridLayout_5.setObjectName("gridLayout_5")
        self.label_31 = QtWidgets.QLabel(self.groupBox_3)
        self.label_31.setObjectName("label_31")
        self.gridLayout_5.addWidget(self.label_31, 0, 1, 1, 1)
        self.label_20 = QtWidgets.QLabel(self.groupBox_3)
        self.label_20.setObjectName("label_20")
        self.gridLayout_5.addWidget(self.label_20, 5, 0, 1, 1)
        self.label_40 = QtWidgets.QLabel(self.groupBox_3)
        self.label_40.setObjectName("label_40")
        self.gridLayout_5.addWidget(self.label_40, 2, 0, 1, 1)
        self.label_41 = QtWidgets.QLabel(self.groupBox_3)
        self.label_41.setObjectName("label_41")
        self.gridLayout_5.addWidget(self.label_41, 3, 0, 1, 1)
        self.label_38 = QtWidgets.QLabel(self.groupBox_3)
        self.label_38.setObjectName("label_38")
        self.gridLayout_5.addWidget(self.label_38, 0, 0, 1, 1)
        self.label_24 = QtWidgets.QLabel(self.groupBox_3)
        self.label_24.setObjectName("label_24")
        self.gridLayout_5.addWidget(self.label_24, 2, 1, 1, 1)
        self.ReadSensor2RegisterPB = QtWidgets.QPushButton(self.groupBox_3)
        self.ReadSensor2RegisterPB.setObjectName("ReadSensor2RegisterPB")
        self.gridLayout_5.addWidget(self.ReadSensor2RegisterPB, 7, 0, 1, 5)
        self.label_25 = QtWidgets.QLabel(self.groupBox_3)
        self.label_25.setObjectName("label_25")
        self.gridLayout_5.addWidget(self.label_25, 6, 1, 1, 1)
        self.label_32 = QtWidgets.QLabel(self.groupBox_3)
        self.label_32.setObjectName("label_32")
        self.gridLayout_5.addWidget(self.label_32, 5, 1, 1, 1)
        self.HighThresholdWindowsLEdit = QtWidgets.QLineEdit(self.groupBox_3)
        self.HighThresholdWindowsLEdit.setObjectName("HighThresholdWindowsLEdit")
        self.gridLayout_5.addWidget(self.HighThresholdWindowsLEdit, 1, 3, 1, 1)
        self.label_28 = QtWidgets.QLabel(self.groupBox_3)
        self.label_28.setObjectName("label_28")
        self.gridLayout_5.addWidget(self.label_28, 1, 2, 1, 1)
        self.label_22 = QtWidgets.QLabel(self.groupBox_3)
        self.label_22.setObjectName("label_22")
        self.gridLayout_5.addWidget(self.label_22, 6, 0, 1, 1)
        self.label_21 = QtWidgets.QLabel(self.groupBox_3)
        self.label_21.setObjectName("label_21")
        self.gridLayout_5.addWidget(self.label_21, 1, 1, 1, 1)
        self.label_26 = QtWidgets.QLabel(self.groupBox_3)
        self.label_26.setObjectName("label_26")
        self.gridLayout_5.addWidget(self.label_26, 3, 2, 1, 1)
        self.label_39 = QtWidgets.QLabel(self.groupBox_3)
        self.label_39.setObjectName("label_39")
        self.gridLayout_5.addWidget(self.label_39, 1, 0, 1, 1)
        self.LowThresholdWindowsLEdit = QtWidgets.QLineEdit(self.groupBox_3)
        self.LowThresholdWindowsLEdit.setObjectName("LowThresholdWindowsLEdit")
        self.gridLayout_5.addWidget(self.LowThresholdWindowsLEdit, 2, 3, 1, 1)
        self.label_23 = QtWidgets.QLabel(self.groupBox_3)
        self.label_23.setObjectName("label_23")
        self.gridLayout_5.addWidget(self.label_23, 2, 2, 1, 1)
        self.label_29 = QtWidgets.QLabel(self.groupBox_3)
        self.label_29.setObjectName("label_29")
        self.gridLayout_5.addWidget(self.label_29, 3, 1, 1, 1)
        self.WhiteChannelOutputHexadecimalLEdit = QtWidgets.QLineEdit(self.groupBox_3)
```

```python

        self.WhiteChannelOutputHexadecimalLEdit.setObjectName("WhiteChannelOutputHexadecimalLEdit")
        self.gridLayout_5.addWidget(self.WhiteChannelOutputHexadecimalLEdit, 6, 3, 1, 1)
        self.label_30 = QtWidgets.QLabel(self.groupBox_3)
        self.label_30.setObjectName("label_30")
        self.gridLayout_5.addWidget(self.label_30, 6, 2, 1, 1)
        self.label_27 = QtWidgets.QLabel(self.groupBox_3)
        self.label_27.setObjectName("label_27")
        self.gridLayout_5.addWidget(self.label_27, 0, 2, 1, 1)
        self.PowerSavingModesLEdit = QtWidgets.QLineEdit(self.groupBox_3)
        self.PowerSavingModesLEdit.setObjectName("PowerSavingModesLEdit")
        self.gridLayout_5.addWidget(self.PowerSavingModesLEdit, 3, 3, 1, 1)
        self.label_33 = QtWidgets.QLabel(self.groupBox_3)
        self.label_33.setObjectName("label_33")
        self.gridLayout_5.addWidget(self.label_33, 5, 2, 1, 1)
        self.AlsHighResolutionOutputHexadecimalLEdit = QtWidgets.QLineEdit(self.groupBox_3)

        self.AlsHighResolutionOutputHexadecimalLEdit.setObjectName("AlsHighResolutionOutputHexadecimalLEdit")
        self.gridLayout_5.addWidget(self.AlsHighResolutionOutputHexadecimalLEdit, 5, 3, 1, 1)
        self.ConfigurationRegisterHexadecimalLEdit = QtWidgets.QLineEdit(self.groupBox_3)

        self.ConfigurationRegisterHexadecimalLEdit.setObjectName("ConfigurationRegisterHexadecimalLEdit")
        self.gridLayout_5.addWidget(self.ConfigurationRegisterHexadecimalLEdit, 0, 3, 1, 1)
        self.line = QtWidgets.QFrame(self.groupBox_3)
        self.line.setFrameShape(QtWidgets.QFrame.HLine)
        self.line.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line.setObjectName("line")
        self.gridLayout_5.addWidget(self.line, 4, 0, 1, 4)
        self.verticalLayout_4.addWidget(self.groupBox_3)
        self.groupBox_4 = QtWidgets.QGroupBox(self.tab_4)
        self.groupBox_4.setObjectName("groupBox_4")
        self.gridLayout_6 = QtWidgets.QGridLayout(self.groupBox_4)
        self.gridLayout_6.setObjectName("gridLayout_6")
        self.label_35 = QtWidgets.QLabel(self.groupBox_4)
        self.label_35.setObjectName("label_35")
        self.gridLayout_6.addWidget(self.label_35, 2, 0, 1, 1)
        self.label_36 = QtWidgets.QLabel(self.groupBox_4)
        self.label_36.setObjectName("label_36")
        self.gridLayout_6.addWidget(self.label_36, 3, 0, 1, 1)
        self.label_37 = QtWidgets.QLabel(self.groupBox_4)
        self.label_37.setObjectName("label_37")
        self.gridLayout_6.addWidget(self.label_37, 1, 0, 1, 1)
        self.PressureLEdit_2 = QtWidgets.QLineEdit(self.groupBox_4)
        self.PressureLEdit_2.setObjectName("PressureLEdit_2")
        self.gridLayout_6.addWidget(self.PressureLEdit_2, 2, 1, 1, 1)
        self.IlluminanceLEdit = QtWidgets.QLineEdit(self.groupBox_4)
        self.IlluminanceLEdit.setObjectName("IlluminanceLEdit")
        self.gridLayout_6.addWidget(self.IlluminanceLEdit, 1, 1, 1, 1)
        self.HumidityLEdit_2 = QtWidgets.QLineEdit(self.groupBox_4)
        self.HumidityLEdit_2.setObjectName("HumidityLEdit_2")
        self.gridLayout_6.addWidget(self.HumidityLEdit_2, 3, 1, 1, 1)
        self.verticalLayout_4.addWidget(self.groupBox_4)
        self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
```



```python
        self.horizontalLayout_2.setObjectName("horizontalLayout_2")
        self.SetUpSensor2PB = QtWidgets.QPushButton(self.tab_4)
        self.SetUpSensor2PB.setObjectName("SetUpSensor2PB")
        self.horizontalLayout_2.addWidget(self.SetUpSensor2PB)
        self.pushButton_3 = QtWidgets.QPushButton(self.tab_4)
        self.pushButton_3.setObjectName("pushButton_3")
        self.horizontalLayout_2.addWidget(self.pushButton_3)
        self.ReadSensor2ValuesPB = QtWidgets.QPushButton(self.tab_4)
        self.ReadSensor2ValuesPB.setObjectName("ReadSensor2ValuesPB")
        self.horizontalLayout_2.addWidget(self.ReadSensor2ValuesPB)
        self.verticalLayout_4.addLayout(self.horizontalLayout_2)
        self.tabWidget.addTab(self.tab_4, "")
        self.tab_2 = QtWidgets.QWidget()
        self.tab_2.setObjectName("tab_2")
        self.gridLayout_3 = QtWidgets.QGridLayout(self.tab_2)
        self.gridLayout_3.setObjectName("gridLayout_3")
        self.ConnectToDbPB = QtWidgets.QPushButton(self.tab_2)
        self.ConnectToDbPB.setObjectName("ConnectToDbPB")
        self.gridLayout_3.addWidget(self.ConnectToDbPB, 1, 0, 1, 1)
        self.ReadSamplePB = QtWidgets.QPushButton(self.tab_2)
        self.ReadSamplePB.setObjectName("ReadSamplePB")
        self.gridLayout_3.addWidget(self.ReadSamplePB, 1, 1, 1, 1)
        self.WriteSamplePB = QtWidgets.QPushButton(self.tab_2)
        self.WriteSamplePB.setObjectName("WriteSamplePB")
        self.gridLayout_3.addWidget(self.WriteSamplePB, 1, 2, 1, 1)
        self.tableView = QtWidgets.QTableView(self.tab_2)
        self.tableView.setObjectName("tableView")
        self.gridLayout_3.addWidget(self.tableView, 0, 0, 1, 3)
        self.tabWidget.addTab(self.tab_2, "")
        self.tab_3 = QtWidgets.QWidget()
        self.tab_3.setObjectName("tab_3")
        self.gridLayout_4 = QtWidgets.QGridLayout(self.tab_3)
        self.gridLayout_4.setObjectName("gridLayout_4")
        self.ChartDEdit = QtWidgets.QDateEdit(self.tab_3)
        self.ChartDEdit.setCalendarPopup(True)
        self.ChartDEdit.setObjectName("ChartDEdit")
        self.gridLayout_4.addWidget(self.ChartDEdit, 1, 1, 1, 1)
        self.label_19 = QtWidgets.QLabel(self.tab_3)
        self.label_19.setObjectName("label_19")
        self.gridLayout_4.addWidget(self.label_19, 1, 0, 1, 1)
        self.DrawChartPB = QtWidgets.QPushButton(self.tab_3)
        self.DrawChartPB.setObjectName("DrawChartPB")
        self.gridLayout_4.addWidget(self.DrawChartPB, 1, 2, 1, 1)
        self.ChartFrame = QtWidgets.QFrame(self.tab_3)
        self.ChartFrame.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.ChartFrame.setFrameShadow(QtWidgets.QFrame.Raised)
        self.ChartFrame.setObjectName("ChartFrame")
        self.verticalLayout_3 = QtWidgets.QVBoxLayout(self.ChartFrame)
        self.verticalLayout_3.setObjectName("verticalLayout_3")
        self.ChartLabel = QtWidgets.QLabel(self.ChartFrame)
        self.ChartLabel.setObjectName("ChartLabel")
        self.verticalLayout_3.addWidget(self.ChartLabel)
        self.gridLayout_4.addWidget(self.ChartFrame, 0, 0, 1, 3)
        self.DrawChartPB.raise_()
        self.label_19.raise_()
        self.ChartDEdit.raise_()
        self.ChartFrame.raise_()
        self.tabWidget.addTab(self.tab_3, "")
        self.verticalLayout.addWidget(self.tabWidget)

        self.retranslateUi(PySensorsDialog)
```



```python
        self.tabWidget.setCurrentIndex(1)

        self.SetUpSensorPB.clicked.connect(PySensorsDialog.GUICB_SetUpSensorPB_Clicked)

        self.ReadSensorValuesPB.clicked.connect(PySensorsDialog.GUICB_ReadSensorValuesPB_Clicked)

        self.ReadSensorRegisterPB.clicked.connect(PySensorsDialog.GUICB_ReadSensorRegisterPB_Clicked)

        self.ConnectToDbPB.clicked.connect(PySensorsDialog.GUICB_ConnectToDbPB_Clicked)
        self.ReadSamplePB.clicked.connect(PySensorsDialog.GUICB_ReadSamplePB_Clicked)

        self.WriteSamplePB.clicked.connect(PySensorsDialog.GUICB_WriteSamplePB_Clicked)
        self.DrawChartPB.clicked.connect(PySensorsDialog.GUICB_DrawChartPB_Clicked)

        self.SetUpSensor2PB.clicked.connect(PySensorsDialog.GUICB_SetUpSensor2PB_Clicked)

        self.ReadSensor2ValuesPB.clicked.connect(PySensorsDialog.GUICB_ReadSensor2ValuesPB_Clicked)

        self.ReadSensor2RegisterPB.clicked.connect(PySensorsDialog.GUICB_ReadSensor2RegisterPB_Clicked)
        QtCore.QMetaObject.connectSlotsByName(PySensorsDialog)

    def retranslateUi(self, PySensorsDialog):
        _translate = QtCore.QCoreApplication.translate
        PySensorsDialog.setWindowTitle(_translate("PySensorsDialog", "PySensors"))
        self.groupBox_2.setTitle(_translate("PySensorsDialog", "Sensor Registers"))
        self.label_7.setText(_translate("PySensorsDialog", "Hr:"))
        self.label_9.setText(_translate("PySensorsDialog", "Over Samplig Temperature"))
        self.label_12.setText(_translate("PySensorsDialog", "Hr:"))
        self.label_11.setText(_translate("PySensorsDialog", "Hx:"))
        self.label_10.setText(_translate("PySensorsDialog", "Over Samplig Pressure"))
        self.label_6.setText(_translate("PySensorsDialog", "Hr:"))
        self.label_14.setText(_translate("PySensorsDialog", "Hx:"))
        self.label_5.setText(_translate("PySensorsDialog", "Hx:"))
        self.label_8.setText(_translate("PySensorsDialog", "Hx:"))
        self.label_13.setText(_translate("PySensorsDialog", "Over Samplig Humidity"))
        self.label_15.setText(_translate("PySensorsDialog", "Hr:"))
        self.ReadSensorRegisterPB.setText(_translate("PySensorsDialog", "Read Sensor Registers"))
        self.label_4.setText(_translate("PySensorsDialog", "Operation Mode:"))
        self.label_16.setText(_translate("PySensorsDialog", "Filter Coefficent"))
        self.label_17.setText(_translate("PySensorsDialog", "Hx:"))
        self.label_18.setText(_translate("PySensorsDialog", "Hr:"))
        self.groupBox.setTitle(_translate("PySensorsDialog", "Sensor Data"))
        self.label_2.setText(_translate("PySensorsDialog", "Pressure"))
        self.label_3.setText(_translate("PySensorsDialog", "Humdity"))
        self.label.setText(_translate("PySensorsDialog", "Temperature"))
        self.SetUpSensorPB.setText(_translate("PySensorsDialog", "1 - SetUpSensor"))
        self.pushButton_2.setText(_translate("PySensorsDialog", "2 - Read Sensor ID"))
        self.ReadSensorValuesPB.setText(_translate("PySensorsDialog", "3 - Read Sensor Values"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab),
```



```python
 translate("PySensorsDialog", "BME680"))
        self.groupBox_3.setTitle( translate("PySensorsDialog", "Sensor Registers"))
        self.label_31.setText( translate("PySensorsDialog", "Configuration Register (16 bit)"))
        self.label_20.setText( translate("PySensorsDialog", "ALS"))
        self.label_40.setText( translate("PySensorsDialog", "ALS WL"))
        self.label_41.setText( translate("PySensorsDialog", "Power Saving"))
        self.label_38.setText( translate("PySensorsDialog", "ALS CONF 0"))
        self.label_24.setText( translate("PySensorsDialog", "Low Threshold Windows Setting"))
        self.ReadSensor2RegisterPB.setText( translate("PySensorsDialog", "Read Sensor Registers"))
        self.label_25.setText( translate("PySensorsDialog", "White Channel Output"))
        self.label_32.setText( translate("PySensorsDialog", "Als High Resolution Output Data"))
        self.label_28.setText( translate("PySensorsDialog", "Hx:"))
        self.label_22.setText( translate("PySensorsDialog", "WHITE"))
        self.label_21.setText( translate("PySensorsDialog", "High Threshold Windows Setting"))
        self.label_26.setText( translate("PySensorsDialog", "Hx:"))
        self.label_39.setText( translate("PySensorsDialog", "ALS WH"))
        self.label_23.setText( translate("PySensorsDialog", "Hx:"))
        self.label_29.setText(_translate("PySensorsDialog", "Power Saving Modes"))
        self.label_30.setText(_translate("PySensorsDialog", "Hx:"))
        self.label_27.setText(_translate("PySensorsDialog", "Hx:"))
        self.label_33.setText(_translate("PySensorsDialog", "Hx:"))
        self.groupBox_4.setTitle(_translate("PySensorsDialog", "Sensor Data"))
        self.label_35.setText(_translate("PySensorsDialog", "White Channel"))
        self.label_36.setText(_translate("PySensorsDialog", "--.--"))
        self.label_37.setText(_translate("PySensorsDialog", "Illuminance"))
        self.SetUpSensor2PB.setText(_translate("PySensorsDialog", "1 - SetUpSensor"))
        self.pushButton_3.setText(_translate("PySensorsDialog", "2 - Read Sensor ID"))
        self.ReadSensor2ValuesPB.setText(_translate("PySensorsDialog", "3 - Read Sensor Values"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_4), _translate("PySensorsDialog", "VEML 7700"))
        self.ConnectToDbPB.setText(_translate("PySensorsDialog", "Connect to DB"))
        self.ReadSamplePB.setText(_translate("PySensorsDialog", "Read Sample"))
        self.WriteSamplePB.setText(_translate("PySensorsDialog", "Write Samples"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_2), _translate("PySensorsDialog", "DataVault"))
        self.ChartDEdit.setDisplayFormat(_translate("PySensorsDialog", "dd/MM/yyyy"))
        self.label_19.setText(_translate("PySensorsDialog", "Giorno"))
        self.DrawChartPB.setText(_translate("PySensorsDialog", "Draw Chart"))
        self.ChartLabel.setText(_translate("PySensorsDialog", "Chart"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_3), _translate("PySensorsDialog", "Charts"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    PySensorsDialog = QtWidgets.QDialog()
    ui = Ui_PySensorsDialog()
    ui.setupUi(PySensorsDialog)
    PySensorsDialog.show()
    sys.exit(app.exec_())
```

14.12 PySensorsDialogImpl.py

