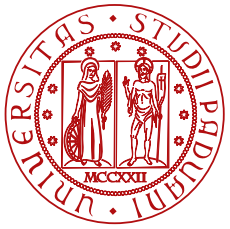

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Tesi di Laurea Triennale in Ingegneria dell'Informazione

Progettazione e sviluppo di un sistema integrato di misura per l'acquisizione di parametri audio e ambientali

Laureando

Giacomo De Toni

Numero di matricola 2009762

Relatore

Prof. Alessandro Pozzebon

Università di Padova

Co-relatore

Ing. Giacomo Peruzzi

Università di Padova

Anno accademico 2023/2024

Data di Laurea 14/03/2024

C'è vero progresso solo quando i vantaggi di una nuova tecnologia diventano per tutti.
Henry Ford

Abstract

Il seguente progetto si occupa dello **sviluppo di un nodo sensore**, o sistema integrato di misura, utile ad acquisire dati che contribuiranno al popolamento di un dataset per un algoritmo di Machine Learning Embedded (TinyML) già sviluppato.
Gli **obiettivi del progetto** sono tre:

1. **Popolamento del dataset**;
2. **Riduzione dei costi** di realizzazione;
3. **Aumento dell'efficienza energetica** per il funzionamento del nodo.

Il nodo dovrà acquisire **dati ambientali** di temperatura, umidità e pressione, e **dati acustici** di monitoraggio del rumore generato dal traffico veicolare in ambiente antropizzato, che contribuiranno al popolamento del dataset prima nominato.
L'algoritmo di TinyML, sfruttando i dati raccolti, **stima la quantità di polveri sottili** presenti nell'aria in aree particolarmente trafficate e probabilmente più inquinate.
Il sistema di misura sarà costituito da:

- un microcontrollore;
- un microfono con adeguato circuito di condizionamento;
- dei sensori digitali per la misurazione dei parametri ambientali;
- un modulo per la comunicazione wireless tramite rete ad ampio raggio (LoRaWAN);
- un sistema di alimentazione.

Indice

Lista delle Figure

Lista dei Frammenti di Codici

Lista degli Acronomi

CS Chip Select

CSS Chirp Spread Spectrum

EDA Electronic Design Automation

GND Ground

I2C Inter Integrated Circuit

IDE Ambiente di Sviluppo Integrato

LoRaWAN rete ad ampio raggio

MAC Media Access Control

MISO Master Input Slave Output

MOSI Master Output Slave Input

PCB circuito stampato

SCL Serial Clock

SDA Serial Data

SPI Serial Peripheral Interface

TinyML Machine Learning Embedded

1 Introduzione

Questa tesi si concentra sulla **progettazione e realizzazione di un nodo sensore**, o sistema integrato di misura. I dati raccolti da questo sistema contribuiranno al **popolamento di un dataset** per un algoritmo di TinyML[1], realizzato in un precedente progetto di tesi [1].
Nella tesi citata si adottano principalmente comandi via software per eseguire le misurazioni audio e si popola il dataset solo con misurazioni acustiche e di particolato tramite il sensore HPMA115S0.
In questo progetto si è voluto **sostituire il software** per le misurazioni audio, **con hardware dedicato** ed **arricchire** il dataset **con dati ambientali** ottenuti da sensori digitali.
Le misure verranno poi inviate ad un server tramite **LoRaWAN**[2], dove si combineranno per formare il dataset e **stimare la quantità di polveri sottili nell'aria**.
Gli obiettivi del progetto sono:

1. **Popolamento del dataset** per l'algoritmo di TinyML;
2. **Riduzione dei costi** di realizzazione;
3. **Aumento dell'efficienza energetica** per il funzionamento del nodo.

[1] Il TinyML, o Machine Learning Embedded, è una particolare branca del Machine Learning e viene sviluppato e pensato per funzionare anche in microcontrollori con ridotte potenze di calcolo rispetto ai processori dei computer standard.

[2] LoRaWAN è in realtà un protocollo Media Access Control (MAC) sul quale il livello fisico di rete LoRa si basa. La rete LoRa sfutta la modulazione di frequenza Chirp Spread Spectrum (CSS) ed è una rete a bassisimi costi.

Per cercare di raggiungere tali obiettivi, rispetto al precedente progetto, sono stati utilizzati i seguenti componenti per il nodo sensore:

- **Microcontrollore ATTiny84**: rispetto ad Arduino Uno o Rasberry PI, garantisce **consumi energetici e costi inferiori**;
- **Sensore microfonico Sparkfun BOB-12758**: sostituisce una scheda arduino con microfono incorporato;
- **Sensori digitali** aggiunti: **MS5803** per pressione atmosferica e **DHT22** per temperatura e umidità;
- **Modulo RFM95**: utile per la **comunicazione wireless** tramite **LoRaWAN**;
- Sistema di **alimentazione a batteria**: garantisce il **funzionamento continuo** del modulo e la possibilità di **installazione anche lontano da fonti di alimentazione** esterne.

