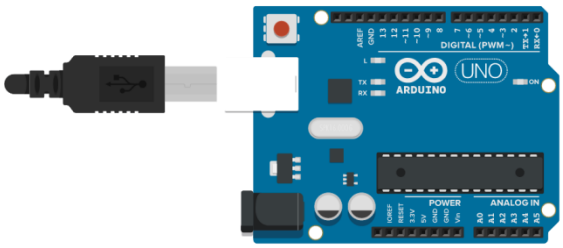


PT_10

Subir y bajar el motor de una persiana mediante mando de radiofrecuencia y wifi.

Autores: Ahmed Chtatou Bougdadi y Juan Carlos Parejo Reja

ÍNDICE

1. INTRODUCCIÓN Y FINALIDAD

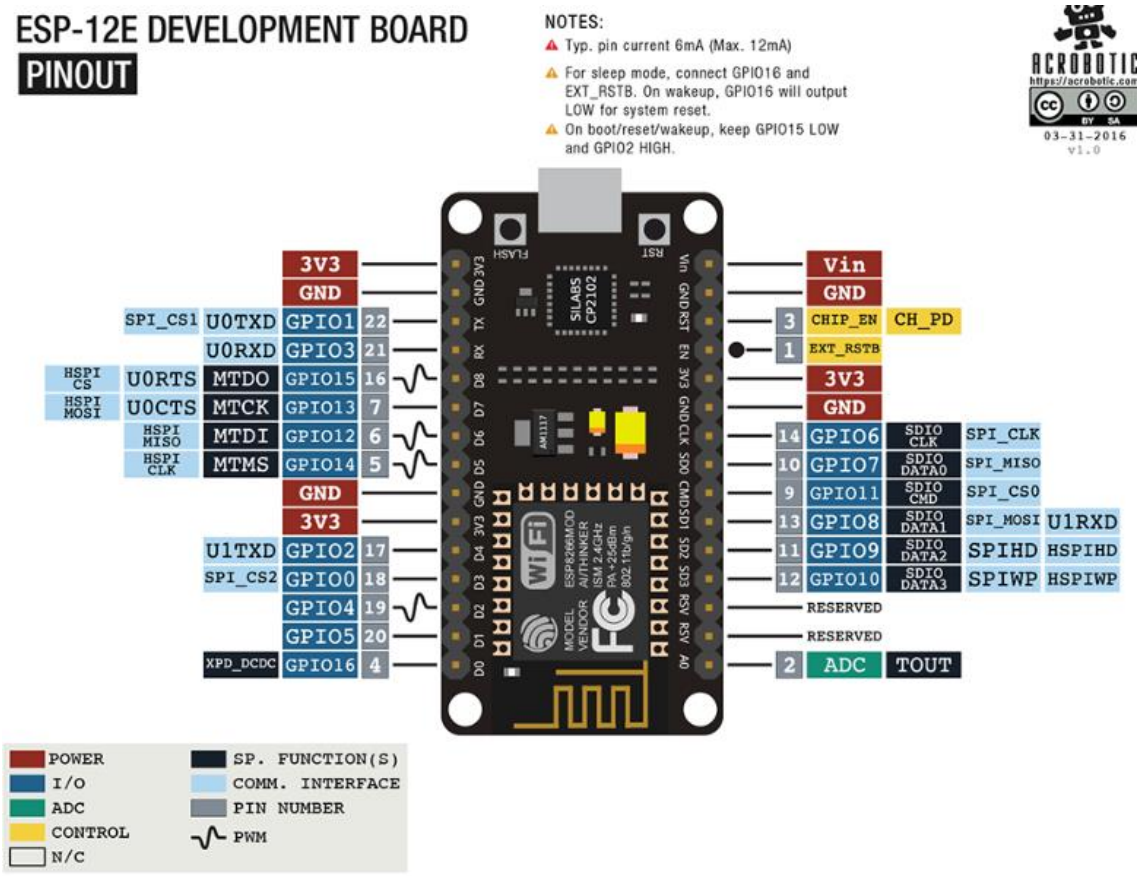
En esta práctica vamos a montar con Arduino el motor de una persiana el cual manejaremos con un mando de radiofrecuencia y con el receptor wifi y teléfono móvil.

