SLN – Smart Light Network

Fabrizio Cara

Nicola Benenati

Elena Marongiu

Corso di Laurea Magistrale in Ingegneria delle Telecomunicazioni

RETI RADIOMOBILI - A.A. 2017/18

Sommario

[Lista dei componenti 3](#_Toc507774235)

[Introduzione 4](#_Toc507774236)

[Telegram 5](#_Toc507774237)

[ThingSpeak 7](#_Toc507774238)

[Realizzazione Smart Light System 9](#_Toc507774239)

[Assemblaggio Hardware 10](#_Toc507774240)

[Funzionamento sistema 13](#_Toc507774241)

[Sketch completo del codice 16](#_Toc507774242)

[Appendice: Immagini 24](#_Toc507774243)

# Lista dei componenti

* 30 diodi led bianchi;
* 30 resistenze da 100 Ω;
* 3 resistenze da 10 KΩ;
* 3 sensori PIR;
* 3 mosfet IRF540;
* 1 fotoresistenza;
* 1 NodeMCU;
* 3 BreadBoard;
* 3 Convertitori di livello logico doppio 3.3V;
* Cavi elettrici per alimentazione e collegamenti;
* Nastro Isolante;
* Alimentatore USB;

# 

# Introduzione

In questo report verrà presentato il progetto per la realizzazione di un sistema smart per l’illuminazione urbana. Il fine è quello di realizzare un sistema che permetta di evitare l’accensione costante delle luci al 100% della potenza anche quando non sono presenti persone e veicoli nella strada, in modo tale da avere un netto risparmio sui costi dell’illuminazione pubblica.

Verranno proposti due algoritmi che, in base alla presenza di persone o veicoli nella strada, permettano la variazione dell’intensità luminosa dei vari apparati di illuminazione, col fine di avere dei consumi di potenza ottimizzati. In entrambi gli algoritmi le luci avranno un’intensità diversa a seconda dell’intensità luminosa presente nell’ambiente in cui il sistema verrà installato. Infatti, nei momenti della giornata in cui si avrà una luminosità ambientale maggiore (pomeriggio), l’intensità luminosa degli apparati di illuminazione sarà minore rispetto all’intensità luminosa che si avrebbe in altri momenti della giornata dove è presente invece una luminosità ambientale minore (sera e notte).

L’accensione delle singole unità di illuminazione sarà supportata dalla rilevazione della presenza, in modo tale che al passaggio di una persona o di un veicolo le luci non si accendano tutte con la stessa intensità luminosa ma varino in funzione della rilevazione della presenza. In linea con ciò si sono sviluppati due modalità di funzionamento:

1. Bassa luminosità dell’ambiente: lampade in prossimità della quale si è rilevata una presenza con un’intensità pari al 100%, mentre le altre con un’intensità dell’80%;
2. Alta luminosità dell’ambiente: lampade in prossimità della quale si è rilevata una presenza con un’intensità pari all’80%, mentre le altre con un’intensità del 50%;

Il sistema sarà poi in grado di trasmettere le informazioni di consumo istantaneo e consumo medio su una piattaforma di IoT (ThingSpeak), dando quindi la possibilità di monitorare le prestazioni del sistema e di confrontare tali prestazioni con quelle di consumo massimo, ossia il caso in cui le singole unità di illuminazione sono costantemente accese al 100% della potenza. Questo è estremamente importante per valutare l’effettivo risparmio energetico. Per questo motivo nell’interfaccia sarà possibile visualizzare l’energia percentuale risparmiata istante per istante, calcolata come differenza tra il consumo massimo e il consumo istantaneo effettivo.

Infine un’ulteriore caratteristica sarà quella di poter utilizzare Telegram (piattaforma di messaggistica molto conosciuta), permettendo al gestore dell’illuminazione di controllare tramite lo scambio di messaggi, l’accensione e lo spegnimento delle luci e di recuperare le informazioni riguardanti i consumi di potenza istantanei dell’apparato di illuminazione.

## Telegram

In questa sezione del report verrà descritto come utilizzare una delle app di messaggistica più famose al mondo come piattaforma di internet of things (IoT).

Telegram è un servizio open source, disponibile per qualsiasi dispositivo mobile e fisso ([telegram.org](https://telegram.org/)), che prevede tra le tante caratteristiche la possibilità di creare dei BOT (entità software) alla quale si può associare un dispositivo hardware.

Questo permette di comunicare con il dispositivo usando dei semplici messaggi, dando la possibilità all’utente di pilotare il sistema o richiedere dati. Tale caratteristica può essere realizzata integrando nel codice una sezione che permetta di accedere tramite un qualsiasi account ai server di Telegram, col quale sarà possibile inviare o ricevere messaggi dal dispositivo nella quale verrà inserito il codice.

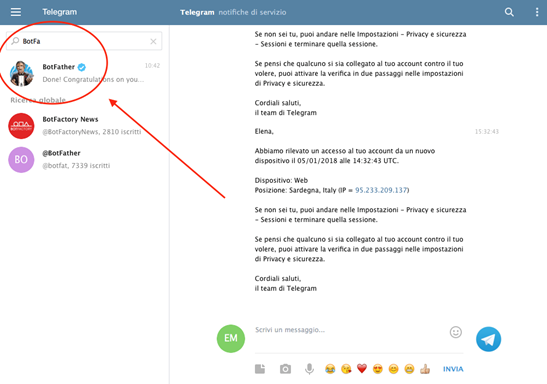
I principali vantaggi nell’utilizzare Telegram sono:

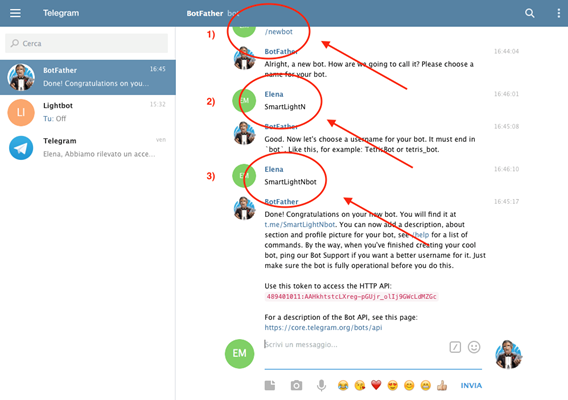
* vasta documentazione facilmente accessibile;
* librerie già implementate in diversi linguaggi di programmazione che permettono l’interfacciamento con Telegram;
* semplicità di connessione col bot tramite rete internet (infatti è necessaria la sola conoscenza delle credenziali);
* completamente implementata e supportata per qualsiasi dispositivo;
* pensata appositamente anche per applicazioni IoT.

