UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA
DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA

REALIZZAZIONE DI UNA STAZIONE METEOROLOGICA CON MICROCONTROLLORE STM32F334R8

RELATORE: prof. Buso Simone

LAUREANDO: Zammattio Mauro

ANNO ACCADEMICO: 2021/2022

22 settembre 2022

Indice

Capitolo 1

Introduzione

Lo scopo di questo progetto sperimentale di tesi è quello di aumentare la familiarità con la programmazione di microcontrollori, in particolare quelli della famiglia STM32 prodotti dalla casa costruttrice STMicroelectronics. Uno di questi (l'STM32F334R8) è stato utilizzato e studiato anche nel corso di elettronica industriale durante il primo semestre del terzo anno.

L'obiettivo del progetto è quella di realizzare una stazione meteorologica in grado di rilevare le principali grandezze e visualizzarle su un display. Si sono utilizzati diversi sensori in grado di rilevare temperatura, umidità, pressione atmosferica, luminosità e un modulo RTC per tenere traccia di data e ora.
L'hardware si compone di due terminali separati, ognuno gestito dal proprio μControllore (montato in una precisa scheda di sviluppo reperibile sul mercato, della famiglia Nucleo-board), che comunicano in maniera wireless tra loro. Da una parte ci sono i veri e propri sensori, mentre dall'altra il display per la visualizzazione. L'idea è quella di poter posizionare la scheda sensori nel punto in cui si vuole effettuare la misura (ad esempio all'esterno di una struttura) e riuscire a visualizzare i dati a distanza (ad esempio all'interno della stessa) senza cablaggio fisico.

Sono state realizzate tre schede PCB per interfacciare le board con le rispettive periferiche in modo compatto, senza breadboard e fili.
Queste verranno chiamate METEO_TX dal lato trasmettitore, invece METEO_RX_TOP e METEO_RX_BOTTOM dal lato ricevitore.

Capitolo 2

Descrizione componenti

2.1 Board Nucleo-F334R8

La scheda Nucleo-F334R8 (visibile in figura 2.1) è una scheda di sviluppo programmabile realizzata dalla STMicroelectronics. Su di essa è montato il vero e proprio microcontrollore STM32F334R8, della famiglia STM32F3 prodotto sempre dalle STM e quindi basato sul core ARM Cortex-M4.(Si rimanda alla sottosezione 2.1.1 per una descrizione più dettagliata).
Essa è molto utilizzata per la creazione di svariati prototipi di sistemi di sviluppo completi a basso costo grazie alle numerose periferiche incorporate nel μcontrollore. In questo progetto è utilizzata per gestire la ricezione dei dati atmosferici dalla scheda esterna e la visualizzazione su display.

Oltre al processore, nella Board sono presenti (visibili anche in figura 2.4):

- diversi connettori per il collegamento delle periferiche e delle alimentazioni (alcuni anche compatibili con la scheda Arduino UNO);
- due pulsanti (Reset, di colore nero, e User, di colore blu);
- tre Led (uno rosso, LD3, indice dell'accensione della scheda, uno Rosso/Verde, LD1, indice del funzionamento della scheda, uno verde, LD2, il cui funzionamento è gestibile dall'utente);
- un oscillatore a cristallo con frequenza di 32.768 kHz per una possibile sincronizzazione del clock del μC;
- un altro oscillatore a cristallo con frequenza di 8 MHz (X1 nella figura 2.4) che può essere preso come sorgente di clock del μC;
- una sezione, addirittura rimovibile, dedicata alla comunicazione del processore con il PC tramite connettore USB mini-B (per la programmazione dello stesso μC, per il debuggin...)

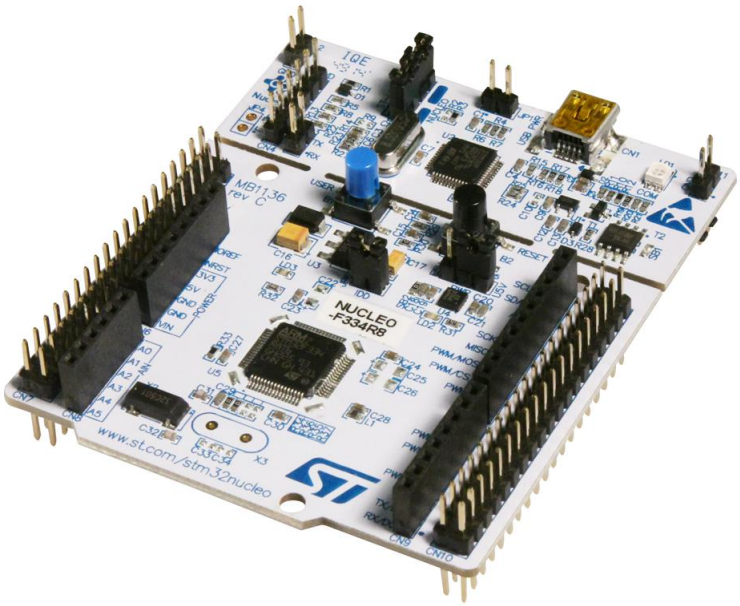


Figura 2.1: La scheda di sviluppo Nucleo-F334R8

Core	ARM Cortex-M4	
Aritmetica	32	bit
Flash	16-32-64	Kbyte
SRAM	12	kbyte
Clock CPU	72	MHz
Voltage in	da 2 a 3.6	V
Package	LQFP64	

Tabella 2.1: caratteristiche principali STM32F334R8

Nelle successive sottosezioni verranno illustrate le principali parti della scheda. Per una descrizione completa si rimanda alla documentazione fornita sul sito `https://www.st.com/en/evaluation-tools/nucleo-f334r8.html`

2.1.1 Microcontrollore STM32F334R8

Le principali caratteristihe del μcontrollore STM32F334R8 sono raffigurate nella tabella 2.1

Questo dispositivo è di medie prestazioni, presenta numerose periferiche e nel suo complesso si può rappresentare con lo schema a blocchi di figura 2.2. Di seguto sono elencate le principali periferiche del μcontrollore:

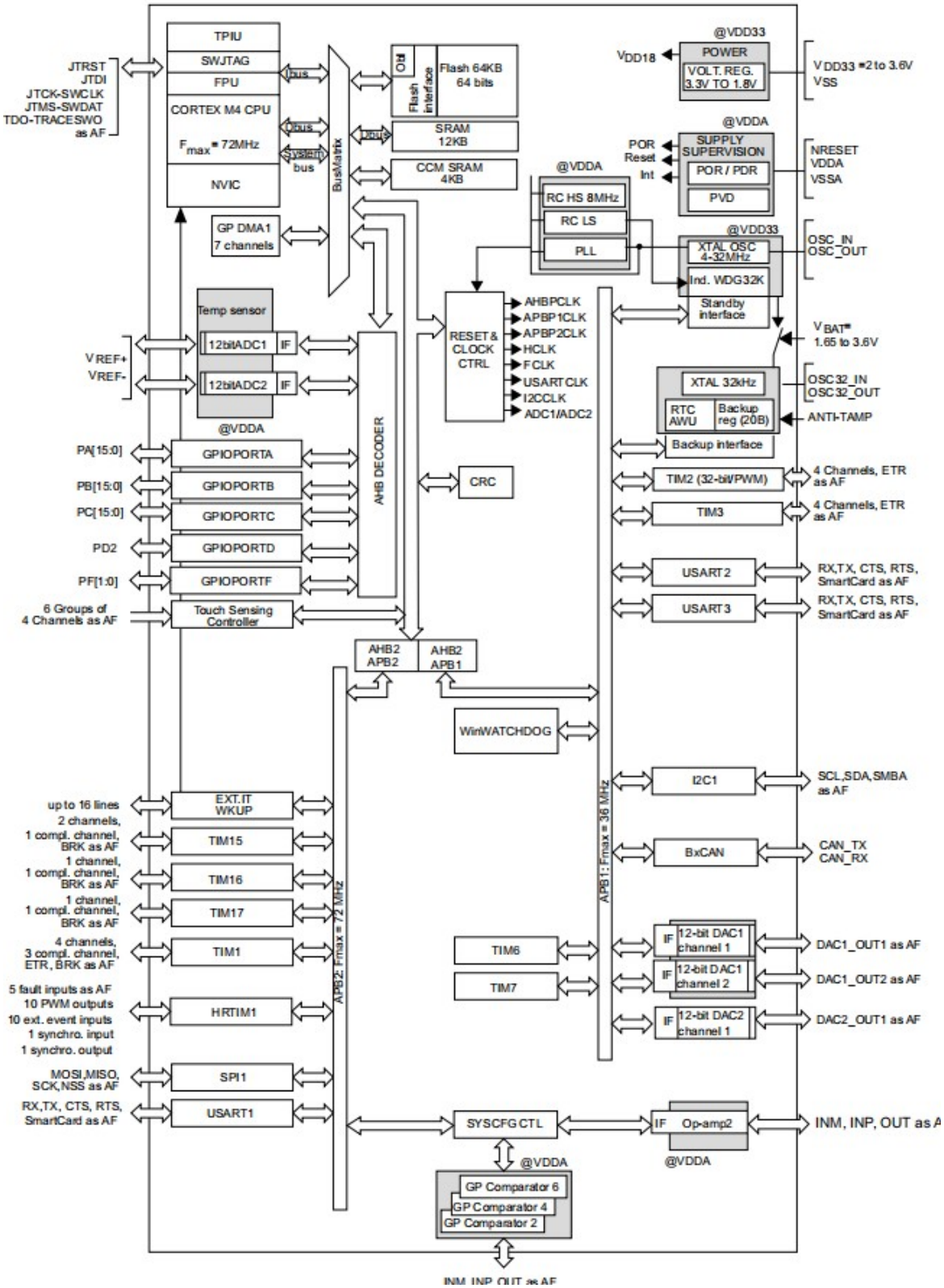


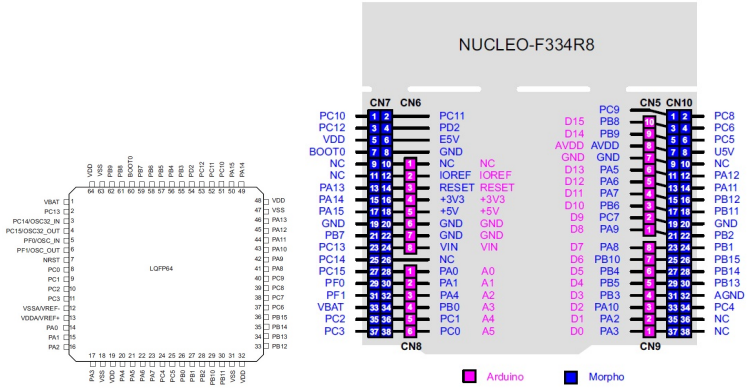
Figura 2.2: Schema a blocchi STM32F334R8

- 2 convertitori ADC con risoluzione di 12 bit con 21 canali ciascuno;
- 2 convertitori DAC, con risoluzione di 12 bit, di cui uno con 2 canali e uno a singolo canale, per un totale di 3 uscite;
- 1 amplificatore operazionale rail-to-rail;
- 3 comparatori ultra-fast-rail-to-rail;
- DMA a 7 canali;
- fino a 51 GPIO suddivisi tra Normal I/O e 5-V Tolerant I/O;
- interfacce di comunicazione:
  - 1 I2C;
  - 1 SPI;
  - 1 CAN;
  - 3 USART.
- diversi tipologie di Timers, tra cui;
  - 1 timer ad alta risoluzione, HRTIM, a 16 bit con 10 canali, in grado di avere una risoluzione temporale massima di 217 ps;
  - 4 timer general purpose a 16 o 32 bit (utilizzabili anche per generazione di segnali PWM);
  - 2 watchdog timer;
  - 1 RTC Timer

Per questo progetto verranno usate quasi esclusivamente la periferica che gestisce la comunicazione I2C in modo da poter ricevere/trasmettere dati da/verso il modulo RTC e display e la periferica USART1 per la ricezione dei dati atmosferici esterni. Per avere maggiori dettagli sul prodotto si rimanda alla consultazione del datasheet scaricabile dalla pagina
`https://www.st.com/en/microcontrollers-microprocessors/stm32f334r8.html`

Il microcontrollore in questione è incorporato in un package con 64 pin come mostrato in figura 2.3a e gli stessi pin sono accessibili anche dalla board di sviluppo Nucleo come in figura 2.3b.

Come succede per molti microcontrollori, il numero di pin fisici è inferiore al numero di periferiche programmabili e di conseguenza diversi piedini hanno funzioni secondarie.
Questo è reso possibile grazie a una complessa rete di multiplexing (chiamata Alternate Function) che programmata a dovere, può assegnare a un piedino una data funzione. Questa funzione deve essere opportunamente definita dal



(a) *Pinout µcontrollore STM32F334R8* (b) *Pinout Board Nucleo-F334R8*

Figura 2.3: pinout del μcontrollore STM32F334R8

programmatore, prestando attenzione a non far sovrapporre più funzioni per uno stesso piedino. Esistono delle mappe predefinite (da AF0 a AF15) a seconda del principale scopo per cui viene usato il μControllore: ad esempio la mappa per l'utilizzo dei timer, oppure quella per l'utilizzo delle periferiche di comunicazione, ecc...

In questo progetto l'assegnazione dei pin viene effettuata attraverso un software di configurazione fornito dallo stesso costruttore del μControllore chiamato STM32CubeMX di cui si parlerà meglio nel capitolo 3.1.

2.1.2 Alimentazione della scheda

La Board Nucleo-F334R8 può essere alimentata in diversi modi modificando le posizioni di alcuni jumper visibili in figura 2.4:

- tramite connettore USB mini-B sia attaccandolo al PC (in modo da poter anche programmare/debuggare) sia attaccandolo ad un alimentatore con uscita 5V tipo USB (in questo caso il jumper JP5 deve essere su U5V [1]);
- tramite pin E5V fornendo appunto 5 Volt (JP5 su E5V, JP1 rimosso);
- tramite il pin VIN con tensione dai 7 ai 12 V (JP5 su E5V, JP1 rimosso);

[1] in questo caso se il JP1 è inserito la scheda avrà un assorbimento di massimo 100mA, altrimenti di 300mA

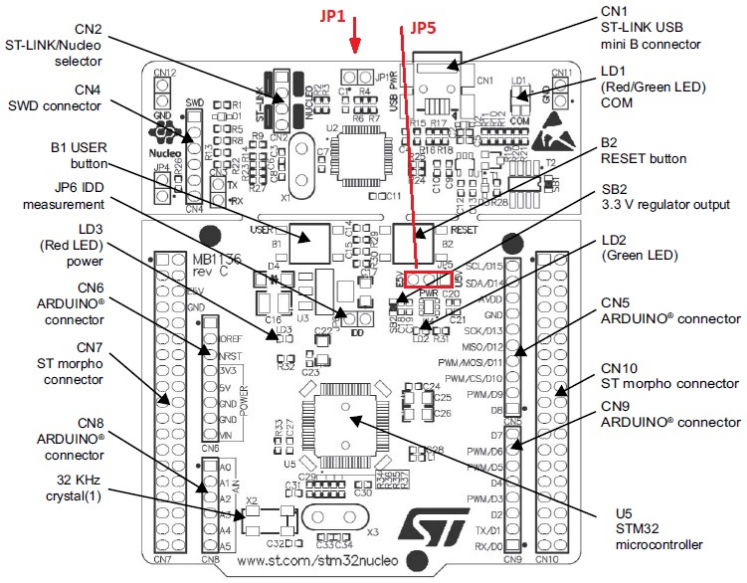


Figura 2.4: Descrizione di alcune parti della scheda

- tramite il pin 3.3V con la medesima tensione (in questo caso però si deve o staccare il modulo per la programmazione della scheda oppure dissaldare le resistenze SB2 e SB12)

Per questo progetto l'alimentazione della board può avvenire tramite connettore USB mini-B oppure tramite il jack di alimentazione presente nella scheda METEO_RX_BOTTOM e connesso al pin VIN [2]. A seconda di come si preferisce procedere si dovrà cambiare la posizione del jumper JP5 come descritto sopra. Di conseguenza i sensori verranno alimentati dai pin presenti nella Board Nucleo (5V e 3.3V a seconda della periferica).

2.2 Board Nucleo-L031K6

La seconda scheda di sviluppo utilizzata è la Nucleo-L031K6 (figura 2.5), molto più compatta della precedente e scelta apposta per ridurre le dimensioni dell'hardware dal lato sensori.