```python
from PyQt5.QtCore import pyqtSlot
from PyQt5.QtCore import QByteArray
from PyQt5.QtCore import QBitArray
from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QWidget
from PyQt5.QtWidgets import QDialog
from PyQt5.QtCore import QDate
from PyQt5.QtCore import QTime
from PyQt5.QtCore import QTimer
from PyQt5.QtCore import QDateTime

from PySensorsDialog import Ui PySensorsDialog

from BME680Sensor import BME680Sensor
from VEML7700Sensor import VEML7700Sensor
from DataVault import DataVault
from PyQt5.QtChart import *

import time

# La Classe "PySensorsDialogImpl" eredita da QDialog ed ha una Variabile di istanza che contiene i Widget disegnati con
# QtDesigner
# usato il metodo della Variabile di Istanza e non dell'Ereditarietà multipla per integrare gli oggetti disegnati
# con QtDesigner


class PySensorsDialogImpl(QDialog):

    # ------------------------------------------------------------
    # Init
    # ------------------------------------------------------------
    def __init__(self, parent=None):
        # Call Warning
            print("PySensorsDialogImpl: Called")
        #

        # Class Attribute Definition
            # Timer for the Sensor Reading
            self.AppTimer = None

            #Sensore BME 680
            self.Sensor1 = None

            # Sensore VEML 7700
            self.Sensor2 = None

            #Repositiry SQL for the Sample
            self.DataVault = None

            # Chart Objects
            self.SampleChart = None
            self.SampleSeries = None
            self.SampleChartView = None
            #
        #

        # Inizializzazione del Parent
            super(PySensorsDialogImpl, self).__init__(parent)
        #
```



```python

        # Dichiarazione della Variabile di Istanza "ui" della Classe "Ui PySensorsDialog"  contiene i Widget grafici
            self.ui = Ui PySensorsDialog()
        #

        # Invocazione del Metodo ui.setupUi della Variabile di Istanza UI passando "self" come argomento
            self.ui.setupUi(self)
        #

        # Warning: SetUp dei Sensori per ora Sospeso!

        # Creazione dell'Istanza del Sensore1 "SensorBME280"
            self.Sensor1 = BME680Sensor()

            # Inizializzazione del Sensore
            # self.Sensor1.SetUpSensor()
        #

        # Creazione dell'Istanza del Sensore1 "SensorBME280"
            self.Sensor2 = VEML7700Sensor()

            # Inizializzazione del Sensore
            self.Sensor2.SetUpSensor()
        #

        # Data Vault Istance Creation
            # Data Vault Instance Creation
            self.DataVault = DataVault()

            #Connect to DB
            self.DataVault.ConnectToDB()
        #

        # Application Timer SetUp
            self.AppTimer = QTimer()

            self.AppTimer.timeout.connect(self.SLOT_ReadSensorsData)

            # Start the Timer
            # Warning : the Time is Suspended for now!
            self.AppTimer.start(30000)
            #
        #

        # Chart Objects initialization
            self.SampleChart = QChart()
            self.SampleSeries = QLineSeries()
            self.SampleChartView = QChartView()
            self.ui.verticalLayout_3.addWidget(self.SampleChartView)
        #
    #

    # CallBack sul SetUp del Sensore1: BME680
    @pyqtSlot()
    def GUICB_SetUpSensorPB_Clicked(self):
        print("GUICB_SetUpSensorPB_Clicked: Called")

        # Inizializzazione del Sensore
        self.Sensor1.SetUpSensor()
```



```python
    #

    # CallBack sul SetUp del Sensore2 VEML 7700
    @pyqtSlot()
    def GUICB SetUpSensor2PB Clicked(self):
            print("GUICB SetUpSensorPB Clicked: Called")

            # Inizializzazione del Sensore
            self.Sensor2.SetUpSensor()
    #

    # CallBack sulla Lettura dei Dati del Sensore 1
    @pyqtSlot()
    def GUICB ReadSensorValuesPB Clicked(self):
        print("GUICB ReadSensorValuesPB Clicked: Called")

        Res=self.Sensor1.GetSensorData()

        if Res == True:
            output = "Temperature Value: {0:.2f} C".format(self.Sensor1.GetTemperatureValue())

            print(output)
            self.ui.TemperatureLEdit.setText(output)
        else:
            print("Read Failed From Sensor")
        # End If
    # End Function

    # CallBack sulla Lettura dei Dati del Sensore 2
    @pyqtSlot()
    def GUICB_ReadSensor2ValuesPB_Clicked(self):
            print("GUICB_ReadSensorValuesPB_Clicked: Called")

            Res = self.Sensor2.GetSensorData()

            if Res == True:
                output = "Illuminance Value: {0:.2f} Lux".format(self.Sensor2.GetIlluminanceValue())

                print(output)
                self.ui.IlluminanceLEdit.setText(output)
            else:
                print("Read Failed From Sensor")
        # End If
    # End Function

    # CallBack sullo Scadere del TimeOut
    @pyqtSlot()
    def SLOT_ReadSensorsData(self):
        print("SLOT_ReadSensorsData: Called")

        Temperature = 0
        Pressure = 0
        Humidity = 0
        Illuminance = 0

        # The Sensor 1 is not present!
        # Res1 = self.Sensor1.GetSensorData()
        Res1 = False

```



```python
        Res2 = self.Sensor2.GetSensorData()

        #
        if Res1 == True:

            # Log
                print("SLOT ReadSensorsData: Read From Sensors 1 succeded")
            #

            #Read Data from Sensors 1
                Temperature = self.Sensor1.GetTemperatureValue()
                Value Str="{0:.1f}".format(Temperature)
                self.ui.TemperatureLEdit.setText(Value Str)

                Pressure = self.Sensor1.GetPressureValue()
                Value Str = "{0:.1f}".format(Pressure)
                self.ui.PressureLEdit.setText(Value Str)

                Humidity = self.Sensor1.GetHumidityValue()
                Value Str = "{0:.1f}".format(Humidity)
                self.ui.HumidityLEdit.setText(Value Str)
            #
        else:
            print("SLOT_ReadSensorsData: Read Failed From Sensors 1")
        # End If

        #
        if Res2 == True:

            # Msg Log
                print("SLOT_ReadSensorsData: Read From Sensors 2 succeded")
            #

            # Read Data from Sensors 2
                Illuminance = self.Sensor2.GetIlluminanceValue()
            #
        else:
            #
                print("SLOT_ReadSensorsData: Read Failed From Sensors 2")
            #
        # End If

        # Write Sample to DataVault
        print()
        #

        # Msg Log
        print("SLOT_ReadSensorsData: Write Data to Vault")
        #

        #
        Today = QDate.currentDate().toString("yyyy-MM-dd")
        Now = QTime.currentTime().toString("hh:mm:ss")
        self.DataVault.WriteSample(Today, Now, Temperature, Pressure, Humidity, Illuminance)
        #
    # End Function

    # CallBack sulla Lettura dei Registri di Controllo del Sensore1
    @pyqtSlot()
    def GUICB_ReadSensorRegisterPB_Clicked(self):
```

```
            print("GUICB ReadSensorRegisterPB Clicked: Called")
            Value Bytes = 25
            Value Str =self.ByteToString(Value Bytes)

            self.ui.OperationModeHexadecimalLEdit.setText(Value Str)

            # Valore del Registro "Operation Mode"
            # Res=self.Sensor1.GetOperationModeRegisterValue()
            # HexValue = Res[0]
            # HumValue = Res[1]
            #

            #Impostazione dei Line Edit
            # self.ui.OperationModeHexadecimalLEdit.setText(HexValue)
            # self.ui.OperationModeHumanReadableLEdit.setText(HumValue)
            #

        # EndFunction

        # CallBack sulla Lettura dei Registri di Controllo del Sensore2
        @pyqtSlot()
        def GUICB ReadSensor2RegisterPB Clicked(self):
            # Call Warning
                print("GUICB ReadSensor2RegisterPB Clicked: Called")
            #

            # ALS CONF 0 Register
                Value Bytes = self.Sensor2.GetAlsConf0RegisterValue()
                Value Str = self.BytesToString(Value Bytes)
                self.ui.ConfigurationRegisterHexadecimalLEdit.setText(Value Str)
            #

            # ALS_WH Register
                Value_Bytes = self.Sensor2.GetHighThresholdWindowsRegisterValue()
                Value_Str = self.BytesToString(Value_Bytes)
                self.ui.HighThresholdWindowsLEdit.setText(Value_Str)
            #

            # ALS_WL Register
                Value_Bytes = self.Sensor2.GetLowThresholdWindowsRegisterValue()
                Value_Str = self.BytesToString(Value_Bytes)
                self.ui.LowThresholdWindowsLEdit.setText(Value_Str)
            #

            # POWER_MODE_SE Register
                Value_Bytes = self.Sensor2.GetPowerSavingModesRegisterValue()
                Value_Str = self.BytesToString(Value_Bytes)
                self.ui.PowerSavingModesLEdit.setText(Value_Str)
            #

            # ALS HIGH RESOLUTION OUTPUT Register
                Value_Bytes = self.Sensor2.GetAlsHighResolutionOutputRegisterValue()
                Value_Str = self.BytesToString(Value_Bytes)
                self.ui.AlsHighResolutionOutputHexadecimalLEdit.setText(Value_Str)
            #

            # ALS HIGH RESOLUTION OUTPUT Register
                Value_Bytes = self.Sensor2.GetWhiteChannelOutputRegisterValue()
                Value_Str = self.BytesToString(Value_Bytes)
                self.ui.WhiteChannelOutputHexadecimalLEdit.setText(Value_Str)
            #
```



```
        # EndFunction

        # ----------------------------------------------------------------
        # Connect To DB
        # ----------------------------------------------------------------
        @pyqtSlot()
        def GUICB ConnectToDbPB Clikced(self):
            self.DataVault.ConnectToDB()
        # End Function

        # ----------------------------------------------------------------
        # Read Sample PB Clicked
        # ----------------------------------------------------------------
        @pyqtSlot()
        def GUICB ReadSamplePB Clicked(self):
            self.DataVault.ReadSamples()
        #

        # ----------------------------------------------------------------
        # Write Sample PB Clicked
        # ----------------------------------------------------------------
        @pyqtSlot()
        def GUICB WriteSamplePB Clicked(self):
            Today = QDate.currentDate().toString("yyyy-MM-dd")
            Now = QTime.currentTime().toString("hh:mm:ss")
            self.DataVault.WriteSample(Today, Now,0,0,0)
        #

        # ----------------------------------------------------------------
        # Draw Charts
        # ----------------------------------------------------------------
        @pyqtSlot()
        def GUICB_DrawChartPB_Clicked(self):
            # Series creation
                print("GUICB_DrawChartPB_Clicked: Called")
            #

            # Estrazione della Data
                Date_Str=self.ui.ChartDEdit.date().toString("yyyy-MM-dd")
            #

            # Estrazione della Query con i Sample per la Temperatura
                Query = self.DataVault.ReadSamples(Date_Str)
            #

            # Series formation
                Query.first()
                while(Query.next() == True):
                    IdSample = Query.value(0)
                    Temperature = Query.value(1)
                    self.SampleSeries.append(IdSample, Temperature)
                # End While
            #

            # Chart Set Up
                self.SampleChart.legend().hide()
                self.SampleChart.addSeries(self.SampleSeries)
                self.SampleChart.createDefaultAxes()
                self.SampleChart.setTitle("Temperature")
            #

```



```
            # SetUp ChartView
                self.SampleChartView.setChart(self.SampleChart)
            #
        #


        # Definizione dello Slot sul Cambio di Valore dello SpinBox2
        # mettere nella lista dei parametri i tipi di dato del Signal
        # @pyqtSlot(int)
        # def on inputSpinBox2 valueChanged(self, value):
        #    self.ui.outputWidget.setText(str(value + self.ui.inputSpinBox1.value()))
        #

        # ritorna una stringa con la rappresentazione in Bits di un Byte
        def ByteToString(self, BytesValue):
            BitsStringRappresentation = ""

            BitsStringRappresentation="{0:b}".format(BytesValue)

            return (BitsStringRappresentation)

            """
            for bIndex in range(0, 8):
                # One Bit Mask
                # 0000 0001
                # 0000 0010
                # ...
                BitValue = BytesValue & (1 << (7 - bIndex))

                if BitValue > 1:
                    BitsStringRappresentation = BitsStringRappresentation + "1"
                elif BitValue == 0:
                    BitsStringRappresentation = BitsStringRappresentation + "0"
                else:
                    BitsStringRappresentation = BitsStringRappresentation + " "
                # End if
            # End For
            return(BitsStringRappresentation)
            """
        # End Function

        # ----------------------------------------------------------------
        # Bytes To String
        # ----------------------------------------------------------------
        # Convert a List of Value [LSB, MSB] to a String
        def BytesToString(self, Value):
            Value_Str = ""
            if isinstance(Value, list):
                iNum = len(Value)
                print("SetRegisterValue: Num. Elements in the List:{0}".format(iNum))
                for i in range(iNum):
                    CurrElement = "{0:0>8b}".format(Value[iNum - 1 - i])
                    print("SetRegisterValue: Element {0} in the List is :{1}".format(iNum - 1 - i, CurrElement))
                    Value_Str = Value_Str + " " + CurrElement
            # End For
            else:
                Value_Str = "{0:0>8b}".format(Value)
            # End If

            return(Value_Str)
```



```
        #

    # End Class

    # ToDoList:
    # Come usare le variabili di Classe nelle fuzioni membro

```

14.13 adapter.py

```python
class Adapter:
    """
    Must implement ` decode()` and ` encode()`.
    """
    def  decode(self, value):
        raise NotImplementedError

    def  encode(self, value):
        raise NotImplementedError


class LookupAdapter(Adapter):
    """Adaptor with a dictionary of values.

    :param lookup table: A dictionary of one or more key/value pairs where the key
    is the human-readable value and the value is the bitwise register value

    """
    def   init  (self, lookup table, snap=True):
        self.lookup table = lookup table
        self.snap = snap

    def  decode(self, value):
        index = list(self.lookup table.values()).index(value)
        return list(self.lookup table.keys())[index]

    def  encode(self, value):
        if self.snap and type(value) in [int, float]:
            value = min(list(self.lookup table.keys()), key=lambda x: abs(x - value))
        return self.lookup table[value]


class U16ByteSwapAdapter(Adapter):
    """Adaptor to swap the bytes in a 16bit integer."""
    def _byteswap(self, value):
        return (value >> 8) | ((value & 0xFF) << 8)

    def _decode(self, value):
        return self._byteswap(value)

    def _encode(self, value):
        return self._byteswap(value)
```

14.14 _init_.py

```python
    version   = '0.0.5'

def  mask width(value, bit width=8):
    """Get the width of a bitwise mask

    ie: 0b000111 = 3
    """
    value >>=  trailing zeros(value, bit width)
    return value.bit length()


def  leading zeros(value, bit width=8):
    """Count leading zeros on a binary number with a given bit width

    ie: 0b0011 = 2

    Used for shifting around values after masking.
    """
    count = 0
    for   in range(bit width):
        if value & (1 << (bit width - 1)):
            return count
        count += 1
        value <<= 1
    return count


def  trailing zeros(value, bit width=8):
    """Count trailing zeros on a binary number with a given bit width

    ie: 0b11000 = 3

    Used for shifting around values after masking.
    """
    count = 0
    for _ in range(bit_width):
        if value & 1:
            return count
        count += 1
        value >>= 1
    return count


def _int_to_bytes(value, length, endianness='big'):
    try:
        return value.to_bytes(length, endianness)
    except AttributeError:
        output = bytearray()
        for x in range(length):
            offset = x * 8
            mask = 0xff << offset
            output.append((value & mask) >> offset)
        if endianness == 'big':
            output.reverse()
        return output


class MockSMBus:
    def __init__(self, i2c_bus):
        self.regs = [0 for _ in range(255)]

```



```python
    def write i2c block data(self, i2c address, register, values):
        self.regs[register:register + len(values)] = values

    def read i2c block data(self, i2c address, register, length):
        return self.regs[register:register + length]


class  RegisterProxy(object):
    """Register Proxy

    This proxy catches lookups against non existent get fieldname and set fieldname methods
    and converts them into calls against the device's get field and set field methods with
    the appropriate options.

    This means device.register.set field(value) and device.register.get field(value) will work
    and also transparently update the underlying device without the register or field objects
    having to know anything about how data is written/read/stored.

