Università della Calabria

Dipartimento di ingegneria informatica, modellistica, elettronica e sistemistica

Corso di studio in

Ingegneria Elettronica

“Smart Indoor Environmental Monitoring and Safeting System”

Studente

Giuseppe Pirilli 237909

Anno Accademico 2023/2024

Sommario

Abstract

In questo elaborato viene affrontata la progettazione di un sistema intelligente, basato su tecnologia IoT (Internet of Things), per il monitoraggio da remoto delle condizioni ambientali indoor in ambito residenziale, rilevando parametri quali: temperatura, umidità, presenza di fiamme, gas pericolosi e/o sostanze tossiche [gas propano (LPG) e monossido di carbonio (CO)], attraverso l'utilizzo di un nodo sensore, componente principale di una Wireless Sensor Network (WSN), in combinazione con altri dispositivi. A questo scopo, viene fornita, innanzitutto, una visione generale di cosa siano le Wireless Sensor Network. In particolare, viene fornita un'analisi completa sull'architettura hardware del componente principale di una WSN: il nodo sensore. La seconda parte dell'elaborato è dedicata all'implementazione dell'applicazione: viene spiegato come è stata realizzata l'applicazione, elencando i vari componenti hardware e software utili allo sviluppo. Nell'ultima parte vengono illustrati i risultati ottenuti ed i possibili sviluppi futuri.

Introduzione

Fino a pochi anni fa, qualsiasi tipo di connessione di rete era costruita con cavi. I vantaggi offerti dalle reti cablate sono ben noti: nessuna limitazione energetica, poiché laddove è possibile portare una connessione è possibile prevedere anche una o più linee di alimentazione, un buon livello di sicurezza, dato che per prelevare informazioni dalla rete è necessario un accesso fisico al canale e prestazioni elevate. Nel contempo però le reti cablate soffrono di gravi limitazioni, tra cui l'evidente difficoltà di realizzazione in ambienti inospitali e gli elevati costi che un cablaggio strutturato comporta. Nell'ultimo decennio, lo sviluppo di tecnologie wireless, quali Bluetooth, Wi-Fi, GSM, LTE, ecc., ha consentito un notevole ampliamento della connettività, instillando l'esigenza non solo di avere una connessione onnipresente, ma di avere connessa in rete qualsiasi cosa. È la cosiddetta $Internet\ of\ Things$, grazie alla quale ogni oggetto si rende intelligente, diviene riconoscibile ed è in grado di comunicare non solo con altri dispositivi della medesima categoria all'interno di una rete isolata, ma con qualsiasi tipologia di dispositivo, ovunque esso si trovi, per mezzo della rete internet. In questo contesto, i dati raccolti dai dispositivi IoT possono essere trasmessi e archiviati su piattaforme cloud per un'elaborazione avanzata, analisi e gestione in tempo reale, utilizzando servizi di $cloud\ computing$. Questo approccio consente di monitorare e attuare procedure di controllo ovunque nel mondo, sfruttando le capacità di accesso remoto e le funzionalità scalabili offerte dai servizi cloud.

L'avanzamento tecnologico delle comunicazioni wireless e dell'elettronica digitale, che ha permesso la realizzazione di dispositivi sempre più piccoli, economici e a basso consumo energetico, hanno reso possibile una nuova prospettiva tecnologica: le $Wireless\ Sensor\ Network$ (WSN).

Uno dei meriti della creazione di una rete di sensori wireless per la raccolta di informazioni è la flessibilità che consente nell'aggiungere sensori e modificare il layout del sistema.

Le WSN sono una particolare tipologia di reti senza fili costituite da un numero variabile di nodi sensori ($mote$) distribuiti nello spazio, caratterizzati da dimensioni contenute, bassi consumi energetici e limitate capacità di elaborazione, i quali comunicano tra di loro attraverso un ricetrasmettitore, per mezzo di tecnologie wireless a corto raggio, per raccogliere dati, analizzarli e, conseguentemente, reagire ad eventi d'interesse in funzione delle informazioni acquisite. La

capacità di elaborazione di questi nodi, seppur limitata, garantisce una distribuzione del carico di lavoro tra i vari nodi sparsi in rete, i quali, anziché inviare dati grezzi, possono svolgere delle operazioni di elaborazione sui dati rilevati/acquisiti ed inviare solo i dati d'interesse già elaborati.
In commercio esistono numerose tipologie di mote in grado di misurare svariate grandezze fisiche: temperatura, umidità, pressione, suono, luminosità, velocità, distanza di oggetti, ecc. Questo fa si che siano molteplici gli scenari applicativi in cui i mote possono essere impiegati.
Tuttavia anche le reti wireless devono affrontare alcune problematiche. Una di queste è indubbiamente la capacità del mezzo trasmissivo, l'etere, che è unico e condiviso da tutti i nodi connessi. L'esistenza di un unico canale limita necessariamente il numero massimo di utenti che possono usufruire del servizio contemporaneamente. Allo stesso modo, la presenza di più utenti determina una riduzione della velocità di trasmissione, in quanto la capacità del canale trasmissivo deve essere suddivisa tra tutti coloro che ne stanno facendo uso. Non da meno è il problema della sicurezza: in assenza di specifici controlli, risulta facile per un attaccante intercettare le informazioni che viaggiano nell'etere o riuscire ad accedere a servizi anche senza autorizzazione. Occorre inoltre considerare che la qualità della comunicazione può venir influenzata anche da fattori esterni, come interferenze elettromagnetiche e ostacoli in movimento. Infine il consumo energetico degli apparati di trasmissione radio è tipicamente più elevato di quelli per la comunicazione via cavo.
Il prototipo sviluppato per questo elaborato consiste in un sistema per il monitoraggio indoor di alcuni parametri ambientali attraverso l'impiego congiunto di un nodo sensore (telosB), ampliato con ulteriori sensori, e della tecnologia Arduino. In particolar modo, il sistema consente il monitoraggio di temperatura, umidità, presenza di fiamme, gas pericolosi e/o sostanze tossiche (propano e monossido di carbonio), rilevando la presenza di incendi ed eventuali fughe di gas che potrebbero provocare esplosioni, garantendo la sicurezza delle persone in vari contesti, come ambienti industriali e/o domestici. Il sistema è stato inoltre equipaggiato di un modulo Wi-Fi per sfruttare la tecnologia IoT, connettendo il sistema alla rete e consentendo l'invio dei dati su una piattaforma IoT (Bolt Cloud IoT), dalla quale sarà possibile, attraverso un'apposita applicazione/dashboard web (interfaccia utente grafica, GUI), il monitoraggio, ed eventualmente il controllo di attuatori, da remoto, in tempo reale.

1 Wireless Sensor Network (WSN)

Le WSN sono una particolare tipologia di reti senza fili composte da un numero variabile di nodi sensori i quali comunicano tra di loro per mezzo di tecnologie wireless, consentendo il monitoraggio dei parametri ambientali in luoghi naturali ma anche industriali e residenziali. I nodi con a bordo i sensori rilevano i dati d'interesse che vengono trasmessi alla base station per poter essere elaborati. Le dimensioni contenute dei sensori fanno sì che lo spazio a disposizione per la sorgente energetica sia molto limitato, con un impatto sulla quantità di energia immagazzinata. Si rende quindi necessario che l'assorbimento energetico dovuto alle fasi di sensing, elaborazione dati e trasmissione radio sia quanto più possibile limitato; ciò impone dei requisiti particolari sull'hardware da utilizzare a bordo. Inoltre, le capacità e le qualità delle trasmissioni tra i sensori di una rete wireless sono fortemente vincolate alle condizioni degli ambienti in cui sono costruite. Infatti tali ambienti nella maggior parte dei casi non possono essere considerati statici in quanto in essi sono

presenti ad esempio ostacoli (muri, mobili, persone, ecc.) e campi magnetici variabili che interferiscono con i segnali trasmessi dai singoli nodi.

1.1 Architettura hardware di un nodo sensore (mote)

Per comprendere a pieno le potenzialità e i limiti di una WSN è necessario descrivere, almeno genericamente, l'architettura del nodo sensore.

Il nodo sensore è in grado di eseguire alcune elaborazioni sui dati acquisiti e comunicare con altri nodi nella rete. Una struttura tipica di un nodo sensore (vedi fig. 1) è costituita da quattro blocchi funzionali:

- *Unità di sensing*, i dati, ovvero le grandezze fisiche o ambientali di diversa natura, vengono rilevati tramite trasduttori, dispositivi in grado di trasformare queste grandezze fisiche in un segnale elettrico analogico (tipicamente un valore di tensione). Poiché questo tipo di operazione è nota come "sensing", i trasduttori vengono chiamati anche sensori. I dati di tipo analogico vengono convertiti in formato digitale tramite un ADC, quindi inviati al processore che si occuperà di elaborarli e trasmetterli in rete. I sensori analogici che si possono trovare in commercio sono numerosissimi e includono trasduttori di temperatura, umidità, intensità luminosa, pressione, nonché rilevatori di fumo o sostanze tossiche (disperse nell'aria o nel suolo), sensori di prossimità, ecc.
  Un nodo sensore può comprendere un numero variabile di sensori. L'equipaggiamento del nodo sensore, quindi, ne determina le funzionalità, ma anche il costo, il consumo energetico e l'ingombro.

- *Unità di elaborazione e controllo*, è costituita da un micro-processore (CPU) che fornisce l'intelligenza necessaria al dispositivo per operare in modo autonomo. Il ruolo della CPU può essere svolto da diverse tipologie di circuiti logici. Comunemente la scelta ricade su un micro-controllore a basso consumo, ma è possibile impiegare anche un FPGA (Field Programmable Gate Array), un DSP (Digital Signal Processing) o un ASIC (Application Specific Integrated Circuit), anche a supporto del micro-controllore per diminuire il carico computazionale, il quale, essendo studiato per contenere il consumo energetico, è dotato di potenza di calcolo limitata.
  Il micro-controllore è affiancato da blocchi di memoria ROM (Read Only Memory) di tipo Flash e RAM (Random Access Memory), utilizzati per immagazzinare il codice eseguibile del sistema operativo e dell'applicazione, nonché i dati acquisiti dai sensori ed elaborati dall'applicazione. La gestione e l'utilizzo delle memorie è una fonte di consumo energetico, pertanto i blocchi di memoria integrati hanno capacità ridotte, limitate a poche decine di kbyte.

- *Unità di comunicazione*, costituita da un chip ricetrasmettitore che consente la comunicazione radio con altri nodi sensori presenti nella rete. Solitamente, tra tutti i componenti del nodo sensore, il chip radio è quello che consuma la maggior parte dell'energia durante la fase di active. Per ridurre il costo e il consumo energetico, si utilizzano tipicamente modulazioni ben consolidate e di bassa complessità, a discapito della capacità

trasmissiva che è spesso limitata a qualche decina di kbit/s. Spesso l'interfaccia è di tipo *half-duplex*, ovvero, non è in grado di ricevere pacchetti durante la fase di trasmissione.

- *Unità di alimentazione*, utilizzata per fornire l'energia richiesta al nodo sensore. I possibili metodi per fornire energia al nodo possono essere classificati in tre gruppi: immagazzinare energia (ad esempio, batterie), distribuire energia al nodo (ad esempio, un filo) e recuperare energia ambiente (ad esempio, una cella solare). È anche possibile una combinazione di questi metodi.
  Le limitazioni di un nodo sensore, in termini di consumo energetico e memoria disponibile, rendono particolarmente difficile l'utilizzo di un sistema operativo *general purpose* per la programmazione, in quanto le risorse richieste da tale tipologia di sistemi operativi non sono compatibili con le limitate prestazioni hardware tipiche dei nodi sensori. Pertanto, è richiesto lo sviluppo di un apposito sistema operativo in grado di soddisfare i seguenti requisiti:

  – Ridotta occupazione di memoria;
  – Basso consumo di energia durante l'esecuzione dei processi;
  – Consumo pressocché nullo durante lo stato di inattività (idle).