Su di essa è montato il μControllore STM32L031K6 sempre prodotto dalla STMicroelectronics basato sul Core ARM Cortex-M0+ (si rimanda alla sottosezione 2.2.1 per maggior dettagli), alcuni led indicatori, un altro microcontrollore per la programmazione e il debugging da computer (tramite il connettore USB micro-B)[3], due oscillatori esterni (X1 a 32 KHz e X2 a

[2] nella sottosezione 4.2 viene descritta in modo migliore questa parte
[3] Tutta questa sezione viene chiamata ST-Link

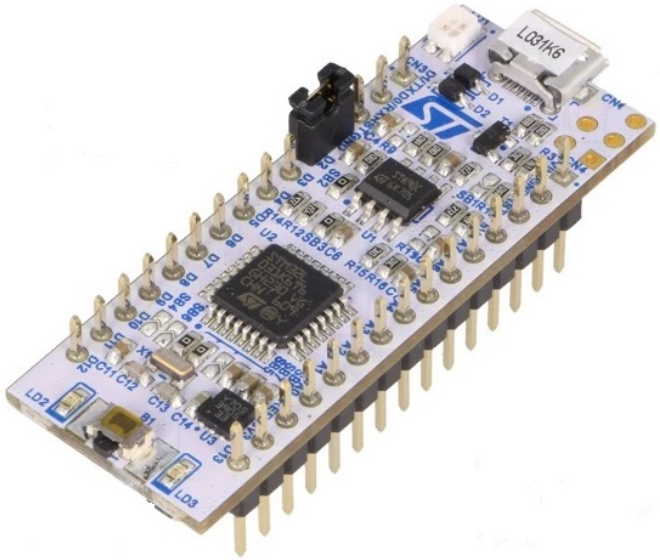


Figura 2.5: La scheda Nucleo-L031K6

8 MHz), un pulsante di reset e altro ancora[4]. In figura 2.6 sono visibili i principali componenti montati sulla board.

Come si può notare, essendo più piccola, il numero di pin del processore è minore, quindi anche la quantità di periferiche e di piedini input/output. Per gli scopi di questo progetto, tuttavia, le risorse fornite dalla scheda sono più che sufficienti.

2.2.1 Microcontrollore STM32L031K6

Dalla tabella 2.2 si possono notare alcune differenze rispetto all'altro integrato, riguardo le principali caratteristiche di funzionamento. In particolare si osservano delle minori prestazioni in termini di velocità massima e di quantità di memoria.

La complessità dello schema a blocchi equivalente è minore, come visibile dalla figura 2.7. Inoltre, come intuibile, anche dal punto di vista del numero di periferiche si hanno limitazioni.
Infatti sono presenti:

- 1 ADC a 12 bit composto da 10 canali;
- 2 comparatori a basso consumo di potenza;

[4]Per una più dettagliata descrizione si rimanda al manuale dell'utente reperibile dal sito del costruttore `https://www.st.com/en/evaluation-tools/nucleo-l031k6.html`

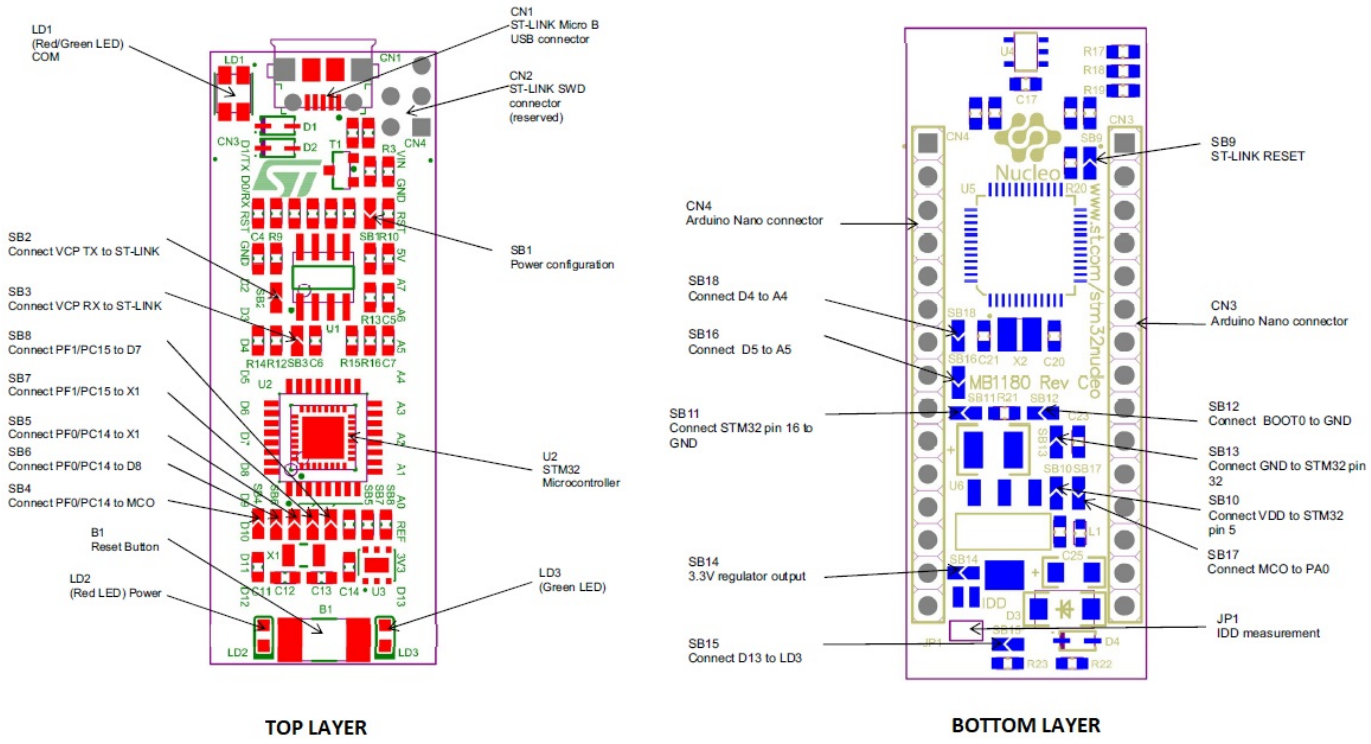


Figura 2.6: Layout della board

Core	ARM Cortex-M0+	
Aritmetica	32	bit
Flash	16-32	Kbyte
SRAM	8	kbyte
Clock CPU	32 (max)	MHz
Voltage in	da 1.65 a 3.6	V
Package	LQFP32	

Tabella 2.2: caratteristiche principali STM32L031K6

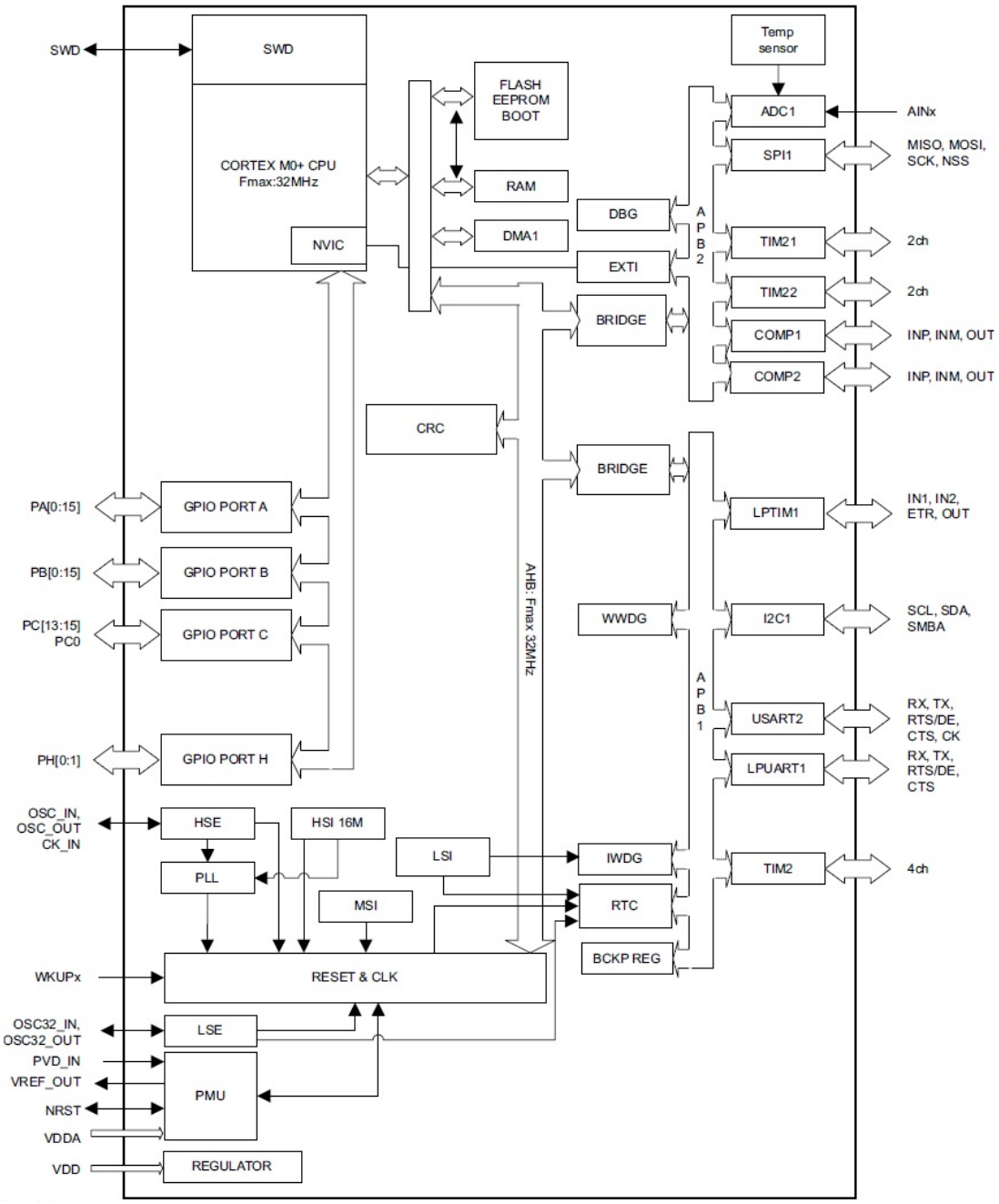


Figura 2.7: Schema a blocchi STM32L031K6

- DMA a 7 canali;
- interfacce di comunicazione:
  - 1 USART, 1 UART a bassa potenza (LPUART);
  - 2 interfacce SPI;
  - 1 interfaccia I2C (normal, fast, ultra fast mode);
- diversi Timers tra cui:
  - 1 timer a 16 bit a 4 canali;
  - 2 timers a 16 bit a 2 canali;
  - 1 timer a 16 bit ultra Low Power;
  - 1 RTC;
  - 2 watchdogs.

Ovviamente per maggiori informazioni si rimanda al datasheet sul sito `https://www.st.com/en/microcontrollers-microprocessors/stm32l031k6.html`.

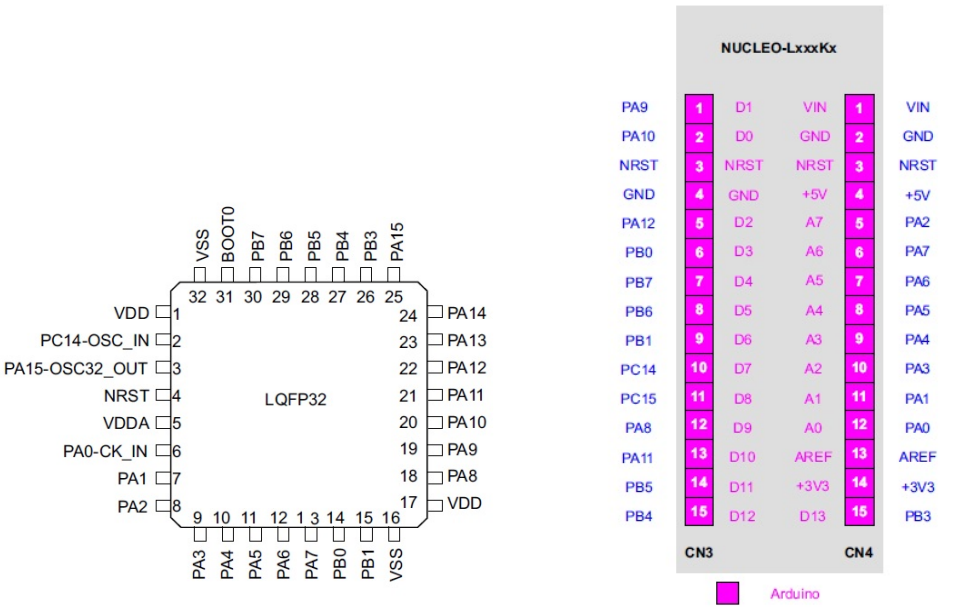
Per questo μcontrollore si è utilizzata una modalità di funzionamento a bassa potenza. Quando non viene richiesta nessuna operazione, l'STM32L031K6 entra in modalità STOP. In questa fase quasi tutti i clock sono disabilitati, e di conseguenza le periferiche associate; vengono mantenuti i valori dei registri e della RAM; il regolatore di tensione è in modalità Low Power.
Il risveglio può avvenire attraverso degli interrupt (configurati come GPIO) forniti in ingresso nelle linee EXTI (da EXTI0 a EXTI15) oppure attraverso specifiche periferiche abilitate (USART, I2C, LPUART, LPTIMER). In questa esperienza si è utilizzato l'interrupt generato quando viene ricevuto un qualsiasi dato dalla LPUART.

Anche in questo caso si è utilizzato il software STM32CubeMX per la configurazione dei pin necessari. Si rimanda alla sottosezione 3.1.2 per maggiori dettagli.
Tutto il μControllore è montato in un package a 32 pin come mostrato in figura 2.8a.Questi pin poi sono accessibili all'utente tramite la Board di sviluppo seguendo lo schema in figura 2.8b.

2.2.2 Alimentazione della scheda

La Nucleo-L031K6 può essere alimentata in diversi modi:

- attraverso il connettore USB connesso a un PC (se si usa il connettore usb solo come alimentazione si deve chiudere il ponte SB1);
- attraverso il pin +5V fornendo appunto da 4.75V a 5.25V;



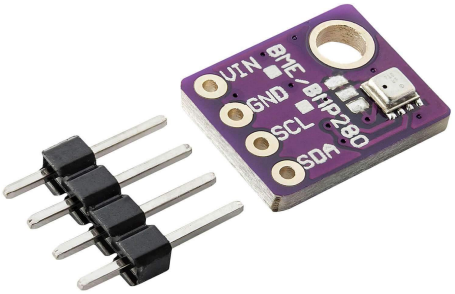
(a) *Pinout µcontrollore STM32L031K6* (b) *Pinout Board Nucleo-L031K6*

Figura 2.8: pinout del µcontrollore STM32L031K6

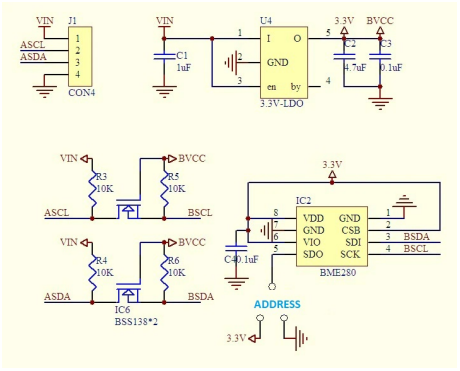
- attraverso il pin VIN fornendo dai 7V a 12V[5];
- attraverso il pin +3.3V con la stessa tensione, ma in questo caso la parte ST-LINK per la programmazione e debugging non viene alimentata.

In questo progetto, per venire incontro alle esigenze di basso consumo, in quanto si è pensato di implementare un'alimentazione a batteria per il trasmettitore, si sono attuati ulteriori accorgimenti (oltre a sfruttare la modalità STOP).
Si è alimentata la board tramite il pin +5V e dissaldando il ponte SB9 presente sulla stessa si è esclusa tutta la parte riguardante l'ST-Link. Così facendo, solo il µcontrollore è attivo e l'assorbimento di corrente è ridotto. La tensione di 5V viene fornita grazie a una sezione di alimentazione (basato sul regolatore di tensione L7805CV) presente nella scheda sviluppata METEO_TX, la quale provvede a fornire tensione per i sensori.
La batteria è collegata alla scheda (e quindi al regolatore di tensione). I dettagli riguardanti la scheda METEO_TX sono visibili alla sottosezione 4.1.
In ogni caso è sempre possibile alimentare l'intera scheda direttamente dal connettore USB micro-B della board Nucleo (ad esempio per la programmazione del micro), che a sua volta provvede a fornire tensione ai sensori.