Per poter creare un bot Telegram, il primo passo è quello di installare Telegram su un qualunque dispositivo e creare un account, successivamente bisogna creare il bot. Per poterlo fare bisogna cercare tramite la barra di ricerca l’utente “BotFather” (che è un bot a sua volta), questo permette l’istanziazione di un account di tipo bot.

Una volta avviato il BotFather con il messaggio “/start” si riceveranno una serie di comandi che permetteranno la creazione o la modifica del bot. Inviando il comando “/newbot” verrà creato un nuovo bot a cui succederà la richiesta del nome e dello username del bot da parte del BotFather. Infine, dopo questi tre passaggi il BotFather creerà il Token di accesso HTTP API, che dovrà poi essere utilizzato insieme al name e allo username per poter accedere dal dispositivo al bot Telegram e poter inviare e ricevere messaggi.

Nelle figure successive verrà mostrato un esempio di creazione di un bot Telegram seguendo i passaggi appena descritti.





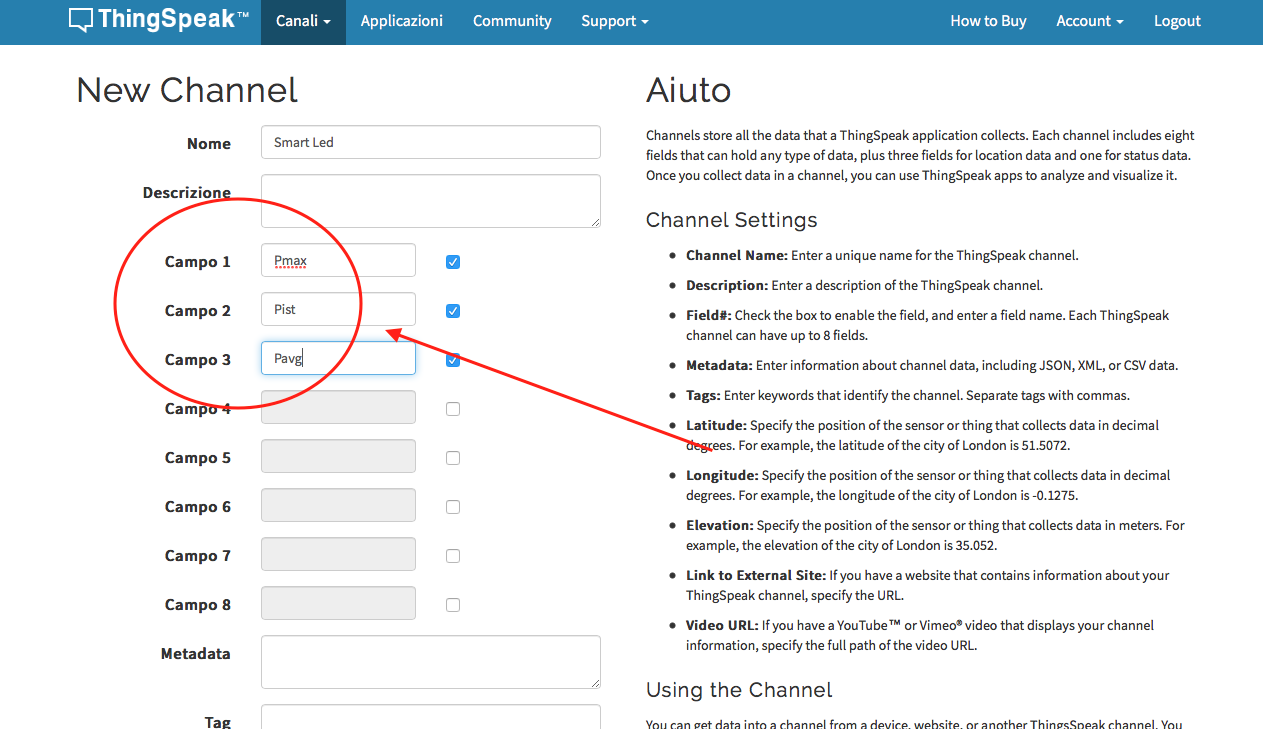
## 

## ThingSpeak

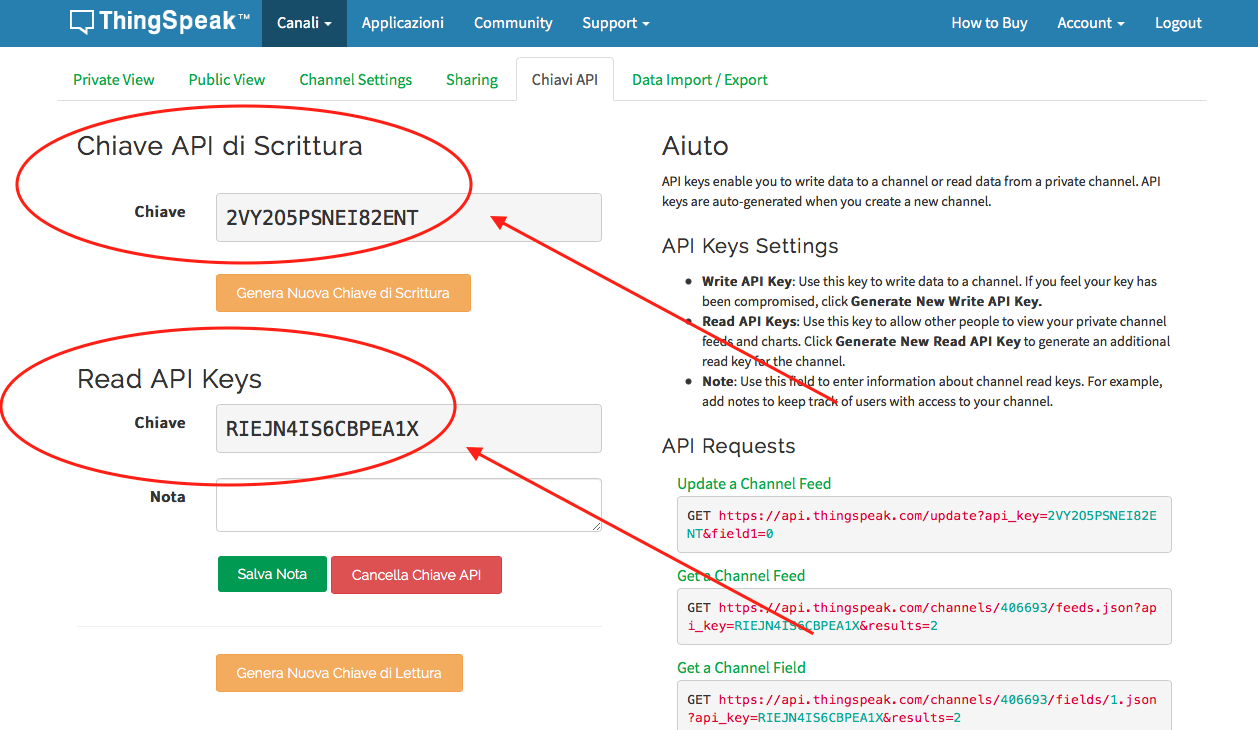
ThingSpeak è una piattaforma molto famosa e utilizzata di Internet of Things che permette il salvataggio e l’interfacciamento di dati ricevuti da dei dispositivi fisici che li rilevano e li inviano alla piattaforma tramite apposite API.