2. MATERIALES Y COMPONENTES

- Protoboard
- Cables (macho/macho , macho/hembra) Arduino
- Placa Arduino UNO
- Control remoto inalámbrico IC 2262/2272 4 CH (4canales)
- Placa de desarrollo nodemcu esp8266
- Motor de persiana 2 direcciones.
- 2 relés
- Fuente de alimentación de arduino

3. DATASHEETS DE LOS COMPONENTES

- Placa de desarrollo nodemcu esp8266



Especificaciones técnicas:

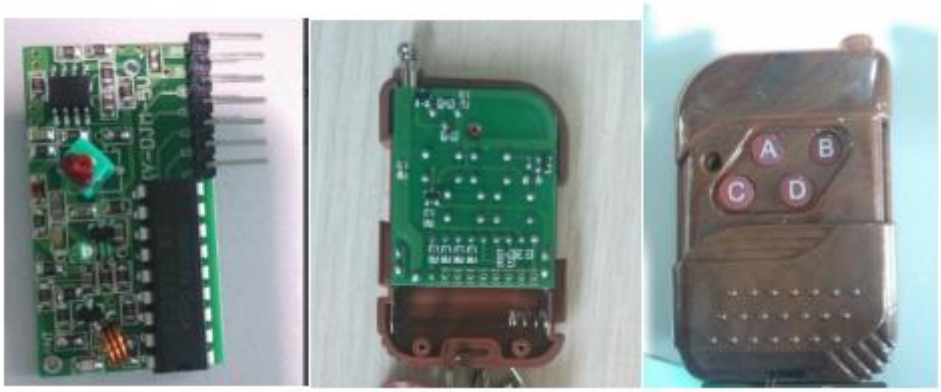
- Voltaje de Alimentación (USB): 5V DC
- Voltaje de Entradas/Salidas: 3.3V DC
- SoC: ESP8266 (Módulo ESP-12)
- CPU: Tensilica Xtensa LX3 (32 bit)
- Frecuencia de Reloj: 80MHz/160MHz
- Instruction RAM: 32KB
- Data RAM: 96KB
- Memoria Flash Externa: 4MB
- Pines Digitales GPIO: 17 (pueden configurarse como PWM a 3.3V)

- Pin Analógico ADC: 1 (0-1V)
- UART: 2
- Chip USB-Serial: CP2102
- Certificación FCC
- Antena en PCB
- 802.11 b/g/n
- Wi-Fi Direct (P2P), soft-AP
- Stack de Protocolo TCP/IP integrado
- PLLs, reguladores, DCXO y manejo de poder integrados
- Potencia de salida de +19.5dBm en modo 802.11b
- Corriente de fuga menor a 10uA
- STBC, 1×1 MIMO, 2×1 MIMO
- A-MPDU & A-MSDU aggregation & 0.4ms guard interval
- Wake up and transmit packets in < 2ms
- Consumo de potencia Standby < 1.0mW (DTIM3)

Interface:

- SDIO 2.0, SPI, UART
- Integra RF switch, balun, 24dBm PA, DCXO y PMU
- Posee un procesador RISC, memoria en chip e interface para memoria externa
- Procesador MAC/Baseband integrado
- Interface I2S para apliaciones de audio de alta calidad
- Reguladores de voltaje lineales "low-dropout" en chip
- Arquitectura propietaria de generacion de clock "spurious free"
- Módulos WEP, TKIP, AES y WAPI integrados

- Control remoto inalámbrico IC 2262/2272 4 CH (4canales)



Especificaciones:

El mando a distancia y el módulo receptor ya contienen 2262/2272 chip de codec.Cuando se pulsa el botón de mando a distancia, se libera de alto nivel de salida hasta que el botón del terminal correspondiente del módulo receptor.Que la antena de la primavera se sueldan a la almohadilla ANT del módulo receptor puede aumentar la distancia de comunicación.