[5] in questo caso e nel caso dei +5V, il modulo ST-Link, per funzionare, deve essere connesso (con l'USB) solo dopo aver già alimentato la scheda



(a) *scheda completa GY-BME280*



(b) *schema circuitale GY-BME280*

Figura 2.9: Scheda con sensore BME280

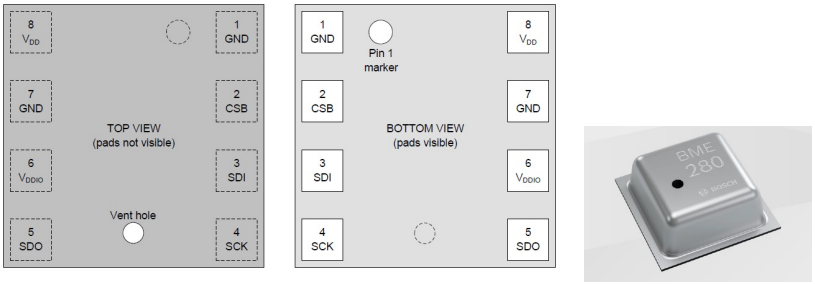
2.3 Sensore di Temperatura-Pressione-Umidità

2.3.1 Panoramica

Il sensore utilizzato per rilevare le grandezze di temperatura, pressione atmosferica e umidità è il BME280[6], prodotto dalla Bosch. Per questo progetto, si è acquistata la scheda GY-GME280 (figura 2.9a) in cui il sensore è già montato, insieme ad altri componenti per il diretto interfacciamento con un μControllore tramite protocollo I2C, senza ausilio di altra componentistica elettronica. Lo schema circuitale della scheda completa è visibile in figura 2.9b e si può notare:

- un regolatore di tensione che permette di alimentare tutto l'apparato da 3.3V a 5V e avere in uscita l'alimentazione di 3.3V stabili per il sensore;
- un traslatore di livello (tramite due mosfet), in grado di interfacciare le due linee del protocollo I2C dal livello di tensione fornito dal microcontrollore a quello in ingresso al sensore (3.3V);
- connettore a 4 pin per il collegamento (VCC, GND, SDA, SCL);
- possibilità di collegare a +VCC il piedino SDO del sensore per cambiare l'indirizzo I2C della periferica (di default il pin SDO è collegato a GND).

[6]Per avere una descrizione completa del sensore e del suo funzionamento si rimanda alla consultazione del datasheet



(a) *pinout BME280 GY-BME280* (b) *immagine BME280*

Figura 2.10: sensore BME280

Il sensore vero e proprio è mostrato in figura 2.12 e ha le seguenti caratteristiche:

Parametro	Valore	Unità di misura
Tensione alimentazione	da 1.71 a 3.6	V
Corrente in sleep mode	0.1	μA
Corrente per lettura umidità, pressione e temperatura (1 Hz)	3.6	μA

Il BME280 può essere programmato ad operare in una delle tre seguenti modalità di funzionamento da cui dipende il suo consumo di potenza:

Sleep Mode: Il sensore non compie operazioni; i suoi registri sono accessibili, basso consumo di potenza;

Forced Mode: Il sensore compie una rilevazione delle tre grandezze, salva i risultati nei registri e ritorna nello stato sleep;

Normal Mode: esegue continuamente un ciclo composto da una lettura e un tempo inattivo.

Per questo progetto il sensore viene fatto operare in "Forced Mode" in modo che dopo ogni rilevazione delle grandezze (eseguito con periodo abbastanza grande rispetto al tempo di misurazione) si risparmi sul consumo di potenza.

Il processo di misurazione può essere ritoccato in base all'utilizzo e alle necessità dell'utilizzatore. Sono presenti infatti dei settaggi che riguardano il sovracampionamento della misura (utile per ridurre il rumore sovrapposto alla stessa) e all'inserimento di un possibile filtro IIR (solo per misura di pressione e temperatura) con lo scopo di ridurre le fluttuazioni sulla misura.

Sensor Mode	forced mode, 1 sample/minute
Sovracampionamento	pressione x 1, umidità x 1, temperatura x 1
filtro IIR	OFF

Tabella 2.3: Settaggi suggeriti per il Wheather Monitoring

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

Figura 2.11: Mappa dei registri del BME280

Nel datasheet del componente viene consigliato come impostare questi parametri a seconda dall'utilizzo che si vuole fare e per questo caso, wheather monitoring, ossia monitoraggio meteorologico, vengono date le specifiche mostrate in tabella 2.3.

2.3.2 Descrizione Registri

In figura 2.11 viene mostrata tutta la mappa dei registri presenti nella memoria del sensore, tuttavia solo alcuni verranno settati e letti per questa esperienza.

Verranno letti innanzitutto i registri in cui sono memorizzati i coefficienti di calibrazione per il calcolo delle tre grandezze in unità leggibili. Questi coefficienti sono salvati a partire dal registro 0x88 fino al 0xA1 e dal registro 0xE1 al 0xF0 nel formato specificato dalla tabella 2.4.

Dopodichè si imposteranno a dovere i registri di indirizzo 0xF2 e 0xF4 per il settaggio del sovracampionamento e della modalita di funzionamento (scrivendo su di essi rispettivamente 0x01 e 0x25). Infine verranno letti i vari registri in cui sono memorizzati i valori acquisiti di temperatura, umidità e pressione (da 0xF7 a 0xFE)[7]. Tutti gli altri registri vengono lasciati

[7]Come verrà ridetto più avanti, il valore dei registri temp_xlsb e press_xlsb, ossia quelli che contengono i 4 bit meno significativi dei 20 bit totali della lettura, è nullo in quanto si è impostato il sovracampionamento a x 1 sia per la temperatura che per la pressione

Register Address	Register Content	Data type
0x88 / 0x89	dig_T1 [7:0]/[15:8]	unsigned short
0x8A / 0x8B	dig_T2 [7:0]/[15:8]	signed short
0x8C / 0x8D	dig_T3 [7:0]/[15:8]	signed short
0x8E / 0x8F	dig_P1 [7:0]/[15:8]	unsigned short
0x90 / 0x91	dig_P2 [7:0]/[15:8]	signed short
0x92 / 0x93	dig_P3 [7:0]/[15:8]	signed short
0x94 / 0x95	dig_P4 [7:0]/[15:8]	signed short
0x96 / 0x97	dig_P5 [7:0]/[15:8]	signed short
0x98 / 0x99	dig_P6 [7:0]/[15:8]	signed short
0x9A / 0x9B	dig_P7 [7:0]/[15:8]	signed short
0x9C / 0x9D	dig_P8 [7:0]/[15:8]	signed short
0x9E / 0x9F	dig_P9 [7:0]/[15:8]	signed short
0xA1	dig_H1 [7:0]	unsigned char
0xE1 / 0xE2	dig_H2 [7:0]/[15:8]	signed short
0xE3	dig_H3 [7:0]	unsigned char
0xE4/0xE5[3:0]	dig_H4 [11:4]/[3:0]	signed short
0xE5[7:4]/0xE6	dig_H5 [3:0]/[11:4]	signed short
0xE7	dig_H6 [7:0]	signed char

Tabella 2.4: Tabella Coefficienti di calibrazione

al loro valore di default, in quanto non serviranno per questa tipologia di misurazione.

2.3.3 Comunicazione I2C

Da datasheet si ricava l'indirizzo a 7 bit del sensore. Esso è composto da una parte fissa di 6 bit (111011) e il bit meno significativo impostabile a 0 o 1 a seconda che il pin SDO sia rispettivamente a GND o VCC. Nella scheda GY-BME280 il collegamento è di default verso GND (è modificabile) quindi l'indirizzo del sensore risulta 1110110 oppure 0x76.

La comunicazione in lettura e scrittura avviene secondo lo standard I2C. Per la scrittura, dopo aver inviato l'indirizzo della periferica, il Master invia l'indirizzo del registro su cui vuole scrivere e nel byte successivo il contenuto di tale registro. Dopodichè il sensore si aspetta di ricevere sempre due byte: uno con l'indirizzo del registro su cui scrivere e successivamente il byte da memorizzare. In figura 2.12 è visibile questo passaggio.

Per la lettura, invece, si parte prima con un invio, da parte del Master, dell'indirizzo del registro da leggere nello Slave. A questo punto il μControllore comincia la procedura di lettura vera e propria dei registri a partire da quello selezionato al passo precedente. Dopo ogni byte letto il sensore incrementa automaticamente di una unità il valore del registro da leggere, senza che

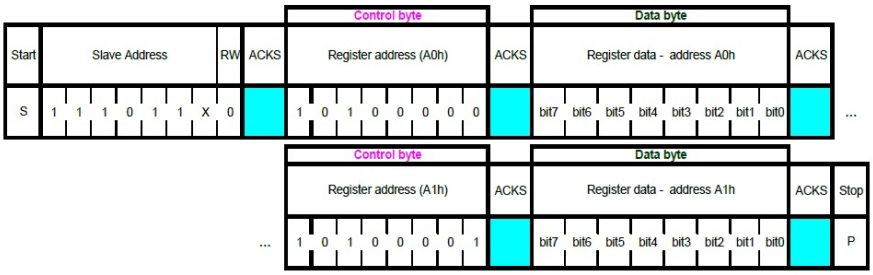


Figura 2.12: Procedura di scrittura nei registri

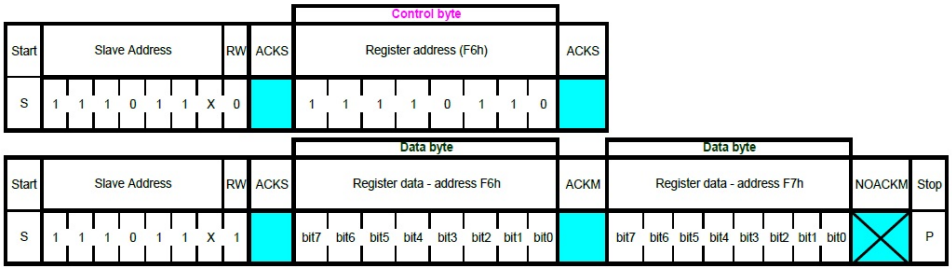


Figura 2.13: Procedura di lettura dei registri

il Master debba ri-selezionarlo di nuovo. In figura 2.13 è visibile questo meccanismo.

2.3.4 Calcolo grandezze

I valori di temperatura, pressione e umidità letti dai registri del sensore sono numeri convertiti dai convertitori ADC interni al modulo, tuttavia non sono scritti in modo da essere letti da una persona, ma vanno calibrati tramite i coefficienti di calibrazione della tabella 2.4. Il datasheet fornisce inoltre uno specifico algoritmo per calcolare i valori effettivi delle grandezze mostrato in figura 2.14. Con i settaggi fatti come al 2.3.2 i valori non compensati di pressione e temperatura vengono forniti con due byte in cui il bit più significativo del MSB ha un valore di 2^{20} e il resto a scalare. Il valore quindi è scritto su 20 bit, di cui i 4 meno significativi (che corrispondono al XLSB) sono tutti a zero.
L'umidità è invece scritta su due byte e quindi il valore non compensato totale è su 16 bit.
In ogni caso i tre valori letti dagli ADC vengono salvati su dei registri a 32 bit in modo da poter svolgere i calcoli.

Tramite la misura di pressione si può ricavare un valore più o meno plausibile di altitudine, in metri sul livello del mare, a cui si trova il sensore. Si utilizza l'espressione seguente:

```
// Returns temperature in DegC, resolution is 0.01 DegC. Output value of "5123" equals 51.23 DegC.
// t_fine carries fine temperature as global value
int32_t t_fine;
var1_t = (((adc_T>>3) - (((int32_t)dig_T1)<<1)) * ((int32_t)dig_T2)) >> 11;
var2_t = (((((adc_T>>4) - ((int32_t)dig_T1)) * ((adc_T>>4) - ((int32_t)dig_T1)))>> 12) * ((int32_t)dig_T3)) >> 14;
t_fine = var1_t + var2_t;
temp = (t_fine * 5 + 128) >> 8;

// Returns pressure in Pa as unsigned 32 bit integer in Q24.8 format (24 integer bits and 8 fractional bits).
// Output value of "24674867" represents 24674867/256 = 96386.2 Pa = 963.862 hPa
int64_t var1_p, var2_p,  p;
var1_p = ((int64_t)t_fine) - 128000;
var2_p = var1_p * var1_p * (int64_t)dig_P6;
var2_p = var2_p + ((var1_p*(int64_t)dig_P5)<<17);
var2_p = var2_p + (((int64_t)dig_P4)<<35);
var1_p = ((var1_p * var1_p * (int64_t)dig_P3)>>8) + ((var1_p * (int64_t)dig_P2)<<12);
var1_p = (((((int64_t)1)<<47)+var1_p))*((int64_t)dig_P1)>>33;
if (var1_p == 0) return 0; // avoid exception caused by division by zero
p = 1048576-adc_P;
p = (((p<<31)-var2_p)*3125)/var1_p;
var1_p = (((int64_t)dig_P9) * (p>>13) * (p>>13)) >> 25;
var2_p = (((int64_t)dig_P8) * p) >> 19;
p = ((p + var1_p + var2_p) >> 8) + (((int64_t)dig_P7)<<4);
press = (int32_t)p / 256.0;

// Returns humidity in %RH as unsigned 32 bit integer in Q22.10 format (22 integer and 10 fractional bits).
// Output value of "47445" represents 47445/1024 = 46.333 %RH
int32_t v_x1_u32r;
v_x1_u32r = (t_fine - ((int32_t)76800));
v_x1_u32r = (((((adc_H << 14) - (((int32_t)dig_H4) << 20) - (((int32_t)dig_H5) * v_x1_u32r)) + ((int32_t)16384)) >> 15) *
            (((((((v_x1_u32r *((int32_t)dig_H6)) >> 10) * (((v_x1_u32r * ((int32_t)dig_H3)) >> 11) +((int32_t)32768))) >> 10) +
            ((int32_t)2097152)) * ((int32_t)dig_H2) +8192) >> 14));
v_x1_u32r = (v_x1_u32r - (((((v_x1_u32r >> 15) * (v_x1_u32r >> 15)) >> 7) *  ((int32_t)dig_H1)) >> 4));
v_x1_u32r = (v_x1_u32r < 0 ? 0 : v_x1_u32r);
v_x1_u32r = (v_x1_u32r > 419430400 ? 419430400 : v_x1_u32r);
humid = (int32_t)(v_x1_u32r>>12) /1024.0;
```

Figura 2.14: Algoritmo di compensazione delle grandezze rilevate

$$altitudine = 44330 \times (1 - (\frac{press}{p0})^{\frac{1}{5.255}}) \text{ [m slm]} \tag{2.1}$$

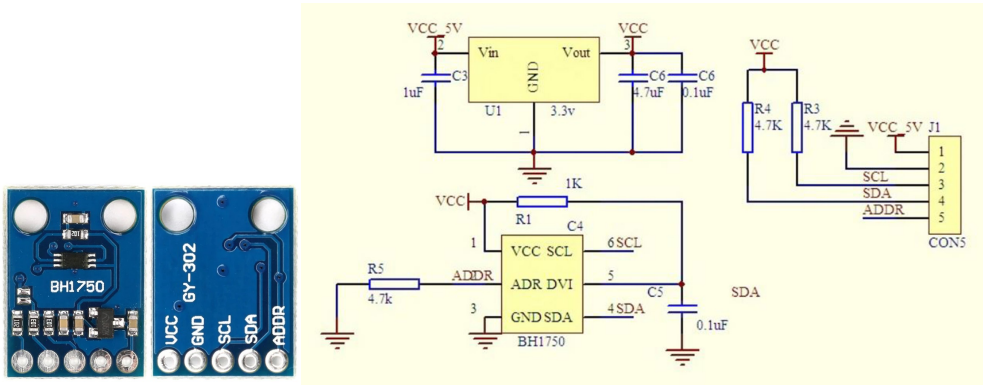
dove $press$ è il valore misurato e $p0$ è la pressione a livello del mare che è stata ipotizzata al suo valore standard, ossia 101325 Pa.
Il risultato che si ottiene è, tuttavia, molto dipendente dalle condizioni atmosferiche del momento (nuvoloso, soleggiato,...) e quindi può discostarsi abbastanza dal valore reale.