Per poter usufruire dei servizi di tale piattaforma, bisogna per prima cosa creare un account tramite il seguente [link](https://thingspeak.com/users/sign_up).

Dopo aver creato un account e aver effettuato l’accesso, bisogna creare un canale, per fare ciò si deve cliccare su “new Channel”. Dopo aver inserito il nome e aver impostato i vari parametri di interesse, bisogna inserire i vari campi che corrispondono ai grafici che saranno poi presenti nel canale. Nel nostro caso andremo ad inserire i valori Pmax (Potenza massima), Pist (Potenza istantanea) e Pavg (Consumo medio) rispettivamente nel Campo 1, 2 e 3. Dopodiché si deve salvare il canale cliccando su “Save channel” situato alla fine della pagina in basso.



Una volta creato il canale bisogna recuperare le chiavi di scrittura e lettura che serviranno al nostro dispositivo per la connessione al canale e il successivo invio (o lettura) dei dati, per fare ciò si deve cliccare su “Chiavi Api”, dove sarà possibile trovare le due Chiavi API di scrittura e lettura.



# Realizzazione Smart Light System

Per realizzare un prototipo del sistema di illuminazione smart si è optato per dei led bianchi che ci consentono di avere una lampada con un’intensità abbastanza forte da vedere le differenze tra le varie intensità luminose.

La presenza viene invece rilevata con dei sensori PIR, mentre la luminosità viene rilevata con una fotoresistenza direttamente collegata al controllore che in questo progetto è il NodeMCU. Tale controllore è particolarmente adatto al nostro scopo perché oltre ad avere svariati pin digitali e un pin analogico per la fotoresistenza, ha anche già integrata una scheda Wifi di tipo ESP8266 ESP12. Poichè tale controllore ha solo l’uscita a 3.3V, sia come VCC che nei pin digitali, per poterlo usare con la stragrande maggioranza dei dispositivi in commercio si rende indispensabile l’uso di un convertitore logico a doppio livello 3.3V-5V.

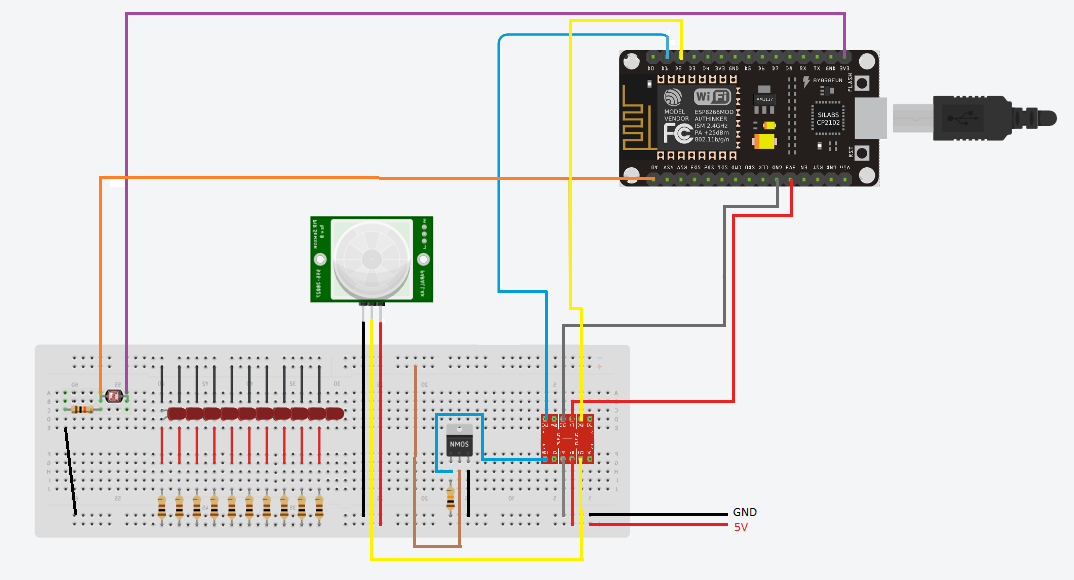
Per la rilevazione della potenza istantanea assorbita dal sistema si è ricorsi all’uso di un multimetro, con il quale si è misurata la potenza che le singole lampade (create con i led) assorbono quando sono al 50%, all’80% e al 100% dell’intensità luminosa. Una volta conosciute le potenze per le singole intensità, è possibile risalire alla potenza massima assorbita dal sistema andando a vedere quali sono i sensori PIR che rilevano una presenza. Infatti le lampade dove si rileva una presenza sono quelle che si devono accendere al massimo mentre le altre invece staranno al minimo, dove minimo e massimo della potenza dipendono dalla luminosità ambientale.

Per quanto riguarda l’invio dei dati alla piattaforma ThingSpeak si ha il vincolo, imposto dalla piattaforma stessa, di massimo un invio ogni 15 secondi che è esattamente il rate che abbiamo scelto di usare per l’invio dei dati.

Infine, per quanto riguarda Telegram, si avrà la possibilità di spegnere o di accendere tutte le luci e di recuperare informazioni quali potenza istantanea, potenza media, potenza massima e energia percentuale risparmiata tutto espresso rispetto ai mW.

Questa funzionalità è stata pensata per un uso privato del sistema in ambienti sia indoor che outdoor, più che per l’illuminazione pubblica.

