

Textul si imaginile din acest document sunt licentiate

Attribution-NonCommercial-NoDerivs

CC BY-NC-ND



Codul sursa din acest document este licentiat

Public-Domain

Esti liber sa distribui acest document prin orice mijloace consideri (email, publicare pe website / blog, printare, sau orice alt mijloc), atat timp cat nu aduci nici un fel de modificari acestuia. Codul sursa din acest document poate fi utilizat in orice fel de scop, de natura comerciala sau nu, fara nici un fel de limitari.

# Cloud's Lights

Controlul iluminatului dintr-o locuință sau chiar dintr-o incintă mai mare prin intermediul serviciilor de cloud a devenit o soluție întâlnită din ce în ce mai des în aplicațiile de tip Home Automation sau Building Automation. Problema majoră a dispozitivelor ce permit acest tip de control este prețul foarte mare. Fie că alegem dispozitive conectate direct la Internet (becuri WiFi) fie că alegem soluții bazate pe controlere specializate de tip routere IoT, costul dispozitivelor necesare face ca aceste soluții să nu fie larg accesibile. În cadrul materialului de față vom încerca să oferim o soluție fiabilă și accesibilă la problema controlului iluminatului într-o locuință prin intermediul rețelei Internet utilizând un serviciu de cloud.



Pentru controlul elementelor de iluminare vom utiliza dispozitive comutator pentru becuri din seria Conrad RSL comercializate în România de German Electronics SRL:



<https://www.germanelectronics.ro/casa-gradina/sistem-automatizare-casa-conrad-rsl/soclu-wireless-pentru-becuri-rslr2-640456.html>

Avantajul acestor dispozitive este faptul că elementul de iluminare este separat, pot fi folosite cu orice tip de bec cu soclu E27 (max. 100W) și în momentul în care becul se arde nu trebuie să schimbăm și elementul de comandă (situație des întâlnită la becurile

WiFi). Aceste dispozitive permit comanda radio de la distanță în bandă ISM de 433MHz – distanță de comandă de până la 25m (suficientă pentru majoritatea apartamentelor). Comanda se poate efectua cu orice telecomandă din seria Conrad RSL dar pe lângă telecomandă se pot folosi și comutatoare de perete din seria Conrad RSL:

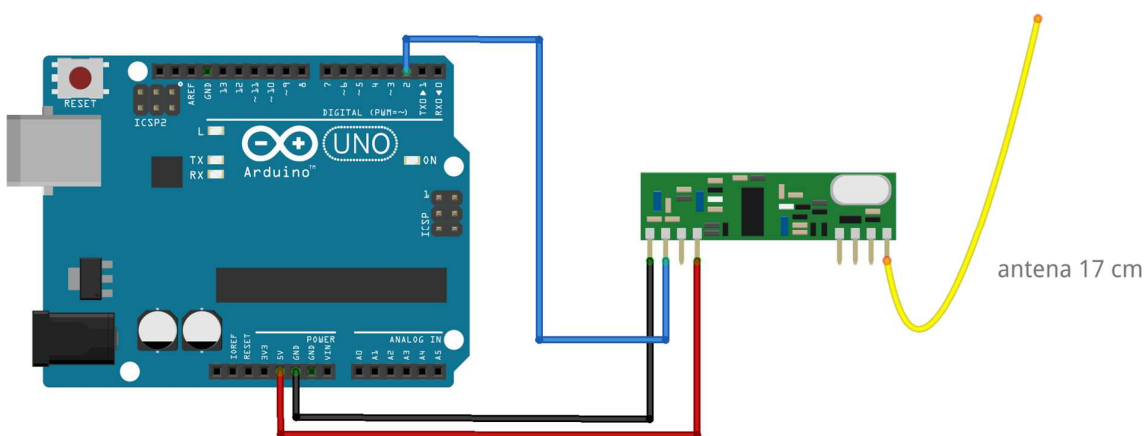


<https://www.germanelectronics.ro/casa-gradina/sistem-automatizare-casa-conrad-rsl/intreupator-de-perete-wireless-rslt4-2-canale-646655.html>