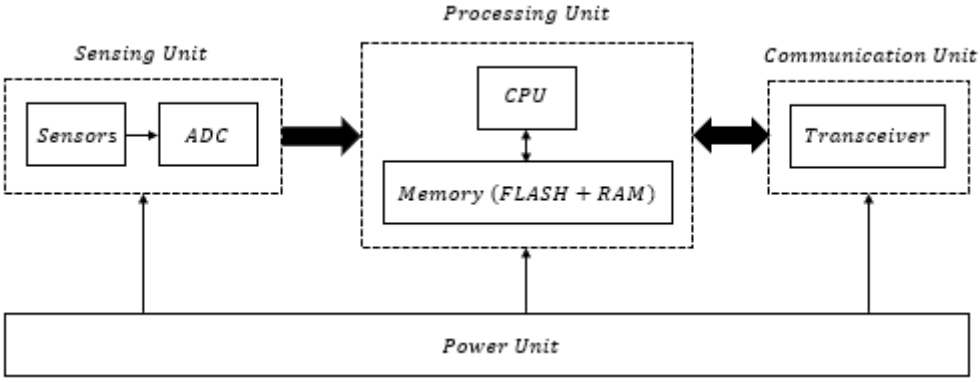


Figure 1- Schema a blocchi di un nodo sensore

1.2 Piattaforma hardware e Software di riferimento: TelosB e TinyOS

Uno dei requisiti fondamentali di un nodo sensore è il basso assorbimento energetico. Questo requisito può essere soddisfatto tramite l'utilizzo di hardware con prestazioni limitate. Non è pertanto possibile utilizzare un sistema operativo *general purpose* in quanto le risorse richieste da tale tipologia di sistemi operativi non sono compatibili con le limitate prestazioni hardware tipiche di questi nodi sensori. Si rende quindi necessario rivolgersi a sistemi operativi appositamente studiati. Negli ultimi anni sono state sviluppate numerose alternative, quelle di maggior successo sono: TinyOS e Contiki.
Per lo sviluppo di questo progetto, la piattaforma hardware prescelta come nodo sensore, in combinazione con altre tecnologie hardware, è la TelosB, mentre il sistema operativo utilizzato per la programmazione del nodo sensore, per la specifica applicazione, è TinyOS.

Il sistema operativo, sviluppato dall'università californiana di Berkeley, ha licenza *open source* ed è scritto nel linguaggio di programmazione *nesC*, che può essere considerato un dialetto del linguaggio *C*. Si tratta di un linguaggio ad eventi, basato sui componenti ed ottimizzato per supportare piattaforme con limitate risorse hardware (quantità di memoria limitata e bassa potenza di calcolo della CPU). Il componente principale (l'unico sempre presente in ogni applicazione) è lo *scheduler*, questo manda in esecuzione i task dei diversi componenti secondo una politica FIFO *run to completion* (un task non può interrompere un altro task). Lo scheduling ha due livelli di priorità: normale, per i task, e quello più alto per gli eventi, che possono interrompere i task.
Il punto in cui NesC differisce da C è nel modello di collegamento. La complessità non è scrivere componenti software, ma combinare un set di componenti e trasformarli in un'applicazione funzionante.
Gli aspetti caratterizzanti di nesC sono:

- Separazione della costruzione e della composizione: i programmi sono costituiti da componenti che vengono assemblati, tramite una procedura di *wiring*, per formare il programma completo (tra i componenti che compongono un'applicazione ci sono anche quelli del sistema operativo stesso).
- Specifica dei componenti per mezzo di interfacce: le interfacce possono essere usate o fornite dal componente; le ultime indicano le funzionalità che il componente fornisce all'utente, le prime rappresentano le funzionalità di cui il componente necessita per svolgere il proprio lavoro.
- Interfacce bidirezionali: da un lato specificano una serie di funzioni che devono essere implementate da chi fornisce l'interfaccia (i comandi) dall'altra una serie di funzioni che devono essere implementate dall'utilizzatore dell'interfaccia (gli eventi).
- I componenti sono collegati fra di loro tramite interfacce.
- Definisce un'elaborazione *event-driven*, ovvero, i componenti di un'applicazione vengono mandati in esecuzione solo quando si verificano gli eventi associati a ciascun componente.
- Esistono due tipi di implementazione di componenti: i *moduli* e le *configurazioni*.
  I moduli implementano le funzionalità delle interfacce del modulo (definiscono gli handler dei comandi e degli eventi). Il modulo implementa inoltre gli eventi e i comandi in modo molto simile a come vengono implementati i sottoprogrammi in C, ad esclusione del fatto che, prima della definizione dell'evento o del comando, bisogna inserire il nome dell'interfaccia relativa.
  Le configurazioni implementano componenti dichiarando una serie di sottocomponenti e definendo i collegamenti tra le interfacce di questi sottocomponenti. Ogni applicazione è composta da una configurazione globale che definisce la connessione dei vari componenti.

Come piattaforma hardware si è optato, come già accennato, per il TelosB (vedi fig. 2).
Questo "mote" è caratterizzato da un basso consumo energetico e, conseguentemente, limitate prestazioni hardware. Presenta, inoltre, la caratteristica di poter mantenere lo stato di idle per la maggior parte del tempo e di riattivarsi molto velocemente per funzionare, tornando allo stato di idle una volta completato il processo. Può essere alimentato da due batteria AA o tramite la porta USB. Se è collegato alla porta USB per la programmazione o la comunicazione seriale (debug), l'alimentazione viene fornita dal pc (non è richiesta alcuna batteria). La tensione operativa del mote, quando connesso alla porta USB, è pari a 3V. Il cuore della piattaforma TelosB è costituito da un

micro-controllore MSP430 prodotto dalla Texas Instruments (MSP430F1611), che presenta un'architettura a 16-bit di parallelismo ed una frequenza di clock pari a 8 MHz. È dotato di 10 kB di memoria RAM e 48 kB di memoria ROM di tipo Flash per l'esecuzione dei programmi, di un convertitore ADC a 12-bit e di un controllore DMA (Direct Memory Access). L'interfaccia radio è costituita da un circuito integrato Chipcon CC2420 compatibile con lo standard 802.15.4 in grado di trasmettere ad un rate di 250 kb/s. Il mote presenta un'antenna a 2.4 GHz integrata nel circuito stampato che permette di trasmettere fino a 50 m indoor e fino a 125 m outdoor. Il nodo è equipaggiato di un sensore di luminosità Hamamatsu S1087 in grado di rilevare lunghezze d'onda nello spettro del visibile e dell'ultravioletto e di un sensore di temperatura/umidità SHT11 prodotto da Sensirion (vedi fig. 3).

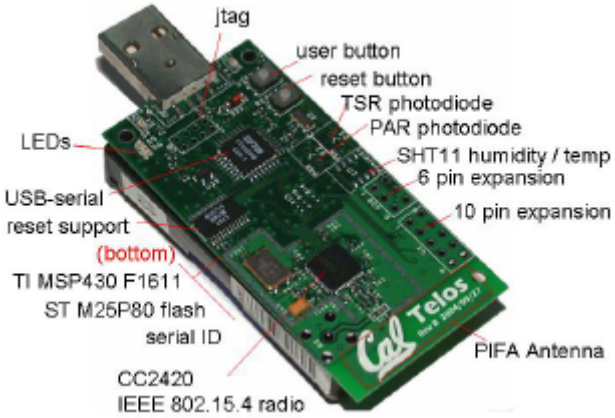


Figure 2 - Nodo sensore: TelosB

Figure 3 - Sensori Hamamatsu S1087 (a sinistra) e Sensirion SHT11 (a destra)

Il TelosB dispone di un connettore di espansione a 16 pin (vedi fig. 4), 6 dei quali sono utilizzabili come canali ADC per l'aggiunta di sensori analogici, inoltre, 2 dei 6 canali ADC (pin 7 e pin 10) possono essere utilizzati come digital I/O solo se R14 ed R16 vengono popolate con una resistenza da 0 ohm. Ciò può tornare utile qualora si decidesse di implementare una comunicazione seriale SPI. Vi è anche la possibilità di realizzare comunicazioni seriali di tipo UART (pin 2 e pin 4) e I2C (pin 5 e pin 8).

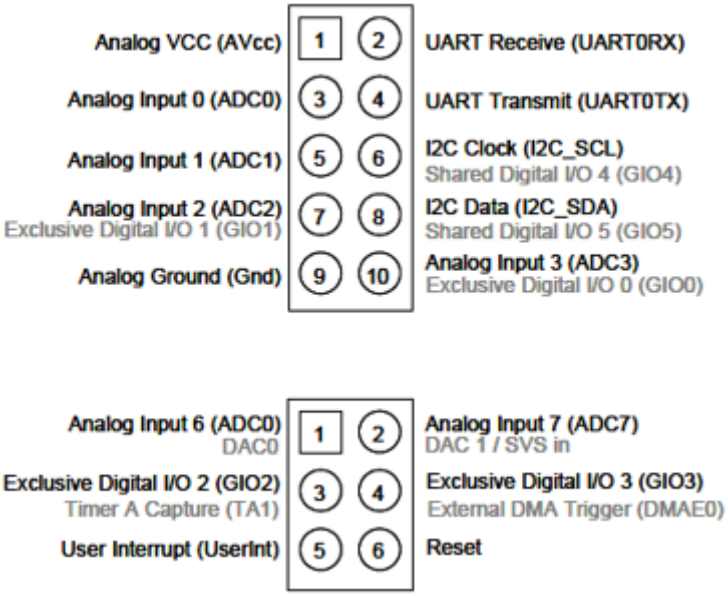


Figure 4 - Connettore di espansione a 16 pin

1.3 Esempio di applicazione NesC

Un'applicazione in TinyOS può essere semplicemente suddivisa in componenti, che possono essere di due tipi: modulo o configurazione. In questa particolare applicazione denominata $Blink$, i LED sul nodo di programmazione sono fatti lampeggiare in modo ordinato. Un codice di esempio del modulo e del configuration file è mostrato qui per una migliore comprensione. In generale, il modulo e le configurazioni sono denominati in base all'applicazione stessa per evitare confusione. Il modulo Blink è il seguente:

```
module BlinkM{
            provides {
                      interface StdControl;
                     }
                uses {
                      interface Timer;

                      interface Leds;
                     }
                     }
```

Nella prima riga, BlinkM implementa l'interfaccia StdControl. Il modulo BlinkM utilizza anche due interfacce: Led e Timer. Ciò significa che BlinkM può chiamare qualsiasi comando dichiarato nelle interfacce che utilizza e deve anche implementare qualsiasi evento dichiarato in tali interfacce. La configurazione per l'applicazione Blink è trattata di seguito:

```
configuration Blink {
                    }
                implementation {
                        components Main, BlinkM, SingleTimer, LedsC;
            Main.StdControl -> BlinkM.StdControl;
            Main.StdControl -> SingleTimer.StdControl;
            BlinkM.Timer -> SingleTimer.Timer;
            BlinkM.Leds -> LedsC;
                            }
```

La parola chiave configuration indica che questo è un file di configurazione. La riga che segue la parola chiave implementation indica il set di componenti qui referenziati. In questo caso Main, BlinkM, SingleTimer e LedsC sono i componenti utilizzati. In tutte le applicazioni TinyOS Main è il primo componente che viene eseguito per primo. Il –> viene utilizzato per collegare le interfacce utilizzate dal componente BlinkM (applicazione principale) ai componenti che forniscono tali interfacce. In questo modo i sotto-componenti sono cablati insieme per sviluppare un'intera applicazione. Anche l'applicazione sviluppata in questo lavoro è cablata in modo simile.

2 Metodologia di sviluppo

Questo paragrafo è diviso in tre parti: panoramica generale del sistema, progettazione hardware, comunicazione e progettazione software. La panoramica generale del sistema descrive l'idea progettuale del sistema, a partire dal nodo sorgente fino alla visualizzazione dei dati sulla piattaforma IoT. La progettazione hardware e software descrivono, rispettivamente, i componenti hardware ed i programmi/routine software, di supporto all'hardware, implementati nel sistema. La comunicazione fa riferimento al protocollo di tipo seriale utilizzato per lo scambio dei dati tra i vari dispositivi del sistema.