2.4 Sensore di luminosità

2.4.1 Panoramica

Per rilevare la luminosità ambientale si è utilizzato il sensore BH1750, prodotto da Rohm Semiconductor[8]. Esso è molto utilizzato per la gestione dell'illuminazione degli schermi LCD negli smartphone ed è dotato di un'interfaccia I2C (normal oppure fast) per la comunicazione.
Anche in questo caso si è acquistata una schedina pre-assemblata (GY-302) su cui è montato il sensore insieme a un circuito di interfacciamento per il collegamento diretto con un qualsiasi μControllore (comprende quindi regolatore di tensione, resistenze e condensatori). La scheda e il corrispondente schema circuitale sono visibili in figura 2.15.

[8] Anche in questo caso si rimanda al datasheet per maggiori informazioni sul componente



(a) *La scheda GY-302* (b) *Schema circuitale della scheda*

Figura 2.15: Scheda GY-302

Parametro	Valore	Unità di misura
Tensione alimentazione	da 2.4 a 3.6	V
Corrente assorbita	120 (VCC=3V, 100lux, 25°C)	μA
Risoluzione H-Res	1	lux
Risoluzione L-Res	4	lux

Tabella 2.5: Principali caratteristiche del BH1750

Le principali caratteristiche del sensore sono riassunte in tabella 2.5.

Una volta alimentato il sensore, esso si trova nello stato di Power Down; a questo punto si possono scegliere due modalità di funzionamento:

Continuously Il sensore continuamente rileva il valore di luminosità e lo salva nei registri;

One Time Il sensore effettua una misurazione e dopo ritorna nello stato di Power Down fino a un prossimo comando.

Nella modalità One Time si riduce notevolmente il consumo di potenza e per questo verrà utilizzata questa modalità.
Il tempo di misurazione non è costante e varia a seconda della risoluzione che si vuole ottenere. La tabella 2.6 illustra le diverse tipologie di risoluzione e i relativi tempi di misurazione.
Da datasheet viene consigliato di utilizzare la modalità H-Resolution in modo da riuscire a distinguere variazioni di luminosità anche con poca luce e quindi verrà utilizzata anche in questo progetto.

Modlità di Misurazione	Tempo	Risoluzione
H-Resolution Mode2	120 ms	0.5 lux
H-Resolution Mode	120 ms	1 lux
L-Resolution Mode	16 ms	1 lux

Tabella 2.6: Risoluzioni possibili

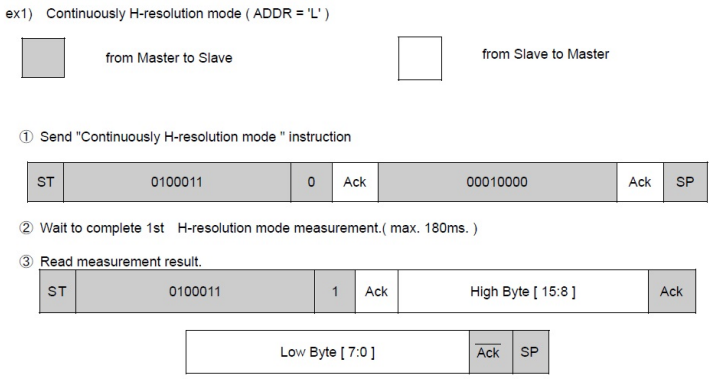


Figura 2.16: Procedimento per la lettura da sensore BH1750

2.4.2 Comunicazione I2C

L'indirizzo a 7 bit della periferica è ricavabile da datasheet e può assumere due configurazioni in base al collegamento del pin ADDR (presente anche sulla scheda GY-302):

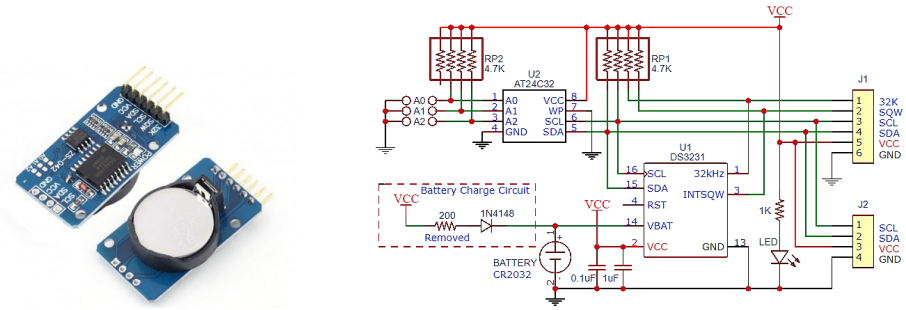
ADDR = 'HIGH' $\Rightarrow$ 1011100

ADDR = 'LOW' $\Rightarrow$ 0100011 (usato in questo progetto)

La lettura del valore di luminosità è piuttosto semplice e si compone prima da una fase di scrittura, in cui viene definita la modalità di funzionamento, dopodichè si attende il tempo di misurazione e infine si inizia con la fase di lettura dei due byte inviati dal sensore.
Il valore rilevato di luminosità, infatti, occupa 16 bit ed è quindi inviato con due byte. Prima viene inviato il byte MSB e dopo l'LSB. La figura 2.16 illustra il procedimento di lettura della luminosità nella modalità Continuously H-Resolution Mode[9]. Come ultimo passo, per calcolare il valore effettivo di lux, il datasheet fornisce un esempio con la seguente formula:
Avendo High Byte = "100000011" e Low Byte = "10010000" si avrà

$$valore = \frac{(2^{15} + 2^9 + 2^8 + 2^7 + 2^4)}{1.2} = 28067[lux] \tag{2.2}$$

[9] in questo progetto si è utilizzata la One time H-Resolution Mode. Sostituendo il comando 0x10 con il byte 0x20 si ottiene la modalità voluta



(a) *Modulo RTC* (b) *schema circuitale Modulo RTC*

Figura 2.17: Modulo RTC con DS3231

2.5 Modulo RTC

2.5.1 Panoramica

Per avere memorizzato in ogni momento la data e l'ora e così poterla visualizzare a display si è scelto di includere nel progetto un modulo RTC. Si è scelto il modulo DS3231 (figura 2.17a) anch'esso montato in una scheda pre-assemblata per facilitarne il collegamento con la Board di sviluppo. Il modulo DS3231 supporta una tensione di alimentazione continua all'interno del range 2.3V - 5.5V. É un prodotto a basso costo ma possiede un'elevata precisione. Sfrutta un protocollo di comunicazione I2C (Normal e Fast) e consente il collegamento di una batteria "tampone" per poter continuare a tenere attivo lo scorrere del tempo anche quando l'alimentazione principale viene tolta. L'alloggio per la batteria (del tipo CR2032) è installato nella scheda pre-assemblata come si nota appunto dalla figura 2.17a.
Oltre a questo e a una serie di resistenze e condensatori, nella schedina è presente anche una memoria EEPROM da 32k (con sigla AT24C32) che comunica sempre con protocollo I2C, ma non ha uno scopo ben preciso: è messa a disposizione in caso si abbia bisogno di salvare dati di qualunque genere. In questo progetto non verrà utilizzata e quindi non viene approfondita ulteriormente.
In figura 2.17b è possibile vedere lo schematico di tutta la scheda, distinguendo i vari componenti illustrati precedentemente. Si nota che sono presenti altri due pin nella schedina. Essi sono collegati ai medesimi pin del modulo DS3231 e svolgono funzioni diverse (che però non verranno sfruttate per questo progetto):

32K fornisce in uscita un clock con frequenza di 32kHz;

SQW può portare in uscita un'onda quadra di frequenza impostabile oppure può fungere come interrupt quando viene impostato un allarme in un certo orario. La funzione del pin è impostabile con un bit a registro.

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–31
05h	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06h	10 Year				Year				Year	00–99
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09h	A1M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + $\overline{AM}$/PM 00–23
0Ah	A1M4	DY/DT	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0Ch	A2M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + $\overline{AM}$/PM 00–23
0Dh	A2M4	DY/$\overline{DT}$	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31
0Eh	$\overline{EOSC}$	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12h	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

Figura 2.18: Mappa dei registri del modulo RTC DS3231

2.5.2 Registri e Comunicazione I2C

Il modulo DS3231 possiede una serie di registri per l'impostazione e visualizzazione della data, ora, temperatura e altri parametri, come visibile in figura 2.18. Quelli che verranno usati in questa esperienza riguarderanno l'ora, la data e la temperatura, in modo da riuscire a misurare anche i gradi centigradi presenti all'interno della stanza. Di conseguenza saranno utilizzati quelli compresi tra l'indirizzo 0x00 a 0x06 e i registri 0x11 e 0x12. I registri riguardanti l'ora e la data andranno impostati una volta (all'accensione del modulo), dopodichè, essendoci la batteria tampone, verranno aggiornati automaticamente dall'integrato stesso.
Si può notare che viene utilizzata la notazione BCD esadecimale per la visualizzazione di ora e data; ad esempio se sono le 13:24:45 i registri 0x00, 0x01, 0x02 conterranno rispettivamente 0x45, 0x24, 0x13[10].
Il bit 7 del registro 0x05 (Century) serve per capire quando il valore dell'anno (registro 0x06) supera 99. Quando questo succede, il bit Century cambia valore e il conteggio Year riparte.

La rilevazione della temperatura avviene in modo automatico ed è sufficiente leggere il valore salvato nei due registri. Essa ha una risoluzione di 0.25 °C e può oscillare nel range operativo del modulo DS3231. Non è una rilevazione

[10] il bit 6 del registro 0x02 viene settato a 0 per impostare la visualizzazione dell'orario nel formato a 24 ore e non a 12 ore

molto precisa, infatti il costruttore dichiara un'incertezza di ±3°C, tuttavia verrà utilizzata come un'indicazione in linea di massima della temperatura presente lato ricevitore.
Dal datasheet si capisce che nel registro 0x11 è memorizzato il valore intero (1 byte con segno) di temperatura, mentre nel successivo registro è salvato il valore decimale. Questo secondo registro può assumere solo 4 valori: 0x00, 0x40, 0x80, 0xC0 che corrispondono ad accostare rispettivamente 0.00 °C, 0.25°C, 0.50°C, 0.75°C al valore intero.

L'indirizzo I2C a 7 bit del modulo si ricava da datasheet. Esso è unico e fisso: 1101000. La comunicazione Master-Slave in scrittura e lettura avviene nel seguente modo:

Scrittura Il master, dopo l'invio dell'indirizzo dello slave e del bit di Write, invia l'indirizzo del registro in cui vuole scrivere. Il byte successivo verrà scritto nel corrispondente registro. Si possono inviare più byte consecutivi sapendo che dopo la ricezione di ogni byte, il modulo incrementa di uno il puntatore al registro. La sequenza è visibile in figura 2.19 (primo diagramma)

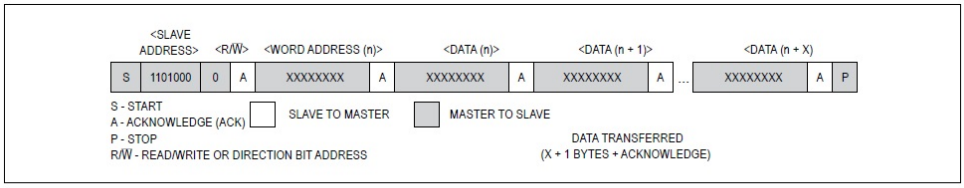
Lettura Può essere fatta in due modi:

- Dopo che il master invia l'indirizzo dello slave e il bit di Read, esso inizia a leggere byte partendo dal registro puntato nell'ultima operazione fatta e via via incrementandone il valore, come illustrato in figura 2.19 (secondo diagramma).
- Il master invia (modalità Write) prima un byte che rappresenta il puntatore al registro voluto, dopodiché inizia una nuova sequenza di lettura di un numero specificato di byte partendo dal registro appena puntato e incrementando di una unità ad ogni lettura. Questo è visibile in figura 2.20

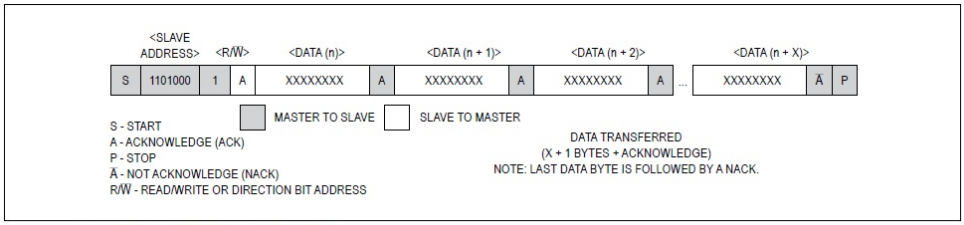
Per essere sicuri di stare leggendo i dati corretti, in questo progetto si è sempre utilizzata la procedura nella quale prima di leggere viene impostato il puntatore voluto.

2.6 Display

Per la visualizzazione dei dati atmosferici rilevati viene utilizzato il display oled SSD1306 da 0.96" (figura 2.21). Anche esso sfrutta il protocollo di comunicazione I2C con il Master.
Il core per la logica del componente supporta una tensione di alimentazione di 3.3V, ma sul retro della scheda (che comprende anche un circuito esterno al vero e proprio display) è montato un regolatore di tensione per potersi interfacciare con tensioni più elevate, ad esempio 5V, come in questo progetto.

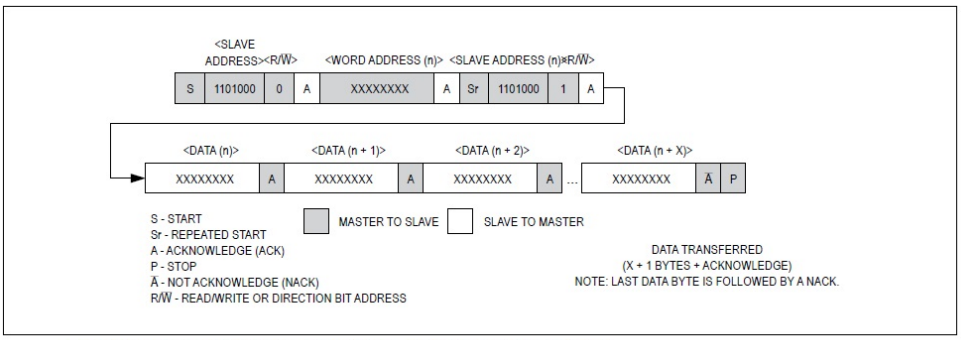


Data Write—Slave Receiver Mode



Data Read—Slave Transmitter Mode

Figura 2.19: Procedura di scrittura e lettura



Data Write/Read (Write Pointer, Then Read)—Slave Receive and Transmit

Figura 2.20: Procedura di lettura con inizializzazione del puntatore a registro

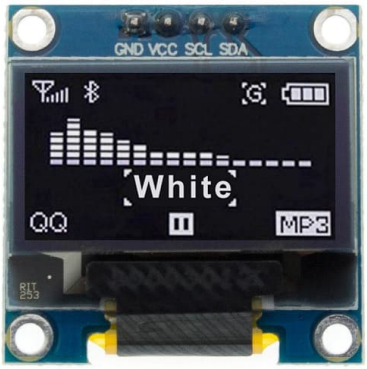


Figura 2.21: Display oled utilizzato con un esempio di possibile visualizzazione

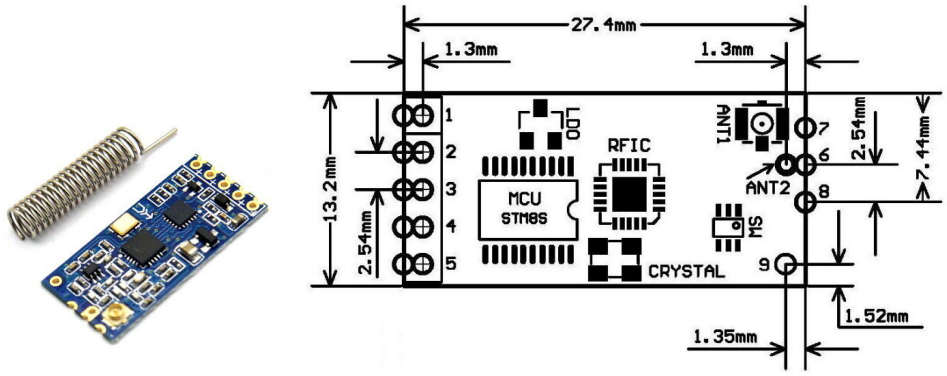
Consultando il datasheet del componente si osserva che la comunicazione "manuale" (ossia inviando manualmente pacchetti di byte) per visualizzare testo e numeri è assai complicata. In parole povere, infatti, si deve "accendere manualmente" ogni singolo pixel alzando il valore di un bit nella memoria del display, che è organizzata come una matrice 128 x 64 (che corrispondono ai 0.96").
Per questa ragione si è utilizzata una libreria apposita per semplificarne la programmazione. Essa contiene dei file .c (che contengono le funzioni utili alla scrittura di testo sul display), dei file .h (per la definizione dei caratteri) e un file di configurazione di tutte le caratteristiche dello stesso componente (indirizzo I2C, tipo di microcontrollore utilizzato come master...). Tutti questi File sono stati scritti dal creatore della libreria e quindi sono funzionanti e compilabili.
Quello che si deve fare è quindi includere nel progetto le libreria descritta e utilizzare le funzioni per la visualizzazione dei dati. In questo progetto si sono utilizzate quasi esclusivamente le funzioni *ssd1306_ SetCursor(uint8_ t x, uint8_ t y)* e *ssd1306_ WriteString(char* str, FontDef Font, SSD1306_ COLOR color)* rispettivamente per impostare la posizione del cursore dove scrivere e per scrivere una stringa di caratteri.
Infine la funzione *ssd1306_ UpdateScreen(void)* è stata utilizzata per aggiornare la visualizzazione in base al contenuto della memoria, la quale varia non appena cambiano i valori ricevuti.

