

Smart Lighting System

Simone Leoni

`simone.leoni@studenti.unimi.it`

1 Introduzione

La continua evoluzione delle tecnologie attuali e, l'inizio della progettazione delle *smartcity*, ha reso necessaria l'evoluzione di tutti i dispositivi all'interno del contesto urbano. Attualmente, tra le varie evoluzioni proposte, una tipologia di dispositivi che ha visto un grosso sviluppo verso una versione *smart* sono gli impianti di illuminazione.

Lo stato dell'arte attuale vede sistemi di illuminazione che presentano:

- Controllo automatico del livello di luce emessa in base alla luminosità ambientale.
- Incremento dell'intensità luminosa al passaggio di individui nelle vicinanze.
- Controllo remoto dello stato di funzionamento, accensione e spegnimento dei singoli dispositivi.

Tutte queste caratteristiche permettono di avere una gestione più efficiente dal punto di vista energetico dell'illuminazione urbana, non risulta infatti necessario mantenere livelli di illuminazione elevate nelle aree poco trafficate. La possibilità inoltre di conoscere lo stato di funzionamento dei singoli dispositivi permette una manutenzione più efficiente.

La versione del progetto realizzato prevede una serie di feature presenti anche nello stato dell'arte attuale aggiungendo alcune piccole modifiche per una migliore integrazione all'interno delle *smart city* e un tentativo di migliorare il contesto di sicurezza urbana. Il sistema realizzato prevede le seguenti feature:

- Controllo del sistema di illuminazione da remoto con accensione e spegnimento delle varie features tramite rete LoRa.
- Livello di illuminazione controllato automaticamente in base al livello di luce ambientale.
- Incremento dell'intensità luminosa al passaggio di individui nei pressi del sistema.

- *Panic button* per la segnalazione di eventuali situazioni di emergenza alle forze dell'ordine e emissione di un segnale sonoro per attirare l'attenzione dei passanti.

L'aggiunta di un *panic button*, come evidenziato in precedenza, è stato pensato per migliorare il livello di sicurezza urbana. Grazie a questo elemento, infatti, adeguatamente integrato con telecamere di sicurezza, sarebbe possibile ottenere una segnalazione tempestiva delle situazioni di emergenza aggiungendo inoltre la possibilità di attirare l'attenzione dei passanti e cercare quindi di sventare le situazioni di pericolo e/o emergenza anche tramite l'aiuto di cittadini presenti in loco.

2 Hardware utilizzato

Il sistema realizzato presenta diverse componenti hardware:

- Scheda di sviluppo STM32F411e-disco: essendo il progetto un proof of concept è stata utilizzata una development board commerciale; per poter rendere il progetto in questione, un possibile dispositivo atto alla vendita, sarebbe necessaria la realizzazione di un PCB custom contenente tutto l'hardware necessario e una forma consona alla dissipazione del calore.
- Matrice led CharlieWing: elemento utilizzato per generare l'illuminazione necessaria. La matrice utilizzata viene comandata tramite *i2c* all'indirizzo 0x74.
- Scheda di sviluppo Raspberrypi Pico: per realizzare la comunicazione wireless senza aggiungere un eccessivo overhead sull'MCU è stato introdotto un coprocessore in modo da gestire autonomamente la comunicazione wireless tramite LoRa. Per effettuare la comunicazione tramite LoRa viene utilizzato un dispositivo WaveShare LoRa Shield basato su SX1262.
- Sensori di movimento SR602: per poter aumentare l'intensità luminosa al passaggio di cittadini nei pressi del sistema di illuminazione sono stati introdotti dei sensori di movimento per rilevarne la presenza.
- Fotorisistore: la presenza di un fotorisistore permette di rilevare il livello di luminosità ambientale e regolare l'intensità della luce emessa conseguentemente.
- Panic button: la presenza di questo bottone è pensata per la segnalazione delle situazioni di emergenza.
- Panic buzzer attivo: utilizzato per l'emissione di un segnale sonoro atto ad attirare l'attenzione in situazioni di emergenza.

In Figura 1 è evidenziata la schematica di collegamento dei dispositivi hardware coinvolti.