# Assemblaggio Hardware

****

Nell’immagine precedente è riportata la rappresentazione del sistema con una sola lampada. I componenti usati per la realizzazione del sistema sono i seguenti:

* Scheda NodeMCU: usata per il controllo della luminosità delle lampade, la ricezione dei segnali di luminosità, la rilevazione delle presenze e l’invio dei dati in rete.
* MOSFET IRF540: usato per far variare l’intensità luminosa della lampada.
* Led per l'illuminazione.
* Sensore PIR: per la rilevazione di presenza di persone o veicoli.
* Convertitore logico 3.3V-5V a due canali
* Resistenze: necessarie per non rischiare di rompere i componenti a causa di voltaggi troppo elevati per quest'ultimi.

I colori dei cavi sono stati raggruppati per utilizzo:

* ROSSO usato per l’alimentazione;
* NERO usato per la massa;
* AZZURRO usato per collegare il gate del mosfet col NodeMCU;
* MARRONE usato per collegare il drain del mosfet col catodo del LED;
* VERDE usato per collegare il Gate col source del mosfet;
* GIALLO usato per collegare l’out del PIR al NodeMCU;

**Collegamento convertitore logico con NodeMCU.**

Questo serve per convertire tutti i collegamenti che partono dal NodeMCU verso i vari sensori dalla tensione di 3.3V a 5V.

Per fare ciò verrà collegato il PIN 3.3V del NodeMCU al PIN LV del convertitore logico, mediante cavo rosso, mentre il PIN GND del NodeMCU va collegato al PIN GND del convertitore, mediante cavo nero. Successivamente collegare il PIN HV del convertitore logico alla linea di alimentazione esterna a 5V e il secondo PIN GND alla linea GND, come da figura.

**Collegamento con i LED.**

Per prima cosa prendiamo l’alimentazione(esterna), di 5V, e la colleghiamo alla linea orizzontale della breadboard, in corrispondenza del ‘ + ’.

Per fornire la stessa tensione a tutti i led, questi vanno collegati in parallelo. Per collegare i 10 led in parallelo questi dovranno essere tutti collegati a due capi comuni, per fare ciò colleghiamo uno dei capi delle resistenze da 100Ω tutti in corrispondenza della linea d’alimentazione in modo tale che abbiano tutti il capo in comune.

All’altro capo di ogni singolo resistore, da 100Ω, colleghiamo l’anodo (il terminale più lungo) dei led, mediante cavetti rossi, come da figura.

Infine il catodo (terminale più corto) dei led va collegato alla seconda linea orizzontale della breadboard, in corrispondenza del ‘ – ‘mediante cavetti neri come da figura, in questo modo anche l’altro capo sarà in comune a tutti i led, per cui essi sono tutti in parallelo.



**Collegamento NodeMCU con il MOSFET.**

Il collegamento non può essere fatto direttamente perché il Mosfet ha bisogno di 5V in ingresso, per cui il gate del Mosfet è stato collegato prima al PIN TX0 del CHAN2 del convertitore logico, mediante il filo azzurro, e poi dal PIN TX1 del CHAN2 del convertitore logico al PIN D1 del NodeMCU, sempre con cavo azzurro, come da figura.

Il terminale source del Mosfet va collegato invece alla linea di GND, mediante cavi NERI come da figura.

Bisogna fare attenzione al fatto che tra il gate e il source bisogna inserire una resistenza di 10KΩ, collegando un pin della resistenza alla linea di GND e l’altro pin al pin gate.

**Collegamento MOSFET con i LED.**

Il pin DRAIN è stato collegato alla linea orizzontale della breadboard, mediante cavo marrone, insieme al catodo dei led in modo tale da chiudere il circuito tra alimentazione e massa.

**Collegamento sensore PIR con il NodeMCU.**

Prima di tutto il sensore PIR va alimentato, per alimentarlo bisogna collegare il pin VCC del PIR all’alimentazione e il pin GND del PIR alla linea dedicata per la massa, come da figura.

Mentre il pin OUT del PIR verrà collegato al PIN D2 del NodeMCU mediante il cavo GIALLO.

**Collegamento Fotoresistenza con il NodeMCU.**

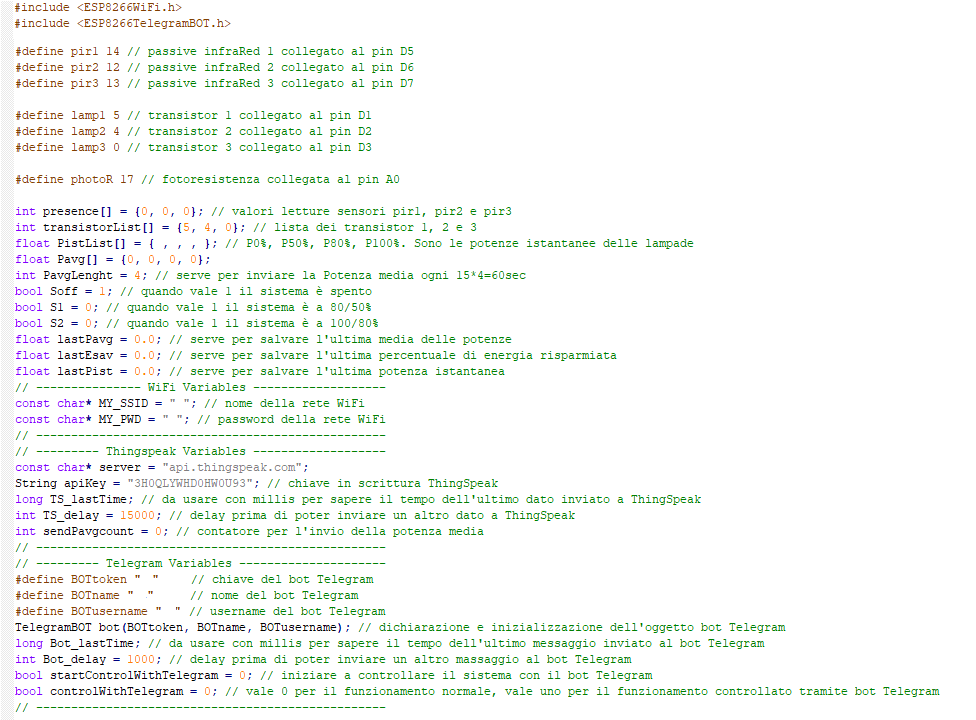
Posizionare la fotoresistenza sulla board e la resistenza da 10KΩ, come da figura, collegare un pin della resistenza alla GND. Far partire un cavo dal pin analogico A0 del NodeMCU e collegarlo a resistenza e fotoresistenza come da figura, infine collegare il pin libero della fotoresistenza all’alimentazione ricavata, in questo caso, da un pin a 3,3V del NodeMCU stesso.