Si è deciso di **rimuovere il sensore di particolato HPMA115S0** affinché si possa ora **stimare la quantità di polveri sottili** sulla base delle nuove misurazioni.
Nel corso di questa relazione, verranno affrontati i seguenti argomenti:

1. **Progettazione dell'hardware**, sviluppo del PCB e realizzazione del case;
2. **Descrizione del codice** utilizzato per l'acquisizione dei dati e l'invio degli stessi tramite LoRaWAN
3. **Idee aggiuntive e progetti futuri**.

2

Progettazione e Realizzazione Hardware

Come introdotto, il sistema di misura integrato è composto da:

1. Microcontrollore **ATTiny84**;
2. Sensore microfonico **BOB-12758**;
3. Sensore di temperatura e umidità **DHT22**;
4. Sensore di pressione **MS5803**;
5. Modulo **RFM95**;

In questo capitolo verranno descritti in modo esaustivo i componenti utilizzati ed esposti i circuiti relativi per il corretto funzionamento del sistema motivando le scelte adottate ed i collegamenti effettuati.

2.1 Sensore microfonico BOB-12758

Il sensore microfonico utilizzato è costituito da un semplice **microfono a condensatore** per microelettronica, come ad esempio il microfono CMA-4544PF-W [2] dell'azienda CUI Devices, e da una **breakout board**[1] progettata e realizzata dall'azienda Sparkfun [3].

[1]Una breakout board è un PCB sviluppato per facilitare il collegamento ed il corretto funzionamento di sensori a microcontrollori o processori.

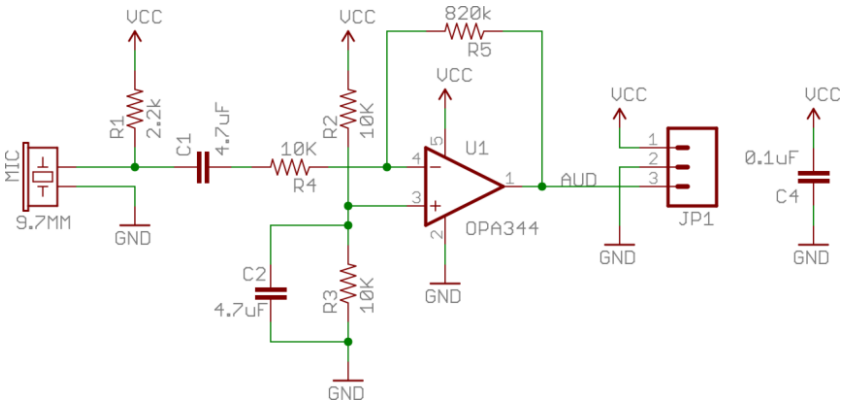

Figura 2.1: Circuito della breakout Board Sparkfun BOB-12758

Il circuito di figura 2.1 rappresenta i collegamenti e lo schematico della breakout board BOB-12758 e si compone di:

1. **Resistore R1** e **condensatore C1**, per **alimentare** il microfono e **acquisire** correttamente il segnale prodotto;
2. **Resistori R2 e R3**, utili ad inserire una **componente continua** nel segnale affinché esso sia completamente compreso nell'intervallo di tensione [0,VCC];
3. **Integrato OPA344** [4] e **resistori R4 e R5**, utili ad **amplificare**[2] il segnale acquisito dal microfono.

La board restituisce un **segnale adattato alla tensione di alimentazione** permettendo di semplificare il circuito per il collegamento al microcontrollore che si compone della sola **rivelazione dell'inviluppo**.

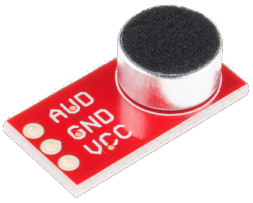

Figura 2.2: Sensore BOB-12758

Il rivelatore d'inviluppo approssima la sequenza dei massimi di un segnale. Il suo uso più comune è nelle telecomunicazioni, come demodulatore [5].

Un rivelatore d'inviluppo è costituito da un diodo, un condensatore ed un resistore. Durante i **picchi positivi** del segnale, il **condensatore si carica** molto rapidamente perché il diodo è in conduzione e la sua resistenza è molto bassa. Nella **fase discendente** del segnale il diodo entra in zona d'interdizione senza più condurre, ed il **condensatore si scarica** sul resistore finché non giunge un nuovo picco positivo del segnale.

[2] L'integrato OPA344 è un amplificatore operazionale e in questo caso viene utilizzato nella sua configurazione lineare di amplificatore invertente.

Nelle telecomunicazioni, la progettazione di questi circuiti si basa su formule matematiche legate alla frequenza dell'onda portante per demodulare segnali analogici. Questo approccio è rigoroso e basato su principi teorici ben stabiliti. Tuttavia, quando l'obiettivo è **rilevare gli inviluppi del segnale** per misurarne il livello medio, si può adottare un **approccio empirico**.
In questo caso, si parte da **valori iniziali plausibili** per i componenti del circuito, senza dipendere da formule teoriche precise, stimandoli in base all'esperienza o a considerazioni pratiche. Successivamente, **attraverso tentativi ed errori**, variando capacità e resistenza, si ottimizzano tali componenti fino a **raggiungere le prestazioni desiderate**.