L'indirizzo I2C è praticamente già fissato a 0x78, ma può essere modificato dissaldando una resistenza e mettendola tra altri due pin della scheda su cui è montato il display. In questa esperienza si è utilizzata la configurazione di default, ossia con l'indirizzo sopra citato.

2.7 Moduli RF

Per instaurare la comunicazione wireless tra le due schede, si è optato per l'utilizzo di due moduli a radiofrequenza. In particolare due HC-12 (figura 2.22a).
Su ognuno di essi è montato un ricetrasmettitore (Si4463), un microcontrollore (STM8S003FS) e altri componenti.
Le principali caratteristiche e il significato dei pin sono visibili nella tabella 2.7 e in figura 2.22b.

Il μControllore è programmato ad hoc per gestire il ricetrasmettitore, inviare/ricevere dati da esso e interfacciarsi verso l'esterno con un protocollo UART. La discreta complessità di questi moduli rende molto facile il loro utilizzo: è sufficiente inviare i dati al trasmettitore in modo seriale e leggerli all'uscita del ricevitore allo stesso modo.



(a) *Il modulo HC-12* (b) *Layout dell'HC-12*

Figura 2.22: Modulo RF

PIN	NOME	DESCRIZIONE
1	VCC	da 3.2V a 5.5V
2	GND	Ground
3	RX	UART input, 3.3V
4	TX	UART output, 3.3V
5	SET	pin per entrare in modalità AT
6	ANT	433MHz antenna pin
7	GND	Ground
8	GND	Ground
9	NC	-
ANT1	ANT	connettore antenna coassiale
ANT2	ANT	433MHz antenna hole

Tabella 2.7: Risoluzioni possibili

Su questi moduli, inoltre, è possibile programmare alcuni paramenti per la comunicazione come:

- baud rate per il protocollo UART;
- frequenza della portante del canale radio (da 433.4 MHz a 473.0 MHz a step di 400 kHz);
- modalità di trasmissione UART (a seconda della distanza voluta, e della velocità di comunicazione);
- altri parametri...

Per fare ciò vengono utilizzati i comandi AT, ponendo il pin SET a livello logico basso. Per questo progetto si è quindi impostata un'uscita digitale (in ognuno dei due μcontrollori) in modo da attivare o disattivare la modalità AT in base al suo valore logico (HIGH $\Rightarrow$ modalità di ricezione; LOW $\Rightarrow$ modalità AT).

Nel datasheet è descritta una modalità di trasmissione adatta per avere un basso consumo di potenza e di conseguenza si è optato per essa. Facendo ciò viene impostato automaticamente il baud rate della porta UART (4800 bit/s) e diminuiscono le prestazioni in termini di massimo raggio di trasmissione (si riduce a 100 m alla massima potenza di trasmissione).
Inoltre si è anche ridotta la potenza trasmessa dal modulo radio (di circa 6 dB), in modo da ridurne ulteriormente il consumo. Ovviamente si riduce il raggio di copertura di circa metà (quindi 50 m). La frequenza della portante del canale di comunicazione è 433.4 MHz.

Capitolo 3

Software utilizzati

3.1 STM32CubeMX

Molti costruttori di μControllori vengono incontro ai clienti fornendo programmi in grado di facilitare la programmazione dei loro prodotti. Questo è il caso del software STM32CubeMx. Grazie ad esso è possibile creare da zero un progetto compilabile e funzionante, che comprende numerosi file di codice. Questo insieme di moduli di codice ha lo scopo di inizializzare tutte le funzionalità che verranno utilizzate dall'utente durante il suo progetto. Senza l'aiuto del tool è molto laborioso settare manualmente tutti i parametri di inizializzazione della CPU e delle periferiche.

Mediante l'uso di STM32CubeMx, inoltre, si riesce a configurare la funzione che dovrà svolgere ciascun pin fisico dell'integrato. L'utente, in base a cosa ha bisogno, abilita o meno la funzionalità che gli serve e ne configura le caratteristiche.
Una volta finita questa parte di settaggio iniziale può essere generato il codice sorgente e da questo momento si passa nell'ambiente di sviluppo integrato, IDE (sezione 3.2) per scrivere le funzioni che svolgono il compito voluto.

3.1.1 Procedura di configurazione STM32F334R8

Una volta mandato in esecuzione il software si inizia un nuovo progetto scegliendo il μControllore che si vuole usare (in questo caso STM32F334R8) e dopodichè si aprirà una schermata dove viene visualizzato il modello del componente come in figura 3.1. Ora vanno configurati due pin per permettere la comunicazione Seriale con il PC e quindi anche il debug. Si clicca sul menù a tendina sula sinistra la voce *System Core>SYS* e alla voce debug si cambia da Disable a Serial Wire. Così facendo dovrebbero diventare verdi due pin del μControllore nel modello visibile in centro: PA13 e PA14.
Adesso si abilitano i due pin per la comunicazione I2C; sempre dal menù a sinistra si sceglie *Connectivity>I2C1* e lo si abilita. Si noterà che i pin

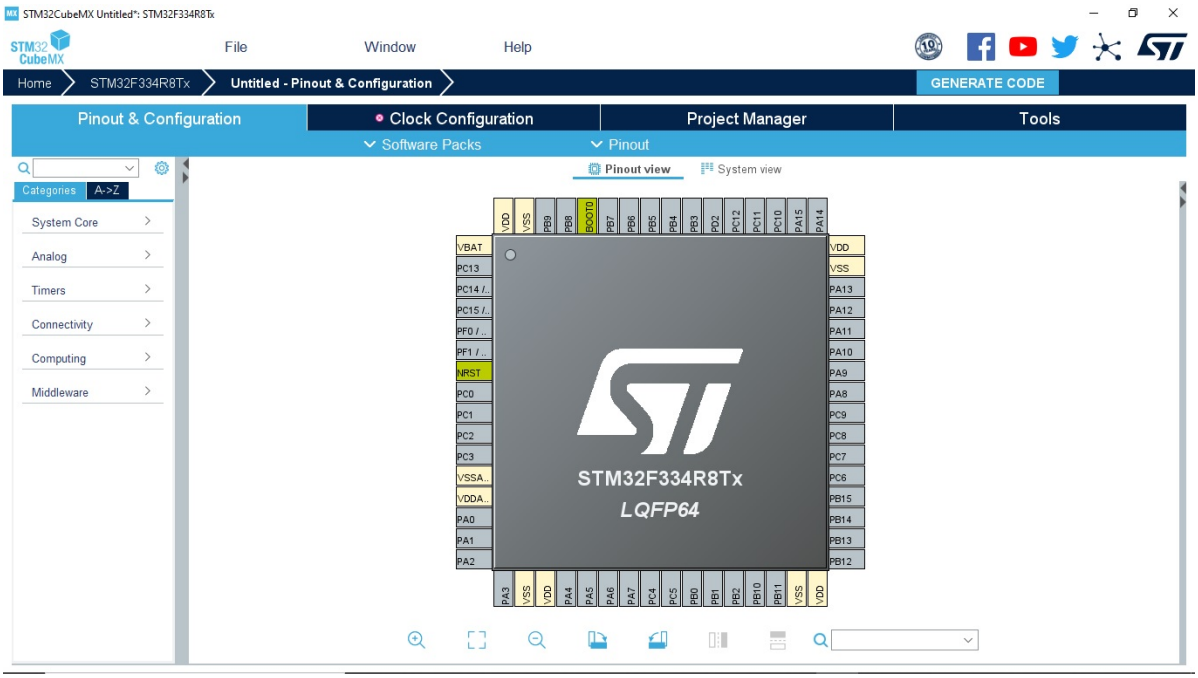


Figura 3.1: Schermata principale che si presenta dopo aver scelto il componente

PA15 e PB7 si accenderanno e assumeranno il ruolo rispettivamente di SCL e SDA. Tuttavia per questo progetto si vuole che i pin di SCL e SDA siano compatibili con la shield di Arduino UNO, quindi si impostano manualmente il pin PB8 come SCL e il pin PB9 come SDA cliccando direttamente sull'immagine dell'integrato. Si è optato per questa configurazione in modo da poter inserire la scheda METEO_RX_TOP nei connettori Arduino-UNO compatibile.
Per impostare la comunicazione UART si seleziona *Connectivity>USART1* e si sceglie la modalità asincrona, impostando poi il baud rate voluto e la modalità di trasmissione dei byte (vengono evidenziati i pin PB6-> USART1_TX e PA10 -> USART1_RX). Si sono assegnati poi due pin (PA5 e PC7) come output digitali, rispettivamente per comandare un led (LD2 presente sulla board) e per impostare o meno la modalità AT del modulo radio, e due pin (PA6 e PA9) di input sul quale sono collegati rispettivamente il pulsante atto a cambiare la visualizzazione della schermata del display e poter vedere le varie grandezze misurate e il pulsante per dare il comando di risveglio al μControllore STM32L031K6 che si trova in modalità di STOP[1] .

Infine si configura il clock abilitando prima di tutto la sorgente risonan-

[1]Questi pin vengono inoltre configurati nella modalità Pull-Down in modo da non dover aggiungere una resistenza verso massa vicino al pulsante. Infatti esso è presente internamente alla Board Nucleo

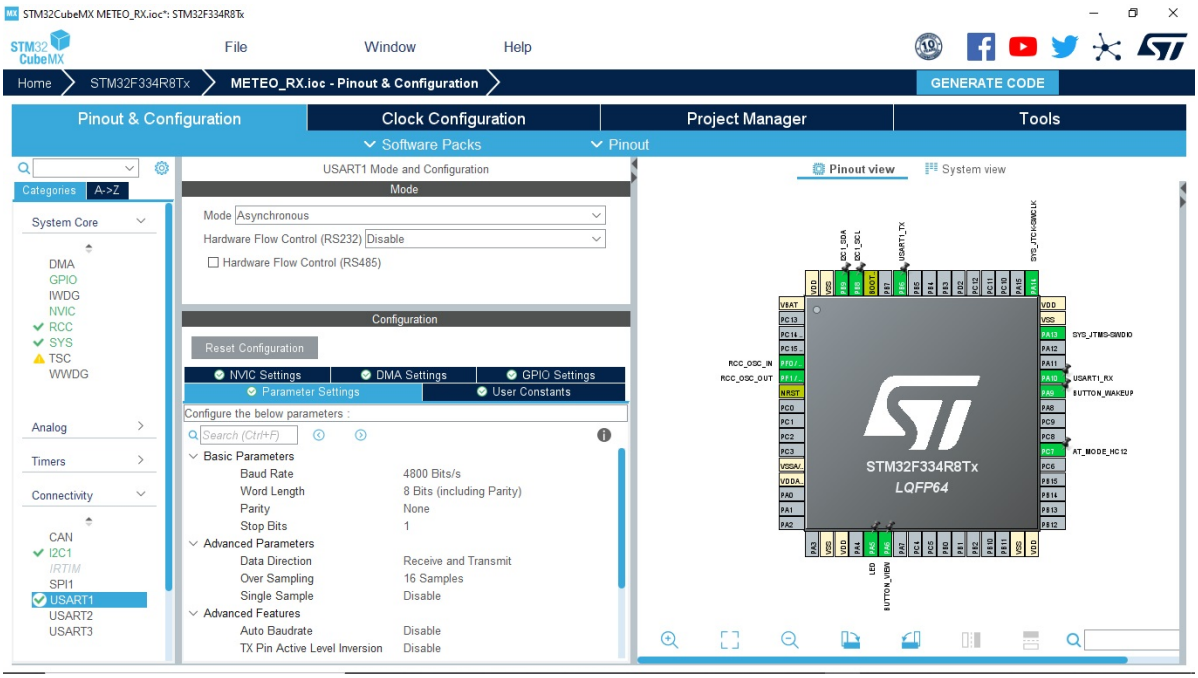


Figura 3.2: Microcontrollore STM32F334R8una volta abilitati i pin necessari

te esterna a 8 MHz andando sempre dal menù a sinistra alla voce *System Core>RCC* e selezionando *Crystal/Ceramic Resonator* alla voce HSE. Si accenderanno i due pin PF0 e PF1. In figura 3.2 si può vedere il modello del microcontrollore una volta impostati i pin necessari.
Per rendere il risonatore esterno effettivamente la sorgente di clock del μC si ricorre ad una specifica schermata di configurazione del clock. Essa è selezionabile nella parte superiore dello schermo alla voce *Clock Configuration*. Verrà visualizzato uno schema a blocchi che comprende diversi multiplexer, moltiplicatori e divisori di CLK in grado di poter scegliere quale sorgente di clock va a alimentare una linea specifica linea di bus e a quale frequenza. Per questo progetto si è configurato il clock come in figura 3.3.

3.1.2 Procedura di configurazione STM32L031K6

In questo caso si è proceduto come segue. Si sono assegnati:

- i pin per il debugging da PC sempre con il passaggio *System Core>SYS>Serial Wire* come nel caso precedente;
- i pin dell'interfaccia I2C (PA9 -> CLK, PA10 -> SDA);

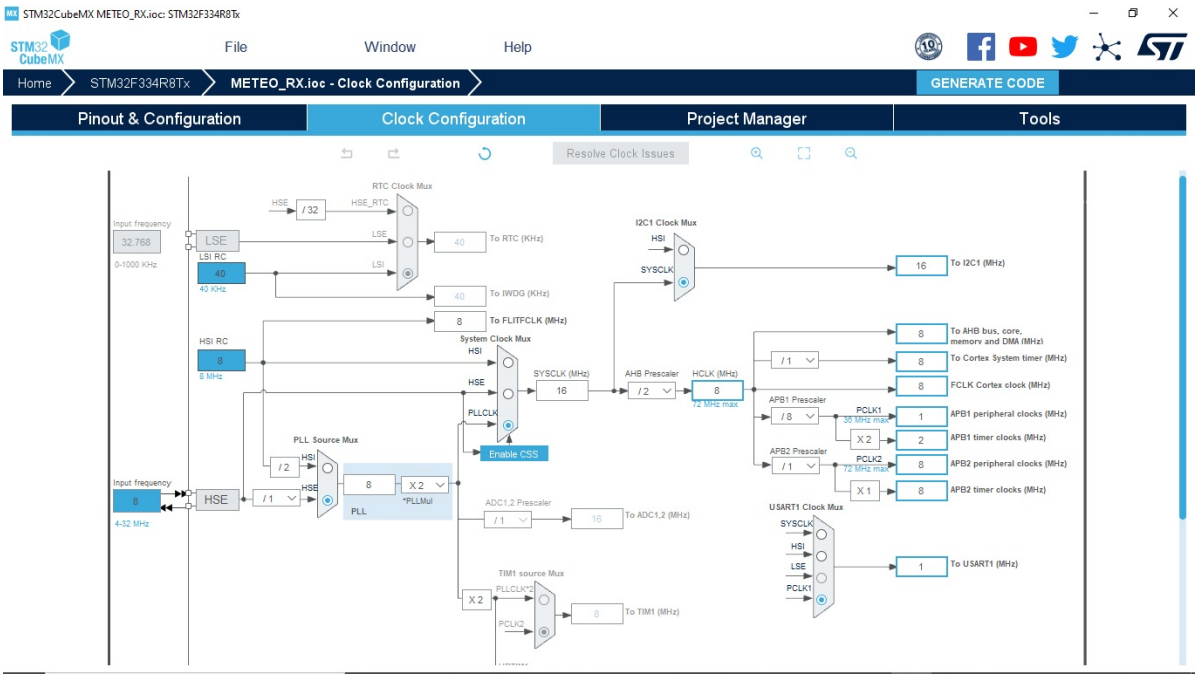


Figura 3.3: Finestra di configurazione del clock utilizzata per l'STM32F334R8

- i pin della la porta seriale LPUART1[2] (PA2 -> LPUART1_TX, PA3 -> LPUART1_RX) per la trasmissione dei dati;
- il pin PA7 come output digitale per selezionare o meno la modalità AT del modulo radio;
- il pin PB3 come output digitale su cui è connesso il led LD3 presente sulla board.