Módulo de receptor:

- Tensión de funcionamiento: 4.5-5.2V DC
- Corriente de funcionamiento: 15mA
- Frecuencia de trabajo: 315MHz
- pines de salida: 4 (D0, D1, D2, D3) que corresponden a la distancia botones de control A / B / C / D
- Nivel de salida: activa alta
- Antena: antena ANT almohadillas se pueden soldar antena de la primavera
- Excelente para DIY

mando a distancia:

- Tensión de funcionamiento: 12V
- Tipo de batería: 23A, 12V

- Corriente de funcionamiento: 10 mA (máx)
- Frecuencia de trabajo: 315MHz
- Number of buttons: 4

Fuente de alimentación:

Especificaciones:

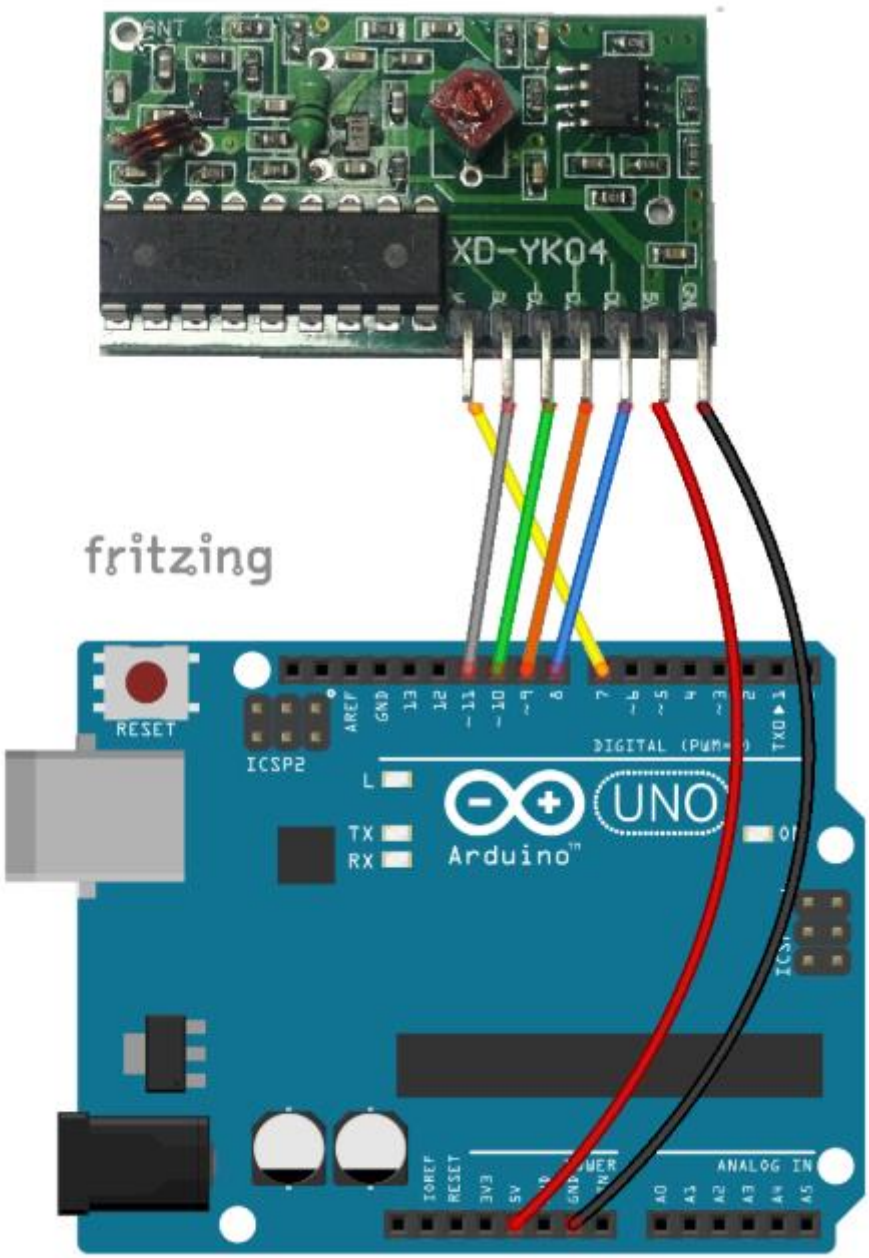
-Voltaje de entrada: 6.5-12 V (DC) o fuente de alimentación USB.