2.1 Panoramica generale del sistema

Il sistema di monitoraggio delle condizioni ambientali indoor, la cui architettura è illustrata in fig. 5, è basato sull'unità centrale principale del mote TelosB (MSP430) in combinazione con l'unità centrale di Arduino (ATMega328P) e del dispositivo Bolt (Tensilica Xtensa LX106), quest'ultimo equipaggiato di un modulo Wi-Fi ESP8266. Il mote è stato interfacciato con sensori analogici/digitali per misurare periodicamente i parametri ambientali, costituiti da: temperatura, umidità relativa, concentrazione di LPG (gas propano), concentrazione di CO (monossido di carbonio) e presenza di fiamme, i quali vengono successivamente elaborati dal mote, confezionati in un frame di pacchetti e inviati al micro-controllore Arduino tramite protocollo UART, il quale, a sua volta, dopo averli ricostruiti e salvati in apposite variabili, li trasmette al modulo Wi-Fi del micro-controllore Bolt (IoT gateway) sempre tramite protocollo UART (l'intervallo di tempo di trasmissione dati viene eseguito ogni secondo). Arduino, inoltre, è responsabile del controllo automatico degli attuatori in funzione delle condizioni

ambientali. Una volta che il modulo Wi-Fi riceve il flusso di dati, questi vengono inviati alla piattaforma Bolt Cloud IoT per il monitoraggio da remoto in tempo reale attraverso un'apposita applicazione web accessibile per mezzo di un browser web. I dati visualizzati vengono aggiornati ogni 30 secondi (limite imposto dalla piattaforma). Oltre al monitoraggio, la piattaforma cloud fornisce ulteriori funzionalità.

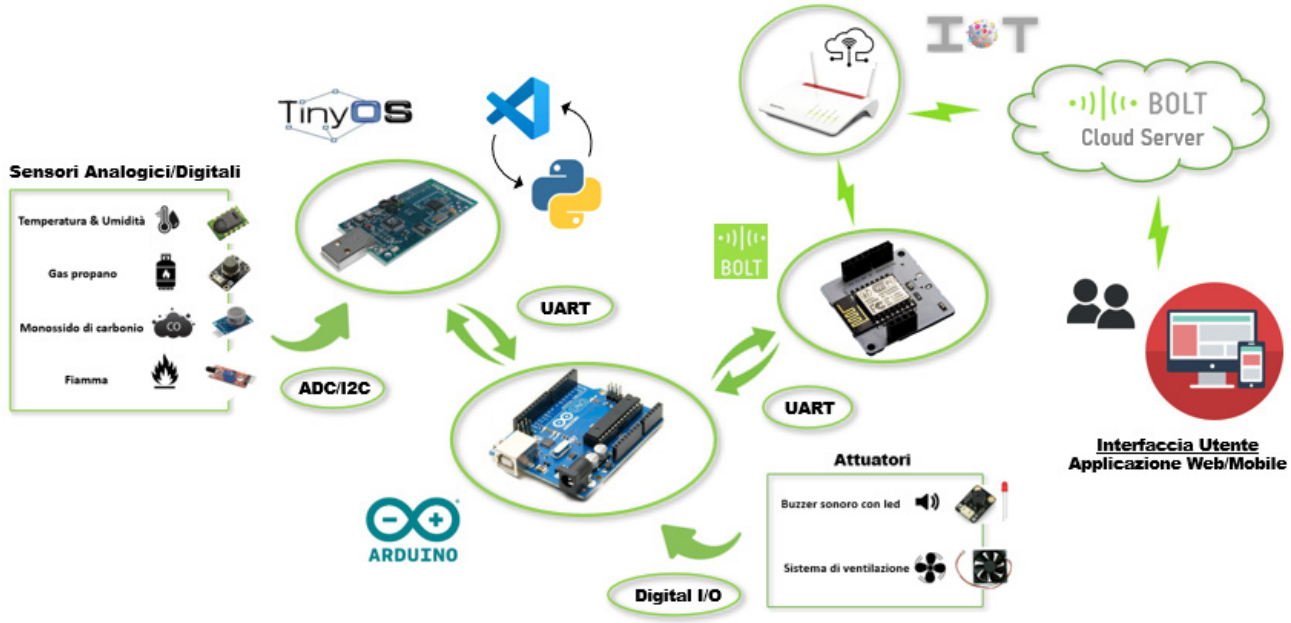


Figure 5 - Architettura del sistema

La fig. 6 fornisce una chiara comprensione della metodologia adottata per raccogliere i dati dai sensori e trasferirli al micro-controllore Arduino, che può a sua volta trasmetterli al dispositivo Bolt, da dove è possibile caricare i dati sul cloud per il monitoraggio e l'analisi da remoto.

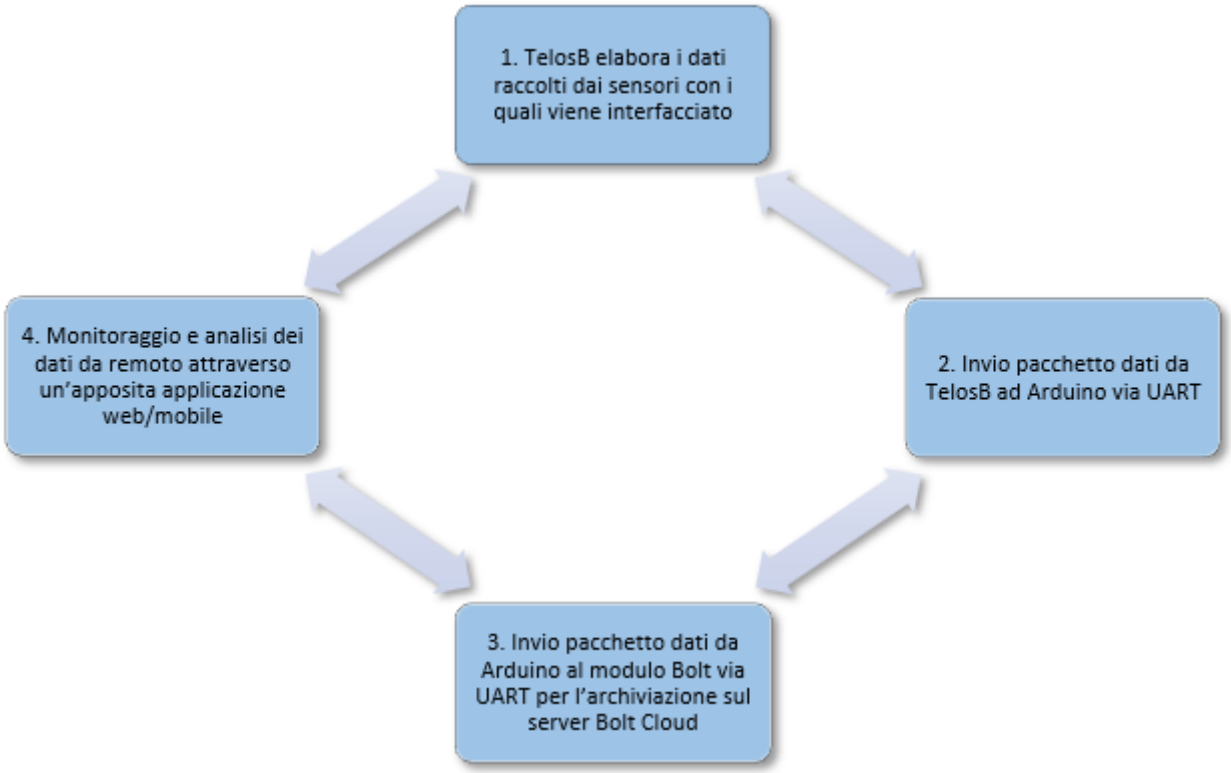
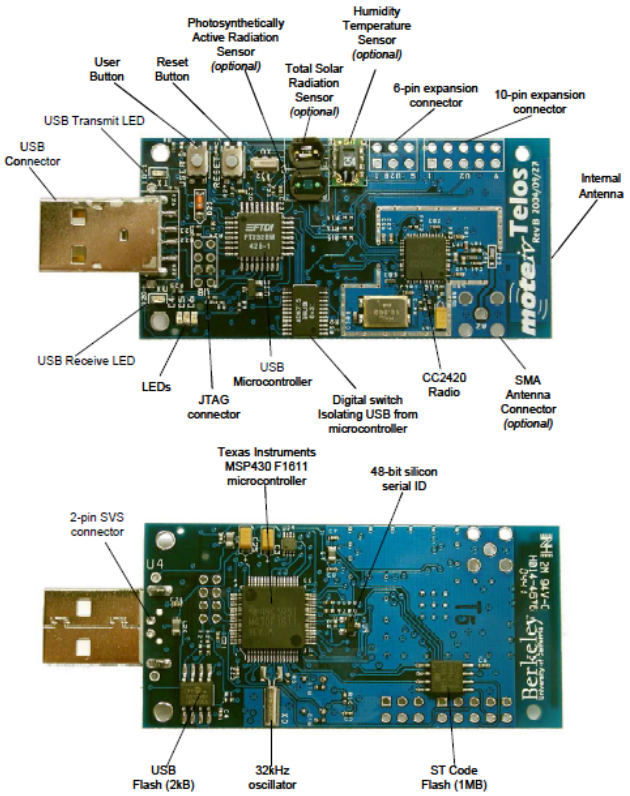


Figure 6 - Flusso dei dati dalla fase di input alla fase di analisi

2.2 Progettazione hardware

L'hardware impiegato per lo sviluppo del sistema è costituito da:

- $Mote\ TelosB$, rappresenta il cuore del sistema ed è responsabile della raccolta ed elaborazione dei dati ambientali per mezzo di vari sensori (sia integrati che esterni) di cui è equipaggiato. Il dispositivo viene programmato nell'ambiente TinyOS, basato sul linguaggio nesC. La programmazione ha previsto l'implementazione di due parti: una relativa al sensing e l'altra al protocollo UART per la comunicazione con il micro-controllore Arduino.

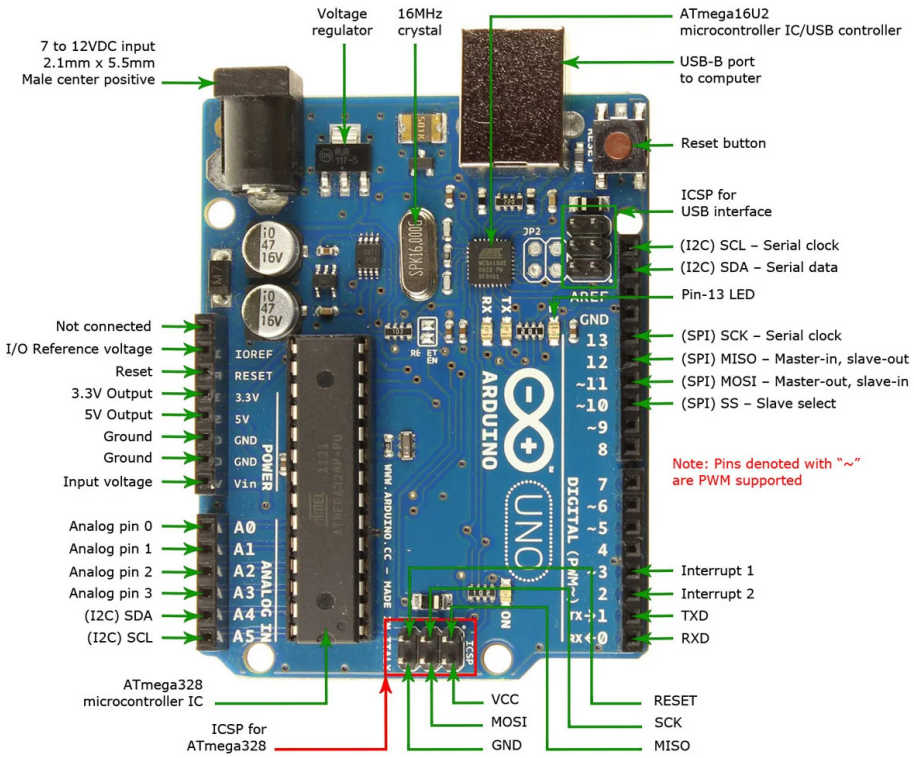


Typical Operating Conditions

	MIN	NOM	MAX	UNIT
Supply voltage	2.1		3.6	V
Supply voltage during flash memory programming	2.7		3.6	V
Operating free air temperature	-40		85	°C
Current Consumption: MCU on, Radio RX		21.8	23	mA
Current Consumption: MCU on, Radio TX		19.5	21	mA
Current Consumption: MCU on, Radio off		1800	2400	µA
Current Consumption: MCU idle, Radio off		54.5	1200	µA
Current Consumption: MCU standby		5.1	21.0	µA