Non si sono assegnati i pin per l'ingresso del clock esterno a 8 MHZ in quanto per fare questo si dovrebbe modificare la configurazione Hardware di alcuni "ponti saldati" come specificato da datasheet. Per questo motivo si è impostata come sorgente di clock l'oscillatore interno all'integrato (HSI). In figura 3.4 e 3.5 si vedono l'assegnazione dei pin e la configurazione del clock dell'STM32L031K6[3].

[2]LPUART sta per LOW POWER UART

[3]Si nota, inoltre, che la sorgente di clock per la LPUART è direttamente l'HSI, in quanto è l'unica sorgente che resta "sveglia" quando il μcontrollore è in stop. Essa è quindi utilizzata per il risveglio dello stesso

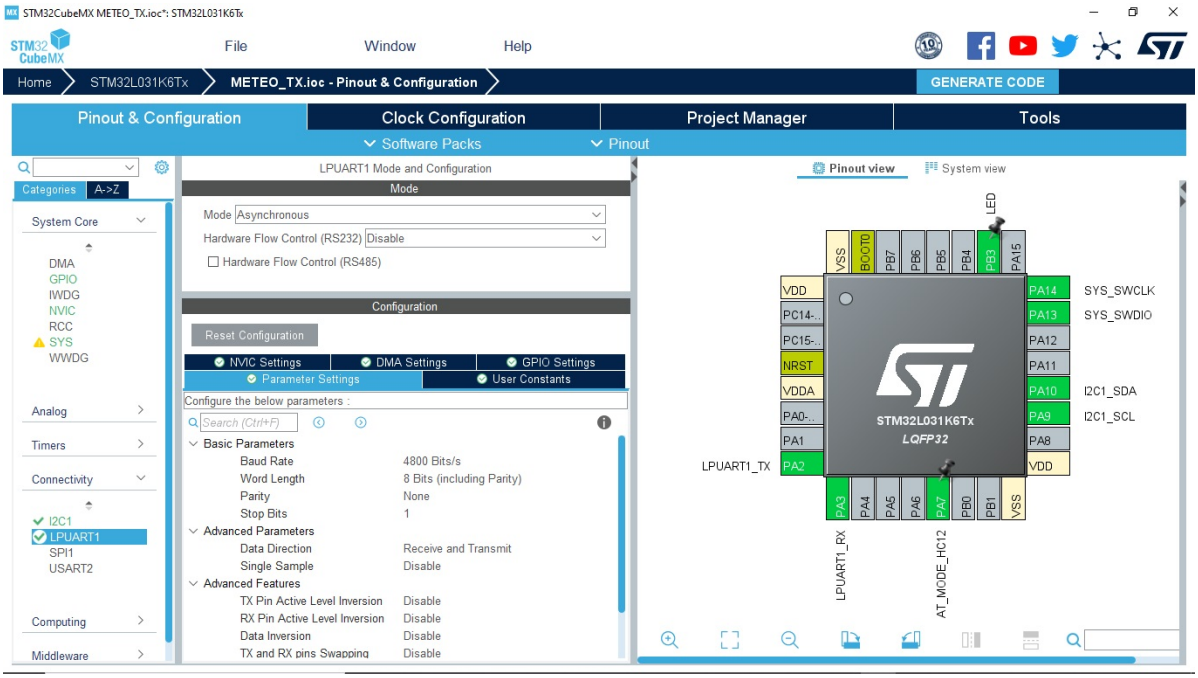


Figura 3.4: Configurazione dei pin del STM32L031K68

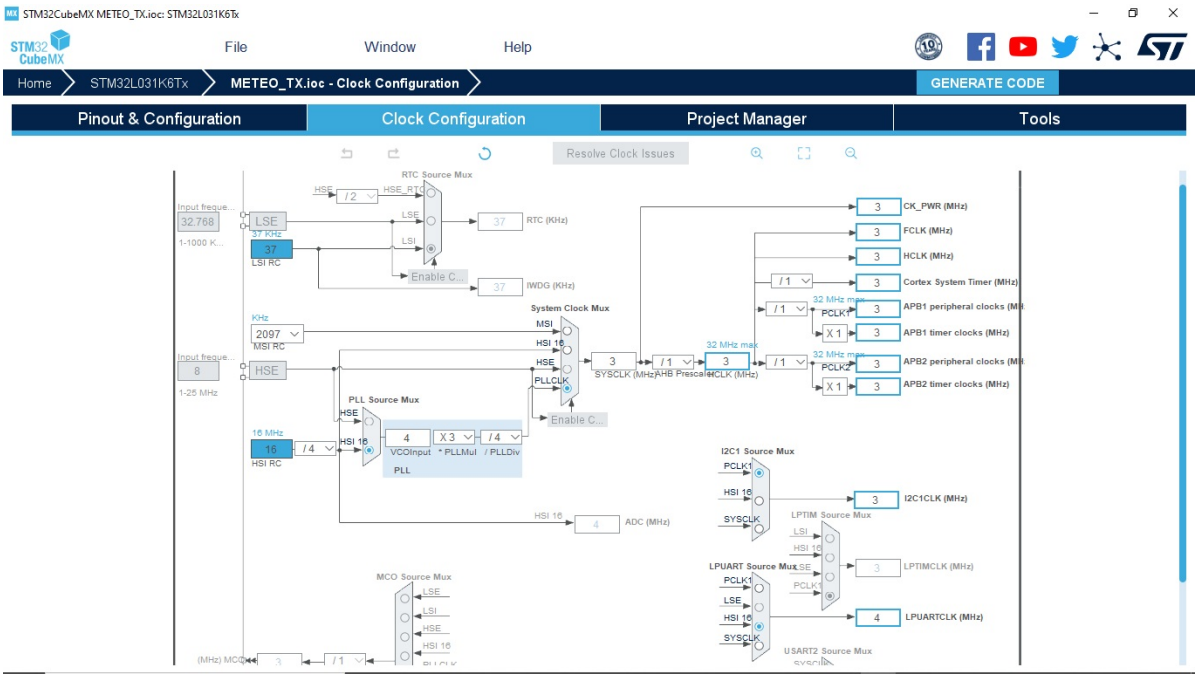


Figura 3.5: Configurazione del clock del STM32L031K68

3.2 Keil-μVision 5

Per la programmazione si possono utilizzare molti ambienti di sviluppo integrato (IDE) presenti sul mercato. Il software μVision fornito dall'azienda Keil è uno dei più diffusi per la programmazione di μControllori basati su core ARM. Si può scaricare gratuitamente dal web tenendo presente che vengono applicate delle limitazioni in termini di quantità di codice compilabile e trasferibile nella memoria flash (o RAM) del μC. Tuttavia in questo progetto le memoria utilizzata è assai inferiore ai limiti
Aprendo il file creato da STM32CubeMX si ottiene la schermata come in figura 3.6. Essa è così composta:

- sulla parte sinistra si osserva l'insieme di file .c (codice C) .h (header) .s (assembly) che compongono l'intero progetto;
- nella parte centrale si vede il contenuto dei file che compongono il progetto;
- nella parte inferiore c'è una finestra che riporta i passaggi effettuati dal software nei momenti di compilazione, linking e caricamento del codice. Vengono visualizzati inoltre eventuali errori nelle tre fasi;
- nella parte superiore ci sono i comandi necessari per la compilazione, caricamento, debugging e altre funzioni.

Il file principale è il main.c il quale viene sempre eseguito dal microcontrollore, dopo aver finito una fase iniziale di presettaggio. Di solito esso contiene diverse chiamate a funzioni di inizializzazione delle periferiche per poi entrare in un ciclo infinito e non uscirne più. In questo ciclo possono esserci delle istruzioni da compiere per il micro oppure no, nel caso in cui la gestione delle periferiche venga fatta attraverso interruzioni o DMA; in questo caso le vere e proprie istruzioni sono inserite nelle corrispondenti ISR (routine di servizio) delle interruzioni.
Sono presenti, poi, tutta una serie di moduli di configurazione delle periferiche in uso. Questi file fanno uso di apposite funzioni di libreria fornite dal costruttore per riuscire a inizializzare, settare e gestire le stesse. Nel caso della famiglia STM32 questo insieme di funzioni (driver) prende il nome di HAL[4].
Si notano inoltre le librerie (aggiunte manualmente) per la gestione della comunicazione con il display OLED.

Tramite il pulsante di debug si apre una nuova pagina di visualizzazione

[4] l'inizializzazione e il settaggio delle periferiche con le funzioni HAL viene fatto automaticamente dal software STM32CubeMX.

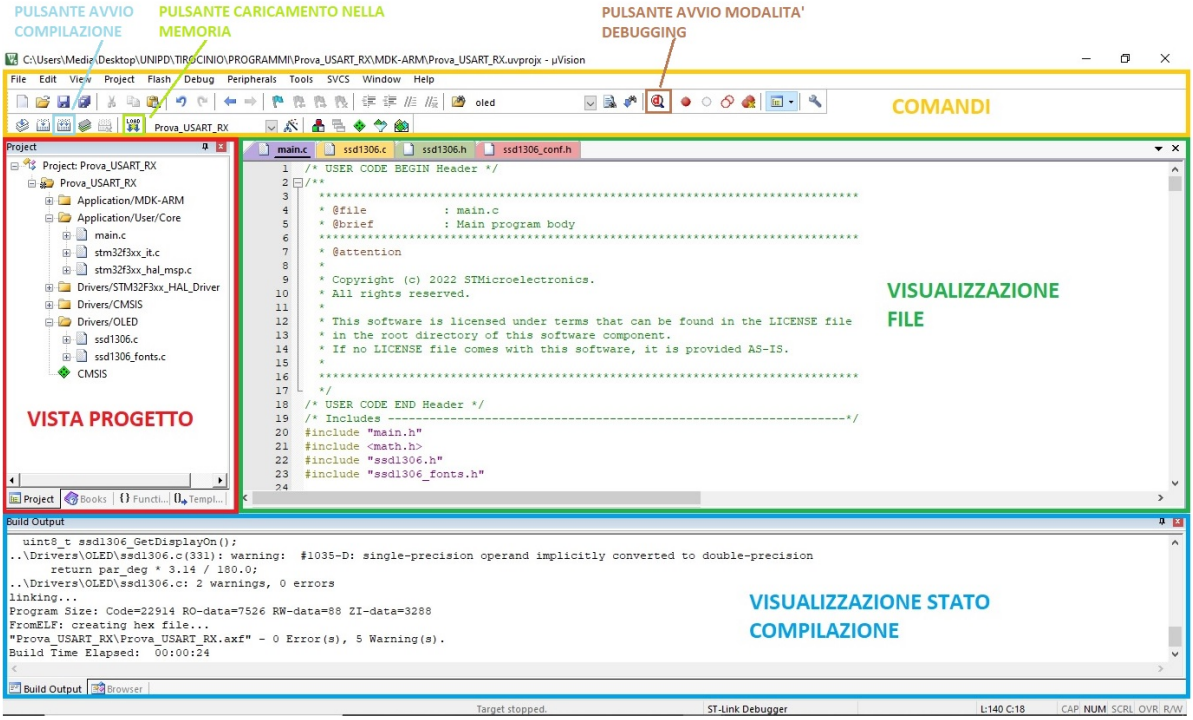


Figura 3.6: Schermata principale del IDE di sviluppo

(figura 3.7). In questa schermata si ha il controllo sullo stato del microcontrollore (run, stop, reset) e si riesce a visualizzare il valore in tempo reale[5] di ciascuna variabile definita nel codice. Si possono inoltre visualizzare: il codice disassemblato e quindi in linguaggio macchina, il valore dei registri principali del processore, il codice sorgente compilato e caricato, in cui è possibile impostare dei punti di blocco del codice (breaking-point) per vedere con maggiore precisione un'eventuale variazione di un dato o altro.

3.2.1 Struttura del codice

Sia per il trasmettitore che per il ricevitore il codice utile alla realizzazione delle funzionalità richieste per il progetto si sviluppa nella funzione main. Essa è composta da una prima parte eseguita sequenzialmente una sola volta ad ogni avvio della scheda e da una seconda parte eseguita ciclicamente infinite volte (loop infinito). Nella prima sezione si eseguono le funzioni di inizializzazione settaggio delle periferiche della scheda e dei componenti (setup), mentre nella sezione di loop si eseguono le attività di misurazione, visualizzazione, trasmissione dei dati periodicamente.

[5] In realtà non è corretto dire in tempo reale, in quanto sussiste un certo tempo di trasmissione tra la board e il PC. Se per esempio una variabile cambia valore troppo velocemente nella sezione di debug non si riescono a visualizzare tutti i cambiamenti

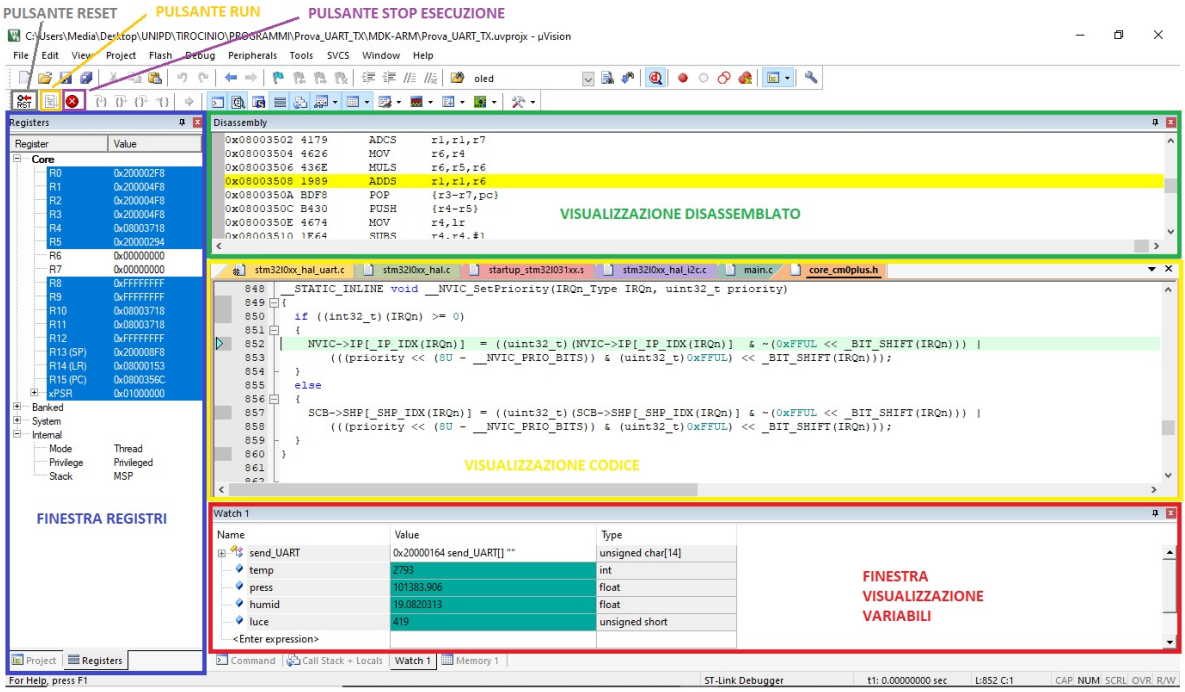


Figura 3.7: Schermata principale finestra di debug

Trasmettitore

Per il trasmettitore, nella parte di setup, dopo l'inizializzazione delle periferiche utilizzate dalla scheda (LPUART, I2C, GPIO, clock), si continua con l'impostazione dei parametri del modulo HC-12 tramite i comandi AT e con la lettura e il calcolo dei coefficienti di calibrazione del sensore BME280.
Nel loop, invece, si procede ciclicamente con:

- la lettura dei sensori;
- calcolo delle grandezze tramite gli algoritmi di compensazione;
- creazione dell'array di byte da inviare al trasmettitore radio;
- trasmissione pacchetto dati tramite protocollo seriale;
- delay di 4 secondi prima del prossimo ciclo.