-Voltaje de salida: 3.3V/5V.

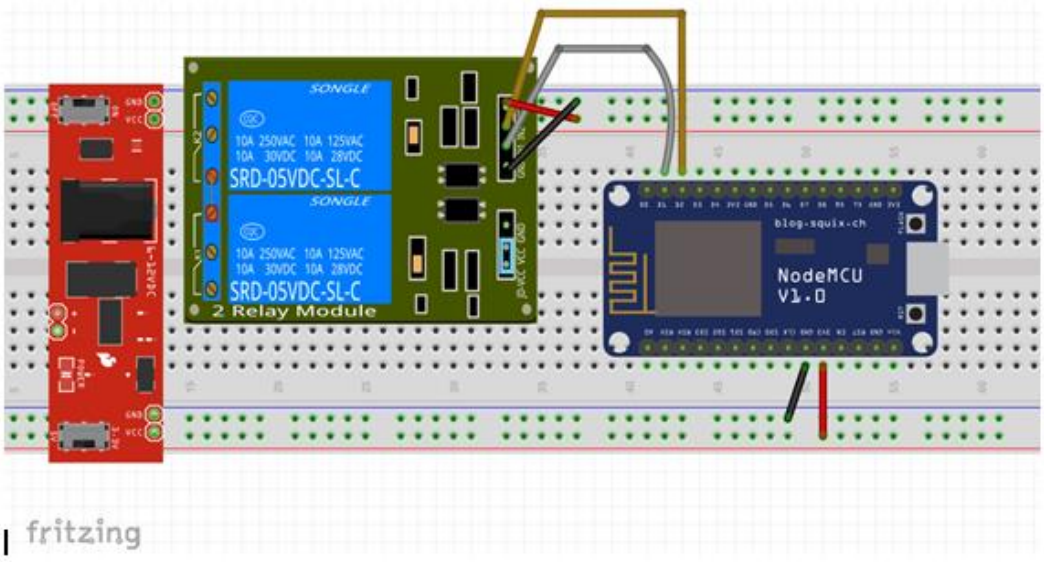
-Máxima corriente de salida: 2 Puertos para alimentar los distintos extremos de la protoboard.

4. Esquemas de los montajes

Esquema Fritzing con mando radiofrecuencia:



Esquema Fritzing Placa de desarrollo nodemcu esp8266


SONGLE
10A 250VAC 10A 125VAC
10A 30VDC 10A 28VDC
SRD-05VDC-SL-C
SONGLE
10A 250VAC 10A 125VAC
10A 30VDC 10A 28VDC
SRD-05VDC-SL-C
2 Relay Module
blog-squix-ch
NodeMCU
V1.0
fritzing