    """
    def   init  (self, device, register):
        object.  init  (self)
        self.device = device
        self.register = register

    def   getattribute  (self, name):
        if name.startswith("get "):
            name = name.replace("get ", "")
            return lambda: self.device.get field(self.register.name, name)
        if name.startswith("set_"):
            name = name.replace("set_", "")
            return lambda value: self.device.set_field(self.register.name, name, value)
        return object.__getattribute__(self, name)

    def write(self):
        return self.device.write_register(self.register.name)

    def read(self):
        return self.device.read_register(self.register.name)

    def __enter__(self):
        self.device.read_register(self.register.name)
        self.device.lock_register(self.register.name)
        return self

    def __exit__(self, exception_type, exception_value, exception_traceback):
        self.device.unlock_register(self.register.name)


class Register():
    """Store information about an i2c register"""
    def __init__(self, name, address, fields=None, bit_width=8, read_only=False, volatile=True):
        self.name = name
        self.address = address
        self.bit_width = bit_width
        self.read_only = read_only
```



```python
        self.volatile = volatile
        self.fields = {}

        for field in fields:
            self.fields[field.name] = field


class BitField():
    """Store information about a field or flag in an i2c register"""
    def   init  (self, name, mask, adapter=None, bit width=8, read only=False):
        self.name = name
        self.mask = mask
        self.adapter = adapter
        self.bit width = bit width
        self.read only = read only


class BitFlag(BitField):
    def   init  (self, name, bit, read only=False):
        BitField.  init  (self, name, 1 << bit, adapter=None, bit width=8, read only=read only)


class Device(object):
    def __init__(self, i2c_address, i2c_dev=None, bit_width=8, registers=None):
        self._bit_width = bit_width

        self.locked = {}
        self.registers = {}
        self.values = {}

        if type(i2c_address) is list:
            self._i2c_addresses = i2c_address
            self._i2c_address = i2c_address[0]
        else:
            self._i2c_addresses = [i2c_address]
            self._i2c_address = i2c_address

        self._i2c = i2c_dev

        if self._i2c is None:
            import smbus
            self._i2c = smbus.SMBus(1)

        for register in registers:
            self.locked[register.name] = False
            self.values[register.name] = 0
            self.registers[register.name] = register
            self.__dict__[register.name] = _RegisterProxy(self, register)

    def lock_register(self, name):
        self.locked[name] = True

    def unlock_register(self, name):
        self.locked[name] = False

    def read_register(self, name):
        register = self.registers[name]
        self.values[register.name] = self._i2c_read(register.address, register.bit_width)
        return self.values[register.name]
```



```python

    def write register(self, name):
        register = self.registers[name]
        return self. i2c write(register.address, self.values[register.name], register.bit width)

    def get addresses(self):
        return self. i2c addresses

    def select address(self, address):
        if address in self. i2c addresses:
            self. i2c address = address
            return True
        raise ValueError("Address {:02x} invalid!".format(address))

    def next address(self):
        next addr = self. i2c addresses.index(self. i2c address)
        next addr += 1
        next addr %= len(self. i2c addresses)
        self. i2c address = self. i2c addresses[next addr]
        return self. i2c address

    def get_field(self, register, field):
        register = self.registers[register]
        field = register.fields[field]

        if not self.locked[register.name]:
            self.read_register(register.name)

        value = self.values[register.name]

        value = (value & field.mask) >> _trailing_zeros(field.mask, register.bit_width)

        if field.adapter is not None:
            value = field.adapter._decode(value)

        return value

    def set_field(self, register, field, value):
        register = self.registers[register]
        field = register.fields[field]
        shift = _trailing_zeros(field.mask, register.bit_width)

        if field.adapter is not None:
            value = field.adapter._encode(value)

        if not self.locked[register.name]:
            self.read_register(register.name)

        reg_value = self.values[register.name]

        reg_value &= ~field.mask
        reg_value |= (value << shift) & field.mask

        self.values[register.name] = reg_value

        if not self.locked[register.name]:
            self.write_register(register.name)

    def get_register(self, register):
```

```
        register = self.registers[register]
        return self. i2c read(register.address, register.bit width)

    def  i2c write(self, register, value, bit width):
        values =  int to bytes(value, bit width // self. bit width, 'big')
        values = list(values)
        self. i2c.write i2c block data(self. i2c address, register, values)