Per l'invio e la ricezione di byte tramite protocollo seriale, si è utilizzata la modalità con interrupt, per evitare di avere dei ritardi nel compiere altre azioni.
Dopo una serie di 10 misurazioni e quindi di 10 invii di dati, il μcontrollore entra in modalità STOP, in attesa del segnale di risveglio da parte dell'altra scheda.

Ricevitore

Anche per il ricevitore, nella sezione di setup, vengono inizializzate le periferiche del μcontrollore, poi il display OLED e infine vengono impostati i parametri del modulo radio.
Dopodiché, ciclicamente ogni 5 ms, vengono rilevati i dati di ora e data dall'RTC e vengono visualizzati i valori atmosferici su display[6]. La ricezione di questi valori tramite UART avviene utilizzando, anche in questo caso, gli Interrupt in modo che l'aggiornamento delle variabili atmosferiche avvenga solo quando viene ricevuto completamente il pacchetto di byte, ossia all'incirca ogni 4 secondi.
Una volta ricevuti i 10 pacchetti di dati, la scheda resta in modalità RUN, con gli ultimi dati memorizzati e visualizzabili sul display. Quando viene premuto il pulsante per il risveglio, viene inviato un byte tramite la UART1 al modulo radio, che provvede a trasmetterlo all'altra board per risvegliarla dallo stop.

[6] Ovviamente i valori atmosferici resteranno fissi per almeno 4 secondi prima di un nuovo invio da parte del trasmettitore

Capitolo 4

Schemi elettrici - Schede

Per rendere più compatto l'intero Hardware del progetto e per non avere fili "volanti" per la connessione tra le schede Nucleo e le varie periferiche, si è optato per la progettazione di tre schede stampate PCB sulle quali verranno alloggiati i vari componenti del sistema.

4.1 Trasmettitore

Di seguito sono riportati lo schema elettrico e il modello dello stampato della scheda trasmettitore METEO_TX (figura 4.1).
Si nota che sono stati predisposti dei jumper in modo da poter collegare in varie posizioni un amperometro per misurare l'assorbimento di corrente totale della scheda, della sola board nucleo e dei soli sensori (compreso il modulo radio). Se non vengono collegati misuratori di corrente, i ponti devono essere cortocircuitati.
L'alimentazione viene fornita da una batteria collegata al connettore a morsetto di ingresso. Subito dopo è presente le sezione con il regolatore di tensione a 5V per fornire tensione alla board e ai sensori. Sono stati inseriti alcuni condensatori di disaccoppiamento su entrambi i lati (ingresso e uscita) del regolatore di tensione. Inoltre, è presente un bottone per il reset del μControllore. Esso è anche presente sulla Board stessa.

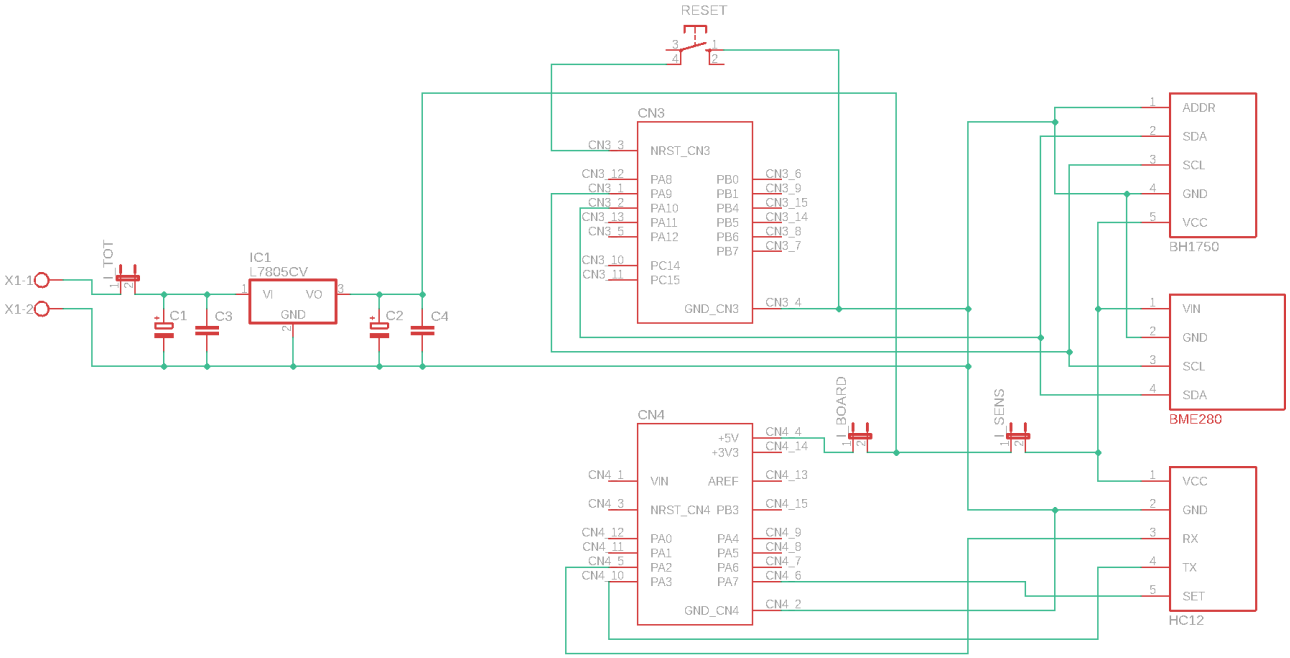
4.2 Ricevitore

Per il ricevitore si è pensato di sviluppare due diverse schede stampate: una che si installa sotto la board Nucleo tramite i connettori CN7 e CN10 (chiamata METEO_RX_BOTTOM) e una più piccola che si monta sopra della stessa tramite i connettori CN5, CN6, CN8, CN9 ossia quelli compatibili con Arduino UNO (chiamata METEO_RX_TOP)[1].

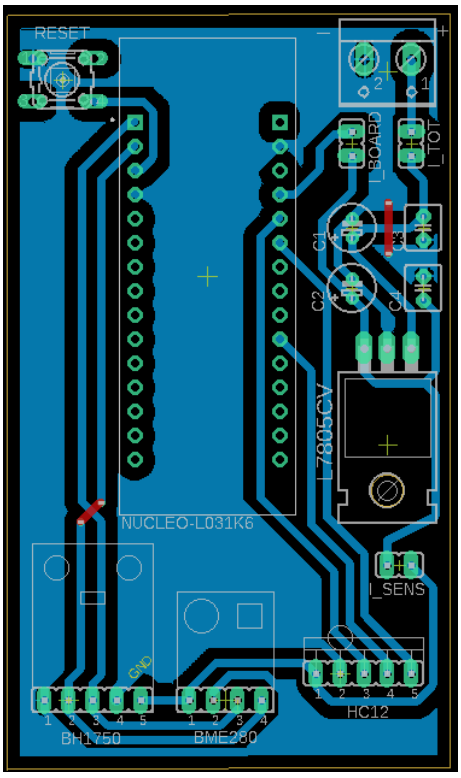
[1] in figura 2.4 sono visibili i diversi connettori menzionati

In figura 4.2 è visibile lo schema elettrico e lo stampato della scheda METEO_RX_BOTTOM, mentre quelli della scheda METEO_RX_TOP sono rappresentati in figura 4.3.
L'alimentazione può essere fornita attraverso il jack J1 con una tensione compresa tra 7V e 12V oppure attraverso il connettore USB (bisogna spostare nella corretta posizione il jumper JP5, come spiegato nella sottosezione 2.1.2). Il connettore J1 è connesso al pin VIN e prima di esso è presente un jumper per la misura della corrente assorbita totale (I_TOT)[2]. Si nota che in questa board non viene esclusa l'alimentazione della parte riguardante l'ST-Link, quindi esso è sempre attivo.
Nella scheda METEO_RX_BOTTOM sono presenti anche due led: LED1 che indica la presenza dei 5V sul pin della Board Nucleo e quindi l'alimentazione attiva della stessa e LED2 che evidenza l'arrivo di un pacchetto di dati al ricevitore.

[2] ovviamente se non viene inserito un amperometro il ponte deve essere cortocircuitato

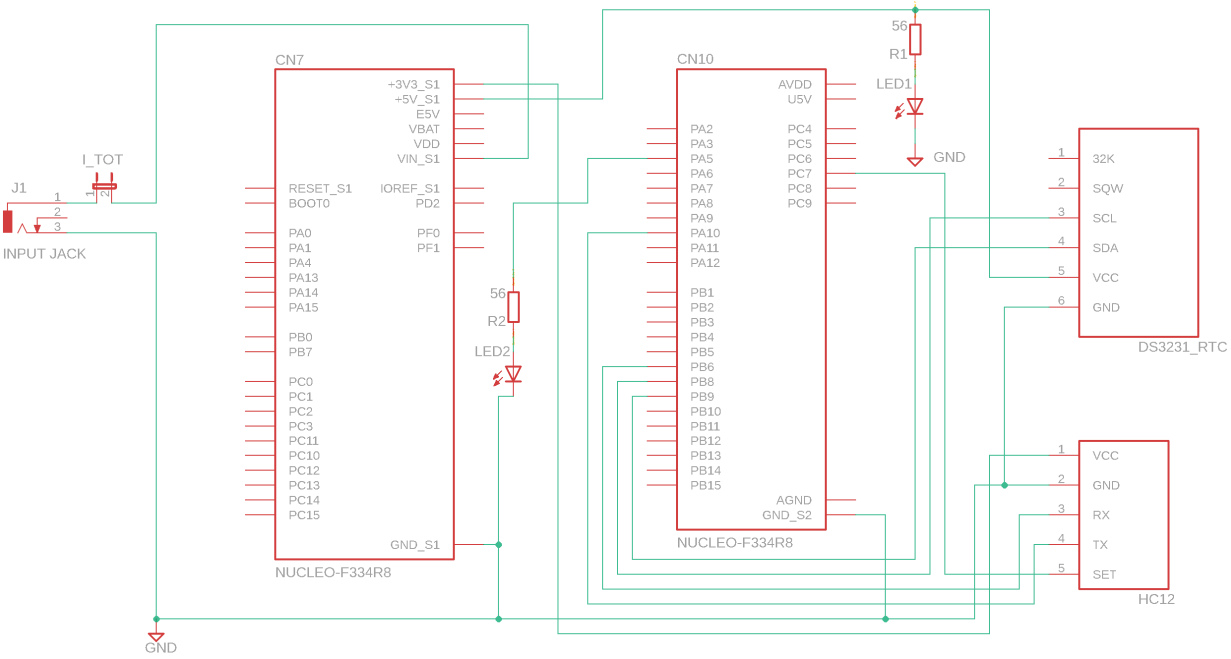


(a)

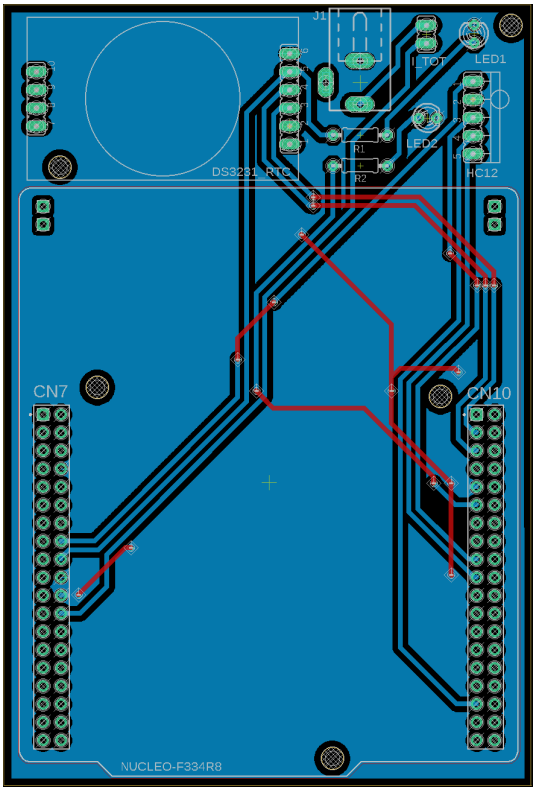


(b)

Figura 4.1: Scheda METEO_TX

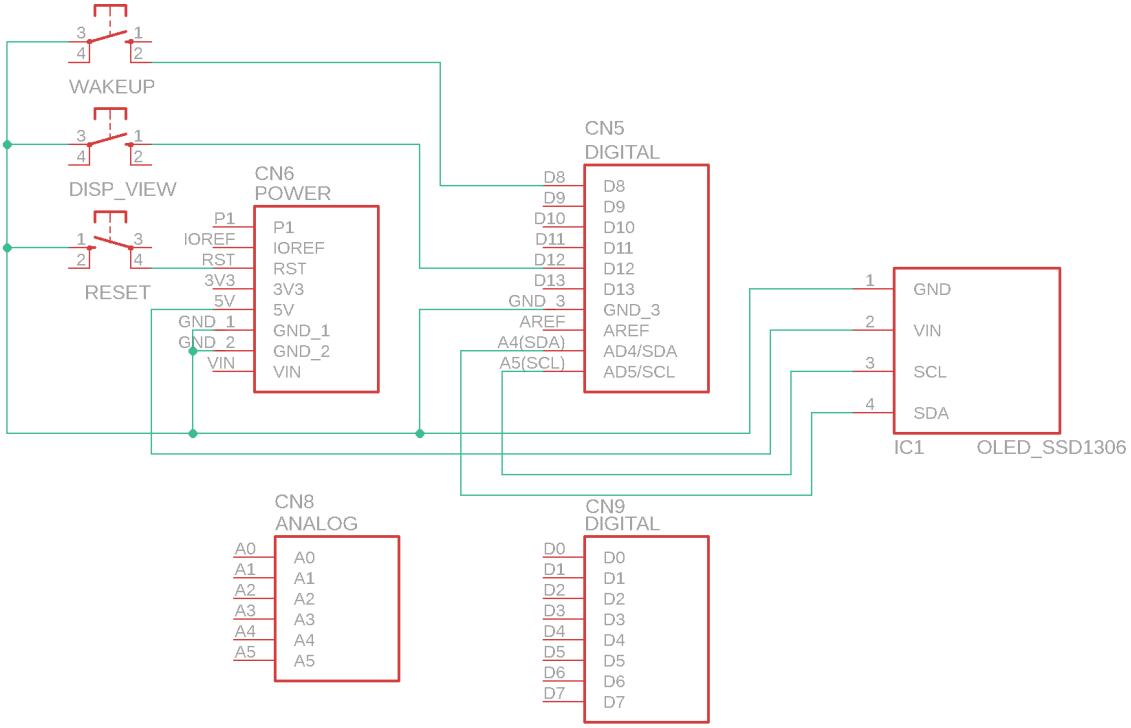


(a)

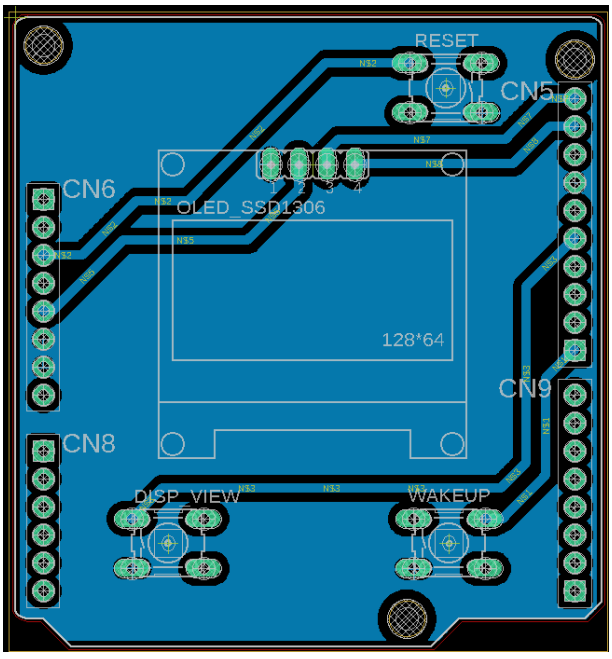


(b)

Figura 4.2: Scheda METEO_RX_BOTTOM



(a)



(b)

Figura 4.3: Scheda METEO_RX_TOP

Capitolo 5

Misure effettuate

5.1 Consumo di potenza

5.1.1 Trasmettitore

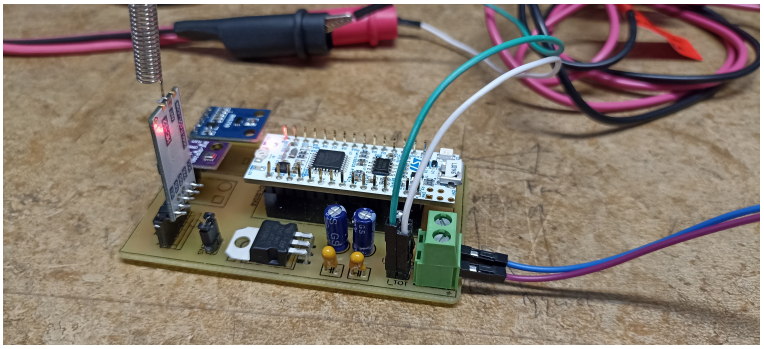
Come si vede dagli schemi nella sottosezione 4.1 è possibile effettuare la misura di corrente in tre diversi punti:

- Corrente totale assorbita dalla scheda METEO_TX tramite il jumper I_TOT;
- Corrente assorbita dalla sola Board Nucleo tramite il jumper I_BOARD;
- Corrente assorbita dai sensori e dal modulo radio tramite il jumper I_SENS.

