Sommario

[Smart Street Lights 2](#_Toc90305144)

[Introduzione 2](#_Toc90305145)

[Obiettivi e contributi del gruppo 2](#_Toc90305146)

[Materiali e strumenti utilizzati 2](#_Toc90305147)

[Collegamenti dei componenti 3](#_Toc90305148)

[Parte web 4](#_Toc90305149)

[Sviluppo del codice 4](#_Toc90305150)

[Sicurezza 5](#_Toc90305151)

# Smart Street Lights

## Introduzione

Il progetto si basa sulla realizzazione di un sistema IoT il cui scopo è la gestione automatica di lampioni stradali che si accendono e si spengono in risposta a determinati eventi.

Il funzionamento del sistema consiste in due modalità: quella automatica che gestisce in modo autonomo i lampioni, ovvero accende e spegne quest’ultimi in base alla luce esterna rilevata e regola inoltre l’intensità della luce, quando accesa, che aumenta al passaggio di veicoli in prossimità dei lampioni grazie all’attivazione dei sensori a infrarossi; l’altra modalità è quella manuale che sfrutta una pagina web dove un addetto può pilotare i lampioni in tre modalità: automatica, accesa alla luminosità massima oppure spenta.–

## Obiettivi e contributi del gruppo

L’obiettivo principale di questo sistema è il risparmio energetico, in quanto riduce i consumi nelle ore in cui le strade sono meno trafficate, mantenendo però sempre una buona visibilità sulla strada. Le luci possono essere comandate singolarmente ed è possibile quindi aumentare la loro intensità in caso di necessità, ad esempio per lavori in corso, attraversamenti pedonali, emergenze ecc.

Il progetto è stato implementato da Leonardo Scandino e Marco Tateo per lo più in collaborazione, ma concentrandosi alle volte singolarmente su vari aspetti del progetto, ovvero il primo sulla parte automatica del sistema e le sue parti fisiche (raspberry, sensori ecc.), e il secondo sulla parte web e la comunicazione tra quest’ultima e la parte fisica.

## Materiali e strumenti utilizzati

* Raspberry pi 4

Immagine che contiene elettronico, circuito

Descrizione generata automaticamente

* Sensori
  + Due sensori a infrarossi IR

Immagine che contiene circuito, elettronico

Descrizione generata automaticamente

* + Fotoresistenza



* Due luci dimmerabili a 12v



* Alimentatore 10 Ampere 12V stabilizzato 220V 120W

Immagine che contiene testo

Descrizione generata automaticamente

* Breadboard, jumper e condensatore

Immagine che contiene testo

Descrizione generata automaticamenteImmagine che contiene forchetta

Descrizione generata automaticamente

* Due moduli mosfet IR520 per regolare le uscite PWM

Immagine che contiene testo

Descrizione generata automaticamente

* Flask

Immagine che contiene testo

Descrizione generata automaticamente

### Collegamenti dei componenti

Il cuore del sistema è un raspberry pi 4, un computer a scheda singola progettata per ospitare sistemi operativi basati sul kernel Linux, utilizzato nel nostro caso per gestire tutti gli elementi del progetto, alle varie GPIO disponibili sono stati collegati tutti i sensori e tutte le luci.

I sensori a infrarossi IR presentano tre pin ovvero, VCC (collegata alla GPIO 3.3v ), GND(collegato al GND) e il segnale OUT collegati rispettivamente alle due GPIO 22 e 23.

La fotoresistenza invece ha bisogno di un condensatore da 0.1 uf per calcolare il livello di luce e come per i precedenti ha bisogno di alimentazione e GND, il segnale passa per la GPIO 4.

La parte più complicata dei collegamenti sono le luci. Queste sono luci dimmerabili e devono essere alimentate a 12V quindi necessitano di un alimentatore, in questo caso di un alimentatore 10 ampere 12V stabilizzato 220V 120W e di un modulo mosfet IRF520 che permette di regolare le uscite PWM. Le due luci sono collegate ai due canali PWM presenti su Raspberry pi ovvero GPIO 13 e 18.

Per semplicità tutto quanto è stato collegato a una breadboard tramite dei jumper.

## Parte web

In questa fase, per sviluppare l’applicazione Python si è fatto uso di un Web framework Open Source di nome Flask. Abbiamo caricato inoltre le seguenti: Flask, render\_templates, request, redirect e session.

L’implementazione di questo framework consiste in poche righe di codice, ma che permettono di controllare le luci a proprio piacimento attraverso le richieste via POST provenienti dalla pagina Web.