Specifications	TPR2420CA	Remarks
Module		
Processor Performance	16-bit RISC	
Program Flash Memory	48K bytes	
Measurement Serial Flash	1024K bytes	
RAM	10K bytes	
Configuration EEPROM	16K bytes	
Serial Communications	UART	0-3V transmission levels
Analog to Digital Converter	12 bit ADC	8 channels, 0-3V input
Digital to Analog Converter	12 bit DAC	2 ports
Other Interfaces	Digital I/O,I2C,SPI	
Current Draw	1.8 mA	Active mode
	5.1 µA	Sleep mode
RF Transceiver		
Frequency band[1]	2400 MHz to 2483.5 MHz	ISM band
Transmit (TX) data rate	250 kbps	
RF power	-24 dBm to 0 dBm	
Receive Sensitivity	-90 dBm (min), -94 dBm (typ)	
Adjacent channel rejection	47 dB	+ 5 MHz channel spacing
	38 dB	- 5 MHz channel spacing
Outdoor Range	75 m to 100 m	Inverted-F antenna
Indoor Range	20 m to 30 m	Inverted-F antenna
Current Draw	23 mA	Receive mode
	21 µA	Idle mode
	1 µA	Sleep mode
Sensors		
Visible Light Sensor Range	320 nm to 730 nm	Hamamatsu S1087
Visible to IR Sensor Range	320 nm to 1100nm	Hamamatsu S1087-01
Humidity Sensor Range	0-100% RH	Sensirion SHT11
Resolution	0.03% RH	
Accuracy	± 3.5% RH	Absolute RH
Temperature Sensor Range	-40°C to 123.8°C	Sensirion SHT11
Resolution	0.01°C	
Accuracy	± 0.5°C	@25°C
Electromechanical		
Battery	2X AA batteries	Attached pack
User Interface	USB	v1.1 or higher
Size (in)	2.55 x 1.24 x 0.24	Excluding battery pack
(mm)	65 x 31 x 6	Excluding battery pack
Weight (oz)	0.8	Excluding batteries
(grams)	23	Excluding batteries

- *Arduino Uno R3*, è responsabile dell'invio dei dati, ricevuti da TelosB, al modulo Wi-Fi e dell'azionamento di attuatori in funzione delle condizioni ambientali. Il dispositivo viene programmato utilizzando il software Arduino IDE.

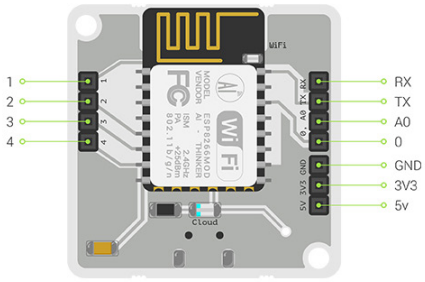


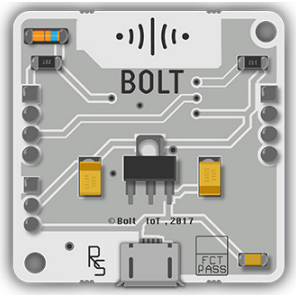
Features

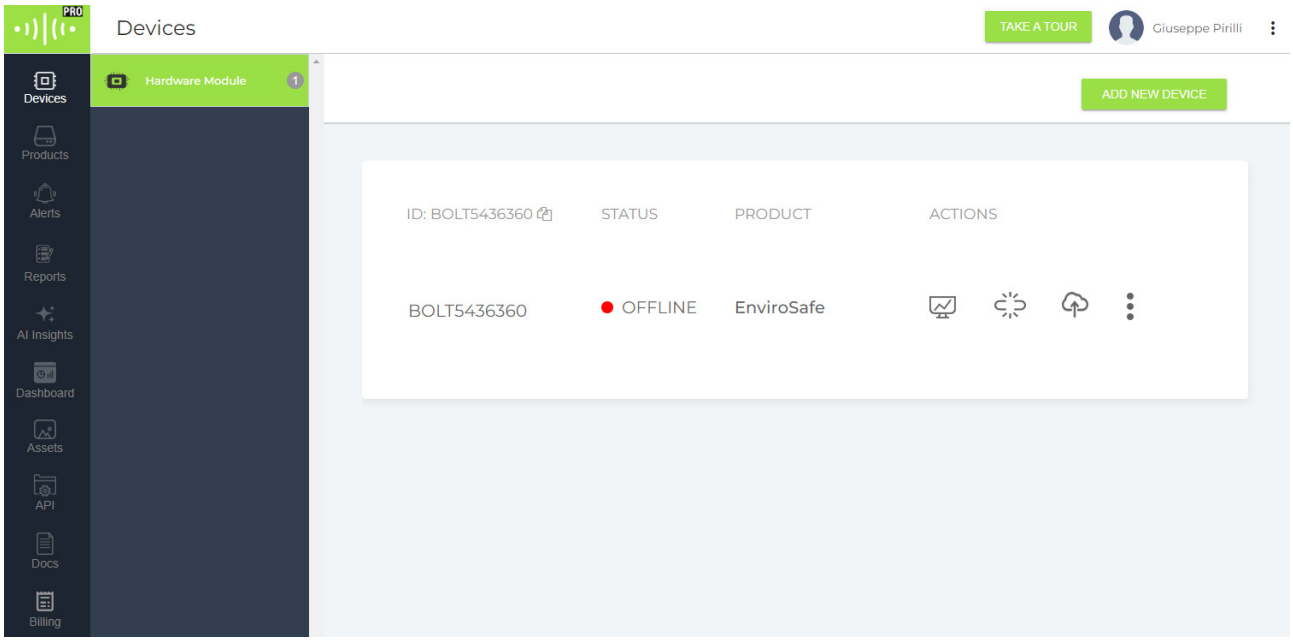
- **ATMega328P** Processor
  - **Memory**
    - AVR CPU at up to 16 MHz
    - 32KB Flash
    - 2KB SRAM
    - 1KB EEPROM
  - **Security**
    - Power On Reset (POR)
    - Brown Out Detection (BOD)
  - **Peripherals**
    - 2x 8-bit Timer/Counter with a dedicated period register and compare channels
    - 1x 16-bit Timer/Counter with a dedicated period register, input capture and compare channels
    - 1x USART with fractional baud rate generator and start-of-frame detection
    - 1x controller/peripheral Serial Peripheral Interface (SPI)
    - 1x Dual mode controller/peripheral I2C
    - 1x Analog Comparator (AC) with a scalable reference input
    - Watchdog Timer with separate on-chip oscillator
    - Six PWM channels
    - Interrupt and wake-up on pin change

- **ATMega16U2 Processor**
  - 8-bit AVR® RISC-based microcontroller
- **Memory**
  - 16 KB ISP Flash
  - 512B EEPROM
  - 512B SRAM
  - debugWIRE interface for on-chip debugging and programming
- **Power**
  - 2.7-5.5 volts

- *Bolt IoT*, risulta equipaggiato di un modulo Wi-Fi ESP8266 che consente di connettere il sistema alla rete internet. Questo garantisce un ampliamento delle capacità del sistema, abilitando il monitoraggio dei parametri ambientali e, eventualmente il controllo di attuatori appropriati, ovvero, dispositivi in grado di interagire con l'ambiente in modi diversi (valvole, irrigatori, allarmi, sistema di ventilazione, ecc.) da remoto. In questo contesto, il sistema sarà in grado di trasmettere, in tempo reale, i dati dal mote alla piattaforma Bolt Cloud, dove possono essere monitorati da remoto e utilizzati per generare, in funzione di un livello di soglia delle condizioni ambientali, avvisi o attivare risposte automatiche.





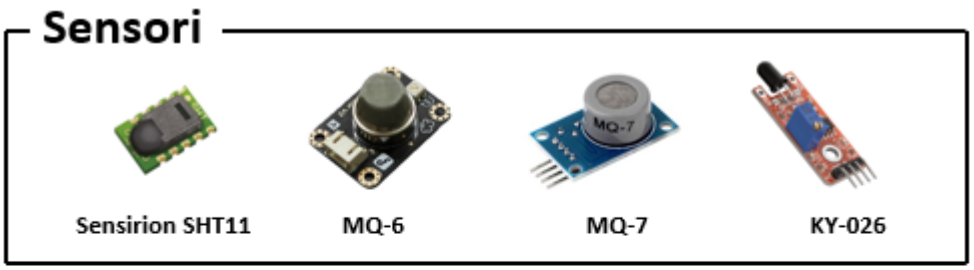


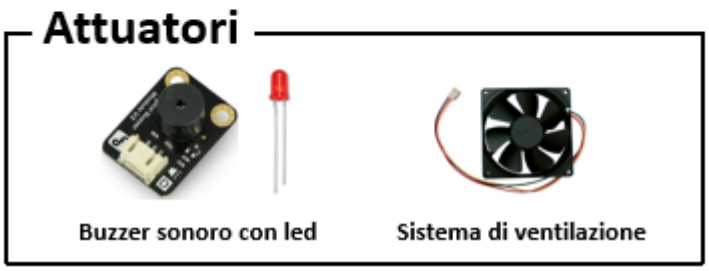
Parameters	Details
Connectivity and Processing Module	ESP8266 with custom firmware
MCU	32-bit RISC CPU: Tensilica Xtensa LX106
Power	5V/1A DC via Micro-USB port or 5V and GND pins
Operating Voltage	3.3V
CPU Clock Frequency	80 MHz
MCU Internal Memory	64 KB of instruction RAM; 96KB of data RAM
MCU External Memory	4 MB Flash memory [QSPI]
GPIO pins	5 Digital pins [3.3V logic]
ADC	1 pin 10 bit ADC [0-1V input]
PWM	All 5 Digital pins capable of PWM [Software PWM]
Connectivity	
WiFi	802.11 b/g/n Automatic AP mode if not connected to WiFi WEP/WPA/WPA2 authentication
UART	8-N-1 3.3V TTL UART [using TX, RX, GND pins] [9600 baudrate]
Cloud	Default: Bolt Cloud (https://cloud.boltiot.com/) Optional: Custom cloud using Bolt APIs
LED Indicators	
WiFi LED - WiFi connectivity	1) Slow blinking: Trying to find and connect to WiFi network 2) Fast blinking: User has connected via Bolt IoT app for setup 3) Stable: Connected to WiFi
Cloud LED – Bolt Cloud connectivity	1) Stable: Connected to Bolt Cloud 2) Off: Not connected to Bolt Cloud 3) Dim: Insufficient power/ incorrect boot
Dimensions	35mm x 35mm
Boot Time	less than 1 second
BOLT Cloud	
Link to Cloud:	https://cloud.boltiot.com/
Features:	1) Remote Configuration 2) Code Editor 3) Smartphone App 4) Notification(SMS & E-Mail) 5) Visualisation & Analysis 6) Remote Control 7) Device Sharing 8) OTA Updates

- $Adattatore\ di\ livello\ logico\ o\ Level\ shifter\ BSS138$, è importante assicurarsi che i livelli di tensione dei segnali TX ed RX relativi alla comunicazione seriale UART tra TelosB ed Arduino e tra Arduino e Bolt siano compatibili, per evitare danni o malfunzionamenti. Poiché TelosB e Bolt sono caratterizzati da una tensione operativa pari a $3V$, mentre Arduino funziona a $5V$, è pertanto richiesto l'utilizzo di un level shifter, il quale è progettato per garantire che i segnali vengano convertiti correttamente tra due diversi livelli di tensione. Può gestire sia la conversione in ingresso che in uscita, quindi è adatto per comunicazioni bidirezionali come UART.