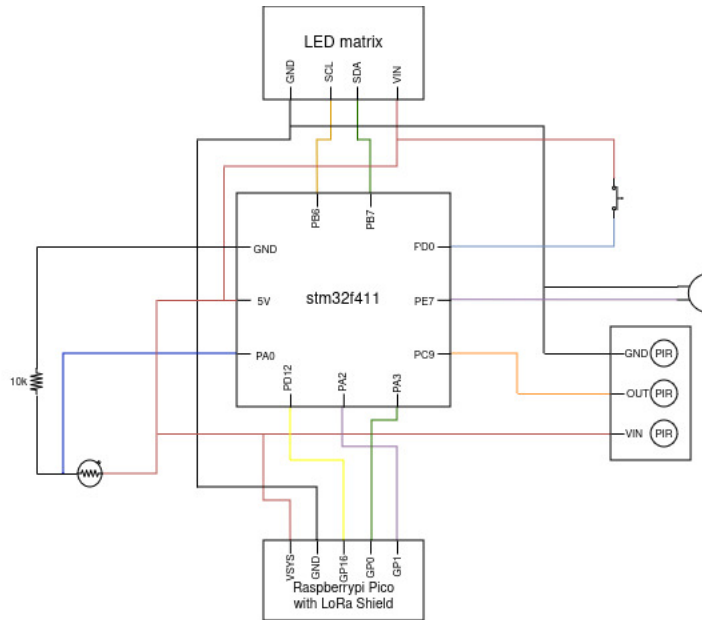


Figure 1: Schematica di collegamento dei dispositivi hardware

3 Architettura Software

Dal punto di vista software è necessario distinguere due diverse architetture: l'architettura dal punto di vista dell'MCU e l'architettura dal punto di vista del coprocessore wireless.

3.1 MCU

Il software presente sull'MCU, stm32f411, responsabile del funzionamento generale del sistema, risulta essere suddiviso in tre elementi principali: un componente di comunicazione con il coprocessore wireless, un componente per la gestione dell'illuminazione e una parte dedicata alle segnalazione di emergenza. Pur essendo tre le componenti del sistema, il software risulta essere suddiviso in quattro moduli, oltre ai tre moduli precedentemente indicati, infatti, è presente un modulo di gestione dei sensori. Questo modulo può essere considerato parte della componente atta alla gestione dell'illuminazione essendo l'unico modulo che ne sfrutta il funzionamento.

3.1.1 Modulo Light

Il modulo *Light*, come si può intuire dal nome, risulta essere il modulo atto alla gestione del sistema di illuminazione. Come si può notare in Figura 2, il modulo presenta un'interfaccia ad alto livello per le varie funzioni supportate e,

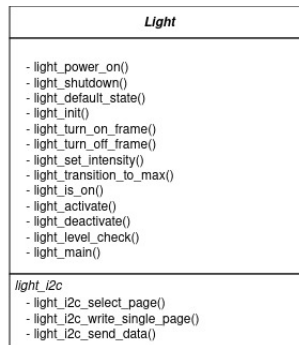


Figure 2: Modulo Lights

come livello di astrazione della comunicazione con la matrice, un sottomodulo *i2c_light*. Il modulo *Light* offre la possibilità di attivare e disattivare il sistema di illuminazione, incrementare in maniera graduale il livello di illuminazione prodotto per migliorare la luminosità in caso di passaggio dei cittadini e regolare il livello di luminosità sfruttando il sensore di luce ambientale ad intervalli di 1 minuto. Per garantire la frequenza di controllo del livello di luminosità ambientale viene utilizzato il timer TIM4. In seguito all'incremento del livello di luminosità dovuto al passaggio di un cittadino, tramite l'utilizzo del timer TIM9, si garantisce che la luminosità verrà mantenuta all'intensità massima per dieci secondi dall'ultima volta in cui è stato rilevato del movimento nei pressi del sistema.