    def  i2c read(self, register, bit width):
        value = 0
        for x in self. i2c.read i2c block data(self. i2c address, register,
        bit width // self. bit width):
            value <<= 8
            value |= x
        return value

```

14.15 Test Software

Al fine di poter estrarre i valori e convalidare la bontà del codice scritto si è realizzata una interfaccia per mezzo delle librerie *Qt* in grado di estrarre e visualizzare i valori presente nei registri dei sensori.

L'interfaccia si compone si una window con quattro schede facenti capo ai quattro sensori utilizzati nell'architettura.

- **BME680** Ambiental Sensor
- **VEML7700** Light Sensor
- **AS7262** Spectrum Sensor
- **VEML6075** UV Sensor

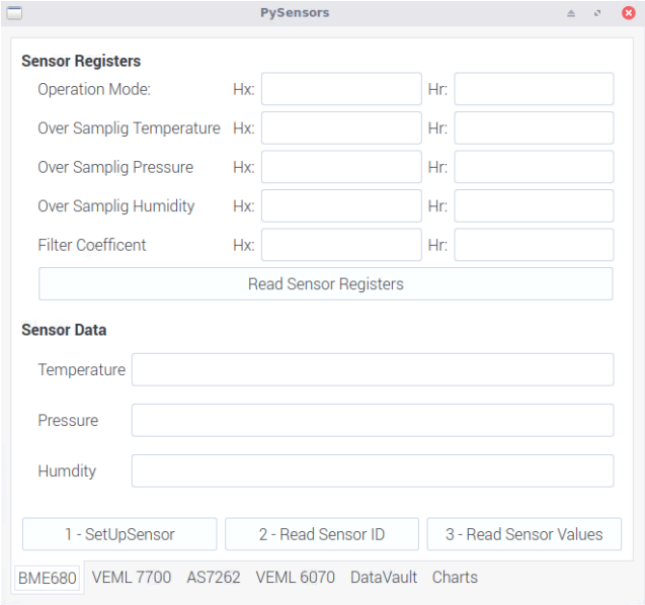


Figure 14.7: BOX 1

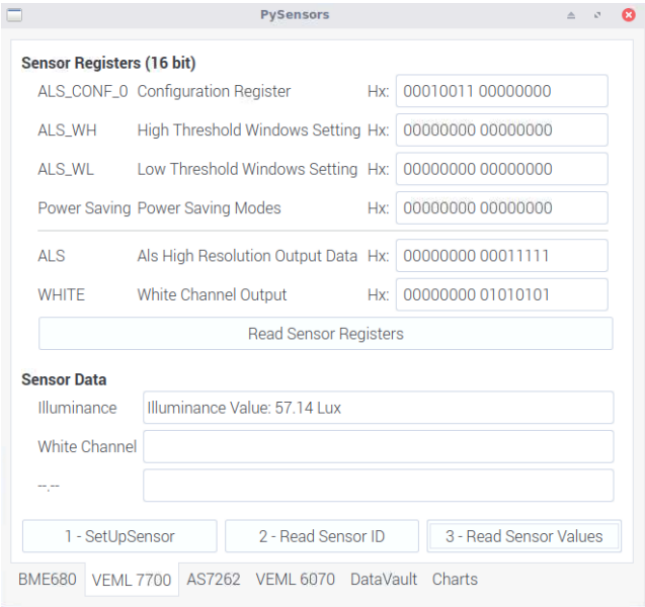


Figure 14.8: BOX 2

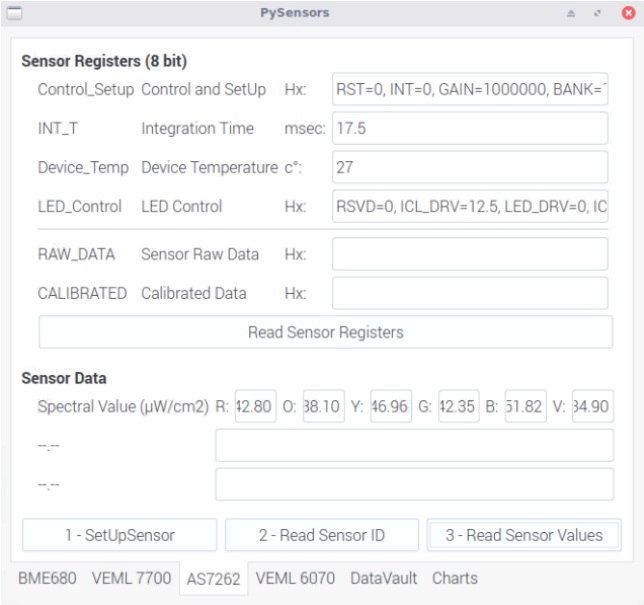


Figure 14.9: BOX 3

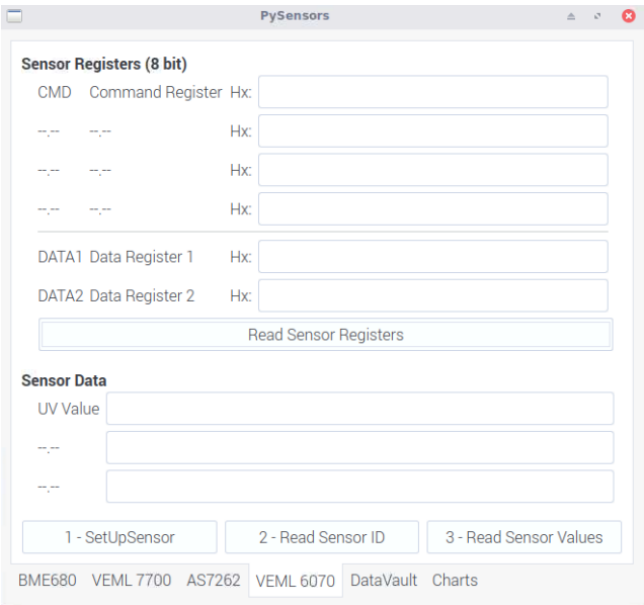


Figure 14.10: BOX 4

Chapter 15

Structure

Per quanto riguarda la struttura, siccome il dispositivo opera in ambiente aperto, occorre che essa sia resistente agli agenti atmosferici in generale. Sarebbe quindi interessante optare per connessioni di tipo IP68 sia a livello di alimentazione che trasmissione dati.

Da ridurre al minimo sono quindi le aperture e le possibilità conseguenti di infiltrazione di acqua.

La struttura integralmente vuole rispettare le dimensioni minime possibili e quindi a fronte di una PCB di 57mm x 83mm si vuole cercare di mantenere grosso modo lo stesso ordine di misura.

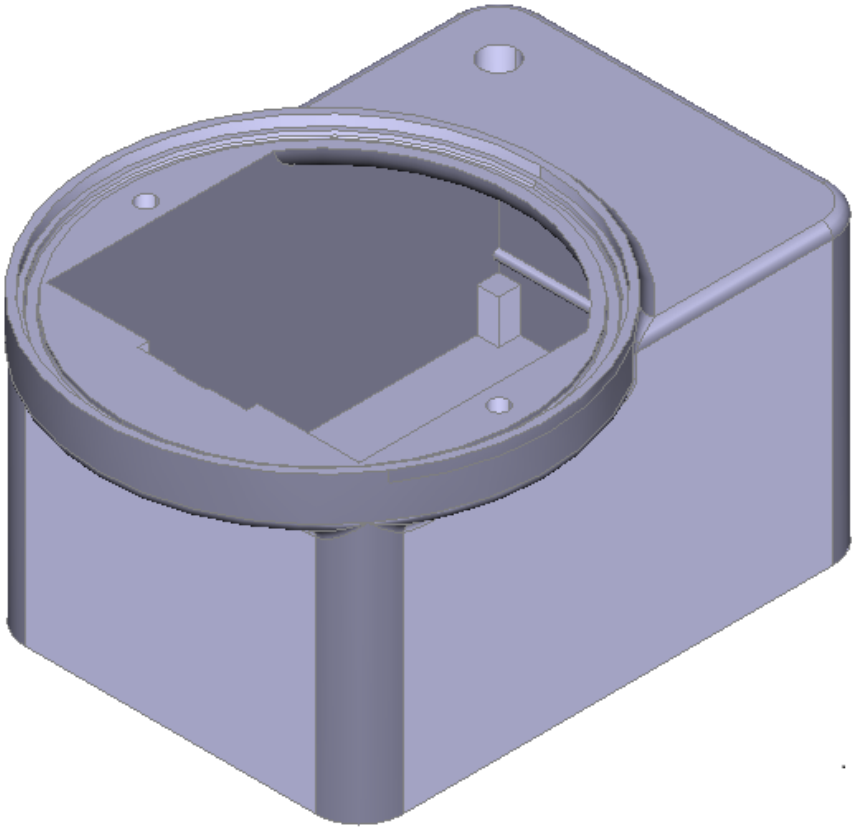


Figure 15.1: General View

15.1 Lenticolar Dome

La luce e l'irraggiamento solare devono poter essere raccolti e analizzati senza attenuazione ne alterazione,quindi si rivela necessario un lenticolare a cupola che opportunamente alloggiato riesca a garantire l'omogeneità della luce all'interno del dispositivo.

Per questo motivo ci si serve di un lenticolare a cupola che ha una risposta piatta lungo l'intero spettro solare *(390nm-1100nm)* e ripari i sensori dagli agenti esterni.

Umidità, raggi UV, neve e pioggia sono infatti cause di possibili malfunzionamenti di un dispositivo non correttamente isolato.

Per questo motivo la struttura presenta al di sotto della lente un O-Ring di materiale plastico in grado di mantenere umidità e infiltrazioni d'acqua lontane dal circuito.

Figure 15.2: Dome

Inoltre, la sua forma a cupola, permette di ricevere l'energia dalla fonte luminosa in modo che sia isotropica.

15.2 Wifi Antenna

Seguendo la strada descritta in precedenza di una comunicazione Wireless si è reso necessario integrare nel design una antenna in modo che possa essere massimizzato il raggio di azione del dispositivo, in quanto esso sicuramente sarà ubicato in una locazione lontana dal *Gateway Wifi*.

Per questo motivo al posto di un controller Wifi con scheda integrata si opta per una antenna esterna.

L'antenna dà sull'esterno del case per mezzo di una connessione SMA J/R qualificata IP66.

Figure 15.3: Antenna

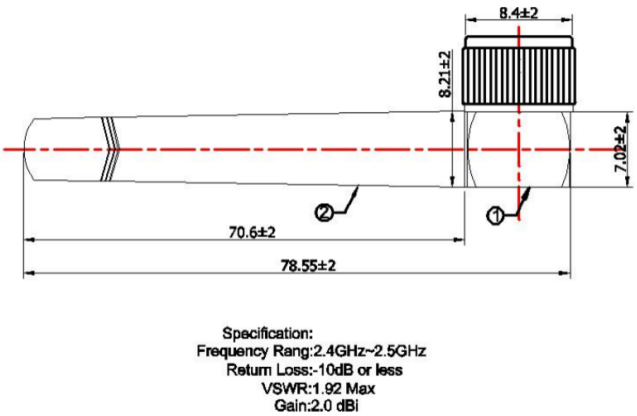
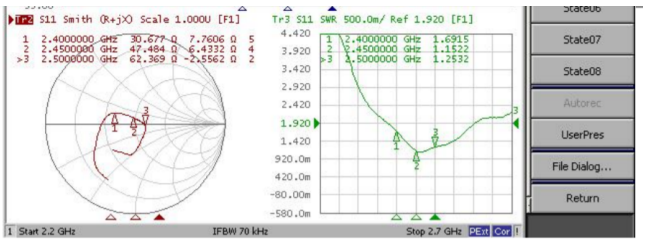
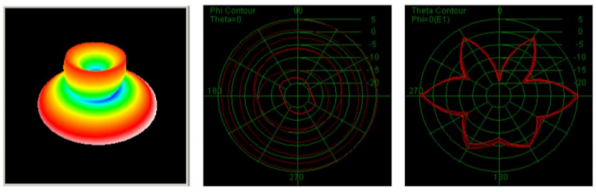


Figure 15.4: Antenna Dimensional Draw



2D、3DRaditation Pattern:



2.4GHz Gain:1.87 Efficiency:93.5%

Figure 15.5: 3D Radiation Pattern

15.3 Antenna Cable

La connessione tra il modulo Wifi e l'antenna stessa è permessa grazie all'uso di un cavo dedicato con le seguenti caratteristiche sulle quali non staremo a dilungarci molto.

Specification

1.Electrical Properties

1.1 Frequency Range----------DC-3GHz

1.2 Impedance---------- 50 Ω

1.3 VSWR---------- ≤1.5

1.4 Cable Loss----------0.1dB / m Max @ 100MHz

1.5 Radiation---------- Omni-directional

1.6 Admitted Power---------- 2W

1.7 Cable---------- Φ1.37mm Gray

1.8 Insulation Resistance----------1000M ohm

2. Physical Properties

2.1 Connector---------- SMA J/R

2.2 Operating Temp---------- -10℃~+60℃

2.3 Storage Temp---------- -10℃~+70℃

2.4 Cable Color---------- Gray

Figure 15.6: Attenuation vs frequency

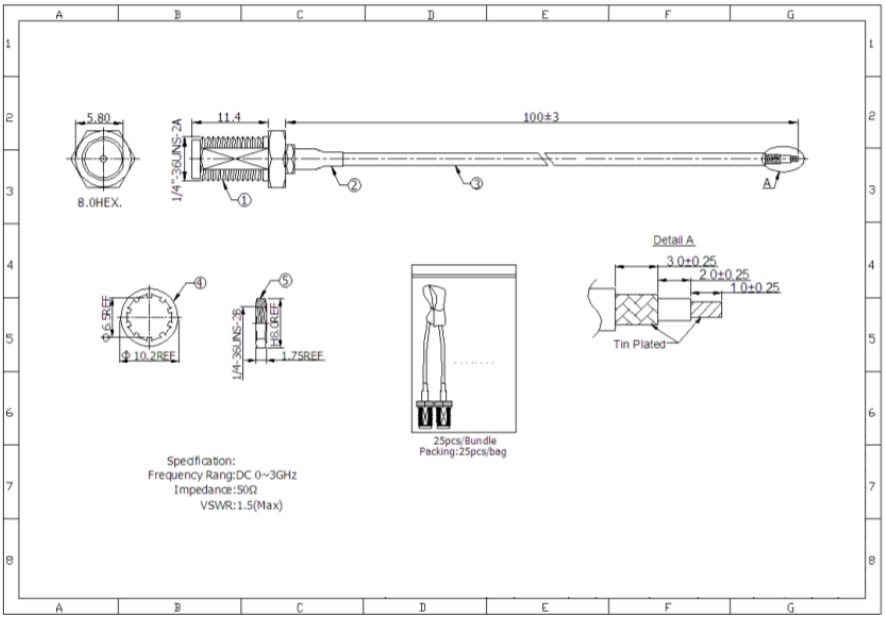


Figure 15.7: Radiation Pattern

15.4 Power Connector e Plug

Al fine anche in questo caso di garantire il massimo isolamento tra il modulo e il mondo esterno è stato scelto un connettore di tipo IP68 USB A, in modo da poter contare sulla reperibilità, semplicità e economicità della connessione senza nessun contro.

Esso ha la funzione di portare la tensione di alimentazione dal dispotivo di supply al modulo.

Figure 15.8: Plug Front

Figure 15.9: Plug Rear

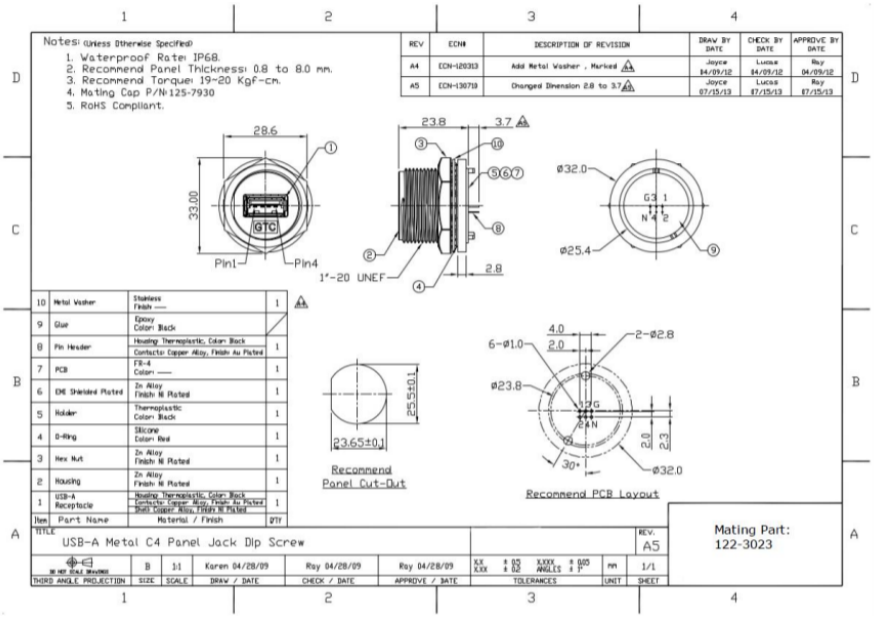


Figure 15.10: Plug Dimensional Draw

Nel caso in cui il dispositivo venga riposto o debba essere scollegato si può coprire la connessione con un *plug* di tipo IP68.

Figure 15.11: Cap Front

Figure 15.12: Cap Rear

15.5 3D Model

La realizzazione è affidata alla conoscenza preliminare del software di modellazione 3d *Catia*.

A livello generale la struttura è stata pensata,essendo un prototipo, per essere stampata con una stampante 3d, dapprima con un prototipo in PLA e successivamente con un prototipo in LongChain Polyamide nero resistente ai raggi UV.

15.5.1 Material

Per il prototipo iniziale è stato scelto un materiale economico giusto per verificare ingrombri ed evidenziare eventuali problematiche non calcolate.

TREED

FILAMENT DATASHEET

MATERIAL: Longchain PA

WE CARE ABOUT YOUR PRINTS

Suggested printing specs.	Value	Unit	Standard
Extruder Temperature	240° +/-10°	°C	Internal
Bed Temperature	95° +/-10°	°C	Internal
Print speed	40-50	mm/s	Internal
Fan	0-30	%	Internal
Minimum Nozzle Diameter	0,35	mm	Internal