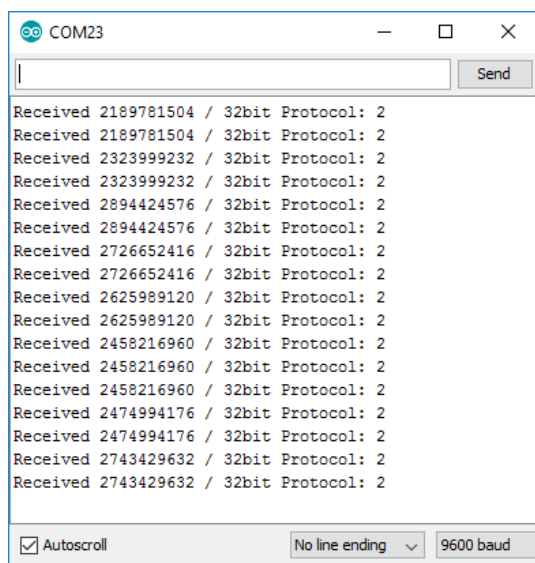
Comutatoarele de perete funcționează pe baza unei baterii de 12V (tip 27A) făcându-le absolut independente de poziționarea rețelei de alimentare cu energie electrică – se pot aplica pe perete oriunde fără a avea nevoie de doză de conectare. Comutatoarele suplinesc telecomenzile permițând comanda a unui sau a două comutatoare de becuri (poziție închis/deschis).

Comutatorul de alimentare a becurilor necesită înregistrarea elementelor de comandă radio (telecomandă, comutator de perete). Butonul prezent la baza soclului poate fi utilizat pentru comanda manuală a becului dar și pentru înregistrarea codurilor de comandă (apăsare prelungă). Pentru operarea corectă a comutatorului este necesară citirea cu atenție a manualului furnizat de producător. Chiar dacă instalarea comutatorului de becuri este asemănătoare cu schimbul unui bec trebuie ca instalarea să se facă cu maximă grijă deoarece există pericol de electrocutare (opriți alimentarea cu energie electrică înainte de instalare!!!).

Pentru a putea reproduce comanda radio a unui astfel de comutator este necesară realizarea unui montaj simplu bazat pe o placă de dezvoltare [Arduino Uno](#) sau echivalentă și un [receptor radio 433MHz](#).

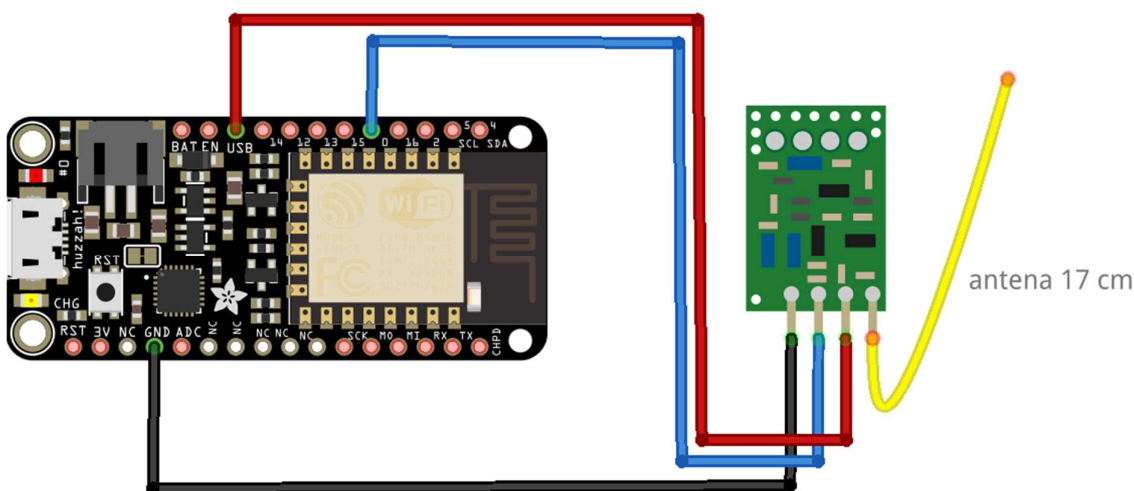


Rulând exemplul *ReceiveDemo\_Simple* al bibliotecii software *rc-switch* vom putea vizualiza tipul de cod și codul emis la fiecare apăsare de buton al comutatorului (sau telecomenzii):



<https://github.com/sui77/rc-switch/>

Codurile obținute (emise de comutator) se vor copia pentru a fi utilizate în program. Sistemul de comandă va fi format dintr-o placă de dezvoltare [Adafruit Feather HUZAH](#) și un [emițător radio 433MHz](#):