Lo schema finale sarà composto, in totale, da 3 lampade uguali composte da:

* convertitore logico a due canali;
* Mosfet;
* 10 LED con rispettive resistenze;
* Sensore PIR.

Tutte e tre le lampade saranno collegate al NodeMCU, unico per tutto il sistema, come la fotoresistenza.

# Funzionamento sistema

****

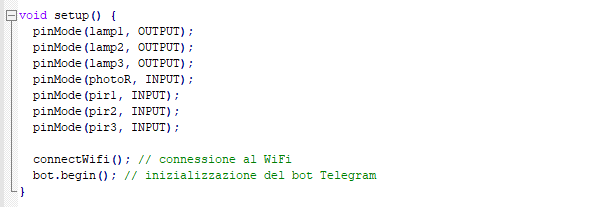
Come mostrato nell’immagine precedente, nelle prime due righe del codice si può vedere che per realizzare tale sistema si ha bisogno di due librerie:

* ESP8266Wifi serve per usare il modulo WiFi per accedere alla rete internet. Tale modulo verrà usato per inviare dati alla piattaforma ThingSpeak e per ricevere ed inviare messaggi al bot Telegram;
* ESP8266TelegramBot serve per comunicare con il bot Telegram.

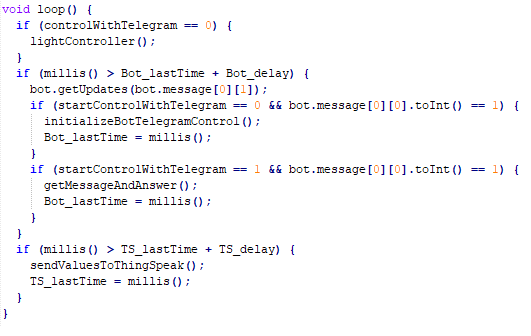
Dopo le librerie, si possono vedere tutte le variabili usate per la realizzazione del sistema con relative descrizioni, partendo da quelle generiche, fino a quelle usate per Telegram e Thingspeak.

Tra le tante variabili usate bisogna prestare particolare attenzione a:

* vettore Pavg: bisogna inserire al suo interno le misurazioni (fatte con il multimetro) della potenza assorbita da una lampada quando questa è al 50%, 80% e 100% della sua intensità:
* MY\_SSID e MY\_PWD: bisogna inserire lo username e la password della propria rete WiFi per poter accedere ed usufruire della connessione alla rete internet;
* BOTtoken, BOTname e BOTusername: bisogna inserire tali parametri presi dal proprio bot Telegram in fase di creazione del bot.



Dopo aver visto le variabili usate dal sistema, nell’immagine immediatamente precedente si riporta il setup. Qua si può vedere l’inizializzazione di tutti i dispositivi usati, partendo dai transistor (indicati come lamp1, lamp2 e lamp3) che permettono di impostare le diverse intensità luminose nei tre apparati di illuminazione, continuando con la fotoresistenza che permette di rilevare il livello di luminosità ambientale, finendo con i PIR (indicati come pir1, pir2, pir3) che permettono invece di rilevare le presenze. Infine si può vedere l’inizializzazione della connessione al WiFi e del bot telegram.



La parte restante per quanto riguarda il codice è il funzionamento effettivo del sistema descritto nel loop.

Il loop è strutturato secondo tre blocchi condizionali if:

* Primo: controlla se il flag “controlWithTelegram” è nullo. Se questo è nullo si richiama la funzione “lightController()” che permette di impostare il funzionamento delle luci a seconda della luminosità dell’ambiente e della rilevazione di una presenza. Le modalità di funzionamento sono: luci spente se vi è alta luminosità; luci dove viene rilevata una presenza all’80% della luminosità e le altre al 50% se vi è media luminosità; luci dove viene rilevata una presenza al 100% della luminosità e le altre all’80% se vi è bassa luminosità.
* Secondo: questo blocco implementa le funzionalità di Telegram. La condizione iniziale è verificata quando passa un secondo dall’ultimo messaggio, in seguito si aggiorna la lista dei messaggi in ingresso, infine sono presenti altri due blocchi condizionali:
  + Primo: in tale blocco si richiama la funzione “initializeBotTelegramControl()” che permette di controllare il sistema e ricevere i comandi disponibili per interagire con il dispositivo. Questo è possibile solo se viene ricevuto il messaggio START come primo messaggio che funziona quindi come una password. Una volta ricevuto tale messaggio tramite un’opportuna gestione dei flag si attiva l’ascolto dei messaggi in ricezione.
  + Secondo: se si riceve il messaggio START si può entrare in questo blocco mentre non si entra più nel precedente finchè non si riceve un EXIT. In questa parte si richiama la funzione “getMessageAndAnswer()” che permette di rispondere alle varie richieste inviate tramite bot Telegram. In questa sezione è inoltre possibile accendere e spegnere il sistema, una volta spento il sistema il flag “controlWithTelegram” viene messo ad 1 e le luci vengono spente impedendo al sistema di riaccendersi.
* Terzo: questo blocco implementa le comunicazioni con la piattaforma IoT ThingSpeak. Ogni 15 secondi viene inviata la potenza massima, la potenza istantanea e l’energia percentuale risparmiata; mentre ogni minuto viene inviata la potenza media.