Material Properties	Value	Unit	Standard
Density	1	g/cm^3	ISO 1183
Hardness	75	sh/D	ISO 868
Tensile Modulus	1440	MPa	ISO 527
Melting Point	180°	°C	ISO 11357
Charpy impact strength, 23°C	NO BREAK	kJ/m^2	ISO 179
Charpy impact strength, -30°C	NO BREAK	kJ/m^2	ISO 179
Yield Stress	43	MPa	ISO 527-2

Figure 15.13: LPA Data

Mentre per quanto riguarda il dispotivo vero e proprio è stato utilizzato un materiale innovativo *Z-ASA Pro* dedicato all'utilizzo in ambienti esterni e a contatto degli agenti atmosferici.

Z-ASA Pro infatti è un materiale termoplastico durevole e mostra una resistenza superiore rispetto ai normali ABS verso fattori esterni, tra cui radiazioni UV e condizioni meteorologiche estreme. Utilizzando Z-ASA Pro, è possibile stampare modelli e prototipi in 3D che resistono al cambiamento di temperatura, umidità o luce solare, mantenendo la loro forma iniziale e un'estetica eccellente.

TECHNICAL DATA SHEET

Date of issue: 01.06.2017 | Update: 03.01.2018 | Version: 2.00

Z-ASA Pro

Mechanical Properties	Metric	English	Test Method
Tensile Strength	24.21 MPa	3510 psi	ISO 527:1998
Breaking Stress	21.94 MPa	3180 psi	ISO 527:1998
Elongation at max Tensile Stress	2.64%	2.64%	ISO 527:1998
Elongation at Break	2.76%	2.76%	ISO 527:1998
Bending Stress	45.60 MPa	6610 psi	ISO 178:2011
Flexural Modulus	1.36 GPa	197 ksi	ISO 178:2011
Izod Impact, Notched	3.81 kJ/m²	1.81 ft-lb/in²	ISO 180:2004
Thermal Properties	**Metric**	**English**	**Test Method**
Glass Transition Temperature	80.99° C	178° F	ISO 11357-3:2014
Other Properties	**Metric**	**English**	**Test Method**
Melt Flow Rate	74.91 g/10 min Load 10 kg Temperature 220° C	0.165 lb/10 min Load 22 lb Temperature 428° F	ISO 1133:2006
Specific Density	1.176 g/cm³	9.81 lb/gal	ISO 1183-3:2003
Shore Hardness (D)	68.0	68.0	ISO 868:1998

Figure 15.14: Z Asa Pro Data

15.5.2 Views

Il sensore ambientale *BME680* necessita di una apertura esterna in modo che possa comunicare direttamente con l'ambiente circostante.

La catena di sensori relativi alla luminosità invece è disposta al di sotto del *Dome* che sta a protezione degli stessi.

Si è scelto di non avere aperture supplementari se non quelle strettamente necessarie quali:

- Usb Plug [Figure 18.16]
- Dome Hole [Figure 18.13]
- Antenna Hole [Figure 18.13,18.18]
- Bme680 Sensor Hole [Figure 18.14]

E' possibile apprezzare la struttura dalle visuali isometriche standard.

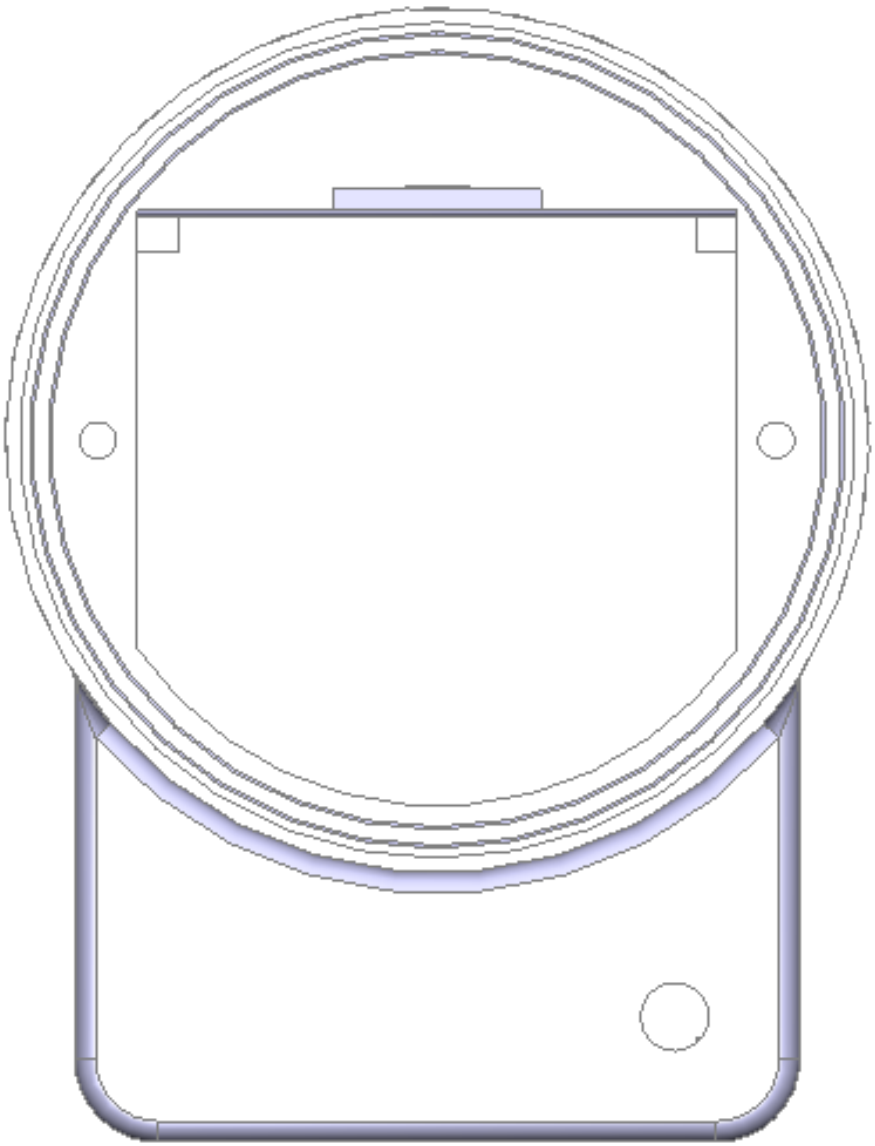


Figure 15.15: Top

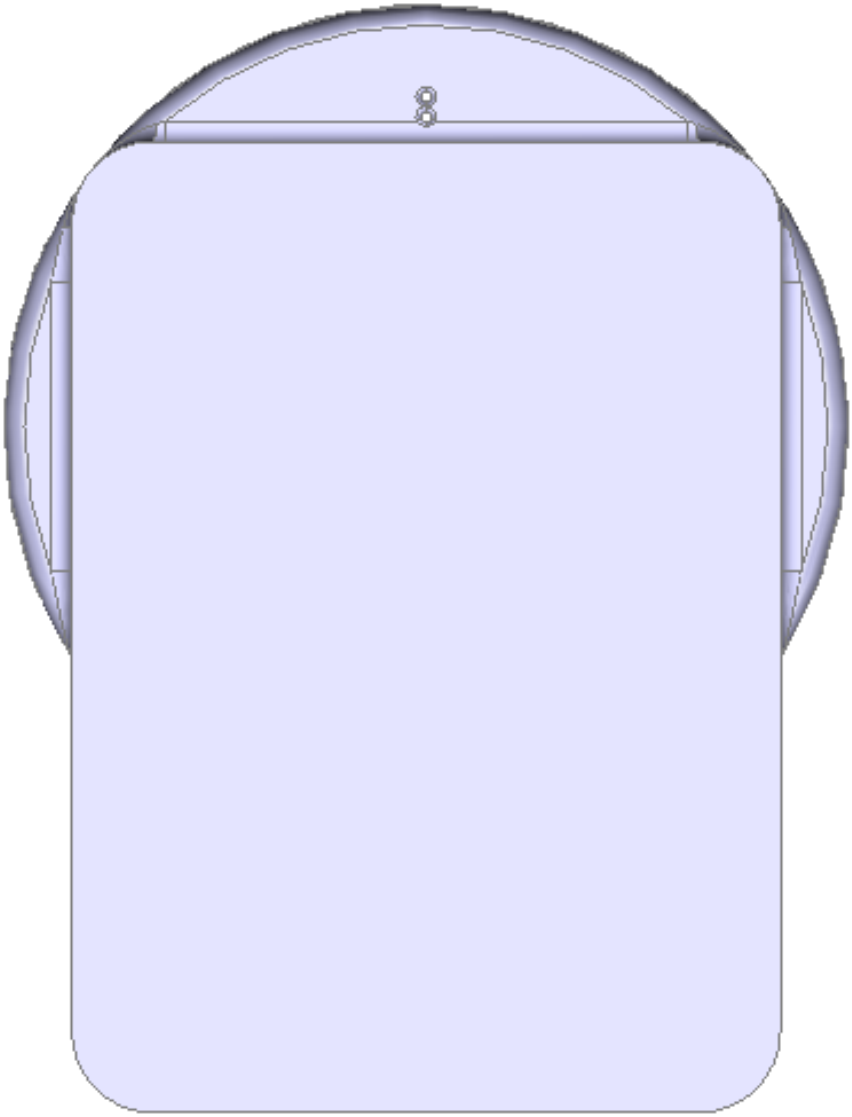


Figure 15.16: Rear

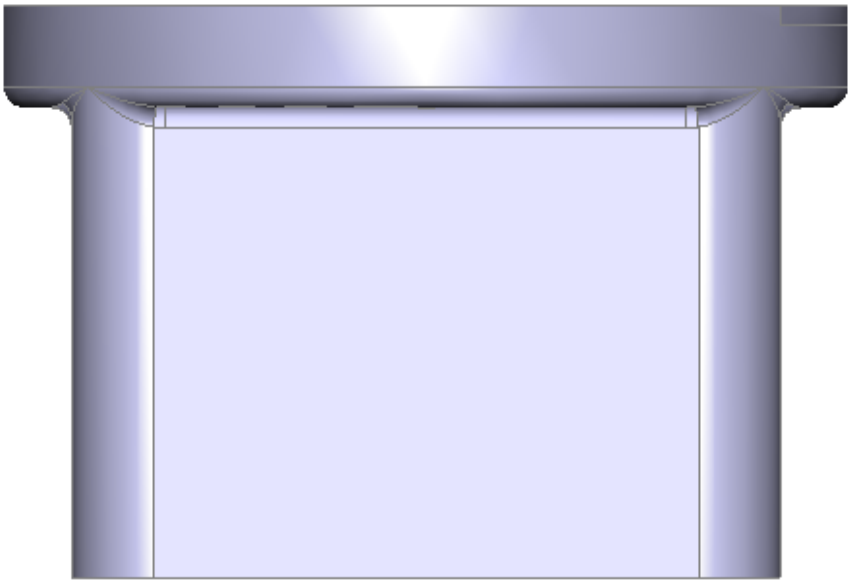


Figure 15.17: Left

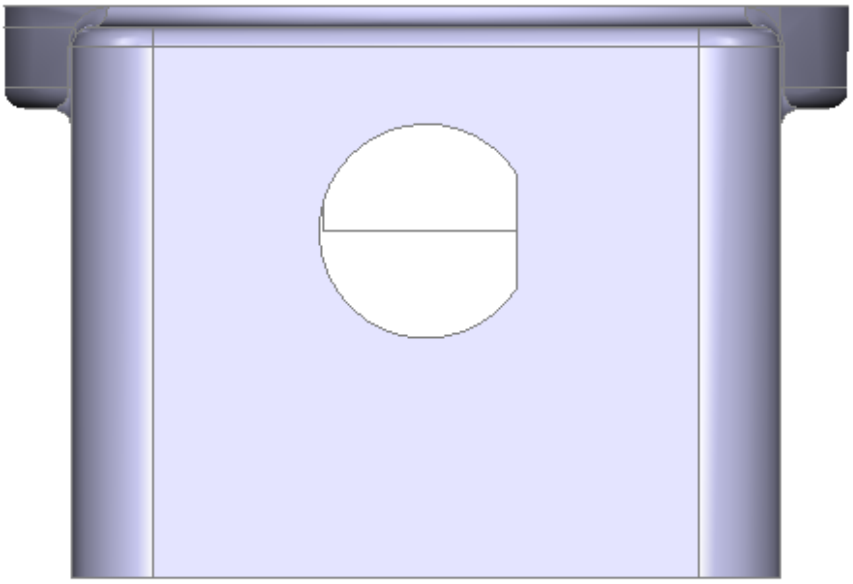


Figure 15.18: Right

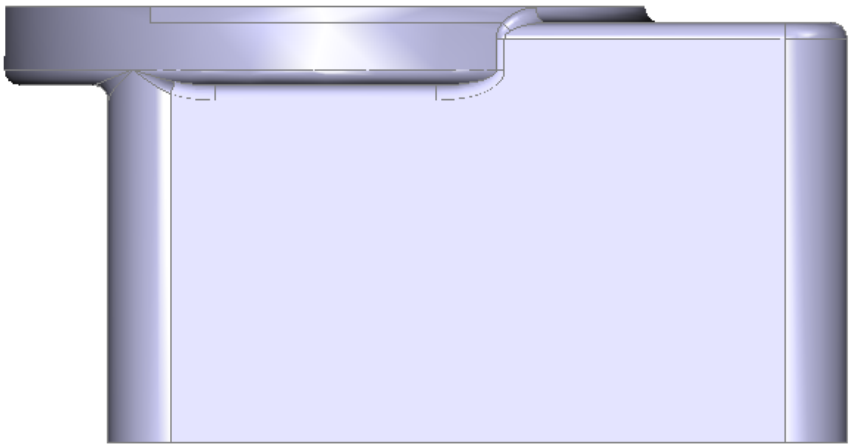


Figure 15.19: Front

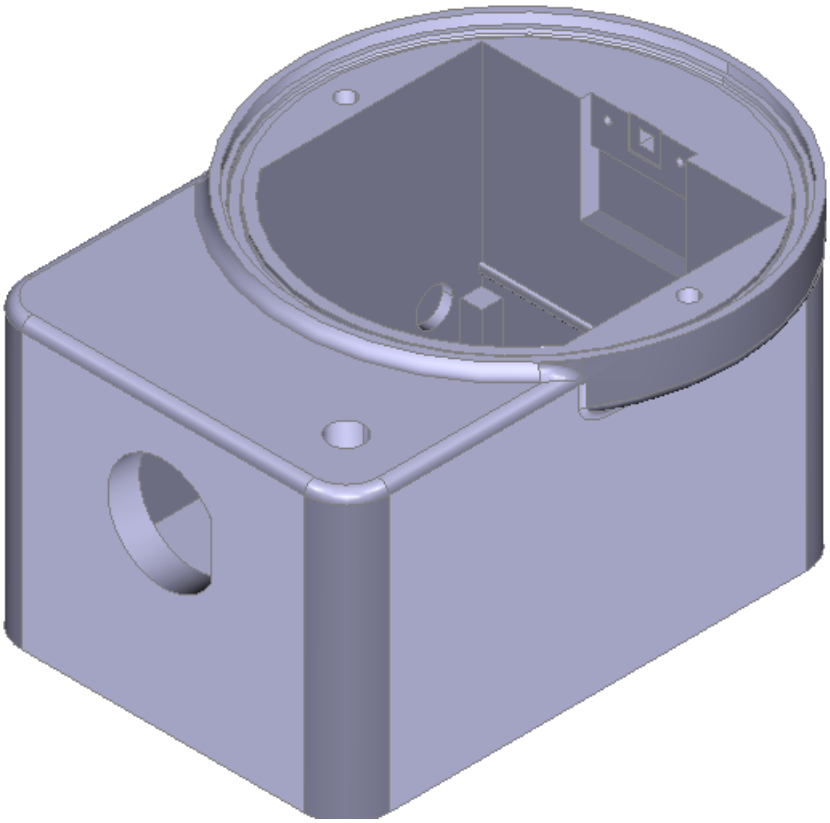


Figure 15.20: N/E

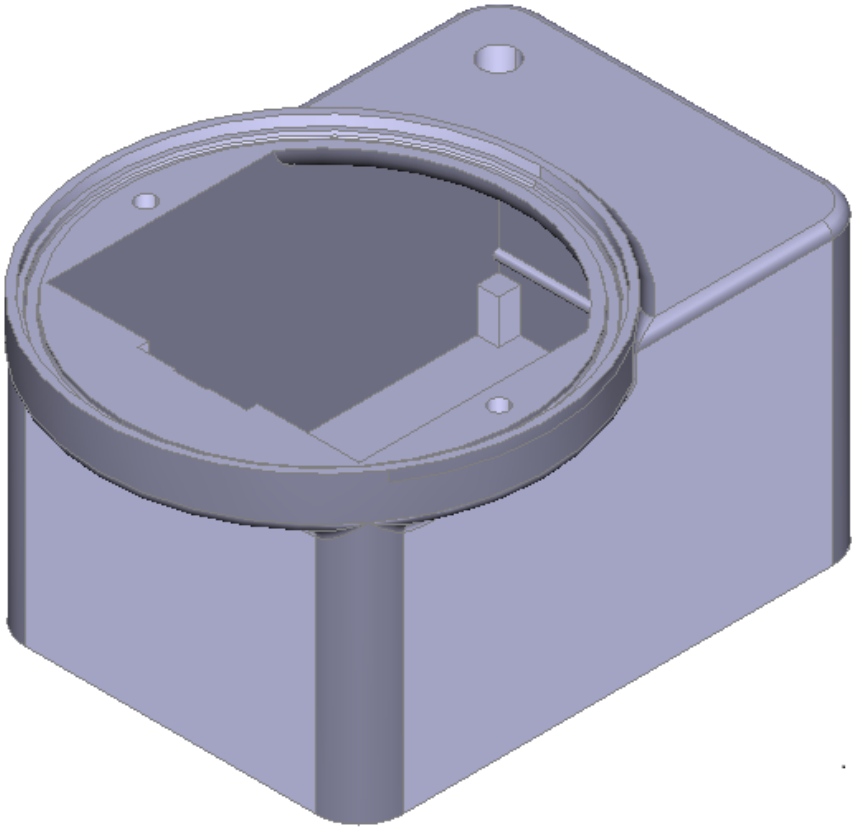


Figure 15.21: S/W

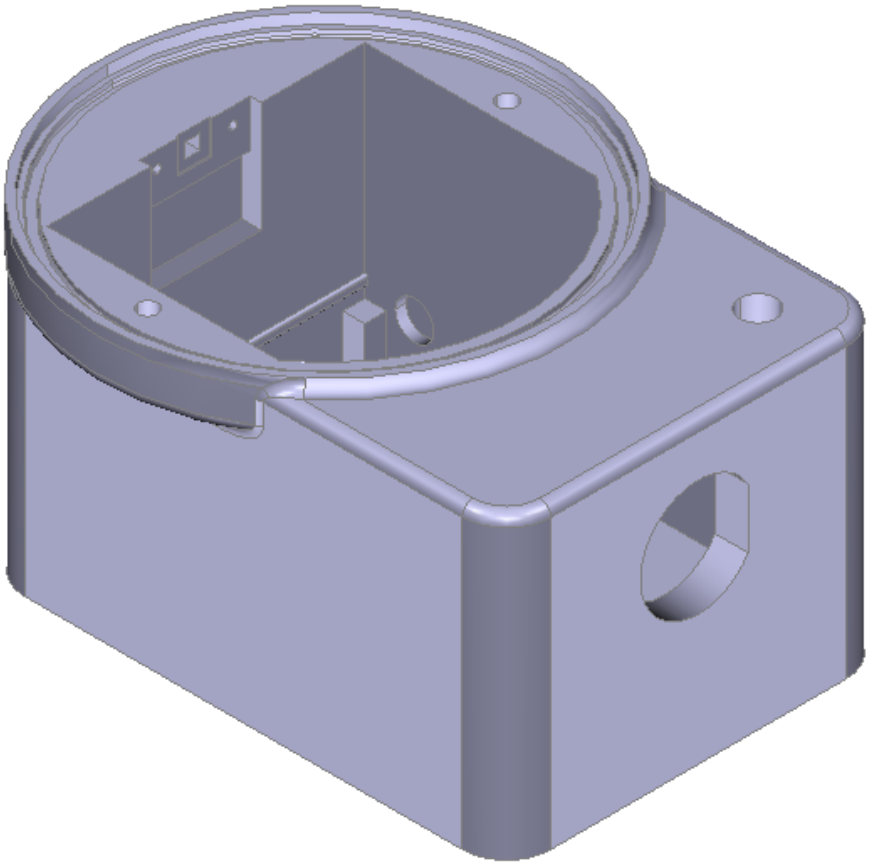


Figure 15.22: S/E

15.6 PLA-Prototipe

La realizzazione del prototipo è affidato a un materiale economico di tipo PLA per mezzo della stampa 3d in scala 1:1.

Figure 15.23: Pla Prototipe 1

Figure 15.24: Pla Prototipe 2

15.7 Final Object

Il risultato è il prodotto finito in *Z-ASA Pro* di cui è stato scelto il color nero.

Figure 15.25: Z Asa 1

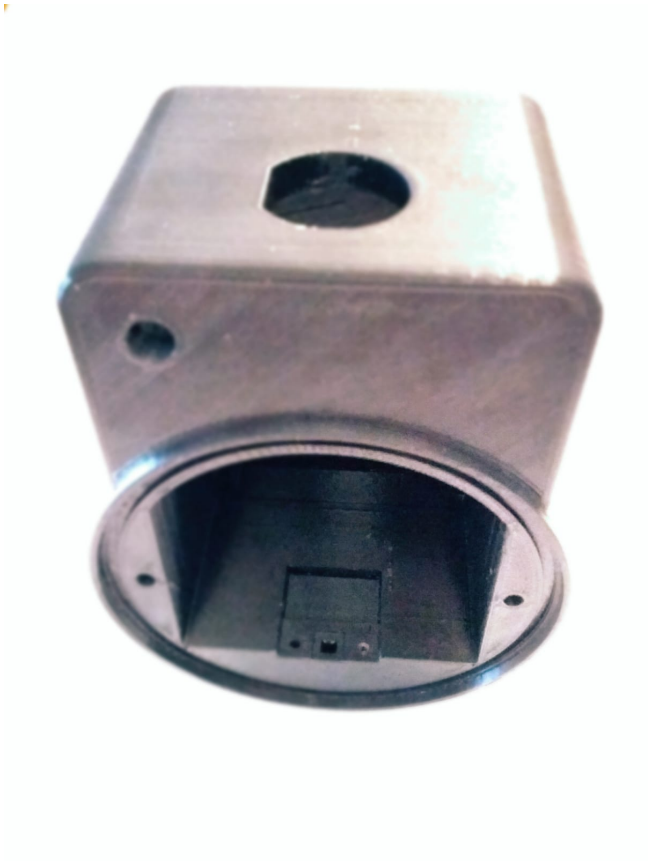


Figure 15.26: Z Asa 2

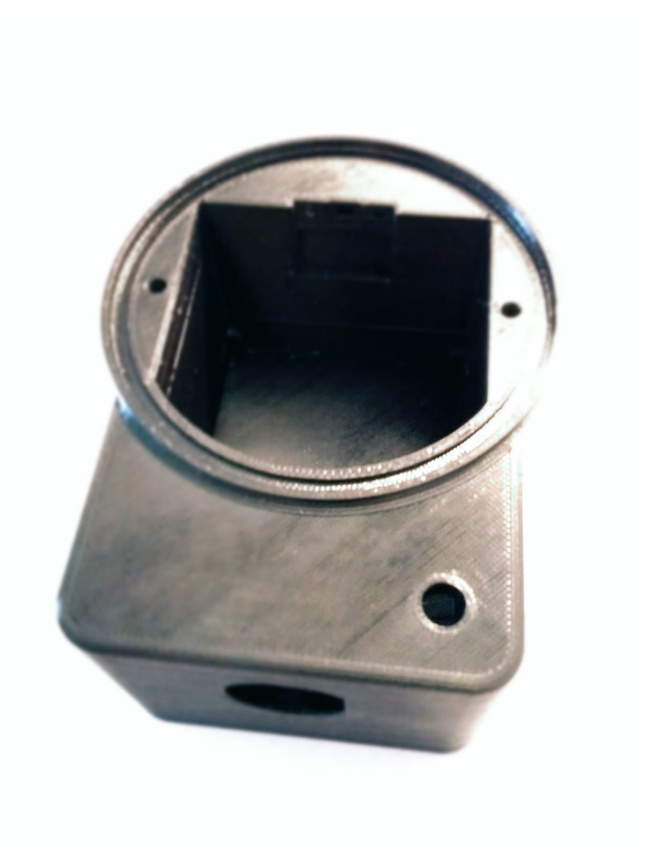


Figure 15.27: Z Asa 3

Chapter 16

DataLogging

Il dispositivo posto all'esterno e montato nella sua integrità è in grado di rilevare i fenomeni che sono stati discussi ampiamente nel documento.

Una volta inizializzato e collegato in remoto, il dispositivo, ha raccolto una serie di dati per quattro giorni consecutivi.

Tramite SQLite è stato creata una struttura ,ad hoc da 21 campi, che contiene i dati rilevati sotto la forma seguente:

'IdSample' Integer
'Date' Testual [dd/mm/aa]
'Year' Integer
'Month' Integer
'Day' Integer
'TimeStamp' Testual
'Hour' Integer
'Minute' Integer
'Second' Integer
'Temperature' Integer
'Pressure' Integer
'Humidity' Integer
'Illuminance' Integer
'White' Integer
'RedChannel' Integer
'OrangeChannel' Integer
'YellowChannel' Integer
'GreenChannel' Integer
'BlueChannel' Integer
'VioletChannel' Integer
'UVChannel' Integer

Se vogliamo parlare in ottica sensoristica,ossia di cosa viene restituito dai vari sensori, la suddivisione dei frame può essere stilata per tipo di sensore.

BME680
'Temperature' 'Pressure' 'Humidity'

VEML7700
'Illuminance' 'White'

VEML 6075
'UVChannel'

AS7262
'RedChannel' 'OrangeChannel' 'YellowChannel' 'GreenChannel' 'BlueChannel' 'VioletChannel'
Una volta raccolti i dati possono essere esportati tramite file CSV ed elaborati.

Di seguito un esempio della struttura del file DataVault con i capi e i valori numerici contenuti

VarName1	VarName2	VarName3	VarName4	VarName5	VarName6	VarName7	VarName8	VarName9	VarName10	VarName11
Number	Datetime	Number	Number	Number	Datetime	Number	Number	Number	Number	Number
5675	2019-08-14	2019	8	14	19:34:27	19	34	27	38.38	969.3
5676	2019-08-14	2019	8	14	19:34:57	19	34	57	38.66	969.31
5677	2019-08-14	2019	8	14	19:35:27	19	35	27	38.94	969.29
5678	2019-08-14	2019	8	14	19:35:57	19	35	57	39.23	969.3
5679	2019-08-14	2019	8	14	19:36:27	19	36	27	39.47	969.27
5680	2019-08-14	2019	8	14	19:36:57	19	36	57	39.76	969.28
5681	2019-08-14	2019	8	14	19:37:27	19	37	27	40.04	969.29
5682	2019-08-14	2019	8	14	19:37:57	19	37	57	40.33	969.29
5683	2019-08-14	2019	8	14	19:38:27	19	38	27	40.56	969.32
5684	2019-08-14	2019	8	14	19:38:57	19	38	57	40.75	969.31
5685	2019-08-14	2019	8	14	19:39:27	19	39	27	40.96	969.34
5686	2019-08-14	2019	8	14	19:39:57	19	39	57	41.17	969.33
5687	2019-08-14	2019	8	14	19:40:27	19	40	27	41.38	969.35