Emițătorul radio funcționează la o tensiune de 5V și din acest motiv este alimentat de la pinul USB (tensiunea furnizată de conexiunea USB a plăcii) – sistemul nu va funcționa corect dacă placa de dezvoltare este alimentată de la un acumulator. Pinul de date al emițătorului se va conecta la pinul 0 al plăcii de dezvoltare. Pentru mai multe informații despre instalarea și utilizarea plăcii Adafruit Feather HUZZAH puteți consulta:

Overview | Adafruit Feather HUZZAH ESP8266 | Adafruit Learning System

<https://learn.adafruit.com/adafruit-feather-huzzah-esp8266/>

Programul sistemului va utiliza mediul de dezvoltare Arduino IDE 1.8.1 și bibliotecile software: rc-switch 2.6.2, PubSubClient 2.6.0 (pentru comunicația MQTT cu serviciul de cloud), ArduinoJson 5.8.3 (pentru interpretarea mesajelor provenite de la serviciul de cloud) și ESP8266WiFi 1.0.0.

```
#include <RCSwitch.h>
RCSwitch mySwitch = RCSwitch();
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
```

Atenție!!! În cadrul bibliotecii *PubSubClient* este necesară modificarea fișierului *PubSubClient.h*: constanta *MQTT\_MAX\_PACKET\_SIZE* trebuie schimbată din 128 în 256.

Pentru a permite conectarea la Internet este necesară personalizarea datelor de conectare la rețeaua WiFi locală (care să permită accesul la Internet):

```
const char* ssid      = "...";  
const char* password = "...";
```

Constantele *API\_KEY*, *PROJECT\_ID* și *DEVICE\_UUID* vor fi modificate conform proiectului înregistrat în serviciul de cloud (etapă explicată în secțiunea următoare).

```
#define API_KEY          "..."  
#define PROJECT_ID      "..."  
#define ACTUATOR_NAME1  "LightSocket1"  
#define ACTUATOR_NAME2  "LightSocket2"  
#define ACTUATOR_NAME3  "LightSocket3"  
#define ACTUATOR_NAME4  "AllLightSockets"  
#define DEVICE_UUID     "..."  
#define sec 1000  
  
char clientId[]          = "Feather_HUZZAH";  
char actuatorTopic1[]    =  
    "/a/"API_KEY"/p/"PROJECT_ID"/d/"DEVICE_UUID"/actuator  
    "/"ACTUATOR_NAME1"/state";  
char actuatorTopic2[]    =  
    "/a/"API_KEY"/p/"PROJECT_ID"/d/"DEVICE_UUID"/actuator  
    "/"ACTUATOR_NAME2"/state";  
char actuatorTopic3[]    =  
    "/a/"API_KEY"/p/"PROJECT_ID"/d/"DEVICE_UUID"/actuator  
    "/"ACTUATOR_NAME3"/state";  
char actuatorTopic4[]    =  
    "/a/"API_KEY"/p/"PROJECT_ID"/d/"DEVICE_UUID"/actuator  
    "/"ACTUATOR_NAME4"/state";  
char server[]            = "mqtt.devicehub.net";  
char message_buffer[150];
```

```
WiFiClient apiClient;
```

Procedura callback este necesară în cadrul comunicației MQTT – ea preia mesajele venite de la serviciul de cloud și le transpune în comenzi radio destinate comutatoarelor de becuri.

```
void callback(char* topic, byte* payload,
              unsigned int length)
{
    StaticJsonBuffer<200> jsonBuffer;
    for(int i=0; i<length; i++)
    { message_buffer[i] = payload[i]; }
    JsonObject& root =
        jsonBuffer.parseObject(message_buffer);
    if (!root.success()) {
        Serial.println("parseObject() failed");
        return;
    }
    boolean onoff = root["state"];
    if(String(topic) == String(actuatorTopic1)){
        Serial.println("message arrived: " + String(onoff) +
            " from Light Switch 1");
        if (onoff) mySwitch.send(2189781504,32);
        else if (!onoff) mySwitch.send(2323999232,32);
    }else if ((String(topic) == String(actuatorTopic2))){
        Serial.println("message arrived: " + String(onoff) +
            " from Light Switch 2");
        if (onoff) mySwitch.send(2424662528,32);
        else if (!onoff) mySwitch.send(2558880256,32);
    }else if ((String(topic) == String(actuatorTopic3))){
        Serial.println("message arrived: " + String(onoff) +
            " from Light Switch 3");
    }
}
```

```

        if (onoff) mySwitch.send(2625989120,32);
        else if (!onoff) mySwitch.send(2458216960,32);
    }else if ((String(topic) == String(actuatorTopic4))){
        Serial.println("message arrived: " + String(onoff) +
            " from All Light Switch");
        if (onoff) mySwitch.send(2474994176,32);
        else if (!onoff) mySwitch.send(2743429632,32);
    }
}

PubSubClient client(server, 1883, callback, apiClient);