5. CÓDIGO

5.1 Código mando radiofrecuencia.

```
int ledPin = 13; // Conectamos el led 13 y asignamos variable ledPin
int ledPin2 = 12;
int inPin = 7; // el botón de lectura es el 7. Lo llamamos inPin
int val = 0; // creamos variable val que leerá estado pin 7
int motor = 0;// variable interna que nos viene a decir cómo está el motor
void setup()
{
Serial.begin (9600);
pinMode (13, OUTPUT) ;
pinMode (12, OUTPUT) ;
for (int i = 7 ; i < 12 ; i++)
pinMode(i, INPUT) ;
}


void loop()
{
if (digitalRead(7) )     // Si hay señal de radio válida
{  Serial.print("Nos llega un valor. \t"); // se imprime Nos llega un valor
if (digitalRead(8))  // leemos el puerto 8 a ver si llega señal
```

```
Serial.print( "Se ha pulsado el Botón D"); // Se imprime, Se ha pulsado ...
 if (digitalRead(9)) // se hace lo mismo con el 9
 Serial.print( "Se ha pulsado el Botón B");
 if (digitalRead(10)) // repetimos para el 10
 {
 Serial.print( "Ahora se ha pulsado el Botón A.");
 digitalWrite (12, ! digitalRead(12)) ; // Sacamos por el puerto 12 el valor de 12
 delay (500) ; // esperamos 0.5 segundos
 val = digitalRead(inPin); // el valor val toma la lectura del pin 7
 if (val ==HIGH && motor ==LOW )
 {
 motor = 1; // se pone variable  motor a 1
 digitalWrite(ledPin, HIGH); // se activa motor
 delay (500);// delay de un segundo
 }
 val = digitalRead(inPin); // volvemos a leer el valor
 if (val ==HIGH && motor == 1 )
 {
 motor = 0; // se pone variable motor a cero
 digitalWrite(ledPin, LOW); // se desactiva motor
 delay (1000);
 }
 }
 if (digitalRead(11))
```

```
{
Serial.print( "Se ha pulsado el botón C");
digitalWrite (13, ! digitalRead(13)) ; // se activa la salida 13
delay (500) ; // se espera 0,5 segundos
}
Serial.println("\t"); // Imprime un tabulador
}
}
```

5.2 Código Wifi.

```
#include <ESP8266WiFi.h>
#include<FirebaseArduino.h>
#define FIREBASE_HOST "home-smart-control.firebaseio.com"
//Aqui pondremos la URL del proyecto de Firebase sin"http:" , "\"
and "/"
#define FIREBASE_AUTH
"bEJRRkXLD7yTx8awq8nHA0SwwVgyiV2uC3vvoVE4"  //Codigo
de autenticacion cliente de Firebase "Database Secret"
#define WIFI_SSID "Adriphone"                    //SSID de
nuestra red WIFI para que el "ESP8266" se conecte a la red
#define WIFI_PASSWORD "vpk5xs9623cse"
//ponemos las password de nuestra red WIFI

#define Relay2 5 //Pin D1 NodeMCU
int val2;
```