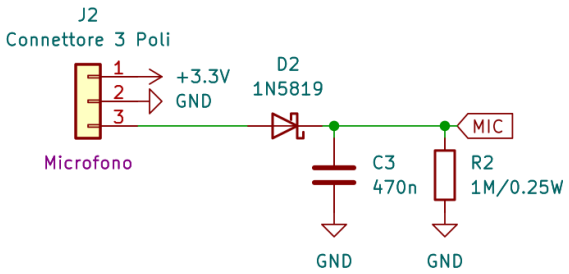

Figura 2.3: Rivelatore d'inviluppo

In figura 2.3 è mostrato il circuito di rivelazione utilizzato costituito da:

- Il **diodo D2 1N5819** [6] che appartiene alla famiglia dei diodi schottky: assicura **elevata velocità di commutazione** e **bassa caduta di potenziale**;
- Un **circuito RC**, costituito da resistore R2 e condensatore C3, con costante di tempo $\tau = 0{,}47$ s: **mantiene il valore** durante la fase di scarica del condensatore.

Il rivelatore d'inviluppo consente di impiegare una frequenza di campionamento bassa e, quindi, **abbassare i consumi energetici**. Questo è permesso dal fatto che del segnale audio misurato è necessaria la sola rivelazione del suo livello medio. Il circuito di rivelazione è stato collegato al pin fisico 11 (PA2) del microcontrollore[3].

2.2 Sensori digitali

I due sensori adottati per misurare i parametri ambientali non richiedono circuiti di condizionamento. Sono dotati di breakout board che trasformano il

[3]Tutti i collegamenti successivi sono da riferirsi ai pin del microcontrollore.

segnale misurato in formato digitale, consentendo un collegamento diretto al microcontrollore.
Questi sensori arricchiscono il dataset con **dati ambientali** che, combinati con le misurazioni del livello medio del rumore del traffico veicolare, **migliorano la precisione** delle stime sulla quantità di polveri sottili nell'aria.

2.2.1 Sensore DHT22

Il sensore di temperatura e umidità DHT22 [7], rappresentato in figura 2.4, è un dispositivo digitale che comunica con il microcontrollore tramite un **unico bus digitale**. Richiede un'alimentazione[4] di 3.3 V o 5 V, ed offre le seguenti caratteristiche tecniche [8]:

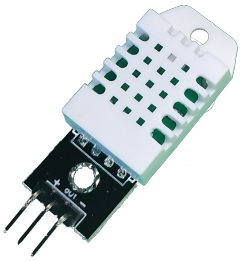

Figura 2.4: Sensore DHT22

- Misura della **percentuale d'umidità**: copre un intervallo **dallo 0% al 100%** con una **precisione di 2-5%**;
- Misura della **temperatura**: copre un intervallo **dai −40° C agli 80° C** con una **precisione ±0.5° C**.

In questo progetto si è collegato il sensore al pin 10 (PA3).

2.2.2 Sensore MS5803

Il sensore di pressione MS5803 [9], visibile in figura 2.5, è un dispositivo digitale che comunica con il microcontrollore tramite il **protocollo Inter Integrated Circuit (I2C)** che necessita di due linee digitali:

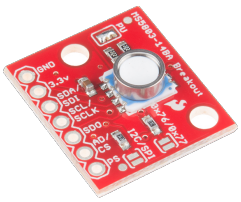

Figura 2.5: Sensore MS5803

- **Serial Data (SDA)** per i dati;
- **Serial Clock (SCL)** per la sincronizzazione.

Da datasheet [10] si legge che deve essere alimentato a 3.3V e, nella misura della **pressione atmosferica**, copre un intervallo che va **da 0 a 14 bar** con una **risoluzione di 1mbar**.
Si è collegato il sensore ai pin 7 per il bus SDA e al pin 9 per il bus SCL.

[4] L'alimentazione in elettronica va sempre intesa come una differenza tra due livelli di potenziale e quindi occupa sempre due pin: uno collegato a massa o Ground (GND) (potenziale elettrico nullo) e l'altro alla linea di alimentazione.

2.3 Modulo RFM95W

I ricetrasmettitori RFM95W [11] (figura 2.6) sono dispositivi che utilizzano il **protocollo LoRaWAN** per comunicare wireless a **lunga distanza** con **bassi consumi energetici**.
In questo caso si è deciso di utilizzare il ricetrasmettore montato su una breakout board sviluppata dalla DiyconElectronics [12].
Il modulo sfutta il **protocollo Serial Peripheral Interface (SPI)** per la comunicazione con il microcontrollore e dunque utilizza **4 linee digitali** quali:

Figura 2.6: Modulo RFM95W

1. **SCL**: per sincronizzarsi con il microcontrollore, collegato al pin 9;
2. **Master Output Slave Input (MOSI)**: come linea di trasmissione collegato al pin 8;
3. **Master Input Slave Output (MISO)**: come linea di trasmissione collegato al pin 7;
4. **Chip Select (CS)**: per selezionare ed attivare il modulo, collegato al pin 3.