```

În cadrul secțiunii *setup()* se va inițializa conexiunea WiFi și conexiunea MQTT cu serverul de cloud.

```

void setup() {
    Serial.begin(9600);
    delay(10);
    Serial.println();
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
    Serial.println("\nStarting connection to server...");
    if(client.connect(clientId))

```



```

{
    client.subscribe(actuatorTopic1);
    client.subscribe(actuatorTopic2);
    client.subscribe(actuatorTopic3);
    client.subscribe(actuatorTopic4);
    Serial.println("Successfully connected and running!");
}
else
{
    Serial.println("Connection problem");
}
mySwitch.enableTransmit(0);
mySwitch.setProtocol(2);
mySwitch.setRepeatTransmit(3);
delay(500);
}

```

Secțiunea *loop()* are sarcina de a supraveghea conexiunea MQTT și de a o reface în cazul unei deconectări – prelucrarea mesajelor de comandă se face în procedura *callback*.

```

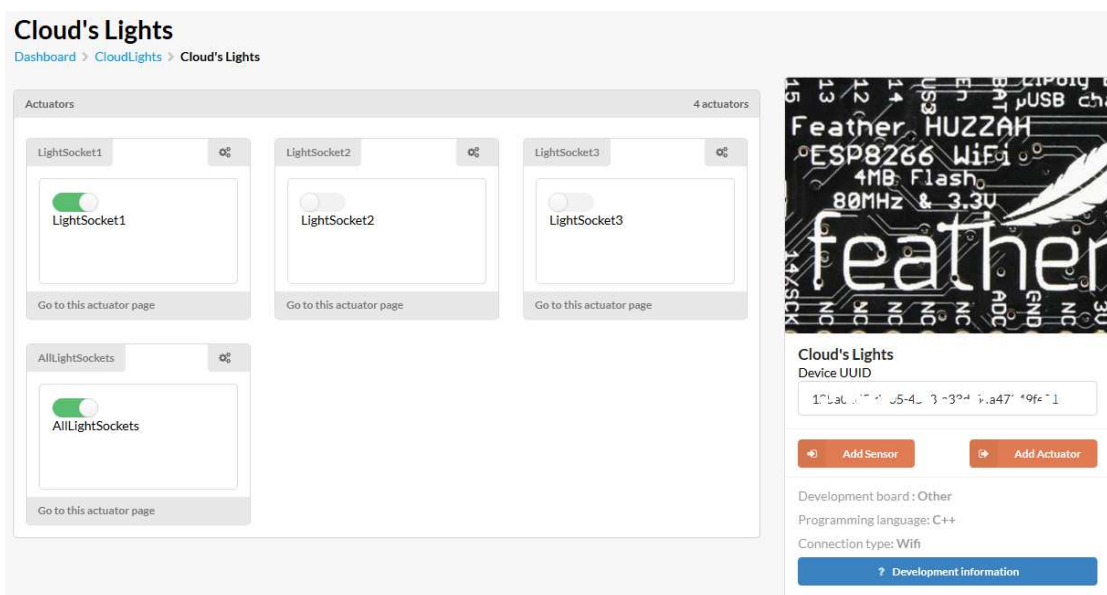
void loop() {
    if (!client.connected())
    { Serial.println("reconnecting ...");
      client.connect(clientId);
      delay(3*sec);
      client.subscribe(actuatorTopic1);
      client.subscribe(actuatorTopic2);
      client.subscribe(actuatorTopic3);
      client.subscribe(actuatorTopic4);
    }
    client.loop();
}