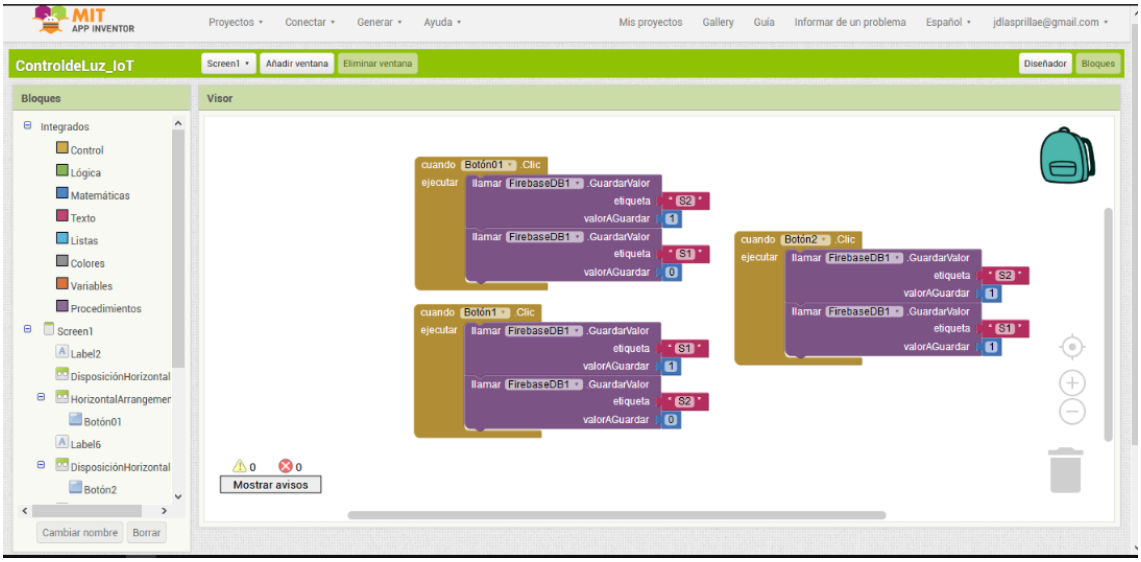
```
#define Relay1 4  //Pin D2 NodeMCU
int val3;


void setup()
{
  Serial.begin(115200);    // Seleccione la misma velocidad en
baudios si desea ver los datos en Serial Monitor

  pinMode(Relay1,OUTPUT);
  pinMode(Relay2,OUTPUT);


  digitalWrite(Relay1,HIGH);
  digitalWrite(Relay2,HIGH);




  WiFi.begin(WIFI_SSID,WIFI_PASSWORD);
  Serial.print("connecting");
  while (WiFi.status()!=WL_CONNECTED){
    Serial.print(".");
    delay(500);
```

```
  }
  Serial.println();
  Serial.print("connected:");
  Serial.println(WiFi.localIP());

  Firebase.begin(FIREBASE_HOST,FIREBASE_AUTH);

  Firebase.setInt("S1",0);                  // Aquí las variables "S1" y
"S2" debe ser la que se usa en nuestro Firebase y MIT App
Inventor (Aplicacion movil)
  Firebase.setInt("S2",0);



}
void firebasereconnect()
{
  Serial.println("Trying to reconnect");
    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  }

void loop()
{
  if (Firebase.failed())
      {
      Serial.print("setting number failed:");
      Serial.println(Firebase.error());
```

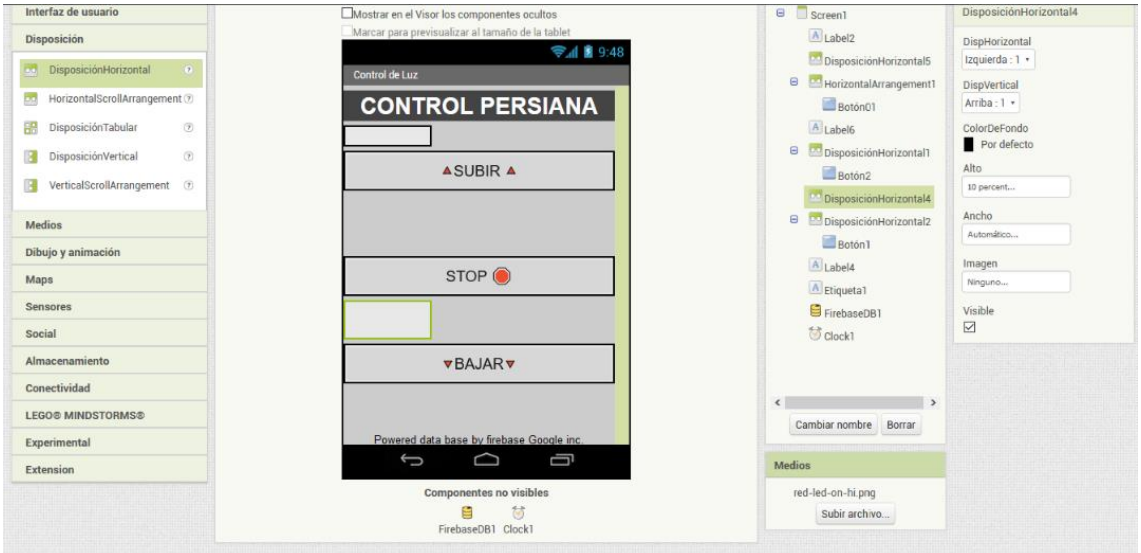
```
    firebasereconnect();
    return;
    }

 val3=Firebase.getString("S1").toInt();    // Lectura del valor de la
variable de estado desde firebase

 if(val3==1)                          // Si, el Estado es 1, enciende el
Relay1
   {

    digitalWrite(Relay1,LOW);
    Serial.println("ON");
  }
  else if(val3==0)                    // Si, el estado es 0, apaga el
relay1
  {

    digitalWrite(Relay1,HIGH);
    Serial.println("OFF");
  }
delay(500);
 val2=Firebase.getString("S2").toInt();   // Lectura del valor de la
variable de estado desde firebase
```

```
 if(val2==1)                    // Si, el Estado es 1, enciende el
Relay1
    {

     digitalWrite(Relay2,LOW);
     Serial.println("ON");
   }
   else if(val2==0)              // Si, el estado es 0, apaga el
relay1
   {

     digitalWrite(Relay2,HIGH);
     Serial.println("OFF");
   }

}
```