5688	2019-08-14	2019	8	14	19:40:57	19	40	57	41.59	969.38
5689	2019-08-14	2019	8	14	19:41:27	19	41	27	41.74	969.36
5690	2019-08-14	2019	8	14	19:41:57	19	41	57	41.87	969.38
5691	2019-08-14	2019	8	14	19:42:27	19	42	27	42	969.39
5692	2019-08-14	2019	8	14	19:42:57	19	42	57	42.07	969.38
5693	2019-08-14	2019	8	14	19:43:27	19	43	27	42.07	969.37
5694	2019-08-14	2019	8	14	19:43:57	19	43	57	42.11	969.38
5695	2019-08-14	2019	8	14	19:44:27	19	44	27	42.15	969.4
5696	2019-08-14	2019	8	14	19:44:57	19	44	57	42.15	969.4
5697	2019-08-14	2019	8	14	19:45:27	19	45	27	42.18	969.41
5698	2019-08-14	2019	8	14	19:45:57	19	45	57	42.18	969.43
5699	2019-08-14	2019	8	14	19:46:27	19	46	27	42.13	969.42
5700	2019-08-14	2019	8	14	19:46:57	19	46	57	42.04	969.44

Figure 16.1: SQL 1-11

VarName12	VarName13	VarName14	VarName15	VarName16	VarName17	VarName18	VarName19	VarName20	VarName21
Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
15.63	3686.4	5244	2889.59	3479.43	3479.43	4601.06	2328.36	2443.32	4.78
15.47	3581.34	5047	2879.72	3462.38	3462.38	4563.16	2312.24	2423.54	4.73
15.61	3496.55	4886	2871.49	3447.34	3447.34	4525.26	2293.81	2404.93	4.67
15.28	3428.35	4748	2864.08	3436.31	3436.31	4496.28	2279.99	2387.48	4.67
15.18	3365.68	4640	2845.97	3416.25	3416.25	4456.16	2259.27	2368.86	4.56
14.92	3332.51	4615	2828.69	3382.16	3382.16	4390.4	2230.48	2339.77	4.44
14.78	3310.39	4544	2804.82	3346.06	3346.06	4323.52	2193.63	2309.52	4.28
14.91	3293.8	4517	2777.66	3309.97	3309.97	4259.99	2162.54	2280.44	4.22
14.64	3297.48	4499	2757.91	3275.87	3275.87	4206.49	2136.06	2253.68	4.16
14.71	3220.07	4460	2723.34	3227.74	3227.74	4130.7	2096.9	2218.77	4.05
15.03	3194.27	4430	2691.25	3181.62	3181.62	4062.71	2064.66	2187.36	3.99
15.1	3155.56	4407	2665.73	3147.53	3147.53	4017.01	2040.48	2161.76	3.94
14.78	3124.22	4366	2632.81	3094.38	3094.38	3943.44	2008.24	2129.18	3.83
14.83	3085.52	4311	2595.78	3036.22	3036.22	3860.96	1970.24	2091.95	3.83
15.12	3032.06	4263	2569.44	3005.14	3005.14	3821.95	1955.27	2069.84	3.83
15.25	2985.98	4208	2538.99	2964.03	2964.03	3762.88	1928.78	2041.92	3.71
15.22	2950.96	4156	2485.49	2886.82	2886.82	3645.85	1873.51	1994.22	3.49
14.99	2897.51	4086	2422.12	2802.59	2802.59	3518.78	1807.87	1943.02	3.32
15.67	2844.06	4035	2371.09	2732.4	2732.4	3415.12	1760.66	1898.81	3.26
15.68	2807.19	3979	2330.77	2683.27	2683.27	3348.25	1730.72	1868.56	3.21
15.61	2742.68	3904	2278.92	2623.11	2623.11	3270.23	1693.87	1833.66	3.21
16.01	2678.17	3818	2217.19	2534.87	2534.87	3148.74	1635.15	1783.63	3.04
15.83	2608.13	3721	2126.66	2412.54	2412.54	2970.4	1546.48	1713.82	2.76
16.23	2538.09	3618	2015.55	2242.07	2242.07	2710.7	1418.66	1618.41	2.36
16.02	2455.14	3492	1910.21	2088.66	2088.66	2481.09	1301.21	1529.99	2.14
16.57	2375.88	3370	1822.15	1966.33	1966.33	2316.13	1220.6	1463.67	1.91

Figure 16.2: SQL 12-21

Nella giornata del 14 Agosto sono stati raccolti tutti i dati, relativi ai fenomeni esogeni posizionando, il dispositivo sul tetto della abitazione.

Ne risulta l'acquisizione completa tranne che per quanto riguarda il canale green del chip AS7262 che presenta un disturbo di tipo saturazione.
La misura è stata prontamente filtrata.

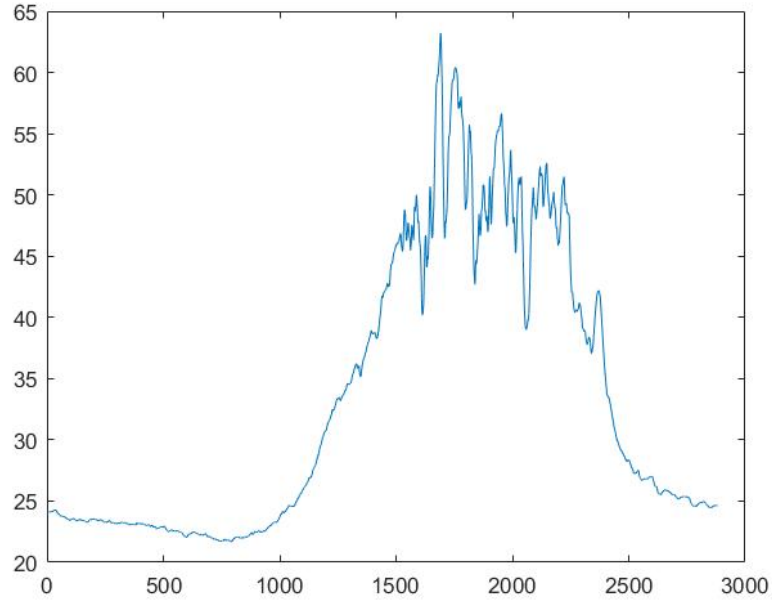


Figure 16.3: TEMPERATURE

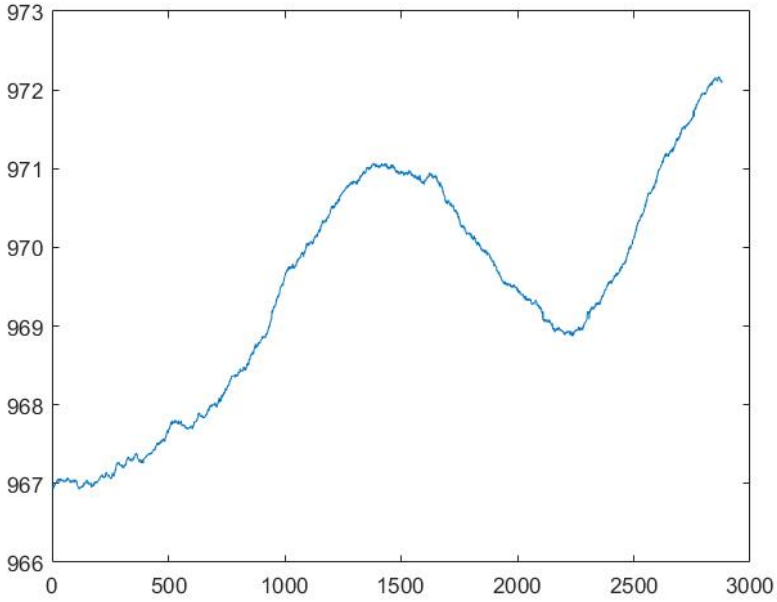


Figure 16.4: PRESSURE

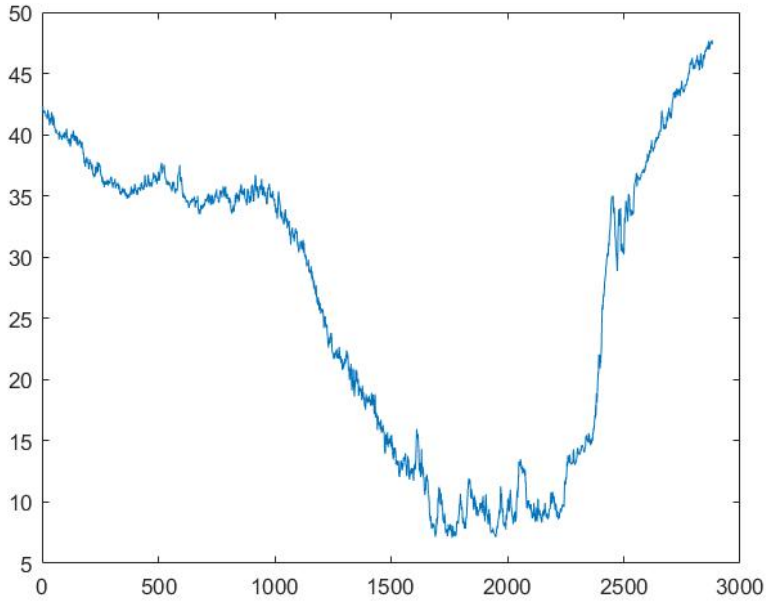


Figure 16.5: RELATIVE HUMIDITY

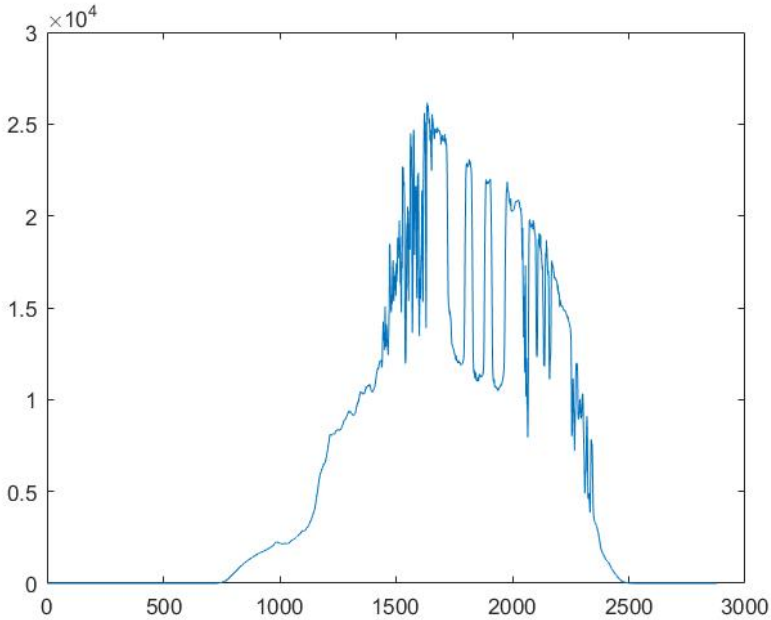


Figure 16.6: VEML 6070 LUX NORMAL CHANNEL

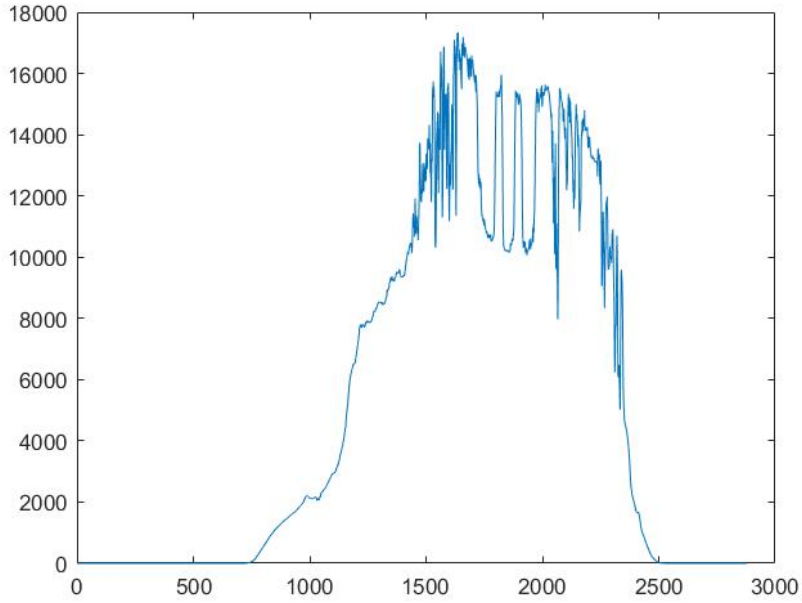


Figure 16.7: VEML7700 WHITE CHANNEL

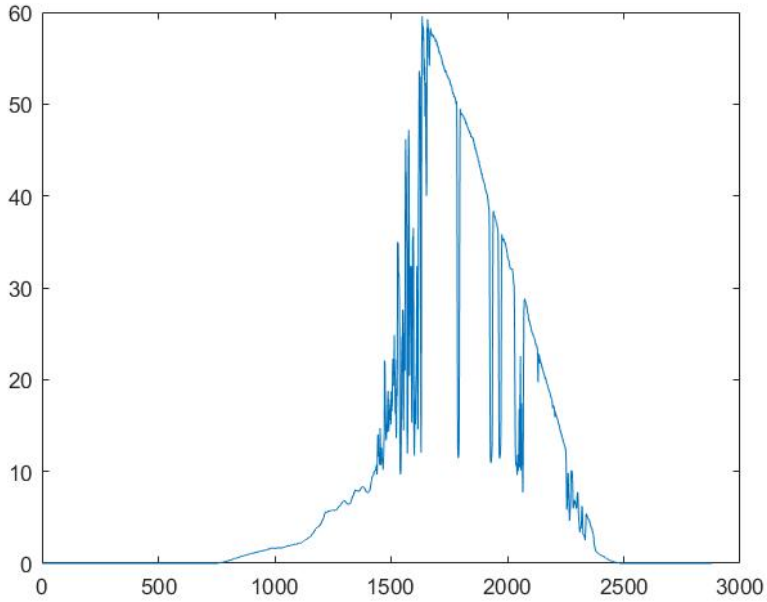


Figure 16.8: VEML 6075 UV CONTENT

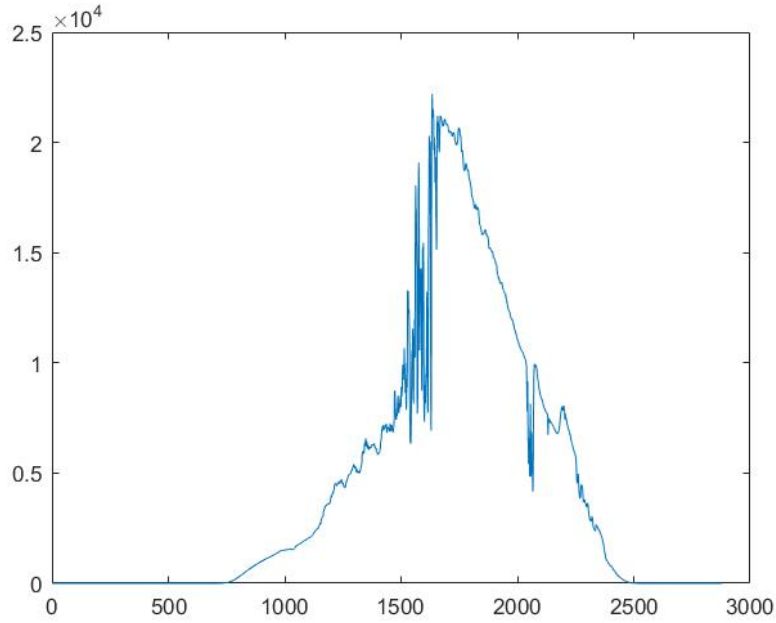


Figure 16.9: AS7262 VIOLET CHANNEL

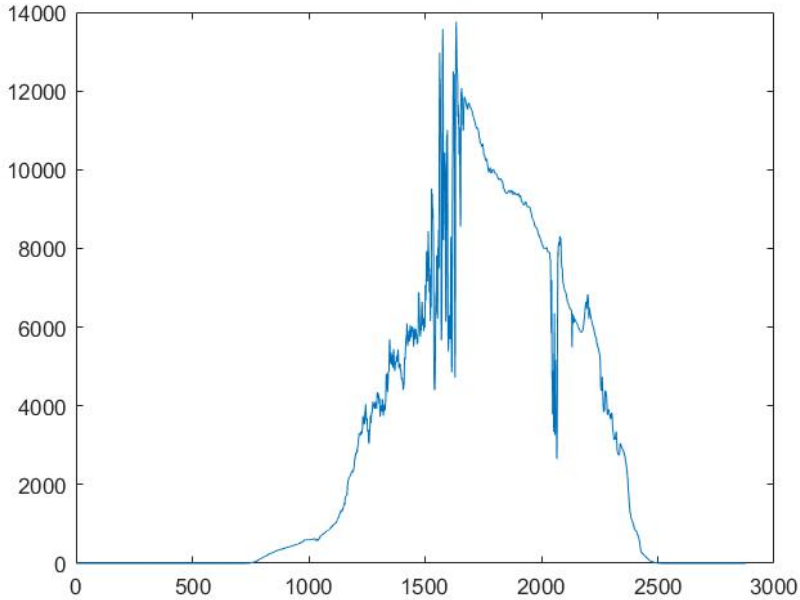


Figure 16.10: AS7262 RED CHANNEL

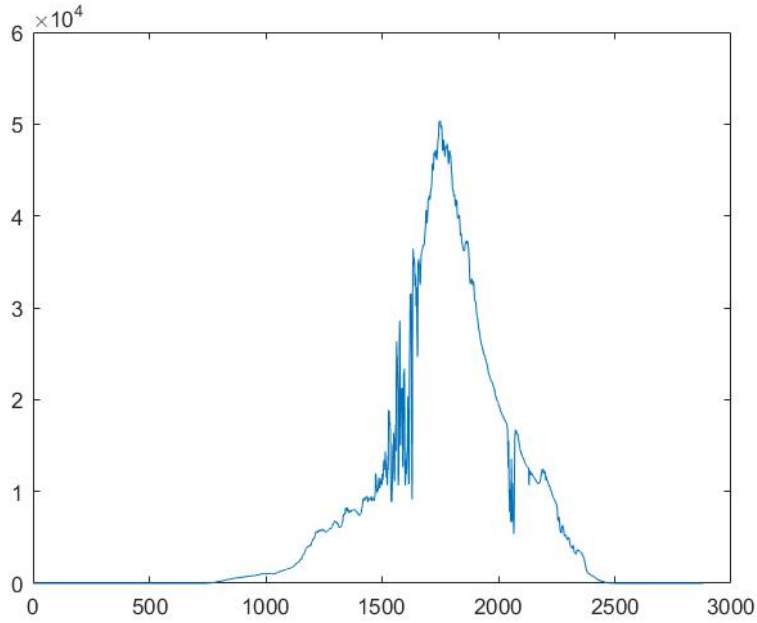


Figure 16.11: AS7262 ORANGE CHANNEL

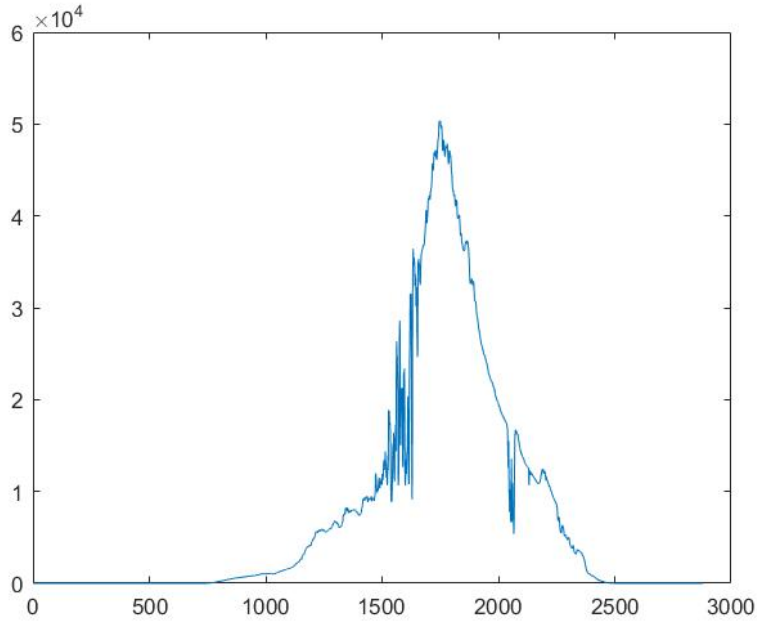


Figure 16.12: AS7262 YELLOW CHANNEL

Il canale verde subisce spesso degli errori di comunicazione imputabili al chip spesso e come raffigurato in rosso porta il valore a saturazione. I valori sono filtrati e interpolati linearmente tramite spline.

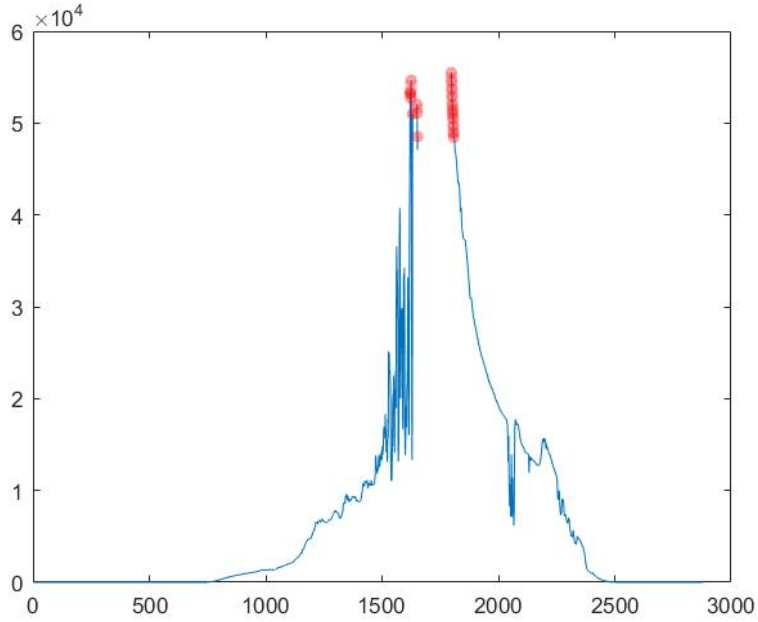


Figure 16.13: AS7262 GREEN CHANNEL - Not Filtered

Al termine della operazione di filtraggio i dati sono i seguenti:

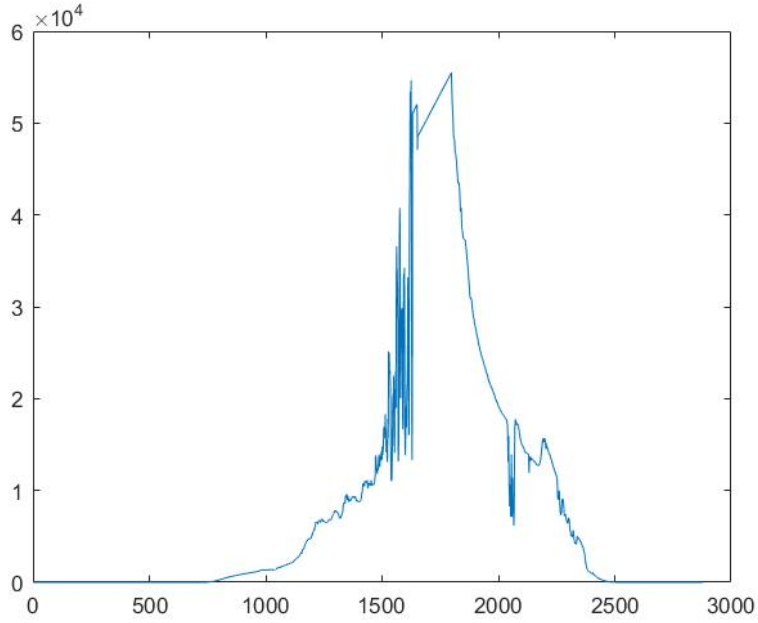


Figure 16.14: AS7262 GREEN CHANNEL

Il dato che sembra presentare una deviazione rispetto ai valori reali è quello della temperatura che ha una dinamica espansa soprattutto quando il sole illumina direttamente il dispositivo, creando così un microclima che non è quello reale.

Detto questo, le temperature minime sono corrette, mentre le massime subiscono un innalzamento rispetto alla vera temperatura massima della giornata.

Per ovviare a cià risulta necessario creare una funzione di correzione in modo che possa riportare la temperatura massima nel range corretto.

Chapter 17

Data Visualization using Hi-Charts

La visualizzazione dei dati raccolti e delle future previsioni è affidato al software Highcharts rilasciato nel 2009 e scritto in Java.

Ogni volta che si pubblica è necessario effettuare un backup del file DataVault che contiene tutti i log raccolti fino a quell'istante in maniera che il server APACHE e HighChart possano attingere e informazioni da un file nuovo senza possibilità di corrompere il vecchio file DataVault.

La gestione è affidata a 4 file:

- SelectGraph.php
- DataVisualizer.php
- Data.js
- GetSampleData.php

Il file DataVisualizer.php usa il file data.js che a sua volta interroga il file values.hph per estrarre i dati contenuti nel database SQLite3.

"SelectGraph.php