Il trasmettitore necessita infine di un'alimentazione a 3.3V e di un ulteriore collegamento tra pin DIO0 del RFM95 e il pin 2 del microcontrollore.
I collegamenti del modulo e del sensore di pressione hanno in comune i pin 7 e 9 del micro[5]. Questo potrebbe portare ad interferenze fra i dispositivi. Si ipotizza di risolvere questa eventuale problematica via software.

2.4 Microcontrollore e Alimentazione

2.4.1 Microcontrollore ATTiny84

Il microcontrollore utilizzato in questo progetto è un ATTtiny84 [13], sviluppato dalla ATmel.
Risulta essere **versatile** ed adatto al progetto per la sua combinazione di **prestazioni elevate, basso consumo energetico e basso costo**.
Il Tiny84 può essere alimentato a 3.3V e per mantenere bassi i consumi energetici si è deciso di utilizzare il **clock interno** impostandolo al minimo.

[5]Micro è l'abbreviazione di microcontrollore

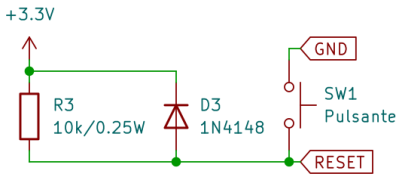

Figura 2.7: Circuito di reset

Dal punto di vista hardware si è realizzato un **circuito per il reset** mediante un pulsante, visualizzabile in figura 2.7, utile anche per il modulo per la comunicazione tramite LoRaWAN.
Il pin di reset (il 4) è un **pin attivo basso** che, per resettare il circuito, deve essere collegato al potenziale di massa. Il **resistore di pull-up** R3 garantisce il corretto funzionamento del microcontrollore **senza reset indesiderati**.

2.4.2 Controllo e uscite

Sono anche stati inseriti dei **led di controllo** e le **uscite** del circuito.
I led, il cui circuito realizzato è mostrato in figura 2.8, sono fondamentali per il **collaudo** del progetto. Durante la fase di setup, il led rosso si accende per verificare se il microcontrollore è configurato correttamente. Una volta completato il setup, il led rosso si spegne e, di conseguenza, si accende quello verde. Questo indica che il microcontrollore sta eseguendo il codice correttamente.

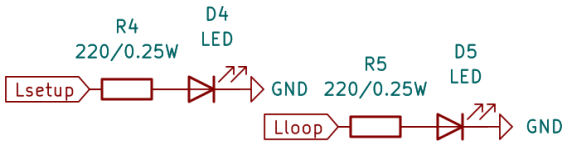

Figura 2.8: Circuito per i led

I due led vengono collegati con il micro al pin 13 (PA0), per il led di setup, e al pin 12 (PA1) per il led verde. È importante utilizzare un resistore per limitare la corrente e garantire il corretto funzionamento dei led.
Le uscite sono i **punti di connessione** per i sensori, che si è deciso di collegare direttamente al PCB per facilitare poi la progettazione e la realizzazione dell'involucro del progetto. I collegamenti sono descritti nelle sezioni precedenti.

2.4.3 Circuito di Alimentazione

In figura 2.9 è mostrato il circuito di alimentazione impiegato.
Esso utilizza il **diodo zener 1N4728** [14] e i due condensatori C1 e C2 in parallelo per stabilizzare la **tensione di alimentazione ai 3.3 V**, l'unico valore, come tensione di alimentazione richiesta, in comune a tutti i dispositivi utilizzati.

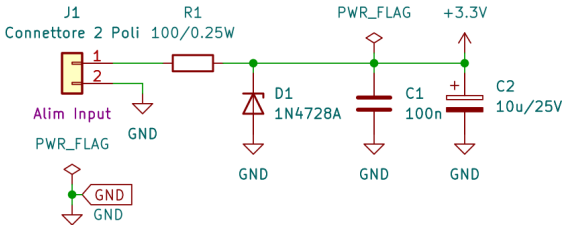

Figura 2.9: Circuito di alimentazione

La scheda presenterà una morsettiera a vite con due connessioni alle quali verranno collegati i poli positivo e negativo di una **batteria**. Per il corretto funzionamento ne serve una in grado di erogare una tensione almeno leggermente maggiore di 3.3 V.

2.5 Progettazione e sviluppo

Per la progettazione del circuito e lo sviluppo del PCB è stato utilizzato KiCAD [15].
KiCAD è una **suite open source** di software per Electronic Design Automation (EDA). Questo strumento facilita la progettazione e la simulazione di hardware elettronico per la produzione di PCB.
Le figure 2.10 e 2.11 mostrano rispettivamente il **circuito completo** ed il **PCB** progettati, tracciati e svilluppati tramite KiCAD.
Per la loro stesura sono state utilizzate le librerie standard di KiCAD e una libreria creata ad hoc per rappresentare correttamente il modulo RFM95 e la sua board.
Lo sviluppo dell'intero progetto di tesi è poi continuato attraverso **TinkerCAD** [16].
TinkerCAD è una **piattaforma online** che consente di creare design digitali in 3D, circuiti e blocchi di codice senza necessità di download o vincoli.
In questo caso è stato utilizzato per realizzare **il modello 3D** della scatola che conterrà il PCB. Il case verrà poi stampato in 3D.