3.1.2 Modulo Sensors

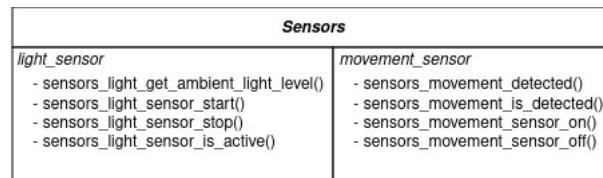


Figure 3: Modulo Sensors

Il modulo *Sensors*, come evidenziato in precedenza, può essere considerato parte della componente per la gestione dell'illuminazione.

Come si può notare in Figura 3 il modulo presenta due sottomoduli:

- *light_sensor* che, tramite ADC, verifica il livello di luce ambientale.
- *movement_sensor* che permette di verificare la presenza di individui nelle vicinanze del sistema.

3.1.3 Modulo Panic

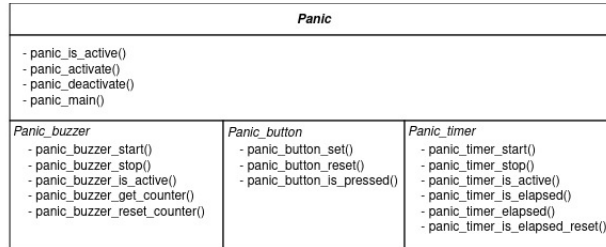


Figure 4: Modulo Panic

Il modulo *Panic* rappresenta l'elemento software che gestisce per intero la componente per la segnalazione delle emergenze. Il modulo, come avviene all'interno delle altre componenti visibile anche in Figura 4, presenta una serie di funzioni ad alto livello che forniscono tutte le funzioni necessarie per il funzionamento di base del componente. Sono presenti inoltre 3 sotto moduli, utilizzati direttamente dal modulo principale, che permettono di gestire singolarmente i vari elementi necessari:

- *panic_buzzer* che si occupa della gestione del buzzer per la generazione di segnali sonori.
- *panic_button* che gestisce le interazioni con il bottone di emergenza.
- *panic_timer* che utilizza il timer TIM3 per gestire l'alternarsi di accensione e spegnimento del buzzer ad intervalli di un secondo.

3.1.4 Modulo Comm

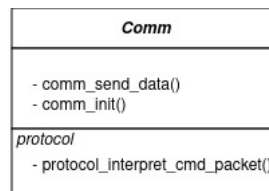


Figure 5: Modulo Comm

Il modulo *Comm* rappresenta l'unico elemento della componente atta alla comunicazione con il coprocessore wireless. Il modulo in questione utilizza la comunicazione **uart** per inviare e ricevere i dati dal coprocessore. Per quanto riguarda la ricezione dei dati, invece, l'interfaccia **uart** viene utilizzata in modalità *interrupt*, in questo modo la reazione alla ricezione dei dati risulta essere immediata.

Per poter interpretare correttamente i dati viene utilizzato il sottomodulo *protocol* che fornisce la possibilità di interpretare i comandi ricevuti dal router. Il modulo *Comm*, inoltre, viene utilizzato per l'invio periodico di un pacchetto, utilizzato per segnalare il corretto funzionamento del sistema, ad intervalli di circa tre secondi tramite l'utilizzo del timer di sistema TIM11. L'invio dei dati avviene utilizzando il DMA per evitare di occupare tempo CPU vista l'elevata frequenza di invio del pacchetto.

3.2 Coprocessore Wireless

Il componente hardware utilizzato come coprocessore wireless è un Raspberry Pico che utilizza il microprocessore *RP2040*. Questo microprocessore presenta due core con architettura Arm-M0+.

La presenza dei due core è stata sfruttata per implementare una suddivisione fisica, oltre che logica, dell'esecuzione dei due moduli presenti sul microcontrollore. L'architettura software presente sul coprocessore wireless è stata utilizzata anche all'interno del router, elemento atto all'invio dei dati e il controllo dello stato di funzionamento del sistema di illuminazione, sostituendo l'interfaccia di comunicazione seriale da *uart* a *usb*.

3.2.1 Modulo LoRa

Il modulo LoRa, utilizzato per la comunicazione radio, sfrutta la libreria RadioLib come driver per la gestione del LoRa shield collegato al Pico. Questo modulo utilizza invio e ricezione dei dati tramite interrupt che, come indicato all'interno della documentazione della libreria stessa, risulta essere il metodo più sicuro per evitare la perdita dei pacchetti.