```

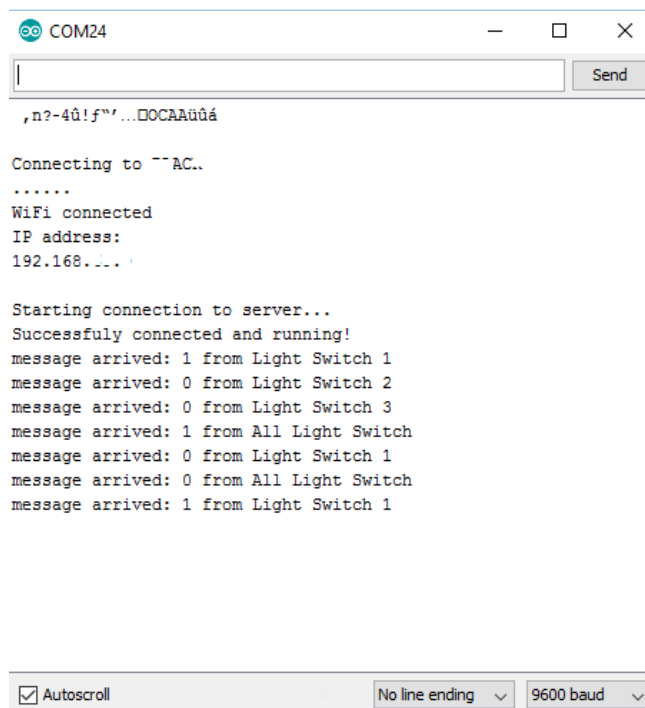
Ca și serviciu de comandă cloud vom utiliza Devicehub.net – un serviciu de cloud specializat IoT care permite înregistrare gratuită a unui cont de dezvoltator (cont cu facilități suficiente pentru proiectul de față: 100 de dispozitive IoT, 100 de senzori . elemente de acționare, 10 mesaje/secundă, 30 milioane de mesaje/lună).



După înregistrarea contului se va genera un nou proiect denumit *CloudLights* (numele se poate modifica). În cadrul acestui proiect se vor adăuga 4 elemente de acționare digitale (Actuator / Digital) pe care le vom numi: *LightSocket1*, *LightSocket2*, *LightSocket3*, *AllLightSockets* (modificarea acestor denumiri necesită modificarea valorii constantelor *ACTUATOR\_NAME* din program). Acum se pot personaliza datele de conectare din program preluând cele necesare din secțiunea *Development Information*.



La execuție programul va avea următoarea raportare pe portul serial (fiecare comandă de închidere / deschidere se va materializa într-un mesaj provenit de la serverul de cloud):



```
COM24
,n?-4û!f""...DOCAAûûá

Connecting to --AC..
.....
WiFi connected
IP address:
192.168.1.1

Starting connection to server...
Successfully connected and running!
message arrived: 1 from Light Switch 1
message arrived: 0 from Light Switch 2
message arrived: 0 from Light Switch 3
message arrived: 1 from All Light Switch
message arrived: 0 from Light Switch 1
message arrived: 0 from All Light Switch
message arrived: 1 from Light Switch 1

[Autoscroll] [No line ending] [9600 baud]
```

La fiecare nouă conectare serviciul de cloud va realiza o inițializare cu starea memorată a comenzilor dar, având în vedere caracterul unidirecțional al comunicației radio, starea comenzilor din serviciul cloud nu oferă certitudinea unei anumite stări pentru comutatoare (starea acestora poate să fie modificată manual sau cu ajutorul unei telecomenzi).

Având în vedere diversitatea de dispozitive telecomandate din familia Conrad RSL aplicația de comandă se poate extinde incluzând și alte dispozitive: prize telecomandate, întrerupătoarea de perete sau comutatoare încastrabile:



<https://www.germanelectronics.ro/casa-gradina/sistem-automatizare-casa-conrad-rsl/comutator-wireless-incestrabil-rslr2-640304.html>

<https://www.robofun.ro/forum/>

Comutatoarele încastrabile pot fi instalate în doze de derivație și pot servi pentru a controla alimentarea cu energie a unor secțiuni din rețeaua electrică (o anexă din gospodărie de exemplu) dar instalarea lor necesită pregătire de specialitate – nu efectuați modificări ale instalației electrice singuri, apălați la un electrician!!!



În cazul în care doriți să dezvoltați sistemul propus puteți parcurge și următoarele proiecte asemănătoare:

ESP8266 Remote Controlled Sockets

<http://randomnerdtutorials.com/esp8266-remote-controlled-sockets/>

Switch a cheap 433Mhz RC-Socket by an Adfafruit HUZZAH ESP8266 WebServer

<http://fritzing.org/projects/esp8266-433mhz-rc-socket-switch>

Using an ESP8266 to Control Mains Sockets Using 433mhz Transmitter and Receiver

<http://www.instructables.com/id/Using-an-ESP8266-to-Control-Mains-Sockets-Using-43>