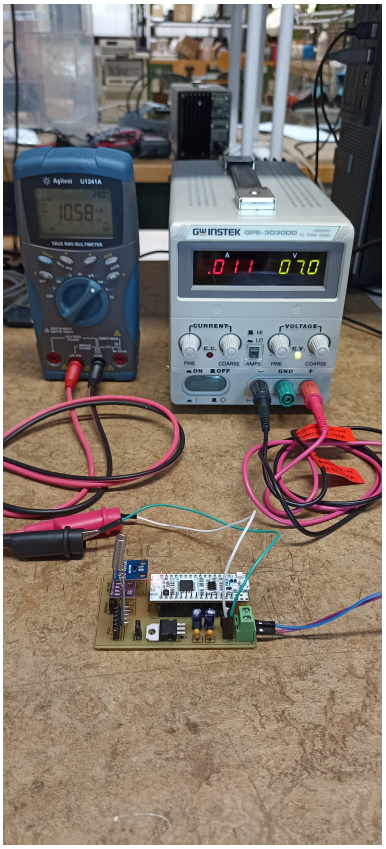
Di seguito sono ripotate valori rilevati alimentando con tensione di 7.0V:

MODE	JUMPER	VALORE	UNITA'
STOP MODE	I_TOT	11.00	mA
	I_BOARD	6.77	mA
	I_SENS	350 (picco)	μA
RUN MODE	I_TOT	11.22	mA
	I_BOARD	6.77	mA
	I_SENS	350 (picco)	μA
PICCO MAX	I_TOT	51	mA
	I_BOARD	7.30	mA
	I_SENS	40	mA

Nelle figure 5.1a e 5.1b si osservano i setup per la misura di corrente nella scheda METEO_TX. In questo caso è stato inserito un amperometro nel jumper I_TOT in modo da misurare l'intera corrente che deve erogare la batteria per alimentare tutti i componenti.
Analogamente si è inserito lo strumento anche negli altri spazi appositi.



(a)



(b)

Figura 5.1: Setup per la misura di corrente assorbita dalla scheda METEO_TX

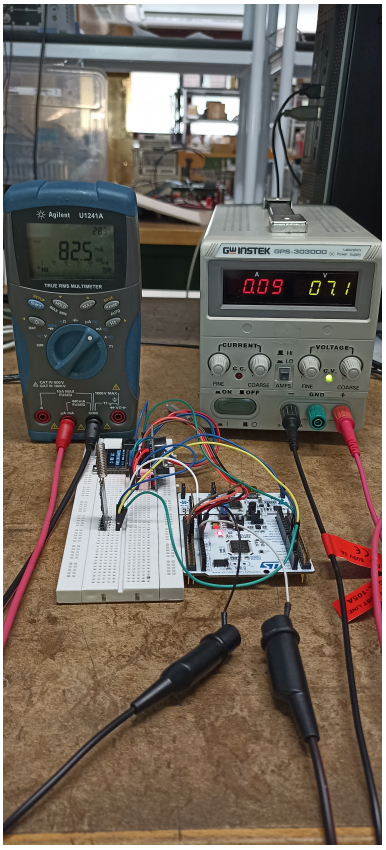


Figura 5.2: Setup di misura per la scheda METEO_RX

Con la misura della I_TOT si può dimensionare la batteria in base alle esigente dell'utente. Si è ipotizzato che il trasmettitore abbia un'autonomia di circa un mese. Considerando solamente il consumo di corrente durante la fase di stop (l'aggiornamento delle misure verrà effettuata solamente quando l'utente ne avrà bisogno e si considera non siano molte in una giornata) si può dimensionare la batteria come segue.
Si sceglie una batteria ai polimeri di litio costituita da più celle in modo da arrivare a una tensione di 7.4V. Si dimensiona poi la capacità:

$$Capacita' = 11.00[mA] * 24[ore] * 31[giorni] = 8184mAh$$

In commercio si trovano celle da 3.2 V a 10Ah. Mettendo due di queste celle in serie si arriverebbe a una tensione di 7.4 V con la stessa capacità.

5.1.2 Ricevitore

La scheda ricevente è stata testata a diverse tensioni di alimentazione comprese tra 7V e 12V senza avere cambiamenti nell'assorbimento di corrente.

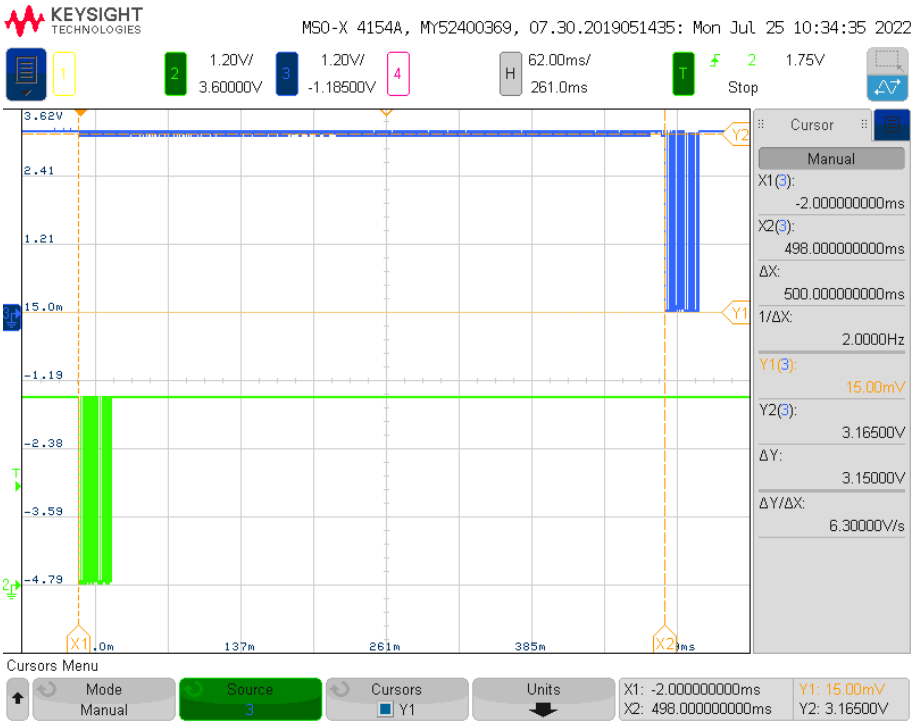


Figura 5.3: Invio e ricezione di un pacchetto di dati

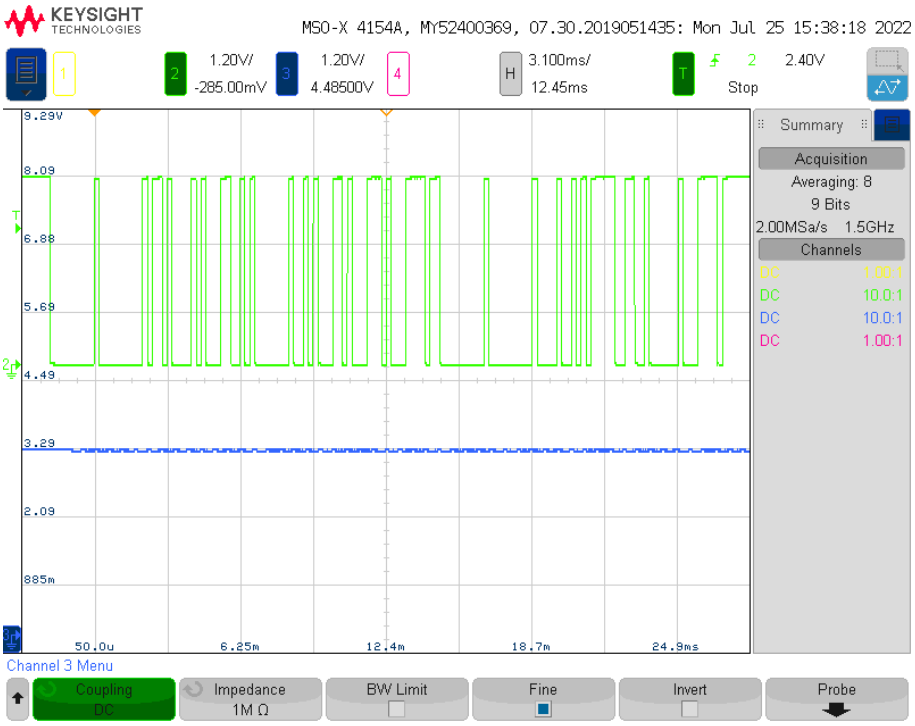
Si è effettuata la misura di assorbimento di corrente totale della scheda:

MODE	CORRENTE	VALORE	UNITA'
ATTESA DATI	I_TOT	61 - 73	mA
RICEZIONE DATI	I_TOT	88 (picco)	mA
INVIO WAKEUP	I_TOT	114 (picco)	mA

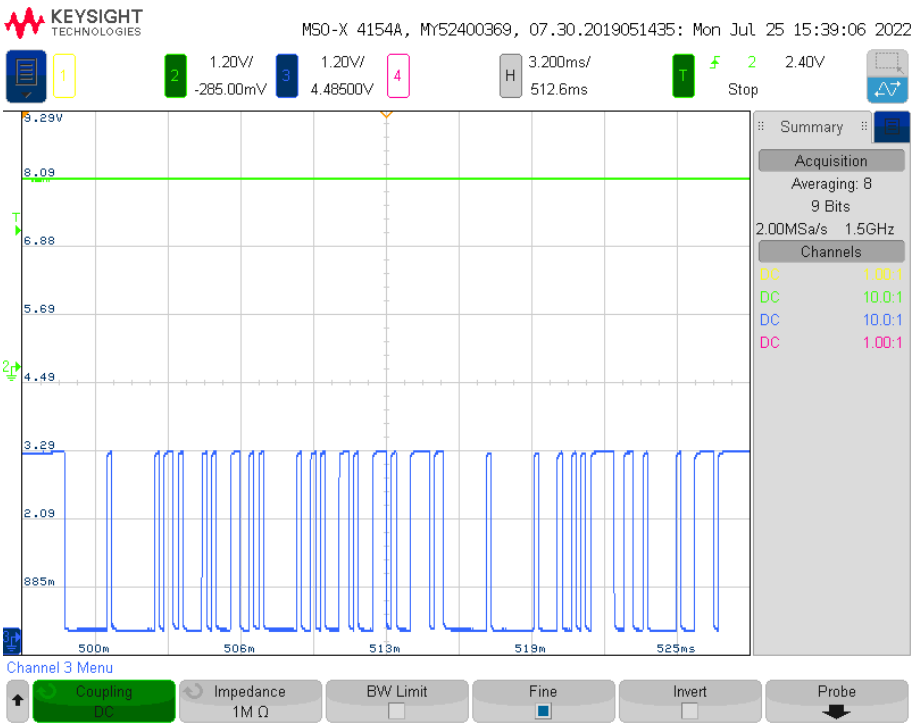
In figura 5.2 si osserva il setup di misura per la scheda ricevente. Si sono effettuate le misure quando ancora l'hardware era provvisorio, quindi si vedono la presenza di cavetti e breadboard.

5.2 Trasmissione dati

Tramite un oscilloscopio si sono visualizzati alcuni segnali di trasmissione dei dati sul canale Seriale (UART). Si sono posizionate due sonde rispettivamente sulla linea TX della Board Nucleo-L031K6 (trasmettitore dei dati atmosferici) e sulla linea RX della Nucleo-F334R8 (ricevitore).
Nella figura 5.3 si osserva il segnale trasmesso in verde e quello ricevuto in blu, ovviamente con un certo ritardo. Quando non viene trasmesso niente la



(a)



(b)

Figura 5.4: Focus sul pacchetto di dati inviato e ricevuto

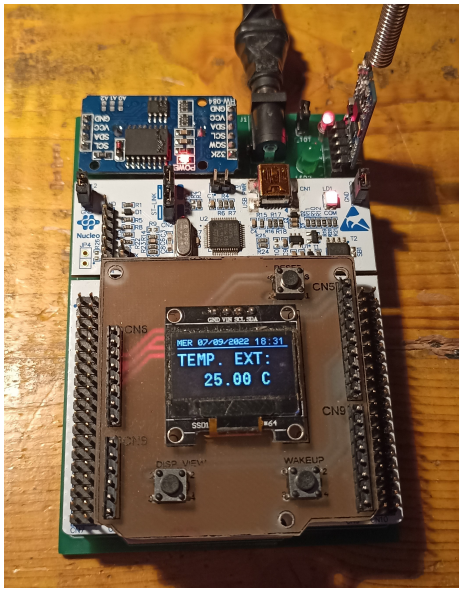
linea resta ad un valore stabile di 3.3V; L'invio di ogni byte inizia con una sequenza di start, portando la linea a un livello basso, e si conclude con un comando di stop, riportandola nello stato alto.

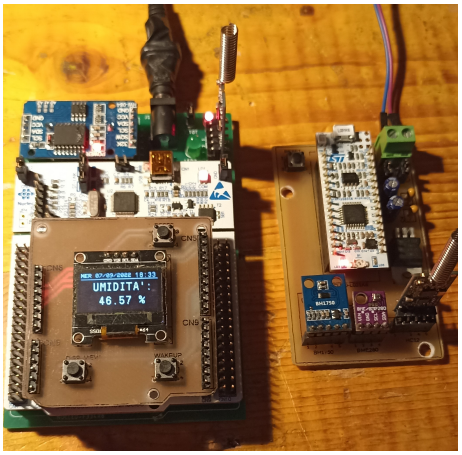
Aumentando la scala temporale si osservano meglio le transizioni di livello dei segnali. Confrontando le due figure 5.4a e 5.4b si nota l'uguaglianza del segnale trasmesso con quello ricevuto, completando la trasmissione wireless.

Capitolo 6

Conclusioni

Con la realizzazione di questo progetto si sono incrementate le conoscenze riguardanti l'impostazione e la programmazione dei microcontrollori della famiglia STM32. In particolar modo l'uso del software STM32CubeMX per l'impostazione dei pin del μC e la definizione delle periferiche.
Si sono poi acquisite delle competenze nella progettazione di PCB ad hoc per l'alloggiamento dei componenti in maniera solida e compatta.
Si è lavorato, inoltre, per dimensionare la batteria di alimentazione del trasmettitore. Questo però si è limitato ad essere un calcolo solamente teorico in quanto, non possedendo una batteria di prova, non è stata verificata nella pratica la reale durata della stessa. Perciò si è studiato il modo per ridurre al minimo il consumo di potenza dell'apparato trasmittente (imparando ad utilizzare le modalità a bassa potenza del μcontrollore e adottando altre strategie di progetto) essendo questo un aspetto fondamentale per riuscire a creare un sistema alimentato a batteria. Le misurazioni fatte, infine, hanno confermato il risparmio di energia voluto.
Il risultato finale del progetto si può vedere dalle seguenti figure.


MER 07/09/2022 18:31
TEMP. EXT:
25.00 C
WAKEUP


UMIDITA':
46.57 %