3.2.2 Modulo Comm

Essendo la libreria utilizzata per la comunicazione wireless scritta interamente in C++, la libreria per la comunicazione seriale, come si può notare in Figura 6 utilizza una gerarchie di classi per introdurre astrazione rispetto all'hardware sottostante. In particolare la libreria realizzata *libcomm* presenta:

- la classe astratta *Protocol*: la quale rappresenta l'insieme delle funzionalità di base del protocollo di comunicazione seriale utilizzato.
- la classe *Uart*: che implementa la classe *Protocol* e fornisce il livello di astrazione rispetto l'interfaccia *uart* utilizzata.
- la class *Comm*: al cui interno è presente un puntatore ad una classe derivata da *Protocol* che gestisce la comunicazione tenendo conto del protocollo di comunicazione utilizzato a livello dell'applicazione. Per la comunicazione dei vari dati, infatti, è stato definito un protocollo di comunicazione per racchiudere le varie informazioni nel minor numero di byte possibile.

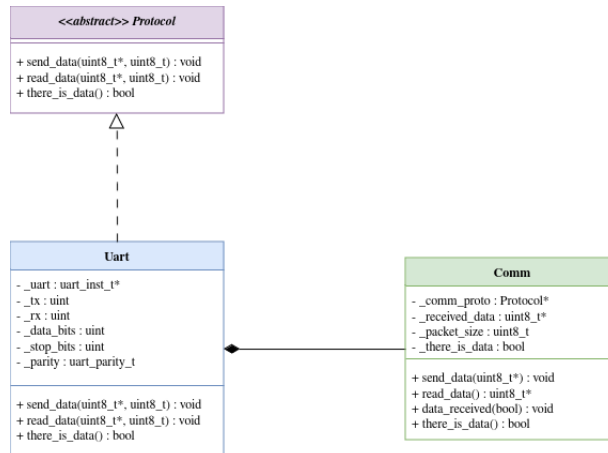


Figure 6: Modulo Comm del coprocessore

Come detto anche in precedenza, l'unico elemento che distingue il funzionamento del coprocessore wireless rispetto al router risulta essere l'interfaccia seriale utilizzata. All'interno del router, infatti, la libreria *libcomm* risulta essere uguale a quella presente sul coprocessore con l'unica differenza che la classe utilizzata per implementare la classe astratta *Protocol* risulta essere la classe *Usb* che fornisce un'astrazione per quanto riguarda la comunicazione tramite la porta microusb presente sul Pico.

3.3 Protocollo di comunicazione



Figure 7: Protocollo di comunicazione

Il protocollo di comunicazione è stato pensato per avere un numero fisso di byte, ogni pacchetto infatti contiene esattamente quattro bytes. Il significato dei singoli byte, come indicato in Figura 7, risulta essere il seguente:

- il primo byte viene utilizzato per indicare il dispositivo di destinazione del pacchetto.
- il secondo byte viene utilizzato per indicare il dispositivo che ha emesso il pacchetto.
- il terzo byte indica il tipo di pacchetto, viene impostato ad uno il bit corrispondente al tipo.

- il quarto byte viene utilizzato per indicare i dati aggiuntivi necessari per il tipo di pacchetto, questo byte viene utilizzato solo dai pacchetti di tipo *command* per indicare l'azione richiesta, in tutti gli altri pacchetti il byte risulta essere 0x00.