# Sketch completo del codice

#include <ESP8266WiFi.h>

#include <ESP8266TelegramBOT.h>

#include <WiFiClientSecure.h>

#define pir1 14 // passive infraRed 1 collegato al pin D5

#define pir2 12 // passive infraRed 2 collegato al pin D6

#define pir3 13 // passive infraRed 3 collegato al pin D7

#define lamp1 5 // transistor 1 collegato al pin D1

#define lamp2 4 // transistor 2 collegato al pin D2

#define lamp3 0 // transistor 3 collegato al pin D3

#define photoR 17 // fotoresistenza collegata al pin A0

int presence**[]** **=** **{**0**,** 0**,** 0**};** // valori letture sensori pir1, pir2 e pir3

int transistorList**[]** **=** **{**5**,** 4**,** 0**};** // lista dei transistor 1, 2 e 3

float PistList**[]** **=** **{** **,** **,** **,** **};** // P0%, P50%, P80%, P100% mW. Sono le potenze istantanee delle lampade // <----- Da modificare

float Pavg**[]** **=** **{**0**,** 0**,** 0**,** 0**};**

int PavgLenght **=** 4**;** // serve per inviare la Potenza media ogni 15\*4=60sec

bool Soff **=** 1**;** // quando vale 1 il sistema è spento

bool S1 **=** 0**;** // quando vale 1 il sistema è a 80/50%

bool S2 **=** 0**;** // quando vale 1 il sistema è a 100/80%

float lastPavg **=** 0.0**;** // serve per salvare l'ultima media delle potenze

float lastEsav **=** 0.0**;** // serve per salvare l'ultima percentuale di energia risparmiata

float lastPist **=** 0.0**;** // serve per salvare l'ultima potenza istantanea

// --------------- WiFi Variables -------------------

const char**\*** MY\_SSID **=** " "**;** // nome della rete WiFi

const char**\*** MY\_PWD **=** " "**;** // password della rete WiFi

// --------------------------------------------------

// --------- Thingspeak Variables -------------------

const char**\*** server **=** "api.thingspeak.com"**;**

String apiKey **=** " "**;** // chiave in scrittura ThingSpeak

long TS\_lastTime**;** // da usare con millis per sapere il tempo dell'ultimo dato inviato a ThingSpeak

int TS\_delay **=** 15000**;** // delay prima di poter inviare un altro dato a ThingSpeak

int sendPavgcount **=** 0**;** // contatore per l'invio della potenza media // <----- Da modificare (anche a riga 274)

// --------------------------------------------------

// --------- Telegram Variables ---------------------

#define BOTtoken " " // chiave del bot Telegram

#define BOTname " " // nome del bot Telegram

#define BOTusername " " // username del bot Telegram

TelegramBOT bot**(**BOTtoken**,** BOTname**,** BOTusername**);** // dichiarazione e inizializzazione dell'oggetto bot Telegram

long Bot\_lastTime**;** // da usare con millis per sapere il tempo dell'ultimo messaggio inviato al bot Telegram

int Bot\_delay **=** 1000**;** // delay prima di poter inviare un altro massaggio al bot Telegram

bool startControlWithTelegram **=** 0**;** // iniziare a controllare il sistema con il bot Telegram

bool controlWithTelegram **=** 0**;** // vale 0 per il funzionamento normale, vale uno per il funzionamento controllato tramite bot Telegram

// --------------------------------------------------

void setup**()** **{**

pinMode**(**lamp1**,** OUTPUT**);**

pinMode**(**lamp2**,** OUTPUT**);**

pinMode**(**lamp3**,** OUTPUT**);**

pinMode**(**photoR**,** INPUT**);**

pinMode**(**pir1**,** INPUT**);**

pinMode**(**pir2**,** INPUT**);**

pinMode**(**pir3**,** INPUT**);**

connectWifi**();** // connessione al WiFi

bot**.**begin**();** // inizializzazione del bot Telegram

**}**

void loop**()** **{**

**if** **(**controlWithTelegram **==** 0**)** **{** // funzionamento con rilevazione di luminosità e presenza

lightController**();**

**}**

**if** **(**millis**()** **>** Bot\_lastTime **+** Bot\_delay**)** **{** // in questa sezione si risponde ai messaggi in ingresso, prima di iniziare bisogna inserire la parola START!

bot**.**getUpdates**(**bot**.**message**[**0**][**1**]);**

**if** **(**startControlWithTelegram **==** 0 **&&** bot**.**message**[**0**][**0**].**toInt**()** **==** 1**)** **{**

initializeBotTelegramControl**();**

Bot\_lastTime **=** millis**();**

**}**

**if** **(**startControlWithTelegram **==** 1 **&&** bot**.**message**[**0**][**0**].**toInt**()** **==** 1**)** **{**

getMessageAndAnswer**();**

Bot\_lastTime **=** millis**();**

**}**

**}**

**if** **(**millis**()** **>** TS\_lastTime **+** TS\_delay**)** **{** // invio dati alla piattaforma ThingSpeak

sendValuesToThingSpeak**();**

TS\_lastTime **=** millis**();**

**}**

**}**

//---------------------------------- Funzioni -----------------------------------------

// permette di controllare il funzionamento delle luci tramite fotoresistenza nelle seguenti modalità:

// - valore fotoresistenza High: non viene accesa nessuna luce

// - valore fotoresistenza Medium: viene accesa la luce in cui c'è una presenza all'80%, mentre le altre al 50%

// - valore fotoresistenza Low: viene accesa la luce in cui c'è una presenza al 100%, mentre le altre all'80%

void lightController**()** **{**

String brightnessValue **=** detectBrightness**();**

**if** **(**brightnessValue **==** "High"**)** **{**

**for** **(**int i **=** 0**;** i **<** 3**;** i**++)** **{**

analogWrite**(**transistorList**[**i**],** 0**);**

**}**

Soff **=** 1**;** // il sistema è spento

S1 **=** 0**;**

S2 **=** 0**;**

**return;**

**}**

**else** **if** **(**brightnessValue **==** "Medium"**)** **{**

detectPresence**();** // aggiorna il vettore delle presenze

**if** **(**oneIsInArray**()** **==** 1**)** **{**

**for** **(**int i **=** 0**;** i **<** 3**;** i**++)** **{**

**if** **(**presence**[**i**]** **==** 1**)** **{**

analogWrite**(**transistorList**[**i**],** 205**);** //80%

**}**

**else** **{**

analogWrite**(**transistorList**[**i**],** 128**);** //50%

**}**

**}**

Soff **=** 0**;**

S1 **=** 1**;** // il sistema è nello stato 1

S2 **=** 0**;**

**return;**

**}**

**else** **{**

**for** **(**int i **=** 0**;** i **<** 3**;** i**++)** **{**

analogWrite**(**transistorList**[**i**],** 128**);** //50%

**}**

Soff **=** 0**;**

S1 **=** 1**;** // il sistema è nello stato 1

S2 **=** 0**;**

**return;**

**}**

**}**

**else** **if** **(**brightnessValue **==** "Low"**)** **{**

detectPresence**();** // aggiorna il vettore delle presenze

**if** **(**oneIsInArray**()** **==** 1**)** **{** // accendo le luci in cui c'è una presenza al 100%, mentre le altre all'80%