```
<!doctype html>
<html lang="it">
<head><title>Data Station</title></head>
<body>
    <FORM action=DataVisualizer.php method=”get”>
    <TABLE align="center" BORDER=0>
        <TR>
            <TD>
                <fieldset>
                    <legend>Giorno</legend>
                    <select name="Giorno">
                    <option value="16">16</option>
                    <option value="17">17</option>
                    <option value="18">18</option>
                    </select>
                </fieldset>
            </TD>

            <TD>
                <fieldset>
                    <legend>Mese</legend>
                    <select name="Mese" >
                    <option value="08">Agosto</option>
                    </select>
                </fieldset>
            </TD>

            <TD>
                <fieldset>
                    <legend>Anno</legend>
                    <select name="Anno">
                    <option value="2019">2019</option>
                    </select>
                </fieldset>
            </TD>
        <TR>
        <TR>
            <TD COLSPAN="3" align="center">
                <button type=”submit”>
                   Carica i Dati dei Sensori
                </button>
            </TD>
        </TR>
    </TABLE>
    </FORM>
</body>
</html>
```

”DataVisualizer.php

```
<html>
<head>
<title>Casoretti DataStation Visualizer</title>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js" type="text/javascript"></script>
<script src="http://code.highcharts.com/highcharts.js"></script>
<script src="http://code.highcharts.com/modules/exporting.js"></script>
<script type="text/javascript" src="data.js" ></script>
</head>
<body>
<?php
    $Giorno=$_GET['Giorno'];
    $Mese=$_GET['Mese'];
    $Anno=$_GET['Anno'];
    echo "<DIV ID='SampleDate'>".$Anno."-".$Mese."-".$Giorno."</DIV><br>";
?>
<div id="TemperatureChart" style="height: 400px; margin: 0 auto"></div>
<br>
<div id="PressureChart" style="height: 400px; margin: 0 auto"></div>
<br>
<div id="HumidityChart" style="height: 400px; margin: 0 auto"></div>
</body>
</html>
```

”Data.js

```
//jQuery Code: Script definition for the Cal Back of the "Document Ready" Event
$(function() {

    var x_values = [];
    var y_values = [];
    var switch1 = true;
    //jQuery Code: Get the URL from the Server "values.php" and invoke che CallBack Function with the loaded document

    // SampleDate Extraction:
    Date_Str=$('#SampleDate').text()

    // Temperature Graph

        Get_Param={SampleData: "Temperature", SampleDate: "2019-08-16"};
        Get_Param_Str="SampleData=Temperature&SampleDate="+Date_Str;

        // Syntax to pass the Data: { SampleData:"Temperature"}
        var Page_URL=encodeURIComponent('GetSampleData.php?SampleData=Temperature&SampleDate=2019-08-16')

        $.get('GetSampleData.php',Get_Param_Str,function(data) {
            switch1 = true;
            data = data.split('/');
            for (var i in data)
            {
                if (switch1 == true)
                {
                // var ts = timeConverter(data[i]);
                var ts = data[i];
                x_values.push(ts);
                switch1 = false;
                }
                else
                {
                y_values.push(parseFloat(data[i]));
                switch1 = true;
                }

            }
            x_values.pop();


            //jQuery Code: Select the DOM Object '#chart' and use highcharts function
            $('#TemperatureChart').highcharts({
                chart : {
                type : 'spline'
                },
                title : {
                text : 'Datalogger Highcharts SQLite'
                },
                subtitle : {
                text : 'Source: Casoretti DataStation'
                },
                xAxis : {
                title : {
                    text : 'Time'
                },
                categories : x_values
                },
```

```
                yAxis : {
                title : {
                    text : 'Temperature'
                },
                labels : {
                    formatter : function() {
                        return this.value + '° C'
                    }
                }
                },
                tooltip : {
                crosshairs : true,
                shared : true,
                valueSuffix : ''
                },
                plotOptions : {
                spline : {
                    marker : {
                        radius : 4,
                        lineColor : '#666666',
                        lineWidth : 1
                    }
                }
                },
                series : [{

                name : 'Temperature',
                data : y_values
                }]
            });
        //
    });
    //End Chart

    // Pressure Graph
        Get_Param_Str="SampleData=Pressure&SampleDate="+Date_Str;
        $.get('GetSampleData.php',Get_Param_Str,function(data) {

        x_values.length=0;
        y_values.length=0;
        switch1 = true;
        data = data.split('/');
        for (var i in data)
        {
            if (switch1 == true)
            {
                // var ts = timeConverter(data[i]);
            var ts = data[i];
                x_values.push(ts);
                switch1 = false;
            }
            else
            {
                y_values.push(parseFloat(data[i]));
                switch1 = true;
            }

        }
        x_values.pop();

        //jQuery Code: Select the DOM Object '#chart' and use highcharts function
```

```
        $('#PressureChart').highcharts({
            chart : {
                type : 'spline'
            },
            title : {
                text : 'Datalogger Highcharts SQLite'
            },
            subtitle : {
                text : 'Source: Casoretti DataStation'
            },
            xAxis : {
                title : {
                    text : 'Time'
                },
                categories : x_values
            },
            yAxis : {
                title : {
                    text : 'Pressure'
                },
                labels : {
                    formatter : function() {
                        return this.value + 'hPa'
                    }
                }
            },
            tooltip : {
                crosshairs : true,
                shared : true,
                valueSuffix : ''
            },
            plotOptions : {
                spline : {
                    marker : {
                        radius : 4,
                        lineColor : '#666666',
                        lineWidth : 1
                    }
                }
            },
            series : [{

                name : 'Pressure',
                data : y_values
            }]
        });
    });
    // End Chart

    //Humidity Graph
        Get_Param_Str="SampleData=Humidity&SampleDate="+Date_Str;
        $.get('GetSampleData.php',Get_Param_Str,function(data) {

        x_values.length=0;
        y_values.length=0;
        switch1 = true;
        data = data.split('/');
        for (var i in data)
        {
            if (switch1 == true)
            {
```

```
                // var ts = timeConverter(data[i]);
            var ts = data[i];
                x_values.push(ts);
                switch1 = false;
            }
            else
            {
                y_values.push(parseFloat(data[i]));
                switch1 = true;
            }

        }
        x_values.pop();

        //jQuery Code: Select the DOM Object '#chart' and use highcharts function
        $('#HumidityChart').highcharts({
            chart : {
                type : 'spline'
            },
            title : {
                text : 'Datalogger Highcharts SQLite'
            },
            subtitle : {
                text : 'Source: Casoretti DataStation'
            },
            xAxis : {
                title : {
                    text : 'Time'
                },
                categories : x_values
            },
            yAxis : {
                title : {
                    text : 'Humidity'
                },
                labels : {
                    formatter : function() {
                        return this.value + '%'
                    }
                }
            },
            tooltip : {
                crosshairs : true,
                shared : true,
                valueSuffix : ''
            },
            plotOptions : {
                spline : {
                    marker : {
                        radius : 4,
                        lineColor : '#666666',
                        lineWidth : 1
                    }
                }
            },
            series : [{

                name : 'Humidity',
                data : y_values
            }]
        });
```

```
    });
    // End Chart
});



function timeConverter(UNIX_timestamp){
  var a = new Date(UNIX_timestamp * 1000);
  var months = ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec'];
  var year = a.getFullYear();
  var month = months[a.getMonth()];
  var date = a.getDate();
  var hour = a.getHours();
  var min = a.getMinutes() < 10 ? '0' + a.getMinutes() : a.getMinutes();
  var sec = a.getSeconds() < 10 ? '0' + a.getSeconds() : a.getSeconds();
  var time = date + ' ' + month + ' ' + year + ' ' + hour + ':' + min + ':' + sec ;
  return time;
}
```

”GetSampleData.php

```php
<?php
    # To Extract the Value from the DB, invoke the PHP as "GetSampleData.php?SampleData=Temperature&SampleDate=2019-08-16
    # echo "Extract Data from Vault<br>";
    # the Parameter to be Extracted from the DB is passed to the PHP page by the Variable "SampleData"
    # $_GET['SampleData']

    $SampleData_AttributeName=$_GET['SampleData'];
    $SampleDate=$_GET['SampleDate'];

    /*
    if($SampleDate=='')
    {
        $SampleDate='2019-08-16';
    }
    */

    # echo "Extract SampleData='".$SampleData_AttributeName."'<br>";
    # echo "Extract SampleData='".$SampleDate."'<br>";

    $DB = new SQLite3('DataVault.db', SQLITE3_OPEN_READONLY);

    /*
    if($DB)
    {
        echo "DataBase 'DataVault' Opened<br>";
    }
    else
    {
        echo "Error in fetch ".$DB->lastErrorMsg();
    }

    # $DB->open("DataVault.db", SQLITE3_OPEN_READONLY);

    # echo "Test DB Opening";
    # echo "Error in fetch ".$DB->lastErrorMsg();

    */

    $QueryText="SELECT Day||' '||Month||' '||Year||' '||TimeStamp AS TS, ".$SampleData_AttributeName." FROM `Sample` WHERE Date='".$SampleDate."'";
    # echo "Executing Query<br>".$QueryText."<br><br>";

    $result = $DB->query($QueryText);

    /*
    if($result)
    {
        echo "Query Executed<br>";
    }
    else
    {
        echo "Error in Query Execution: ".$DB->lastErrorMsg();
    }
    */

    while($row = $result->fetchArray())
    {
        # Warning: use the Attribute Index instead of the Attribute Name
        echo $row['TS'] . "/" . $row[1]. "/" ;
```