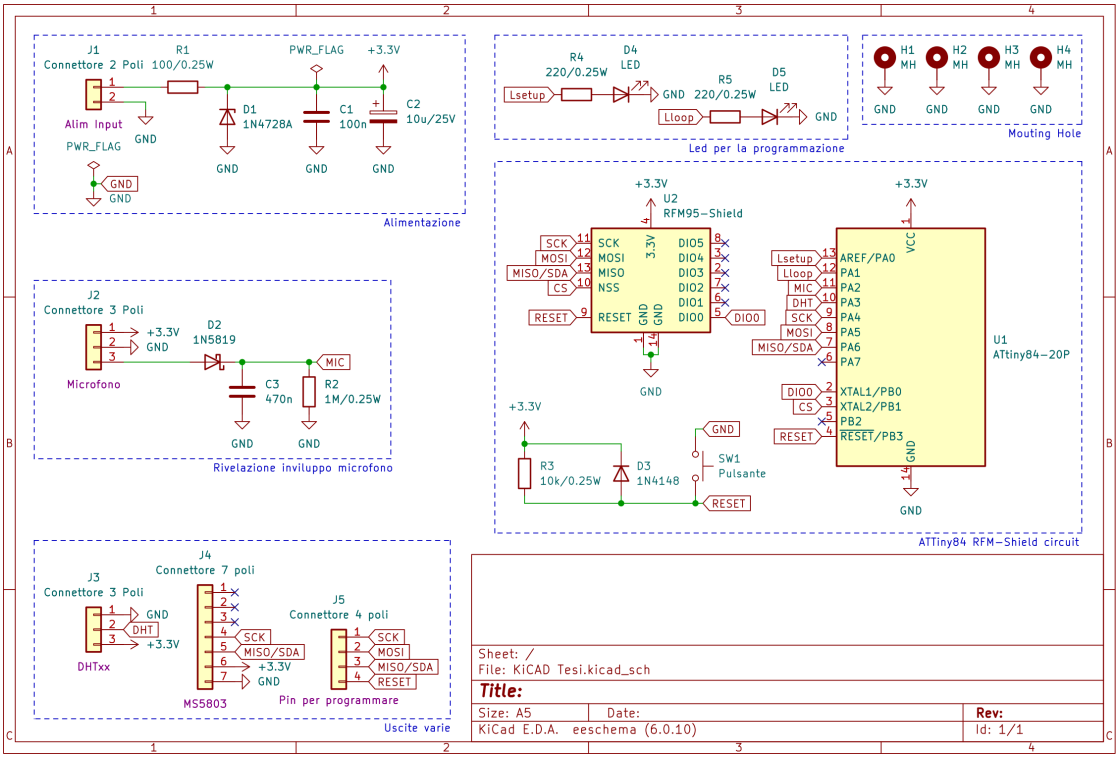

Figura 2.10: Circuito progettato

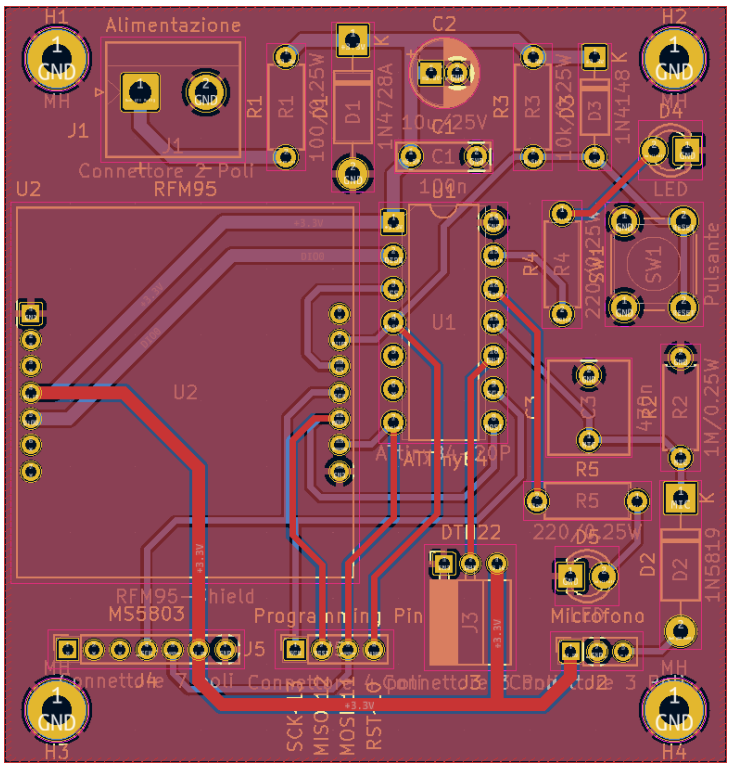

Figura 2.11: PCB sviluppato

3

Software sviluppato

In questo capitolo verrà descritto il codice utilizzato che ha portato alle successive conclusioni, descritte nel prossimo capitolo.
È importante capire come si esegue la **programmazione di microcontrollori** diversi da quelli direttamente implementati nelle schede Arduino, e come **caricare il bootloader** compatibile con Arduino IDE[1]. Questo è ampiamente spiegato nel libro intitolato *"Arduino e le tecniche di programmazione dei microcontrollori Atmel"* [18]. Il bootloader utilizzato è stato trovato attraverso una pagina nel forum di Arduino [19] che nominava una pagina di GitHub [20].
Il codice finale utilizzato per la programmazione del micro comprende:

1. il corretto **funzionamento dei led** di controllo;
2. la **lettura del sensore** di temperatura e umidità DHT22;
3. la **conversione** delle misurazioni acquisite dal sensore;
4. l'**invio** di questi ultimi alla rete LoRaWAN **tramite il modulo RFM95**.

3.1 Intestazione del codice

Nel codice vengono incluse due **librerie**. La prima, chiamata LoRaWAN [21, 22], richiama la libreria RFM95 [23]. Le due collaborano affinchè micro e

[1]Un IDE è un'applicazione che fornisce vari strumenti per lo sviluppo software, come un editor di codice e un debugger. [17]

modulo **comunichino correttamente** fra loro e **si interfaccino** con la rete LoRa, attraverso gli opportuni comandi. La seconda si chiama TinyDHT [24], è stata sviluppata da Adafruit [25] ed è utile ai microcontrollori della famiglia Tiny per **leggere i sensori** di temperatura e umidità DHT11 o DHT22.

```
#include <LoRaWAN.h>
#include <TinyDHT.h>

// pin connected to leds
#define PIN_LED_SETUP 0
#define PIN_LED_LOOP 1

// digital pin connected to RFM95 Module
#define DIO0 10
#define NSS 9

// digital pin connected to the DHT sensor
#define DHTPIN 3
// uncomment whatever type you're using!
//#define DHTTYPE DHT11   // DHT 11
#define DHTTYPE DHT22   // DHT 22

// define RFM95W Module LoRa
RFM95 rfm(DIO0,NSS);

// define LoRaWAN layer
LoRaWAN LoRa = LoRaWAN(rfm);

// define DHTxx
DHT dht(DHTPIN,DHTTYPE);