- $Modulo\ relay\ 5V\ a\ singolo\ canale$, è un dispositivo elettromeccanico che utilizza una corrente elettrica per aprire o chiudere i contatti di un interruttore. Rappresenta un componente elettrico essenziale per quasi ogni attuatore azionato elettricamente. Controlla i circuiti utilizzando segnali indipendenti a bassa potenza. Pertanto, i micro-controllori possono azionare qualsiasi dispositivo indipendentemente dal consumo energetico. In questo contesto, il relay funge da supporto ad Arduino per l'azionamento del sistema di ventilazione.

- $Sensori\ e\ attuatori$, i parametri ambientali, tra cui temperatura dell'aria, umidità relativa dell'aria, concentrazioni di LPG, concentrazioni di CO e presenza di fiamme, sono stati selezionati per determinare le condizioni ambientali indoor in ambito industriale o residenziale. Per misurare questi parametri ambientali sono stati utilizzati sensori quali: Sensirion SHT11 (temperatura e umidità), MQ-6 (LPG), MQ-7 (CO), KY-026 (rilevatore di fiamma).
  Il sensore Sensirion SHT11 risulta già integrato sul TelosB, mentre gli altri sensori esterni vengono interfacciati con il mote attraverso il connettore di espansione a 16-pin, sfruttando i canali ADC.
  Un circuito partitore di tensione viene utilizzato per stabilizzare la tensione, poiché i sensori esterni ed il mote, con cui questi si interfacciano, operano a diversi livelli di tensione.
  Il sistema è stato inoltre dotato di attuatori quali: un allarme sonoro, costituito da un buzzer passivo e un led, che entra in funzione qualora venga rilevata la presenza di fiamme, e un sistema di ventilazione che viene attivato qualora le concentrazioni di LPG e CO superano i valori di soglia impostati. Quest'ultimo, per questioni di indisponibilità e spazio, è stato realizzato attraverso l'utilizzo di un semplice led.





La fig. 7 illustra le principali caratteristiche dei sensori ed attuatori impiegati.

Sensore	Parametri ambientali	Caratteristiche	Applicazioni	Specifiche elettriche
Sensirion SHT11 	Temperatura & Umidità 	— Basso consumo — Non richiede calibrazione — Stabilità e robustezza — Alta sensibilità — Alta precisione ($\pm 3\%\ RH, \pm 0.4\ °C$)	Eccellente per progetti commerciali o domestici che richiedono tali letture	— Tensione operativa: $2.4 - 5.5\ V$ — Corrente operativa: $28\ \mu A$ — Consumo di potenza: $30\ \mu W$ — Range operativo °C: $-40\ -125$ °C — Range operativo $\%RH$: $0 - 100\ \%RH$
MQ-6 	Gas Propano (LPG) 	— Alta sensibilità a LPG — Bassa sensibilità al fumo — Stabilità e robustezza — Risposta veloce	Utilizzati nei dispositivi di rilevamento delle perdite di gas, quali LPG e metano, in ambito domestico e industriale	— Tensione operativa: $5\ V$ — Corrente operativa: $150\ mA$ — Range operativo: $200\ -10000\ ppm$ — Resistenza di carico R_L: Regolabile — Resistenza sensore R_S: $10 - 60\ k\Omega$
MQ-7 	Monossido di Carbonio (CO) 	— Alta sensibilità a CO — Stabilità e robustezza — Risposta veloce	Utilizzati nei dispositivi di rilevamento di monossido di carbonio in ambito domestico, industriale e automobilistico	— Tensione operativa: $5\ V$ — Corrente operativa: $150\ mA$ — Range operativo: $50\ -5000\ ppm$ — Resistenza di carico R_L: Regolabile — Resistenza sensore R_S: $2 - 20\ k\Omega$
KY-026 	Rilevatore di fiamma 	— Il fotodiodo collegato è sensibile allo spettro luminoso prodotto dalle fiamme — Risposta veloce — Alta sensibilità	Tipicamente impiegato nei dispositivi di rilevamento di incendi	— Tensione operativa: $3.3 - 5.5\ V$ — Range operativo: $760\ -1100\ nm$ — Angolo di rilevamento: $60°$

Attuatore	Caratteristiche	Applicazioni	Specifiche elettriche
Buzzer passivo con led 	N/A	Utilizzato in sistemi per la segnalazione di pericoli tramite l'emissione di un segnale acustico e luminoso	— Tensione operativa: $3.3V - 5V$
Sistema di ventilazione 	N/A	Utilizzato in sistemi per raffreddare o purificare l'aria in presenza di alti livelli di sostanze tossiche o infiammabili	— Tensione operativa: $12\ V$

I dati raccolti dal nodo sensore, attraverso i canali ADC, rappresentano dati grezzi. Queste letture ADC grezze dovranno essere pre-elaborate dal nodo sensore, convertendole in unità ingegneristiche idonee in funzione del parametro misurato. La pre-elaborazione dei dati serve a garantire la qualità dei dati per l'analisi. In particolare, per i dati di temperatura e umidità è sufficiente utilizzare una formula di conversione empirica, riportate di seguito, ottenendo in output il valore di temperatura e di umidità relativa, rispettivamente, in gradi Celsius e in termini percentuali.

$$°C = -43.9 + 0.01 \cdot Raw\_value$$

$$\%RH = -2.0468 + 0.0367 \cdot Raw\_value - (1.5955 \cdot 10^{-6} \cdot Raw\_value^2)$$

I sensori utilizzati per misurare i valori di LPG e CO devono essere inizialmente calibrati determinando il valore della resistenza in aria pulita, R_0, in funzione del valor medio di R_S in aria pulita e del rapporto $ratio = R_S/R_0$ in aria pulita. A questo punto, applicando l'interpolazione lineare tra i punti noti della curva di sensibilità dei sensori (vedi fig. 7-8), è possibile determinare la concentrazione di gas in ppm al variare del rapporto.

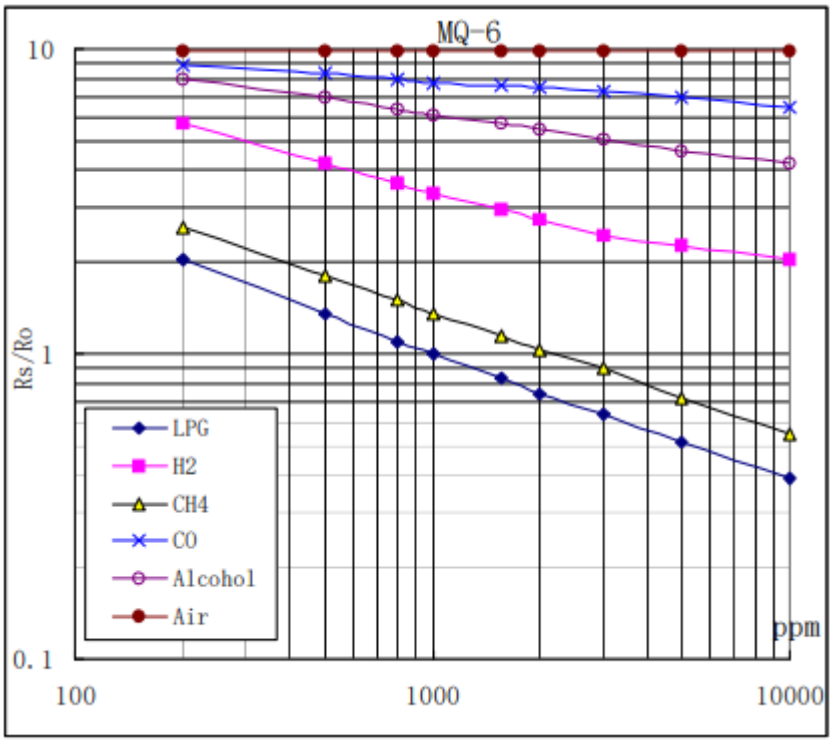


Figure 7 - Curva di sensibilità MQ-6 (LPG)

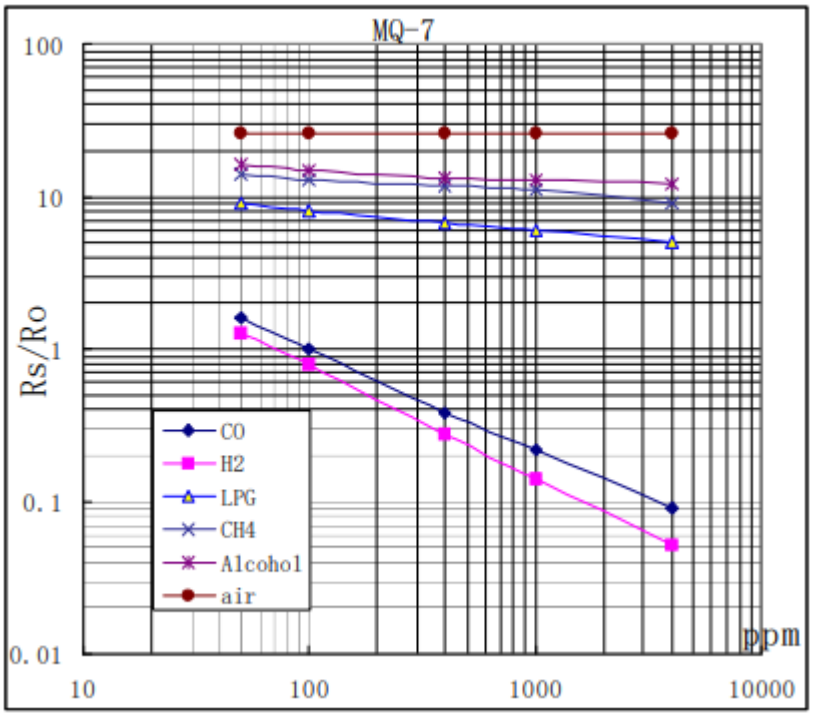


Figure 8 - Curva di sensibilità MQ-7 (CO)

$$V_{out} = \frac{ADC\_data}{4095} \cdot V_{cc}$$

$$R_S = R_L \cdot \left(\frac{V_{cc}}{V_{out}} - 1\right)$$

$$R_0 = \frac{R_{S,clean\_air}^{avg}}{ratio_{clean\_air}}$$

Se il rapporto, $ratio = R_S/R_0$, si trova tra due punti noti $(ppm_1, ratio_1)$ e $(ppm_2, ratio_2)$, è possibile calcolare la concentrazione ppm corrispondente al valore $ratio$, come:

$$ppm = ppm_1 + \frac{(ppm_2 - ppm_1) \cdot (ratio - ratio_1)}{ratio_2 - ratio_1}$$

Infine, l'output analogico del sensore di rilevamento di fiamme, digitalizzato dall'ADC, viene semplicemente convertito in un valore di tensione tramite la seguente relazione:

$$V_{out} = \frac{ADC\_data}{4095} \cdot V_{cc}$$

In presenza di fiamme la tensione in uscita si riduce gradualmente a zero. In caso contrario, la tensione in uscita risulta pari alla tensione di alimentazione del sensore.