```
    }

    # echo "Extract Data from Vault Terminated<br>";
?>
```

Implementando la funzione in maniera ricorsiva è possibile visualizzare tutti i parametri loggati e la possibile previsione dell'irraggiamento solare per mezzo delle chart interattive.

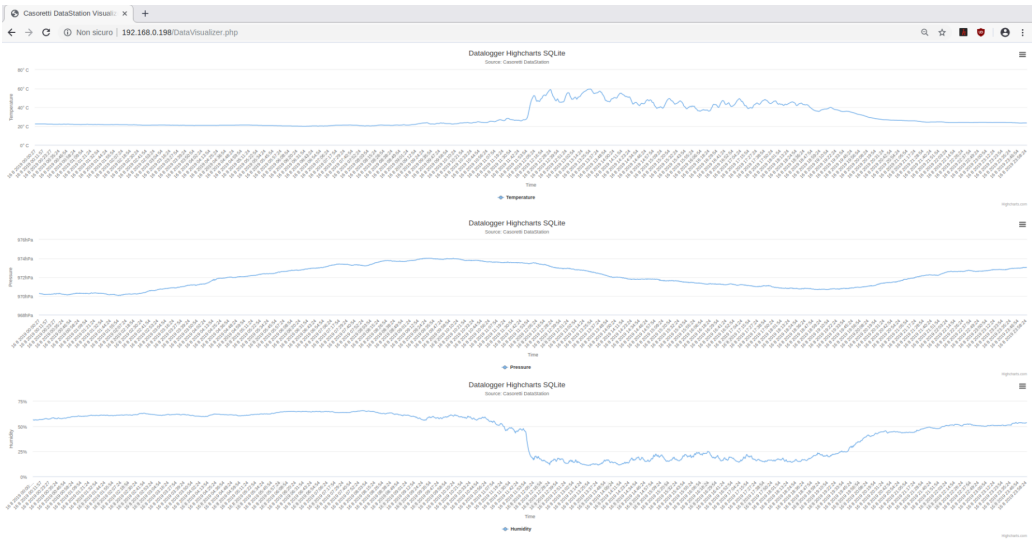


Figure 17.1: HiChart Render

Attraverso il box di scelta è possibile scegliere il giorno su cui viene effettuare la query nel database LiteSQL3.

In questa maniera sono disponibili solo i dati nella finestra temporale scelta.

Figure 17.2: WebChart

Chapter 18

Project Conclusion

Il progetto nel semestre estivo è sicuramente impegnativo e dato il tempo limitato è difficile portare a termine l'intera assegnazione.

Il progetto si compone di più parti che spaziano dal design elettronico al disegno in cad 3D alla scrittura del codice in Python per cui la sua complessità non è assolutamente banale.

Il risultato finale si presenta come di seguito:

Figure 18.1: Final Product

Figure 18.2: Final Product 2

Figure 18.3: Final Product 3

Figure 18.4: Final Product 4

Figure 18.5: Final Product 5

Figure 18.6: Final Product 6

Figure 18.7: Final Product 7

Figure 18.8: Final Product 8

Figure 18.9: Final Product 9

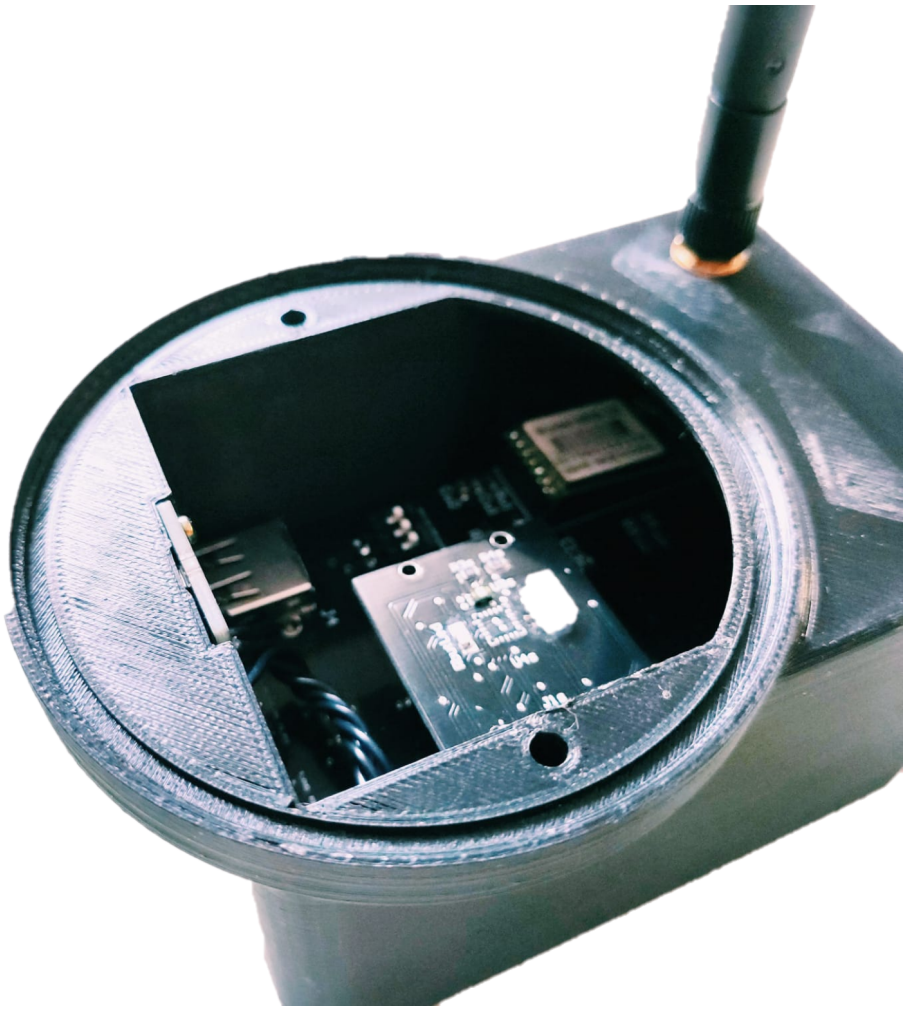


Figure 18.10: Final Product 10

Figure 18.11: Final Product 11

Figure 18.12: Final Product 12

18.0.1 Electronic

Per quanto riguarda la parte elettronica non sono state riscontrate particolari difficoltà se non nel caso della PCB dei sensori dove il protocollo di comunicazione I^2C ha sofferto particolarmente della congiunzione delle due schede elettroniche dovendo far riprogettare quella che contiene i sensori.

Anche l'utilizzo di hardware professionale come il BME680 non ha permesso di integrare lo stesso sensore all'interno della PCB.

Si è resa quindi necessaria la stilatura di due schede elettroniche per poter effettuare le misure che sono riportate all'interno del documento, infatti nella prima versione il bus I^2C portava al rilevamento di dispositivi che non erano presenti fisicamente.

18.0.2 3D CAD

La struttura 3D ha rappresentato una sfida per quanto riguarda la ricerca del materiale e per quanto riguarda la insufficiente conoscienza dei software di modellazione solida.

Tuttavia apparte le difficoltà riscontrate inizialmente si è condotta in positiva la realizzazione della struttura atta a contenere il dispositivo in questione.

La scelta del materiale non ha creato problemi e per mezzoo della stampa 3D è stato possibile realizzare in maniera veloce ed economica un prototipo funzionale.

18.0.3 Software

Data il tempo ridotto non è stato possibile raccogliere una serie di dati abbastanza ampia per poter condurre uno studio veritiero sul predittore.

Anche software ha presentato una sfida non indifferente data la difficoltà di unire tramite codice i moduli che permettono il funzionamento del WIFI e la lettura completa dei sensori.

La scrittura dei moduli in *Python* ha creato non poche difficoltà soprattutto nella parte che riguarda il sensore BME680 che si è dovuta tradurre da Circuit Python, un linguaggio proprietario di Bosch, a Python.

18.0.4 Issues

La parte più ostica è stata sicuramente quella che inizialmente si era valutata più banale, ossia quella della pcb sensori.

Il prolungamento delle piste tra le due pcb presenta sporadicamente alcuni disturbi che si presentano senza preavviso e che disturbano il bus di comunicazione.

Per questo motivo in estremis si è reso necessario produrre una ulteriore pcb per cercare di raccogliere più dati possibili prima della scadenza relativa alla consegna.

Siccome erano state valutate inizialmente le breakout board per poter acquisire velocemente i fenomeni esogeni da loggare si sviluppa una pcb ad hoc per poter contenere le tre tipologie di sensori.

Test Sensors Pcb Breakout Board

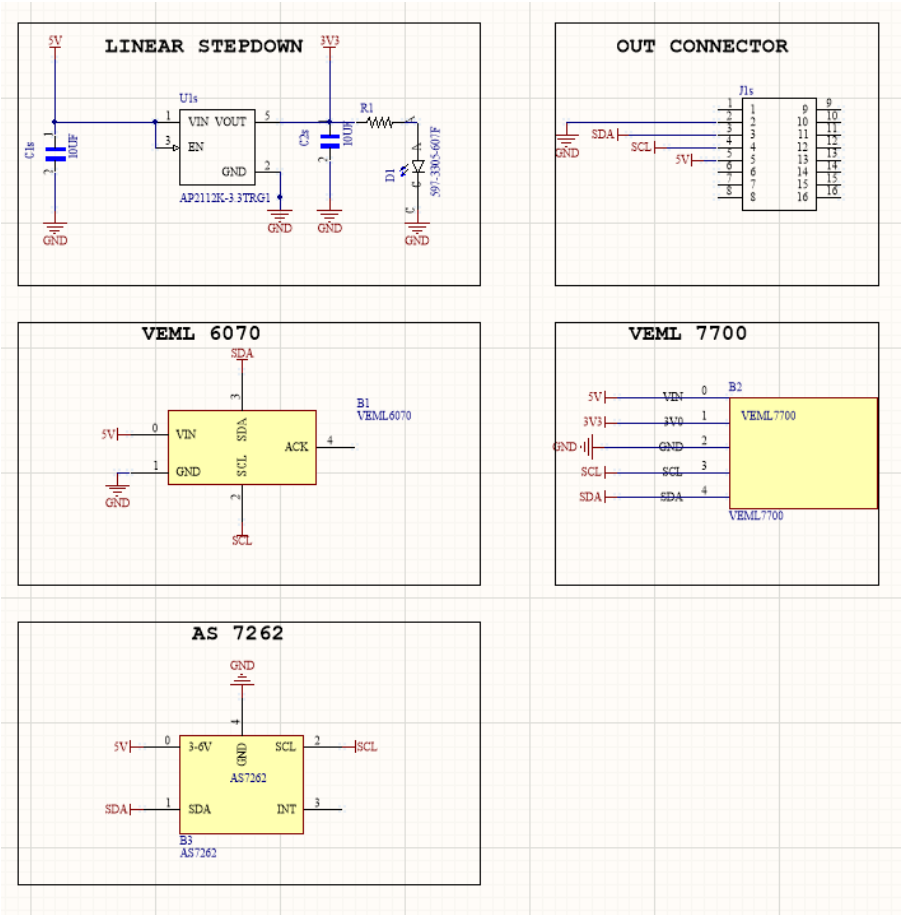


Figure 18.13: New Sensors Pcb Schematic

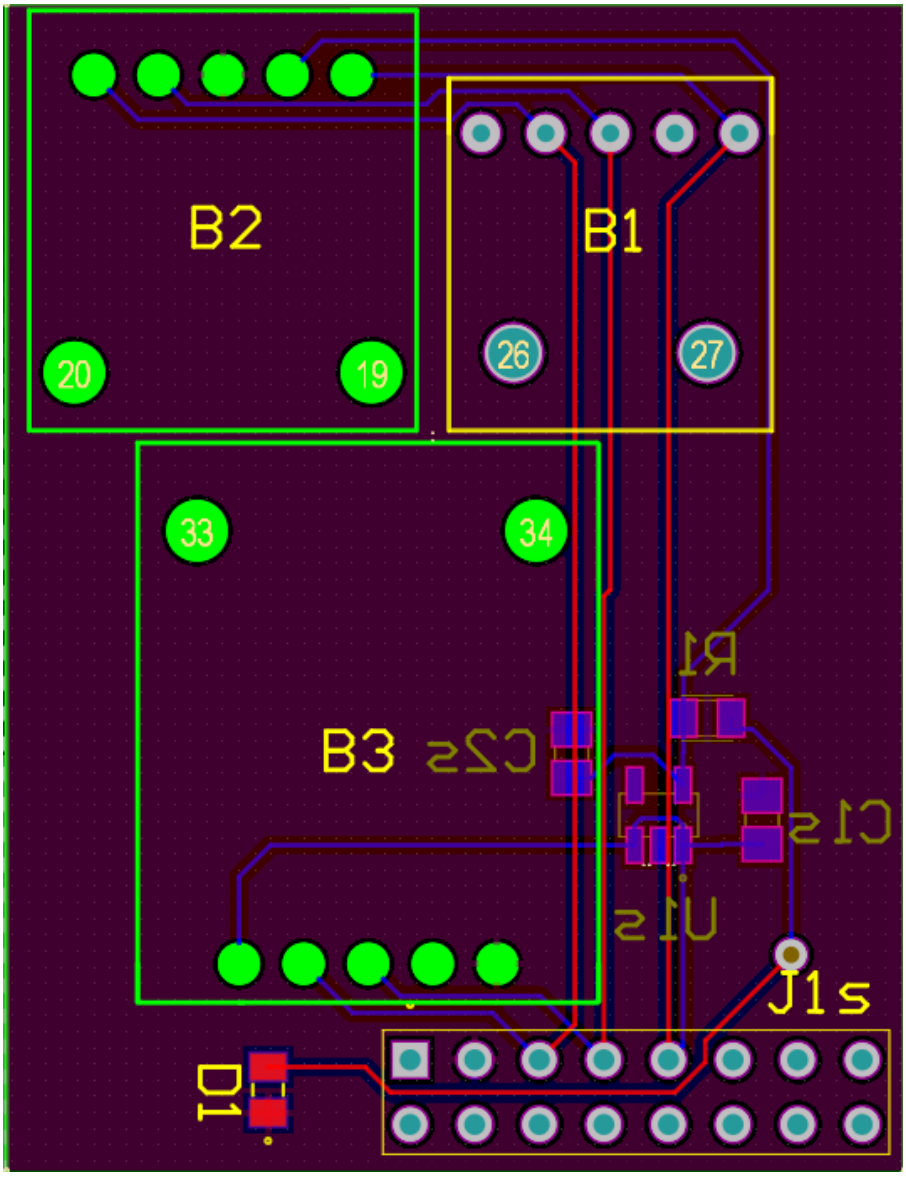


Figure 18.14: New Sensors Pcb 2d

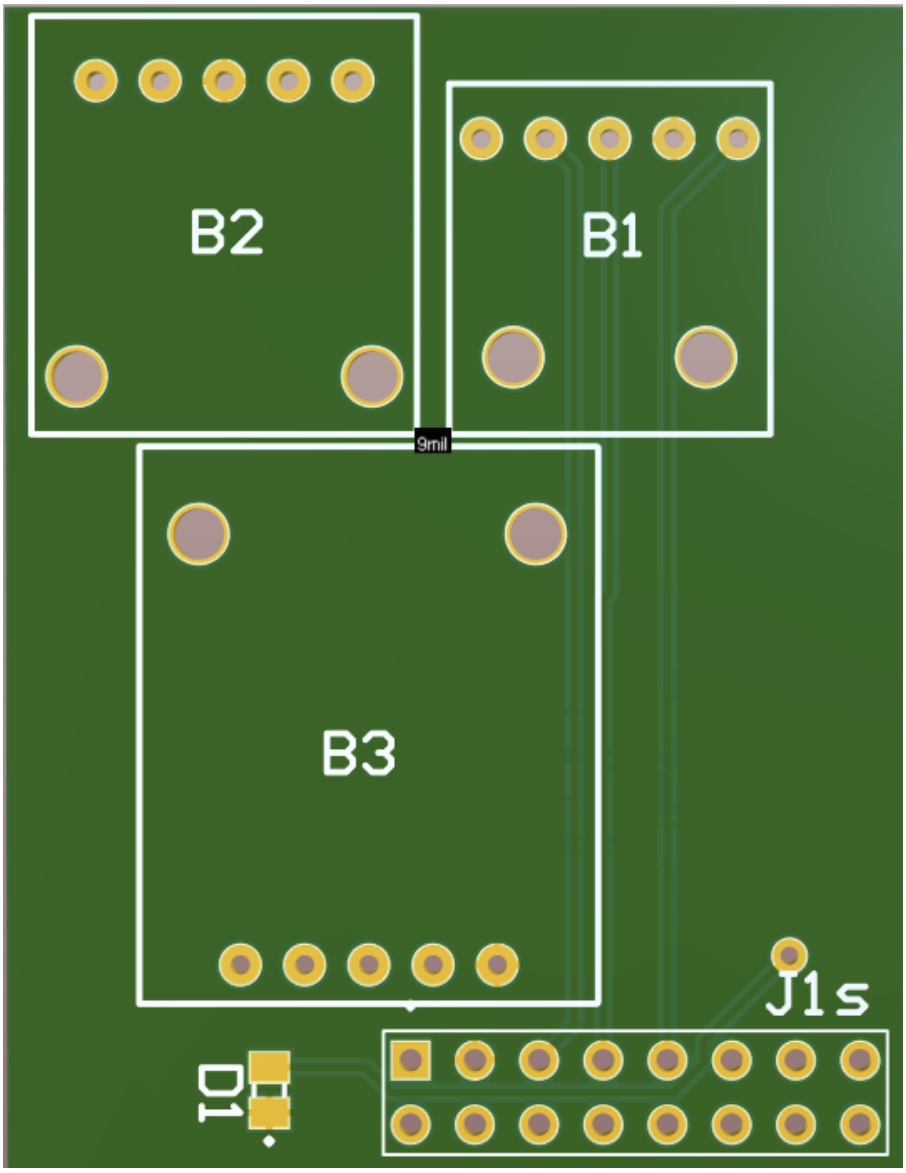


Figure 18.15: New Sensors Pcb 3d front

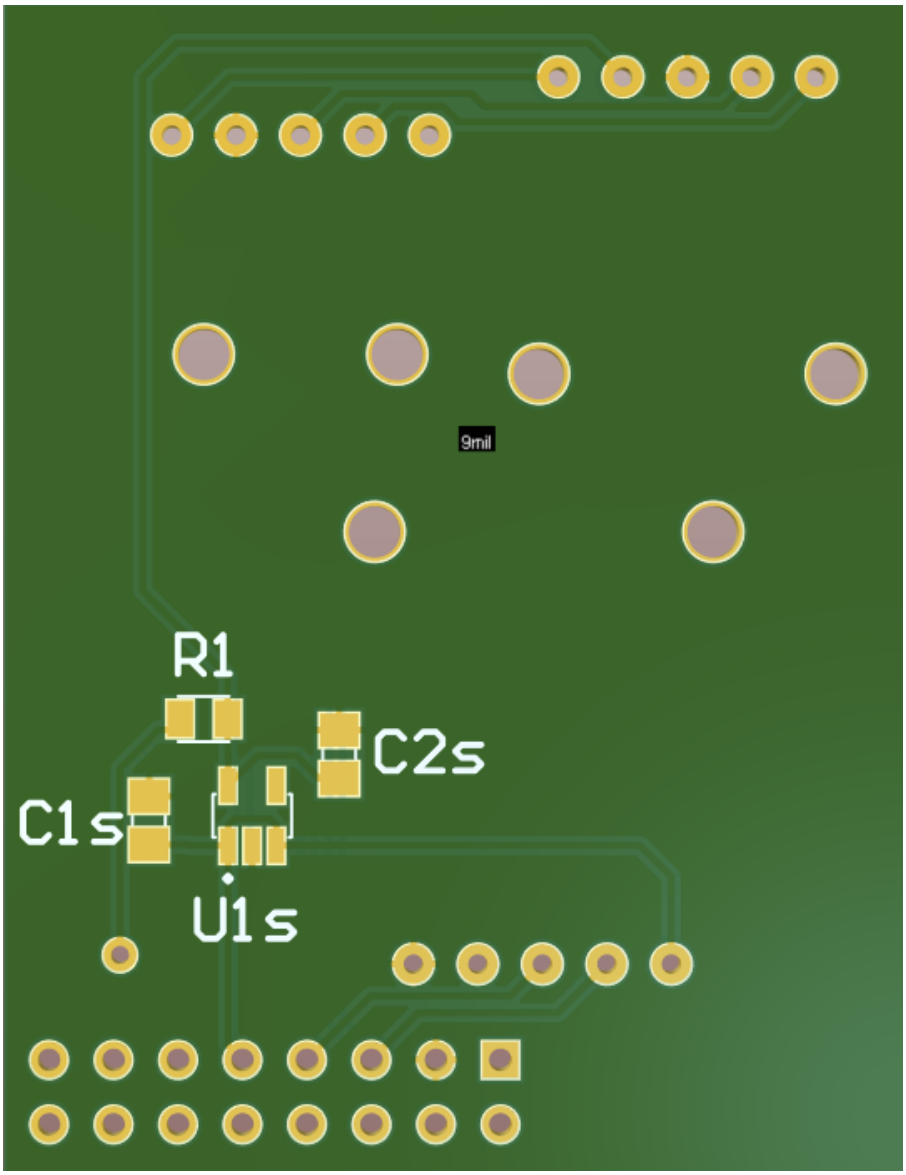


Figure 18.16: New Sensors Pcb 3d back

18.0.5 General Consideration

Il progetto nel complesso ha rappresentato una ottima sfida universitaria , coinvolgendo con notevole interesse lo studente ,creando così un dispositivo non privo di difetti ma che si pone come ottima base per uno sviluppo futuro.

E' possibile tramite di esso raccogliere dati ogni 15 minuti campionando una serie di dati molto interessanti al fine di poter valutare un predittore per la stima dell'irraggiamento solare nei giorni a venire.

Il database successivamente è accessibile tramite remoto per poi essere visualizzato tramite la ferrata piattaforma di visualizzazione High-Charts.

I giorni consecutivi alla stesura del documento prevedono un ulteriore prova della pcb dei sensori con una conseguente raccolta dati che potrebbe portare allo sviluppo del predittore *Persistence* a ridosso della discussione della seguente tesi.

Bibliography

[1] Ankur Kamthe, Tao Liu and Alberto E. Cerpa. *SIPS: Solar Irradiance Prediction System.* The Pennsylvania State University, University Park, PA 16802.

[2] Jared A. Lee, Sue Ellen Haupt. *Solar Irradiance Nowcasting Case Studies near Sacramento.* Research Applications Laboratory, National Center for Atmospheric Research, Boulder, Colorado.

[3] P. Shrivastava, Michael Carmichael. *Cloud Detection with MATLAB.* Sagar Institute of Research & Technology Excellence (SIRTE), Bhopal.

[4] Dilip Kumar Krishnappa, David Irwin, Eric Lyons, Michael Zink. *CloudCast: Cloud Computing for Short-term Mobile Weather Forecasts.* University of Massachusetts Amherst.

[5] Munir Husein and Il-Yop Chung. *Day-Ahead Solar Irradiance Forecasting for Microgrids Using a Long Short-Term Memory Recurrent Neural Network: A Deep Learning Approach.* School of Electrical Engineering, Kookmin University.

[6] Thomas Schmidt. *Small-scale solar irradiance nowcasting with sky imager pictures.* Institute of Physics, Energy Meteorology Group University of Oldenburg.

[7] Miguel Gomes Lopes. *Development of a low-cost, short-term solar irradiance forecasting system.* Instituto Universidario Superior Tecnico, Lisbon, Portugal.

[8] Cyril Voyant, Gilles Notton. *Solar irradiation nowcasting by stochastic persistence: a new parsimonious, simple and efficient forecasting tool.* Universidad De La Paz.

[9] Cristian Crisosto , Martin Hofmann , Riyad Mubarak and Gunther Seckmeyer. *One-Hour Prediction of the Global Solar Irradiance from All-Sky Images Using Artificial Neural Networks.* Institute for Meteorology and Climatology, Leibniz Universität Hannover.

[10] Evangelos Rikos, Stathis Tselepis, Carsten Hoyer-Klick, and Marion Schroedter-Homscheidt. *Stability and Power Quality Issues in Microgrids Under Weather Disturbances.* IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING, VOL. 1, NO. 3, SEPTEMBER 2008.

[11] Brian C. Ancell. *Weather and Forecasting (WAF) (ISSN: 0882-8156; eISSN: 1520-0434)* . Texas Tech University.

[12] Amit Kumar Yadav S.S.Chandel. *Solar radiation prediction using Artificial Neural Network techniques: A review.* Universidad de la Tecnica, New Mexico.

[13] Alexis Fouilloy. *Solar irradiation prediction with machine learning: Forecasting models selection method depending on weather variability.* Universitas Catholica Chilensis, Santiago del Chile.

[14] Mehmet Demirtas, Mehmet Yesilbudak. *Prediction of solar radiation using meteorological data. IEEE.*

[15] Md. Earfan Ali Khondaker. *A Short Term Day-Ahead Solar Radiation Prediction Using Machine Learning Techniques. Danesh Science and Technology University, (HSTU), Dinajpur-5200, Bangladesh*

[16] Mohammad Mehdi Lotfinejad, Reza Hafezi, Majid Khanali, Seyed Sina Hosseini. Mehdi Mehrpooya 5,6 and Shahaboddin Shamshirband 7,8,* I. *A Comparative Assessment of Predicting Daily Solar Radiation Using Bat Neural Network (BNN), Generalized Regression Neural Network (GRNN), and Neuro-Fuzzy (NF) System: A Case Study. Department of Agricultural Machinery Engineering, Faculty of Agricultural Engineering and Technology, University of Tehran, Karaj 4111, Iran; khanali@ut.ac.ir*