// define LoRa adress
unsigned char NwkSkey[16] = { 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0
    x07, 0x08, 0x09, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16 };
unsigned char AppSkey[16] = { 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0
    x07, 0x08, 0x09, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16 };
unsigned char DevAddr[4] = { 0x00, 0x00, 0x00, 0x08 };

// frame counter for LoRa
unsigned int Frame_Counter = 1;
```

Frammento 3.1: Librerie e intestazione

All'interno dell'intestazione del codice (frammento 3.1) vengono definite le costanti, le variabili globali e gli oggetti[2] utilizzati tramite gli appositi comandi delle librerie. Questi sono utili nell'intero codice, dovendoli richiamare spesso.

3.2 Setup

Il setup, mostrato nel frammento 3.2, è quella parte di codice scritto in Arduino IDE che viene **eseguita una sola volta** ed è utile al microcontrollore per inizializzare variabili, configurare pin e impostare parametri, tutto necessario nel resto del codice.

```
// the setup function runs once when you press reset or power the
    board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(PIN_LED_SETUP, OUTPUT);
  pinMode(PIN_LED_LOOP, OUTPUT);

  // blinking setup led
  digitalWrite(PIN_LED_SETUP, HIGH);  // turn the LED on (HIGH is the
    voltage level)

  // initialise LoRaWAN keys and Device Address for ABP joining
  LoRa.setKeys(NwkSkey, AppSkey, DevAddr);
  rfm.init();  // initialize RFM module

  // define bytebuffer length
  uint8_t Data_Length = 0x10;  // modify it if you wanna change the
   length of payload
  // define data string
  unsigned char Data[Data_Length];

  // inizialize sensor
  dht.begin();

  Data_Length = sprintf(Data,"sensor ready");
  delay(10);
```

[2]Arduino IDE è un ambiente di programmazione che si basa sul linguaggio C++ e dunque fa parte dei cosiddetti linguaggi di programmazione a oggetti. In essi svolgono un ruolo fondamentale le librerie che definiscono e possono utilizzare gli oggetti.

```
  LoRa.Send_Data(Data,Data_Length,Frame_Counter);
25  delay(10);

27  delay(5000);                              // wait
  digitalWrite(PIN_LED_SETUP, LOW);    // turn the LED off by making
    the voltage LOW
29 }
```