2.3 Comunicazione: protocollo UART

La fase di debug e la comunicazione TelosB-Arduino e Arduino-Bolt IoT viene gestita dal protocollo $UART$ $(Universal\ Asynchronous\ Receiver-Transmitter)$ per permettere di avere i dati disponibili per la trasmissione Wi-Fi. Si tratta di un protocollo di comunicazione seriale asincrono in cui il formato del pacchetto e la velocità di trasmissione (baundrate) sono riconfigurabili. In altre parole, non richiede un segnale di clock condiviso tra il trasmettitore ed il ricevitore, a differenza delle comunicazioni sincrone (SPI, I2C).
La comunicazione può essere:

- Simplex, utilizzo di una singola linea dati con trasmissione dati unidirezionale;
- Half-duplex, utilizzo di una singola linea con trasmissione dati bidirezionale (ma non simultaneamente).
- Full-duplex, utilizzo di due linee dati separate (TX e RX) con trasmissione bidirezionale simultanea. Rappresenta il tipo di comunicazione adottata dal sistema, ed un esempio è illustrato in fig.9;

Poiché l'UART non utilizza segnali di clock, i bit di dati vengono trasmessi uno per uno, dal meno significativo al più significativo, racchiusi tra un bit di start ed un bit di stop per permettere al ricevitore di sapere quando iniziare a leggere i bit in arrivo, poiché questi due bit rappresentano l'inizio e la fine del payload. Inoltre, entrambe le parti trasmettano allo stesso baundrate prestabilito per garantire la correttezza dei dati ed evitare errori nella trasmissione. Eventualmente è possibile aggiungere un bit di parità alla fine del payload per il rilevamento degli errori. Questo consente al ricevitore di comunicare se dei dati sono cambiati durante la trasmissione. Conta il numero di bit con valore '1' e controlla se il totale è un numero pari o dispari.

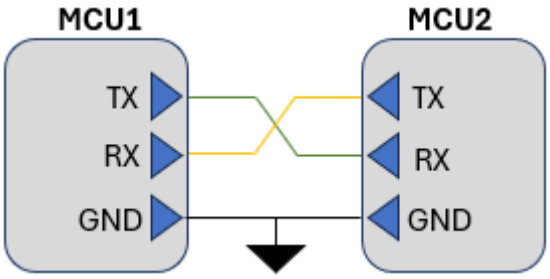


Figure 9 - Comunicazione UART full-duplex

Il formato del pacchetto UART, in riferimento ad un singolo byte di dati, è generalmente strutturato come illustrato in fig.10.

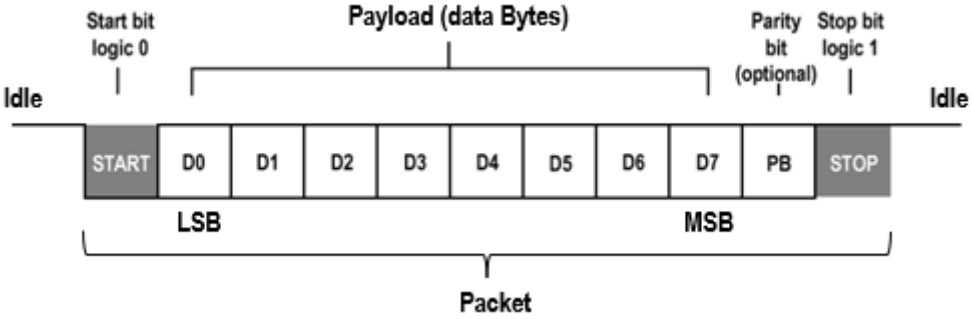


Figure 10 - Pacchetto UART

In questo contesto, il pacchetto UART è stato esteso con un ulteriore bit che fornisce informazioni sulla lunghezza in byte del frame dati complessivo rappresentante le informazioni sui parametri ambientali. Ciò è stato necessario, poiché viene utilizzato un frame dati dinamico che si intervalla periodicamente tra due lunghezze distinte. Ciò è dovuto al fatto che, riguardo ai dati di temperatura e umidità relativa, diversamente ai dati di LPG, CO e Flame, quello che si desidera analizzare e trasmettere è il valor medio su un periodo di 30 secondi, in funzione di misurazioni su questi parametri acquisite ogni secondo.

2.4 Progettazione software

Nei microcontrollori e nei computer, la programmazione si riferisce alla scrittura di una sequenza di direttive, eseguibili dal processore in un ordine dato per svolgere un'attività preimpostata. Comprende il debug e la risoluzione dei problemi delle istruzioni e delle sequenze di istruzioni per garantire l'implementazione dell'attività di raccolta. Esistono determinate parole, grammatica e regole per i linguaggi di programmazione.
Esistono tre tipi di linguaggi di programmazione per i microcontrollori a seconda della vicinanza delle istruzioni nella somiglianza del linguaggio delle operazioni eseguite dal controller. I tre livelli di linguaggi di programmazione sono: codice macchina, linguaggio assembly e linguaggio di alto livello. Nel codice macchina le istruzioni sono scritte in forma binaria (cifre 0 e 1), memorizzate come tensioni "BASSE" e "ALTE". È il livello più basso di linguaggio di programmazione e i microcontrollori comprendono questo linguaggio. Il linguaggio assembly è la rappresentazione inglese del codice macchina, si basa su mnemonici e codici esadecimali. La conoscenza dell'architettura del microcontrollore è fondamentale in questo linguaggio. Il linguaggio di alto livello utilizza parole e istruzioni facilmente comprensibili dall'uomo. Esempi di linguaggi di alto livello sono BASIC, Pascal, C++ e Java. Un programma chiamato compilatore consente la conversione dei programmi in linguaggio di alto livello in formato binario (cifre 0 e 1) che può essere caricato nella memoria del computer per l'esecuzione. I linguaggi di alto livello sono facili da usare, tuttavia, i linguaggi assembly hanno i seguenti meriti:

1. I loro programmi sono più veloci da eseguire e richiedono meno spazio di memoria.
2. Consentono lo sfruttamento diretto delle funzionalità dei microcontrollori.
3. Migliorano il controllo diretto e accurato delle risorse del microcontrollore come RAM, porte ecc.
4. Hanno meno regole e restrizioni.

L’applicazione software del sistema è stata strutturata in tre parti principali:

- Una parte, compilata e installata sul micro-controllore TelosB, è stata sviluppata nell’ambiente TinyOS, basato sul linguaggio di programmazione nesC. I punti chiave riguardano l’acquisizione, l’elaborazione e la trasmissione ad Arduino dei dati.

- Una parte, compilata e installata sul micro-controllore Arduino, è stata sviluppata tramite il software Arduino IDE, scritto nei linguaggi C e C++. I punti chiave riguardo la ricezione, elaborazione e trasmissione sul Cloud Bolt, da parte di Arduino, dei dati trasmessi dal TelosB.

- Una parte, utilizzata per il design dell'interfaccia utente su browser web, è stata sviluppata direttamente sul Cloud Bolt, utilizzando un particolare framework JavaScript che è parte integrante dell'ecosistema Bolt IoT. I punti chiave riguardano la creazione di una pagina web per visualizzare in forma di grafici dinamici, in tempo reale, i valori di tutti i sensori connessi al sistema. Quando si accede alla pagina, tutti i dati effettivi dei sensori vengono recuperati dal server Web.

Viene, inoltre, eseguita una fase di debug tramite interfaccia USB, stampando a schermo i dati, per verificare che le varie fasi di acquisizione, elaborazione, trasmissione e ricezione, siano eseguite correttamente.

Il diagramma del flusso di lavoro del firmware caricato nella memoria ROM di tipo flash del TelosB è illustrato nella fig. 11. Vengono eseguite prima le operazioni di inizializzazione, tra cui l'inizializzazione delle variabili, delle porte ADC, della comunicazione UART e di ciascun sensore. Dopo il completamento dell'inizializzazione, viene avviato, attraverso l'interfaccia boot, il flusso principale del ciclo. Ogni sensore raccoglie periodicamente le informazioni sull'ambiente circostante. Il TelosB elabora e incapsula i dati in un frame di dati dopo il completamento della raccolta. I frame di dati elaborati, una volta verificata la disponibilità del bus UART, vengono trasmessi ad Arduino mettendosi in attesa del segnale di acknowledgment il quale conferma la corretta ricezione del pacchetto. Dopo ogni trasmissione dati, il TelosB ripete il processo di raccolta e trasmissione.

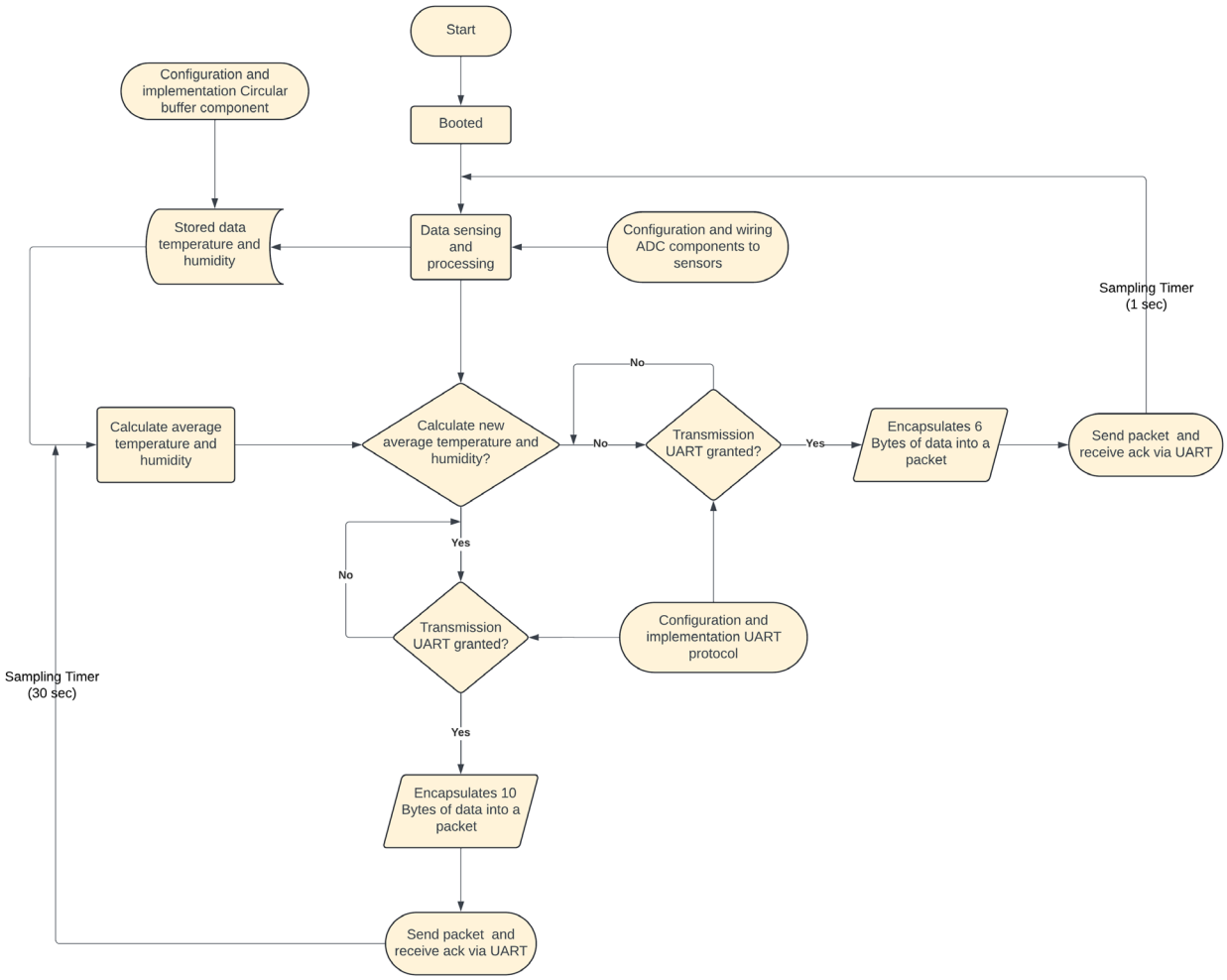


Figure 11 - Flowchart firmware TelosB

Il firmware comprende:

- Un file denominato $Makefile$, utilizzato per dare informazioni al compilatore su regole e proprietà dell'applicazione nesC per generare il file eseguibile. Il Makefile in un'applicazione nesC serve a:
  - Definire il componente principale (in questo caso, $EnviroSafeAppC.nc$) dell'applicazione nesC;
  - Definire le regole generali di compilazione fornite da tinyOS;
  - Specificare le directory dove trovare librerie esterne o componenti aggiuntivi personalizzati richiesti per l'applicazione nesC;
  - Definire delle costanti (es. dimensione buffer, tempi di campionamento, ecc.).