La funzione utile di Flask è che riesce ad accedere ai vari templates e file attraverso il comando ***app= Flask(\_\_name\_\_)***, dove ***name***si riferisce al modulo python dove l’istanza ***app*** è stata chiamata.

Abbiamo posto la riga di codice ***App.route(‘ ’)*** al di sopra di alcune funzioni, la quale permette di eseguirle o meno in base al percorso URL selezionato. Inserendo degli specifici parametri è possibile quindi eseguire delle specifiche funzioni di codice, come ad esempio:

* Il parametro ***(‘/’)*** permette di eseguire la funzione ***index,*** la quale carica la pagina web iniziale di accesso.
* Il parametro ***(‘/login’, methods =[‘POST’,’GET’])*** permette di controllare i dati inseriti dall’utente, ovvero username e password, per verificare se c’è o meno una corrispondenza con i valori del dizionario ***user.*** Tutto questo è inserito in una definizione **login**.
* I parametri ***(‘/dashboard’)*** e ***(‘/logout’)*** eseguono rispettivamente le funzioni ***dashboard*** e ***logout,*** le quali inseriscono e tolgono dal dizionario **session** l’utente autenticato. All’uscita dell’utente dalla sessione, si viene infine riportati alla pagina di login iniziale.
* Il parametro ***(‘/dashboard’)*** permette di eseguire anche la funzione ***main***, la quale apre la pagina Web main.html e carica nel template ,di quest’ultima pagina, il dizionario delle luci;
* Infine, il parametro ***(‘/light’, methods = [‘POST’])***, che attiva la funzione più articolata del codice. Difatti, quando l’addetto seleziona le varie modalità dei lampioni (quindi: automatica, acceso o spento) tramite i pulsanti predisposti via Web, viene eseguita la funzione ***click,*** la quale attiverà la modalità desiderata sulle specifiche luci.

Tutti i parametri riportati sopra, fatta eccezione di /***dashboard***, fanno riferimento ad uno specifico valore di un parametro ***action*** di un form dei codici html.

## Sviluppo del codice

L’intero codice è suddiviso in tre file, ovvero mainApp.py che è il codice principale scritto in python, main.html e login.html i codici scritti invece in html per le pagine Web.

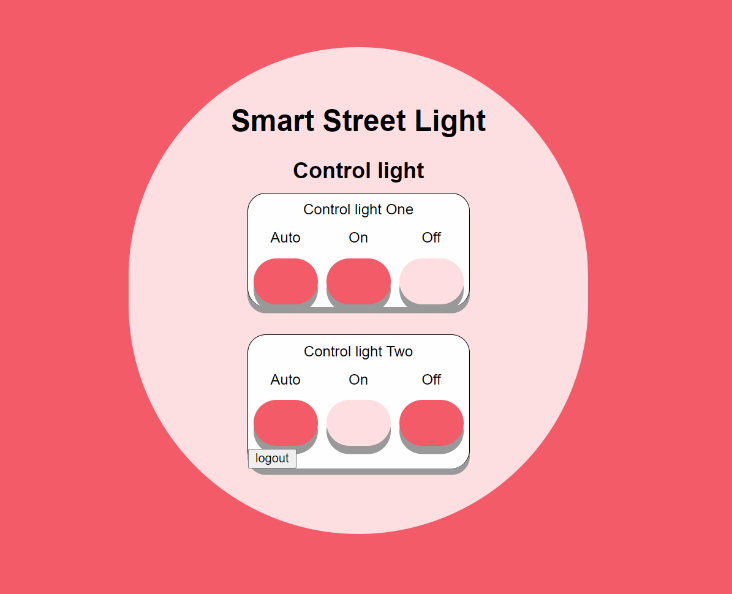
In mainApp.py vengono importate le librerie che serviranno nel corso del progetto ovvero RPi.GPIO, time, multiprocessing e Flask.

Per fare interagire la parte web e la parte automatica viene utilizzato un dizionario denominato lights dove sono inserite le informazioni più importanti ovvero: i pin delle luci, il pulsante selezionato nella pagina web, il pin del sensore a infrarossi, il pwm di riferimento e il processo.

A questo punto vengono settati tutti i sensori e tutte le luci tramite dei cicli e salvati nel dizionario.