Frammento 3.2: void setup

Il frammento 3.2, grazie alla presenta dei commenti, risulta **di facile comprensione**, ma è utile una spiegazione aggiuntiva delle righe di codice utilizzate per la comunicazione via LoRaWAN (frammento 3.3).

```
1  // initialise LoRaWAN keys and Device Address for ABP joining
  LoRa.setKeys(NwkSkey,AppSkey,DevAddr);
3  rfm.init();  // initialize RFM module

5  // define bytebuffer length
  uint8_t Data_Length = 0x10;  // modify it if you wanna change the
    length of payload
7  // define data string
  unsigned char Data[Data_Length];
9  //[...]
  Data_Length = sprintf(Data,"sensor ready");
11 delay(10);
  LoRa.Send_Data(Data,Data_Length,Frame_Counter);
13 delay(10);
```

Frammento 3.3: Comandi LoRa

Dopo l'inizializzazione degli oggetti rfm e LoRa, si trova la definizione della variabile Data_Length e del vettore Data. Questi sono necessari per il corretto invio di pacchetti. Infatti la funzione **Send_Data**, per il corretto funzionamento, necessita della stringa da inviare, della sua lunghezza e di un contatore, definito nell'intestazione.

La funzione **sprintf** di C++ aggiorna propriamente sia il vettore Data, con la stringa, sia la variabile Data_Length, restituendo la lunghezza del vettore aggiornato.

3.3 Loop

Il loop, riportato interamente nel frammento 3.4, è quella parte di codice scritto in Arduino IDE che, al contrario del setup, viene **ripetutamente eseguita** e che costituisce il **cuore** del codice, utile a far funzionare il microcontrollore.

```
// the loop function runs over and over again forever
void loop() {
  // blinking loop led
  digitalWrite(PIN_LED_LOOP, HIGH);   // turn the LED on (HIGH is the
    voltage level)
  delay(1000);                        // wait for a second

  // reading temperature and humidity
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  char *s = "";
  int vt = t;
  if(t<0){
    s = "-";
    vt = -t;
  }
  int ft = trunc((t-vt) * 10);

  int vh = h;
  int fh = trunc((h-vh) * 10);

  // define bytebuffer length
  uint8_t Data_Length = 0x10;  // modify it if you wanna change the
   length of payload
  // define data string
  unsigned char Data[Data_Length];

  // expedition of measurement
  // increment of the counter
  Frame_Counter++;
  Data_Length = sprintf(Data,"%s%d.%d  %d.%d",s,vt,ft,vh,fh);
  delay(10);
  LoRa.Send_Data(Data,Data_Length,Frame_Counter);
  delay(10);

  digitalWrite(PIN_LED_LOOP, LOW);    // turn the LED off by making
```

```
    the voltage LOW
  delay(10000);                    // wait for a second
}
```

Frammento 3.4: void loop

Come per il setup, anche il frammento di codice 3.4 è di facile comprensione, ma è utile spiegare meglio le righe di codice mostrate nel frammento 3.5.
A causa delle capacità di calcolo dei microcontrollori tiny, la funzione sprintf di C++ permette **la sola conversione** di variabili di tipo intero in variabili stringa. Le variabili di tipo float, utilizzate per salvare le misurazioni del sensore, **non sono convertibili direttamente** ed è stato necessario divIDEre la parte intera da quella frazionaria salvandole nelle due variabili di tipo intero.

```
1  // reading temperature and humidity
   float h = dht.readHumidity();
3  float t = dht.readTemperature();

5  char *s = "";
   int vt = t;
7  if(t<0){
     s = "-";
9    vt = -t;
   }
11 int ft = trunc((t-vt) * 10);

13 int vh = h;
   int fh = trunc((h-vh) * 10);
```

Frammento 3.5: Conversione da float a intero

3.4 Osservazioni sul codice

Il codice descritto, come detto nel preambolo, è **incompleto e non completamente funzionante**: manca l'acquisizione del microfono e delle misurazioni del sensore di pressione e, come si vede in figura 3.1, i dati inviati alla rete non corrispondono ad effettivi valori di temperatura e umidità.
Dopo svariati tentavi, ipotesi ed errori si è concluso che tutto ciò dipendesse dalla **ridotta capacità di memoria** del Tiny84[3].

[3]Tiny84 è una abbreviazione del nome ATTiny84 già utilizzato.

SNR	RSSI	Size [B]	Fcnt	Payload
7.2	-89	16	9	0.0
8.8	-87	18	7	255.0
9.5	-87	16	6	0.0
8.5	-82	18	4	255.0
9	-87	16	3	0.0
9	-82	25	1	sensor ready

Figura 3.1: Dati inviati tramite LoRaWAN

Infatti, come si vede nella figura 3.2 che mostra l'output di Arduino IDE alla compilazione del codice, viene occupato il **99% dello spazio in memoria**. Questo porta a **possibili overflow** [26] nei comandi o nella convesione da tipo float a tipo char e inoltre **non permette l'aggiunta** delle istruzioni utili ad acquisire le misurazioni di microfono e sensore di pressione.