  Ciò è fondamentale per consentire al compilatore di generare il file binario eseguibile dell'applicazione nesC.
- Due componenti, ovvero, un file module chiamato $EnviroSafeC.nc$ e un file configuration di primo livello chiamato $EnviroSafeAppC.nc$.

Il modulo $EnviroSafeC.nc$ fornisce l'implementazione effettiva dell'applicazione nesC, mentre la configurazione $EnviroSafeAppC.nc$, che rappresenta la struttura globale dell'applicazione nesC, fornisce il collegamento tra le interfacce utilizzate dall'applicazione nesC ed i componenti che forniscono tali interfacce. Questi possono essere componenti hardware (come sensori o LED) o software (come timer o moduli di comunicazione). In altre parole, ha il compito di descrivere come i diversi componenti del sistema devono essere collegati tra loro, ovvero, definisce l'architettura dell'applicazione, specificando quali componenti interagiscono e come le interfacce tra questi vengono collegate.

Quest'ultimo rappresenta anche il componente, utilizzato dal compilatore nesC per generare il file eseguibile.

Un'interfaccia dichiara un set di funzioni chiamate comandi ed eventi, mentre un componente che fornisce tale interfaccia, definisce il corpo di implementazione dei comandi e notifica gli eventi. Viceversa, un componente che utilizza tale interfaccia invoca i comandi ed implementa i gestori/handlers degli eventi. Pertanto, un'interfaccia è bidirezionale, poiché si può essere fornitori oppure utilizzatori di tale interfaccia.

L'intero codice sorgente utilizzato in questa applicazione è fornito nella sezione appendice.

Il modulo $EnviroSafeC.nc$ utilizza diverse interfacce, quali: Boot, Timer, Leds, Read, CircularBuffer, ecc.

L'interfaccia Leds è utilizzata per accendere o spegnere i diversi leds presenti sul mote invocando i comandi On() e Off() nell'handler di un evento.

L'interfaccia Boot definisce un evento booted() attraverso il quale vengono eseguite delle azioni all'avvio del sistema. In questo contesto, si procede all'inizializzazione dei buffer circolari e all'avvio dei timer periodici.

Le interfacce timer definiscono un comando startPeriodic() per l'avvio dei timer e un evento fired() che viene innescato periodicamente, ogni qualvolta il timer scade. In particolare, l'innesco di un evento fired() (ogni secondo) contribuisce alla lettura dei dati dai sensori, alla loro formattazione e all'accesso del bus UART per la trasmissione dati, mentre l'innesco di un altro evento fired() (ogni 30s) contribuisce al calcolo del valor medio dei parametri di temperatura e umidità e alla loro formattazione per la trasmissione via UART.

Le interfacce Read definiscono il comando read() per la lettura dei dati dai sensori e un evento readDone() che viene innescato ogni qualvolta termina l'operazione di lettura. Questo evento contribuisce alla conversione delle letture ADC in unità ingegneristiche. Vengono configurati diversi canali ADC per poter accedere alle letture dei sensori esterni passando per l'ADC msp430adc12. Quest'ultimo è cablato ai sensori esterni come illustrato nel configuration file $EnviroSafeAppC.nc$.

Le interfacce CircularBuffer definiscono diversi comandi indispensabili per la configurazione ed il corretto utilizzo dei buffer circolari, impiegati per immagazzinare le letture al secondo di temperatura e umidità relativa, al fine di determinare in un secondo momento il valor medio.

Le interfacce di interpolazione definiscono un comando interpolate() che viene chiamato per determinare le concentrazioni in parti per milione di LPG e CO in seguito ad un processo di interpolazione lineare sulla curva di sensibilità dei sensori.

Infine, le interfacce Resource e UartSream vengono utilizzate per la gestione del protocollo di comunicazione UART. Esse definiscono i comandi request(), send(), receive() e release() che vengono chiamati per poter, rispettivamente, richiedere l'accesso al bus UART, trasmettere il pacchetto UART, ricevere l'ack che attesti la corretta ricezione dei dati trasmessi e rilasciare il bus UART. Gli eventi:

granted(), receivedByte(), sendDone() e receiveDone() contribuiscono, rispettivamente, all'accesso al bus UART per l'invio del pacchetto dati, al rilascio del bus UART per, rispettivamente, ogni byte trasmesso, ogni pacchetto trasmesso e ogni ack ricevuto.

Il software di controllo o firmware, convertito in codice macchina e caricato sul microcontrollore ATmega328P di Arduino, è stato sviluppato in Arduino IDE. Questo processo ha coinvolto la seguente procedura:

1. Creazione di un nuovo file.
2. Progettazione di uno sketch.
3. Compilazione del codice sketch.
4. Combinazione del codice sketch con le librerie Arduino.
5. Caricamento del codice sketch.

Arduino si occupa di ricostruire ed elaborare correttamente ciascun pacchetto proveniente dal TelosB, della trasmissione dell'acknowledgment al mote, attuare delle azioni automatiche in funzione dei valori dei parametri ambientali e della trasmissione di quest'ultimi al modulo Bolt IoT per poter essere caricati sul server Bolt Cloud e monitorati in tempo reale da remoto tramite un'apposita interfaccia web.
Flusso di lavoro del firmware (vedi fig. 12):

- Il dispositivo viene inizializzato;
- Si procede alla lettura ed elaborazione periodica del pacchetto dati trasmesso da TelosB via UART;
- In funzione dei valori acquisiti, il dispositivo abiliterá o meno degli attuatori e, procederá alla trasmissione dei dati via UART verso il modulo Bolt IoT, il quale provvederà a caricarli sul server Bolt Cloud per il monitoraggio da remoto in tempo reale.

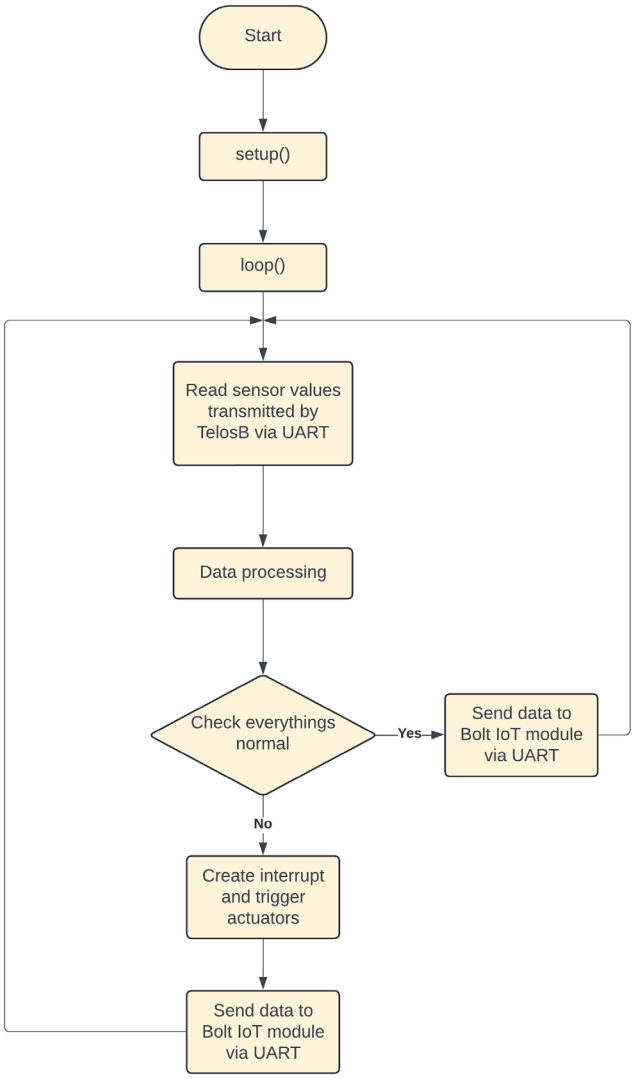


Figure 12 - Flowchart firmware Arduino Uno

Di seguito viene fornita una panoramica delle funzionalità principali del codice:

1. Comunicazione seriale: Il codice utilizza diverse interfacce seriali per comunicare con un dispositivo TelosB tramite la libreria *AltSoftSerial* e con il modulo Bolt IoT attraverso la libreria *boltiot*.
2. Gestione pacchetti: Viene implementato un protocollo per la lettura e il trattamento di pacchetti seriali che iniziano con un delimitatore di inizio (*SOP*, definito come <) e terminano con uno di fine (*EOP*, definito come >).
   La funzione loop() contiene il ciclo principale in cui avviene la ricezione dei pacchetti da TelosB.
   Il codice legge ogni byte proveniente da TelosB:
   - Quando viene letto il delimitatore di inizio (*SOP*), la ricezione del pacchetto inizia.
   - Quando viene letto il delimitatore di fine (*EOP*), la ricezione termina.
   - Durante la ricezione, i dati vengono salvati nell'array *inData* e il primo byte rappresenta la lunghezza del pacchetto.

3. Gestione dei dati: Dopo aver ricevuto un pacchetto completo, i dati vengono elaborati. Le variabili *Propane*, *CO_ppm*, *flame*, *Temperature*, *e humidity* sono estratte e gestite a seconda dei loro valori, e sono utilizzate anche per controllare:
   - Buzzer e LED: Se il valore della fiamma è inferiore a 3, viene attivato un allarme con un buzzer e un LED.
   - Sistema di ventilazione: Se i valori di LPG o CO superano soglie specifiche, viene attivato un sistema di ventilazione.
4. Trasmissione dei dati: I dati ambientali vengono inviati al modulo Bolt IoT per il monitoraggio in cloud, in tempo reale.
5. Invio di ACK a TelosB: Viene inviata un valore pari ad 1 per confermare la corretta ricezione del pacchetto dati, in caso contrario, ack = 0.

Viene eseguita una fase di debug su seriale, tramite interfaccia USB, stampando a schermo i dati per verificare che le varie fasi di acquisizione, elaborazione, trasmissione e ricezione, siano eseguite correttamente.

Per permettere all'utente di poter monitorare il proprio sistema, è stata scelta la piattaforma di analisi Bolt IoT, la quale consente di archiviare (sul proprio server cloud), visualizzare e analizzare flussi di dati da remoto, in tempo reale attraverso un'apposita pagina web o app mobile. I servizi forniti da questa piattaforma sono stati discussi precedentemente, quando è stato descritto il modulo Bolt IoT.
Dunque, in questo contesto la piattaforma Bolt IoT agisce come server, mentre il modulo Bolt (ESP8266-12S) è stato utilizzato come client.