**for** **(**int i **=** 0**;** i **<** 3**;** i**++)** **{**

**if** **(**presence**[**i**]** **==** 1**)** **{**

analogWrite**(**transistorList**[**i**],** 255**);** //100%

**}**

**else** **{**

analogWrite**(**transistorList**[**i**],** 204**);** //80%

**}**

**}**

Soff **=** 0**;**

S1 **=** 0**;**

S2 **=** 1**;** // il sistema è nello stato 2

**return;**

**}**

**else** **{** // spengo tutte le luci

**for** **(**int i **=** 0**;** i **<** 3**;** i**++)** **{**

analogWrite**(**transistorList**[**i**],** 204**);** //80%

**}**

Soff **=** 0**;**

S1 **=** 0**;**

S2 **=** 1**;** // il sistema è nello stato 2

**return;**

**}**

**}**

**}**

// permette di controllare il funzionamento delle luci tramite comando via Telegram

// I comandi disponibili sono:

// ON, OFF per accendere o spegnere le luci

// PMAX, PIST, PAVG, ESAV per conoscere quali sono le varie potenze e l'energia percentuale risparmiata

String lightControllerWithInput**(**String comand**)** **{**

**if** **(**comand **==** "ON"**)** **{**

controlWithTelegram **=** 0**;**

**return** "1"**;**

**}**

**else** **if** **(**comand **==** "OFF"**)** **{**

**for** **(**int i **=** 0**;** i **<** 3**;** i**++)** **{**

analogWrite**(**transistorList**[**i**],** 0**);**

**}**

controlWithTelegram **=** 1**;**

Soff **=** 1**;** // il sistema è spento

S1 **=** 0**;**

S2 **=** 0**;**

**return** "1"**;**

**}**

**else** **if** **(**comand **==** "PMAX"**)** **{**

**return** "La potenza massima vale: " **+** **(**String**)(**PistList**[**3**]** **\*** 3**)** **+** "mW"**;**

**}**

**else** **if** **(**comand **==** "PIST"**)** **{**

**if** **(**Soff **==** 1 **&&** S1 **==** 0 **&&** S2 **==** 0**)** **{**

**return** "La potenza istantanea vale: " **+** **(**String**)**PistList**[**0**]** **+** "mW"**;**

**}**

**else** **if** **(**Soff **==** 0 **&&** S1 **==** 1 **&&** S2 **==** 0 **|** Soff **==** 0 **&&** S1 **==** 0 **&&** S2 **==** 1**)** **{**

**return** "La potenza istantanea vale: " **+** **(**String**)**lastPist **+** "mW"**;**

**}**

**}**

**else** **if** **(**comand **==** "ESAV"**)** **{**

**return** "L'energia risparmiata istantanea vale: " **+** **(**String**)**lastEsav **+** "%"**;**

**}**

**else** **if** **(**comand **==** "EXIT"**)** **{**

controlWithTelegram **=** 0**;**

startControlWithTelegram **=** 0**;**

**return** "1"**;**

**}**

**else** **{**

**return** "0"**;**

**}**

**}**

// riempie l'array delle presenze con 0 (nessuna presenza) o 1 (presenza rilevata) per ogni pir del sistema(3 pir)

void detectPresence**()** **{**

presence**[**0**]** **=** digitalRead**(**pir1**);**

presence**[**1**]** **=** digitalRead**(**pir2**);**

presence**[**2**]** **=** digitalRead**(**pir3**);**

**}**

// rende 1 se nell'array passato è presente almeno un 1, altrimenti rende 0

bool oneIsInArray**()** **{**

**for** **(**int i **=** 0**;** i **<** 3**;** i**++)** **{**

**if** **(**presence**[**i**]** **==** 1**)** **{**

**return** 1**;**

**}**

**}**

**return** 0**;**

**}**

// rende una stringa con le soglie di luminosità: High, Medium, Low

// bisogna testare la luminosità dell'ambiente per decidere le due soglie!!