Codigo para la APP wifi



Control de la APP móvil wifi



6. PROCESSING

¿Qué es Processing?

Processing es un dialecto de Java que fue diseñado para el desarrollo del arte gráfico, para las animaciones y aplicaciones gráficas de todo tipo. Desarrollado por artistas y para artistas.

Explicaremos brevemente en que consiste processing

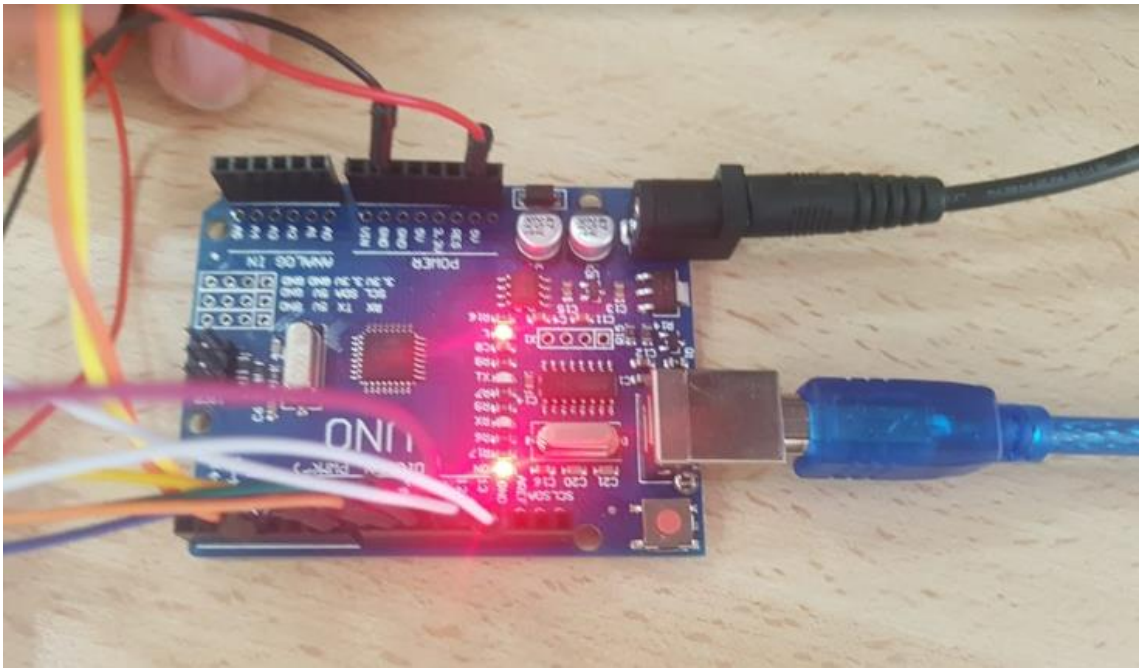
Gracias a este software de visualización y unas librerías que han sido instaladas previamente en dicho programa, podemos visualizar gráficamente los objetos detectados por nuestro sonar así como las medidas a las que se encuentra el mismo, pudiendo ver la distancia y el ángulo a la que se encuentra el obstáculo,

para comunicar el programa con Arduino previamente tendremos que indicar a este en el puerto “COM” donde lo tenemos conectado en nuestro caso será el puerto “COM 3” una vez hecho eso pulsando sobre el botón de ejecutar nos apareará la interfaz gráfica del radar, el cual si todo es correcto nos mostrará lo citado previamente.