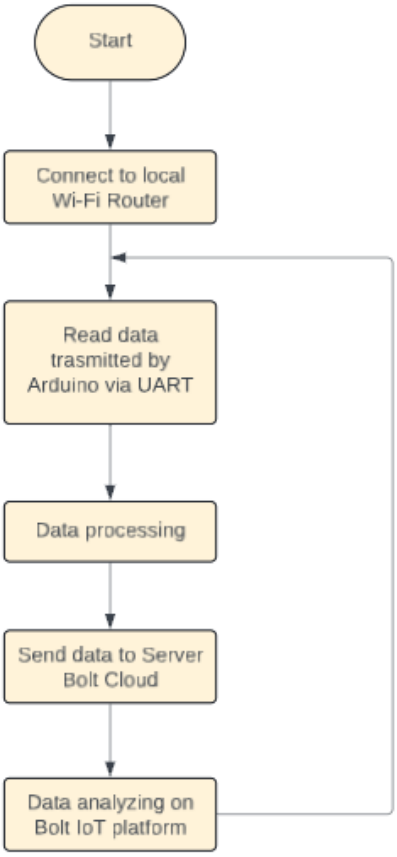
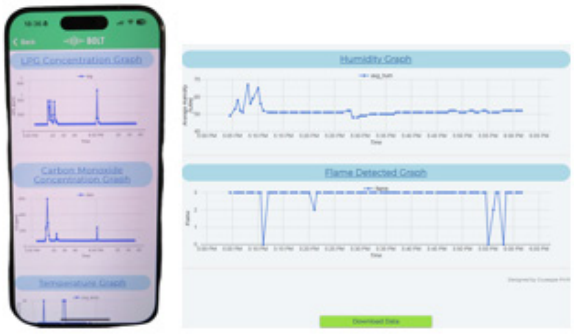


Figure 13 - Flowchart Bolt IoT

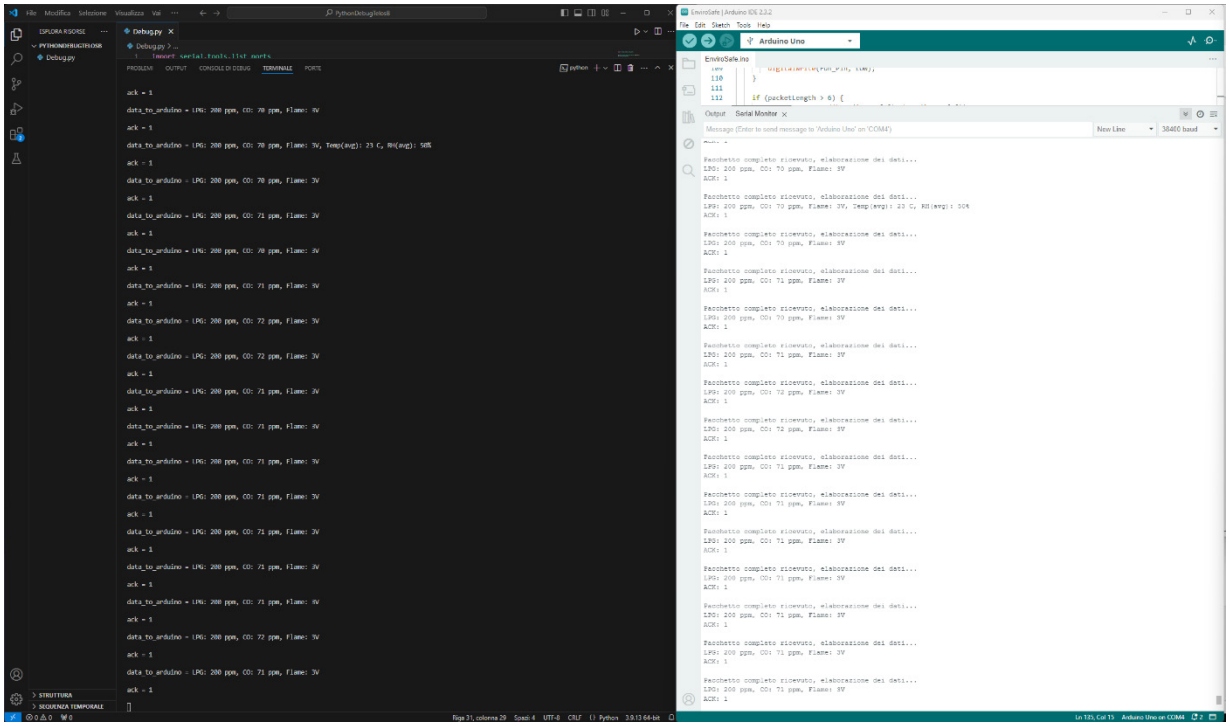
L’applicazione web/mobile, oltre al monitoraggio in tempo reale, consente di visualizzare i dati storici, di tutti i sensori collegati, in quattro diverse prospettive temprali: giorno, settimana, mese e anno.
La pagina web/mobile sviluppata è impostata per mostrare il valore di tutti i sensori connessi al sistema. Quando si accede alla pagina tutti i dati effettivi dei sensori vengono recuperati dal server web (Server Bolt Cloud) e visualizzati sottoforma di grafici dinamici.

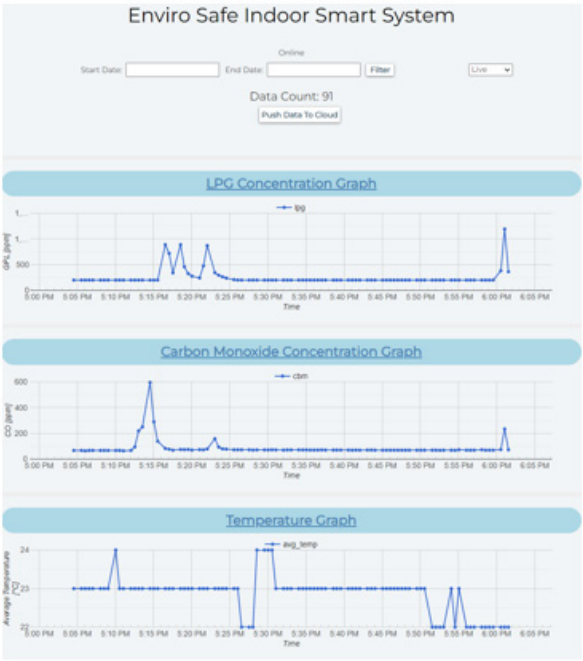
3 Risultati sperimentali

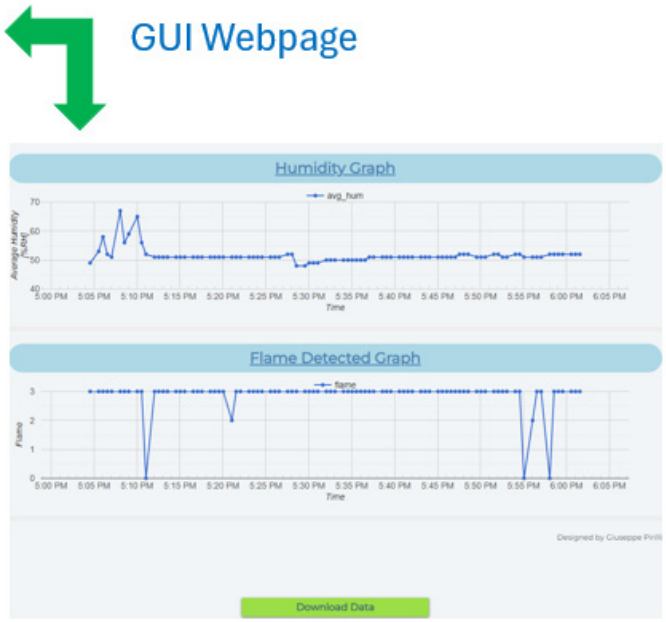
Setup sperimentale



GUI Webpage/App mobile







4 Conclusioni

È importante disporre di sistemi domestici per rilevare tempestivamente la presenza di incendi e gas pericolosi, poiché questi possono rappresentare un rischio per la salute e la sicurezza delle persone. Il prototipo sviluppato implementa sensori per identificare e monitorare diversi parametri ambientali, tra cui gas, quali propano e monossido di carbonio (sostanza legata alla presenza di fumo nell'aria), oltre alla capacità di rilevare la presenza di fiamme e, quindi eventuali incendi nella loro fase iniziale. È inoltre equipaggiato di attuatori azionati automaticamente, quali: allarme sonoro per avvisare le persone di un'emergenza incendio immediata, in modo da evacuare l'ambiente in cui si verifica tale accumulo di incendio o agire tempestivamente per estinguere l'incendio e, un sistema di ventilazione per purificare l'aria in presenza di alte concentrazioni di gas pericolosi.
Un potenziale approccio per migliorare le prestazioni del sistema, in termini di efficienza, accuratezza e scalabilità, è stato identificato nell'integrazione di tecnologie IoT. L'archiviazione, l'elaborazione, l'analisi ed il monitoraggio dei dati da remoto, in tempo reale, sono alcune delle peculiarità rese possibili dall'IoT e dalle tecnologie correlate. Dunque, sistemi intelligenti basati su IoT offrono un miglioramento significativo rispetto al metodo tradizionale. Questi progressi contribuiscono a creare ambienti di vita più sicuri, sottolineando l'importanza della continua ricerca e sviluppo in questo campo.
Nel complesso, il sistema sviluppato fornisce una soluzione economica, affidabile e comoda per rilevare e monitorare in loco o da remoto, in tempo reale, le condizioni ambientali indoor in ambito residenziale. La sua capacità di rilevare incendi e gas pericolosi, nonché la sua accuratezza e facilità d'uso, lo rendono uno strumento prezioso per garantire la sicurezza delle persone.

5 Sviluppi futuri

Possibili upgrade da apportare al sistema:

- Adattare il sistema per uso in ambiente industriale, oltre che residenziale;
- Ricerca sull'uso di un sistema di alimentazione a batteria;
- Integrare ulteriori sensori per il rilevamento di altri gas pericolosi (CO2, NH4, ammoniaca, ecc) e della qualità dell'aria interna in generale (VOC index). L'uso di sensori di misurazione di differenti gas o sostanze tossiche è un vantaggio fondamentale in quanto consente un monitoraggio più accurato e completo dell'ambiente. In definitiva, il metodo proposto può aumentare significativamente la consapevolezza dei livelli di inquinamento nell'ambiente, consentendo alle persone di adottare precauzioni sanitarie e contribuire alla riduzione dei materiali inquinanti;
- Inviare notifiche all'utente via SMS/e-mail qualora venga rilevata la presenza di un incendio o le concentrazioni di gas superano una soglia prestabilita (la piattaforma Bolt IoT fornisce questo tipo di servizio);
- Controllo da remoto degli attuatori tramite piattaforma web/mobile IoT;
- Sviluppo di un'applicazione web/mobile proprietaria, senza usufruire di piattaforme IoT di terze parti;
- Dotare il sistema di modulo bluetooth per consentire il monitoraggio in loco in assenza di connessione internet tramite un'apposita applicazione pc/mobile
- Integrare un algoritmo di machine learning addestrato sul storico dei dati per migliorare l'accuratezza, la reattività e la capacità di previsione del sistema. Ecco alcune aree in cui il machine learning può fare la differenza:
    - Riconoscimento di schemi/tendenze e anomalie nei dati dei sensori: I sensori di gas e di fumo raccolgono continuamente dati sui livelli di sostanze come monossido di carbonio e altri gas pericolosi. Un modello di machine learning addestrato su dati storici potrebbe imparare i modelli normali di emissioni di gas o di fumo e rilevare anomalie che indicano un pericolo imminente. Ad esempio, se ci sono picchi ricorrenti nei livelli di gas in momenti della giornata in cui la cucina è usata o si usa un riscaldamento che emette fumo temporaneo aumentando i livelli di monossido di carbonio, il sistema potrebbe riconoscerli come normali. Tuttavia, un picco anomalo in un momento insolito (ad esempio, durante la notte) potrebbe essere interpretato come un potenziale rischio di fuga di gas o incendio e attivare un allarme. Dunque, utilizzando il machine learning, il sistema potrebbe imparare a distinguere meglio tra situazioni pericolose e non pericolose, riducendo i falsi allarmi.
    - Previsione di pericoli in base a condizioni ambientali: Gli algoritmi di machine learning possono analizzare i dati storici per creare modelli predittivi che stimano il rischio di un evento pericoloso, come una fuga di gas o un incendio, in base a diverse condizioni ambientali. Ad esempio, un modello potrebbe rilevare che aumenti persistenti della temperatura in combinazione con determinati livelli di gas aumentano la probabilità di incendio. Il sistema potrebbe quindi attivare un allarme preventivo prima che la situazione diventi critica.
    - Manutenzione predittiva dei sensori: Il machine learning può essere utilizzato per monitorare lo stato di salute dei sensori stessi. Un sistema può imparare a riconoscere

quando un sensore sta cominciando a fornire dati anomali o rumorosi, suggerendo che il sensore potrebbe essere guasto o che potrebbe richiedere manutenzione o sostituzione. Ad esempio, se un sensore di gas comincia a dare letture errate, il sistema può avvisare l'utente che è necessario intervenire, evitando potenziali malfunzionamenti o mancate rilevazioni future.