String detectBrightness**()** **{**

int brightness **=** analogRead**(**photoR**);**

**if** **(**brightness **<** 700**)** **{** // <--------- Soglie da modificare

**return** "Low"**;**

**}**

**else** **if** **(**brightness **>** 700 **&&** brightness **<** 850**)** **{** // <--------- Soglie da modificare

**return** "Medium"**;**

**}**

**else** **if** **(**brightness **>** 850**)** **{** // <--------- Soglie da modificare

**return** "High"**;**

**}**

**}**

// connessione al WiFi

void connectWifi**()** **{**

WiFi**.**begin**(**MY\_SSID**,** MY\_PWD**);**

**while** **(**WiFi**.**status**()** **!=** WL\_CONNECTED**)** **{**

delay**(**1000**);**

**}**

**}**

// invia i dati sulla potenza alla piattaforma ThingSpeak

void sendValuesToThingSpeak**()** **{**

WiFiClient client**;**

**if** **(**client**.**connect**(**server**,** 80**))** **{**

String postStr **=** apiKey**;**

postStr **+=** "&field1="**;**

postStr **+=** String**(**PistList**[**3**]** **\*** 3**);** // potenza massima

postStr **+=** "&field2="**;**

**if** **(**Soff **==** 1 **&&** S1 **==** 0 **&&** S2 **==** 0**)** **{**

SOffPistList**();**

postStr **+=** String**(**lastPist**);** // potenza istantanea Stato 0

lastEsav **=** computeEsaved**();**

postStr **+=** "&field3="**;**

postStr **+=** String**(**lastEsav**);** // energia risparmiata Stato 0

**}**

**if** **(**Soff **==** 0 **&&** S1 **==** 1 **&&** S2 **==** 0**)** **{**

S1PistList**();**

postStr **+=** String**(**lastPist**);** // potenza istantanea Stato 1

lastEsav **=** computeEsaved**();**

postStr **+=** "&field3="**;**

postStr **+=** String**(**lastEsav**);** // energia risparmiata Stato 1

**}**

**if** **(**Soff **==** 0 **&&** S1 **==** 0 **&&** S2 **==** 1**)** **{**

S2PistList**();**

postStr **+=** String**(**lastPist**);** // potenza istantanea Stato 2

lastEsav **=** computeEsaved**();**

postStr **+=** "&field3="**;**

postStr **+=** String**(**lastEsav**);** // energia risparmiata Stato 2

**}**

**if** **(**sendPavgcount **==** PavgLenght**)** **{** // ogni minuto viene soddisfatta la candizione

Paverage**();**

postStr **+=** "&field4="**;**

postStr **+=** String**(**lastPavg**);** // potenza media ultimo minuto

sendPavgcount **=** 0**;**

**}**

postStr **+=** "\r\n\r\n"**;**

client**.**print**(**"POST /update HTTP/1.1\n"**);**

client**.**print**(**"Host: api.thingspeak.com\n"**);**

client**.**print**(**"Connection: close\n"**);**

client**.**print**(**"X-THINGSPEAKAPIKEY: " **+** apiKey **+** "\n"**);**

client**.**print**(**"Content-Type: application/x-www-form-urlencoded\n"**);**

client**.**print**(**"Content-Length: "**);**

client**.**print**(**postStr**.**length**());**

client**.**print**(**"\n\n"**);**

client**.**print**(**postStr**);**

**}**

client**.**stop**();**

**}**

// calcola il valore medio della potenza

void Paverage**()** **{**

float sum **=** 0**;**

**for** **(**int i **=** 0**;** i **<** PavgLenght**;** i**++)** **{**

sum **=** sum **+** Pavg**[**i**];**

**}**

lastPavg **=** sum **/** PavgLenght**;**

**}**

// aggiorna il valore di potenza quando il sistema è spento

void SOffPistList**()** **{**

lastPist **=** PistList**[**0**];**

Pavg**[**sendPavgcount**]** **=** lastPist**;** // inserisce il valore di potenza nel vettore che verrà usato per fare la media

sendPavgcount**++;** // incrementa il contatore che verrà usato per inviare i dati sulla potenza media

**}**

// aggiorna il valore di potenza come somma dell'80% della potenza per le lampade accese e il 50% per quelle spente

void S1PistList**()** **{**

lastPist **=** 0**;**

**for** **(**int i **=** 0**;** i **<** 3**;** i**++)** **{**

**if** **(**presence**[**i**]** **==** 1**)** **{**

lastPist **=** lastPist **+** PistList**[**2**];**

**}**

**else** **{**

lastPist **=** lastPist **+** PistList**[**1**];**

**}**

**}**

Pavg**[**sendPavgcount**]** **=** lastPist**;** // iserisce il valore di potenza nel vettore che verrà usato per fare la media

sendPavgcount**++;** // incrementa il contatore che verrà usato per inviare i dati sulla potenza media

**}**

// aggiorna il valore di potenza come somma del 100% della potenza per le lampade accese e l'80% per quelle spente

void S2PistList**()** **{**

lastPist **=** 0**;**

**for** **(**int i **=** 0**;** i **<** 3**;** i**++)** **{**

**if** **(**presence**[**i**]** **==** 1**)** **{**

lastPist **=** lastPist **+** PistList**[**3**];**

**}**

**else** **{**

lastPist **=** lastPist **+** PistList**[**2**];**

**}**

**}**

Pavg**[**sendPavgcount**]** **=** lastPist**;**

sendPavgcount**++;**

**}**

// calcola la media del vettore delle potenze

float computeEsaved**()** **{**

**return** **((**lastPist **-** **(**PistList**[**3**]** **\*** 3**))** **/** **(**PistList**[**3**]** **\*** 3**))** **\*** 100**;**

**}**

// permette di controllare il sistema e ricevere informazioni tramite bot se viene ricevuto il messaggio START a cui viene risposto con la lista dei comandi disponibili

void initializeBotTelegramControl**()** **{**

**for** **(**int i **=** 1**;** i **<** 2**;** i**++)** **{**

**while** **(true)** **{**

String mess**;**

mess **=** bot**.**message**[**i**][**5**];** // inserisce il testo del messaggio ricevuto nella variabile mess

mess**.**toUpperCase**();** // rende il testo tutto maiuscolo perché non è case sensitive

**if** **(**0 **==** mess**.**compareTo**(**"START"**))** **{**

bot**.**sendMessage**(**bot**.**message**[**i**][**4**],** "Inizializzo sistema"**,** ""**);**

bot**.**sendMessage**(**bot**.**message**[**i**][**4**],** "Comandi disponibili:"**,** ""**);**

bot**.**sendMessage**(**bot**.**message**[**i**][**4**],** "- ON, OFF permettono di accendere e spegnere il sistema"**,** ""**);**

bot**.**sendMessage**(**bot**.**message**[**i**][**4**],** "- PMAX rende il valore di potenza massima del sistema"**,** ""**);**

bot**.**sendMessage**(**bot**.**message**[**i**][**4**],** "- PIST rende il valore di potenza intantanea del sistema"**,** ""**);**

bot**.**sendMessage**(**bot**.**message**[**i**][**4**],** "- ESAV rende la percentuale di energia risparmiata dal sistema"**,** ""**);**

bot**.**sendMessage**(**bot**.**message**[**i**][**4**],** "- EXIT per uscire dal bot telegram"**,** ""**);**

startControlWithTelegram **=** 1**;**

**break;**

**}**

**else** **{**

bot**.**sendMessage**(**bot**.**message**[**i**][**4**],** "Comando iniziale sbagliato!"**,** ""**);**

**break;**

**}**

**}**

bot**.**message**[**0**][**0**]** **=** ""**;**

**}**

**}**

// permette di eseguire dei comandi dopo l'inizializzazione del controllo del sistema con Telegram. Risponde con l'esito del comando

void getMessageAndAnswer**()** **{**

**for** **(**int i **=** 1**;** i **<** 2**;** i**++)** **{**

String mess**;**

mess **=** bot**.**message**[**i**][**5**];**

mess**.**toUpperCase**();**

String answer **=** lightControllerWithInput**(**mess**);** // richiama la funzione per il controllo del sistema tramite input

**if** **(**answer **==** "1"**)** **{**

bot**.**sendMessage**(**bot**.**message**[**i**][**4**],** "Comando Eseguito"**,** ""**);**

**}**

**else** **if** **(**answer **==** "0"**)** **{**

bot**.**sendMessage**(**bot**.**message**[**i**][**4**],** "Comando non esistente, riprovare!"**,** ""**);**

**}**

**else** **{**

bot**.**sendMessage**(**bot**.**message**[**i**][**4**],** answer**,** ""**);**

**}**

bot**.**message**[**0**][**0**]** **=** ""**;**

**}**

**}**

//------------------------------------------------------------------------------

# Appendice: Immagini