Il funzionamento della parte automatica è essenzialmente sviluppato in due definizioni principali, ovvero dimmerLuce e valueFoto. La prima si occupa di leggere l’output di valueFoto, e se il valore è alto accende le luci al 40%; dal momento che le luci sono accese e un veicolo passa in prossimità di un sensore a infrarossi della luce di riferimento, ovvero quella che illumina la macchina e il suo percorso, viene aumentata l’intensità per qualche secondo al 100% per poi farla tornare al 40%. La seconda si occupa di leggere i dati prodotti dalla fotoresistenza e ricavarne un valore, alto se la luminosità esterna è bassa, basso se c’è abbastanza luce.

In seguito, viene la parte di codice che si interfaccia anche alla parte web descritta in precedenza.

Come prima cosa viene passato il dizionario **lights** alla nostra pagina web che farà visualizzare lo stato iniziale, a questo punto viene inserita una definizione **click** (già citata sopra) che cambia lo stato del nostro dizionario e lo aggiorna.

Se si clicca su Auto viene modificato nel dizionario il valore del pulsante in AUTO; viene spenta qualsiasi luce, per poi essere accesa o meno in base al processo, descritto sopra, che sfrutta le definizioni **dimmerLuce** e **valueFoto** (inserendo nel dizionario la luce di riferimento che ha quel processo in esecuzione).

Se si clicca su OFF viene modificato, sempre nel dizionario, il valore del pulsante in OFF, il codice verifica se il processo è attivo e in caso affermativo lo termina. Il funzionamento di ON è il medesimo, a differenza del fatto che viene accesa la luce al 100%.

Si riesce a tenere traccia in modo efficiente dello stato del sistema grazie al fatto che ad ogni click il dizionario viene aggiornato.

Le pagine login.html e main.html sono invece delle semplici pagine Web, la prima visualizza una delle due parti della sicurezza che verranno spiegate successivamente. La seconda (foto soprariportata) presenta una pagina contenente dei riquadri in cui sono posti i tre pulsanti (auto, on, off) per ciascuna delle due luci. È posto anche un piccolo pulsante **logout** la cui funzione è quella di fare uscire l’utente dalla sessione di lavoro, mantenendo però attive le modalità attuali delle luci.

Sicurezza

Per la parte di sicurezza abbiamo riportato una ricerca per un accesso sicuro al Raspberry pi e implementato invece un metodo per l’accesso all’applicazione Web:

* Il primo è un’autenticazione SSH (Secure Socket Shell) a due fattori su Raspberry pi

Questo procedimento richiede l’immissione di due informazioni per l’accesso, in quanto dovrà essere inserita, oltre agli standard username e password, un codice generato da Google Authenticator.

Google Authenticator è un servizio di generazione di token realizzato da Google distribuito come applicazione mobile.

Nella fase di setup di quest’ultimo vengono forniti un QRcode, una secret key e cinque codici di emergenza in caso di smarrimento del dispositivo associato. Una volta scannerizzato il QRcode, che abilita Google Authenticator sul dispositivo, bisogna includere questa modalità di autenticazione al file di configurazione pam.d/sshd del Raspberry, inserendo la stringa “auth required pam\_google\_authenticator.so”. Dopo aver riavviato Raspberry pi al momento della connessione in ssh viene quindi richiesto anche il codice ricevuto sul dispositivo nell’applicazione Google Authenticator.

In questo modo si intensifica la sicurezza per l’accesso in remoto tramite ssh al Raspberry e quindi proteggere il codice da eventuali attacchi.

* Il secondo, che a differenza del primo è stato implementato, riguarda l’autenticazione per l’accesso alla parte web dove si possono gestire le luci. Questa parte è stata realizzata con Flask. Dopo aver importato da flask le librerie necessarie (session e redirect), si crea un dizionario user dove viene inserito username e password dell’utente abilitato.

Tramite la pagina login.html (già citata) si richiede l’inserimento delle credenziali per l’accesso.

I dati inseriti sono gestiti nel codice in python, dove vengono confrontati con il dizionario user (definizione **login)**.

Se i dati coincidono si viene indirizzati sulla pagina principale main.html per la gestione delle luci, se al contrario sono errati viene mostrato un messaggio di errore: “username o password errati”, in questo modo si blocca l’accesso ad utenti non autorizzati. Per completare il login è posto anche un piccolo pulsante **logout** in main.html, la cui funzione è quella di fare uscire l’utente dalla sessione di lavoro, mantenendo però attive le modalità attuali delle luci.

Immagine che contiene testo, clipart

Descrizione generata automaticamente