3.4 Interazioni tra i moduli

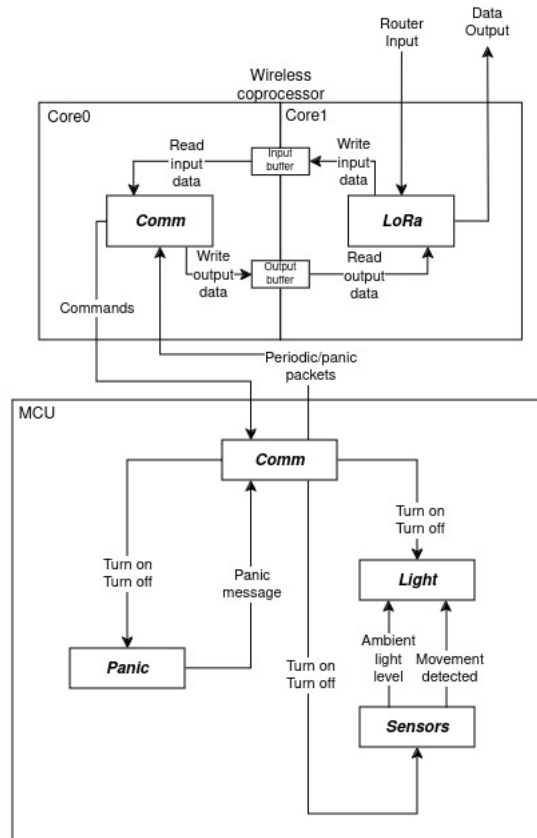


Figure 8: Diagramma di interazione delle componenti

La Figura 8 presenta un diagramma di interazione delle componenti. Come si può notare i moduli *Comm* sia dell'MCU che del coprocessore risultano essere le due componenti che permettono l'interazione tra i due elementi hardware. I dati che vengono inviati in uscita dall'MCU risultano essere i pacchetti generati dal sistema di gestione delle emergenze ed i pacchetti periodici che vengono utilizzati per segnalare il corretto funzionamento del sistema. I pacchetti che vengono inviati dal coprocessore all'MCU, invece, sono solamente i pacchetti contenenti i comandi. All'interno del coprocessore, per favorire lo scambio dei dati tra le due componenti software presenti, sono stati predisposti due buffer

circolari con accesso in mutua esclusione. Questi due buffer permettono di processare i dati ricevuti senza preoccuparsi dell'arrivo imminente di un altro pacchetto. Il modulo *Comm* dell'MCU, oltre a gestire la comunicazione con il coprocessore ha il compito di attivare e disattivare gli altri moduli in base ai comandi ricevuti dal router. Il modulo *Sensors* fornisce al modulo *Light*, solamente quando il modulo *Light* è acceso, i dati catturati dai sensori di luce ambientale e di movimento. Il modulo *Panic*, infine, genera i pacchetti per il modulo *Comm* utilizzati per segnalare al router le situazioni di emergenza.

4 Tempi di esecuzione

La parte del sistema per cui l'analisi dei tempi di esecuzione risulta essere la più significativa è la fase di inizializzazione. Durante questa fase, infatti, viene mandata una grossa mole di dati alla matrice led essendo necessaria l'inizializzazione di tutti i registri presenti sul dispositivo.

Un altro elemento per cui può essere effettuata l'analisi dei tempi di esecuzione sono le tempistiche di reazione ai comandi. Questa misurazione non risulta essere molto significativa perchè il sistema realizzato non è time sensitive ma rappresenta i tempi di reazione del sistema agli impulsi esterni.

I tempi misurati per le due fasi indicate sono i seguenti:

- inizializzazione del sistema: $33\ ms$
- reazione ai comandi: $\leq 1ms$

5 Analisi dei consumi

Dal punto vista energetico, risulta necessario distinguere diverse condizioni. È necessario evidenziare che, essendo un sistema di illuminazione la maggior parte dell'energia consumata viene assorbita dalla matrice led. Nell'eventualità della realizzazione del sistema in dimensioni maggiori adatte all'illuminazione stradale, i consumi risulterebbero maggiori. Come detto in precedenza possiamo distinguere due diverse condizioni di funzionamento:

- quando la matrice led risulta essere accesa vengono registrati dei consumi di circa $500\ mA$.
- quando la matrice led risulta essere spenta vengono registrati dei consumi di circa $200\ mA$.

La presenza del buzzer per la segnalazione delle emergenze e del coprocessore wireless producono delle oscillazioni al consumo registrato. Il coprocessore wireless, infatti, richiede un incremento del consumo di circa $40\ mA$ durante la fase di invio dei dati. Il buzzer, invece, durante l'emissione dei segnali sonori produce un incremento del consumo di circa $25\ mA$.