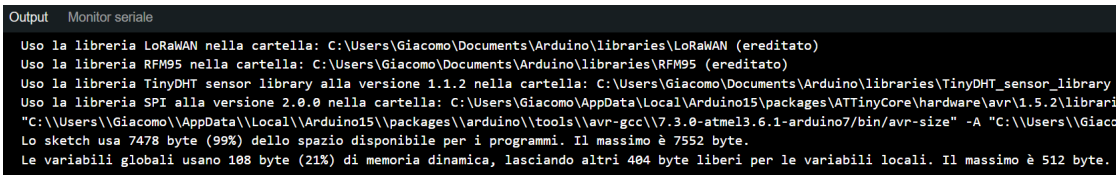

Figura 3.2: Output Arduino IDE

4

Conclusioni

Nonostante gli **ottimi risultati** ottenuti e descritti nel capitolo 2, il capitolo 3 ha evidenziato **l'importante limite** del microcontrollore utilizzato: la memoria. Si conclude che il progetto di tesi studiato, con le caratteristiche descritte, è **la prova della necessità** di un microcontrollore con più memoria.
È importante notare che questa tesi rappresenta una **solida base** dalla quale partire per la realizzazione del progetto completamente funzionante. Il **microcontrollore sostitutivo** individuato è l'ATTiny1604 [27] che, rispetto a quello utilizzato, presenta **memoria più grande** e **capacità di calcolo maggiori**. Da datasheet, inoltre, si legge che i pin per SPI e I2C non sono in comune e la possibile problematica rilevata nel capitolo 2 **si risolve** senza ricorrere a soluzioni via software magari complesse da realizzare.

Sitografia e Bibliografia

[1] Magalini Giorgio. *Tecniche indirette di monitoraggio ambientale tramite embedded machine learning*.

[2] Cui Devices. *Datasheet CMA-4544PF-W*.

[3] Sparkfun. *Pagina prodotto BOB-12758*. URL: https://www.sparkfun.com/products/12758.

[4] Texas Instrument. *Datasheet OPA344*.

[5] Wikipedia. *Rivelatore d'inviluppo*. URL: https://it.wikipedia.org/wiki/Rivelatore_d%27inviluppo#:~:text=In%20elettronica%2C%20il%20rivelatore%20d,i%20segnali%20modulati%20in%20ampiezza..

[6] Vishay. *Datasheet 1N5819*.

[7] Aosong Electronics. *Datasheet DHT22*.

[8] Adafruit. *Specifiche DHT22*. URL: https://www.adafruit.com/product/385.

[9] Sparkfun. *Specifiche e immagine MS5803*. URL: https://www.sparkfun.com/products/12909.

[10] Measurement Specialties. *Datasheet MS5803*.

[11] Hoperf Electronic. *Datasheet RFM95*.

[12] DiyconElectronics. *Informazioni RFM95W*. URL: https://diycon.nl/product/lora-node-pcb-100-shield-only-rfm92-95/.

[13] ATmel. *Datasheet ATTiny84*.

[14] Vishay. *Datasheet 1N4728*.

[15] KiCAD. *Sito ufficiale KiCAD*. URL: https://www.kicad.org/.

[16] TinkerCAD. *Sito ufficiale TinkerCAD*. URL: https://www.tinkercad.com/.

[17] Wikipedia. *IDE*. URL: https://it.wikipedia.org/wiki/Ambiente_di_sviluppo_integrato.

[18] Michele Menniti. *Arduino e le tecniche di programmazione dei microcontrollori Atmel*.

[19] Mike Trantham. *Link al forum sui core ATTiny*. URL: https://forum.arduino.cc/t/attinycore-on-ide-2-2-1/1182910.

[20] Spence Konde. *Link GitHub alla pagina del core ATTiny*. URL: https://github.com/SpenceKonde/ATTinyCore.

[21] jamct. *Link alla cartella di librerie*. URL: https://github.com/jamct/radio-mailbox/tree/master.

[22] jamct. *Link alla libreria LoRaWAN*. URL: https://github.com/jamct/radio-mailbox/tree/master/03_lora_test/lib/LoRaWAN.

[23] jamct. *Link alla libreria RFM95*. URL: https://github.com/jamct/radio-mailbox/tree/master/03_lora_test/lib/RFM95.

[24] Adafruit. *Link alla libreria TinyDHT*. URL: https://github.com/adafruit/TinyDHT.

[25] Adafruit. *Link sito ufficiale Adafruit*. URL: https://www.adafruit.com/.

[26] Wikipedia. *Overflow*. URL: https://it.wikipedia.org/wiki/Overflow.

[27] Microchip Technology. *Datasheet ATTiny1604*.

Ringraziamenti

Un ringraziamento speciale va alla mia famiglia e alla mia ragazza Margherita che mi hanno sempre supportato e aiutato tanto in questo percorso di formazione. Ringrazio particolarmente il mio relatore per la sua totale comprensione e disponibilità. Ringrazio anche tutti i professori di questa triennale, che sono stati fonte di ispirazione e conoscenza. Infine ringrazio in generale parenti e amici che mi sono stati vicini in questi anni.