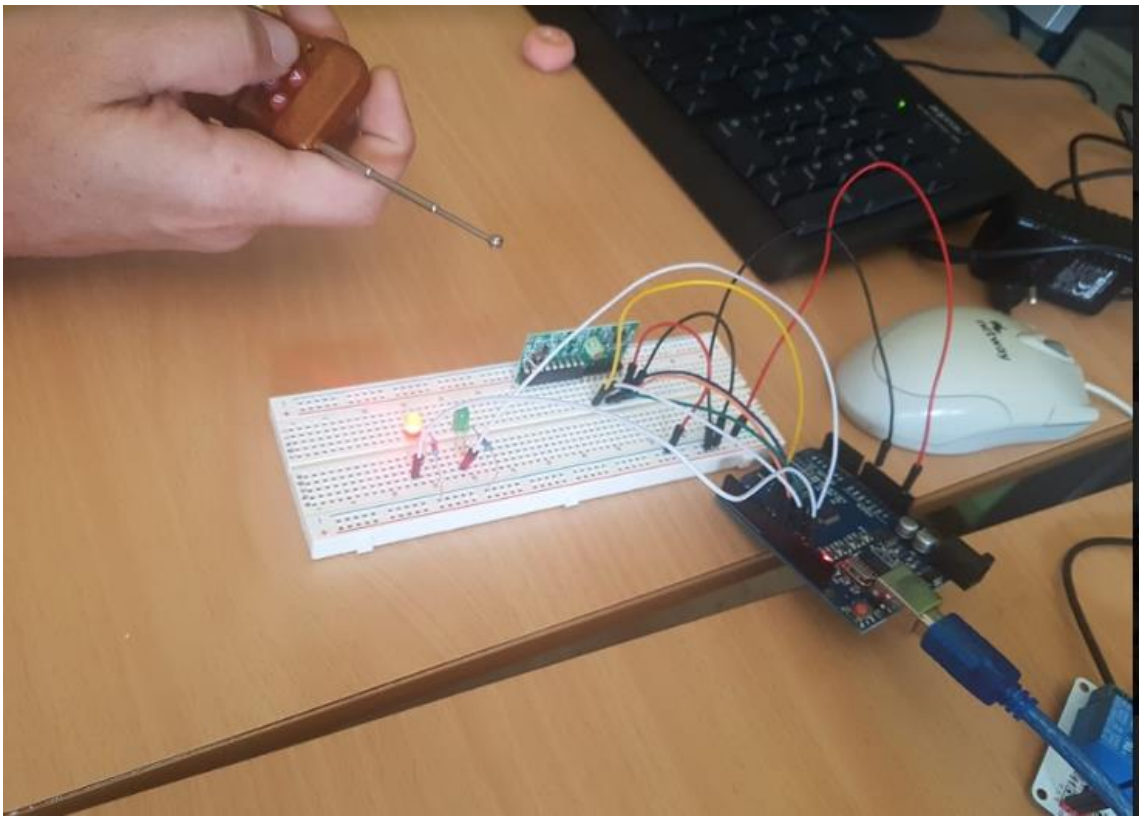
7. MONTAJE EN FÍSICO CON FOTOS Y VIDEOS

Para el montaje físico he utilizado los componentes citados previamente los cuales con la ayuda de cables he conexionado dando sentido al proyecto, estos han sido conectados en la protoboard para facilitar la labor de unión de componentes, una vez finalizado el mismo podremos proceder a su programación y alimentación, para ello simplemente conectamos el USB a Arduino y a PC, a continuación vamos a mostrar las fotos de la practica:

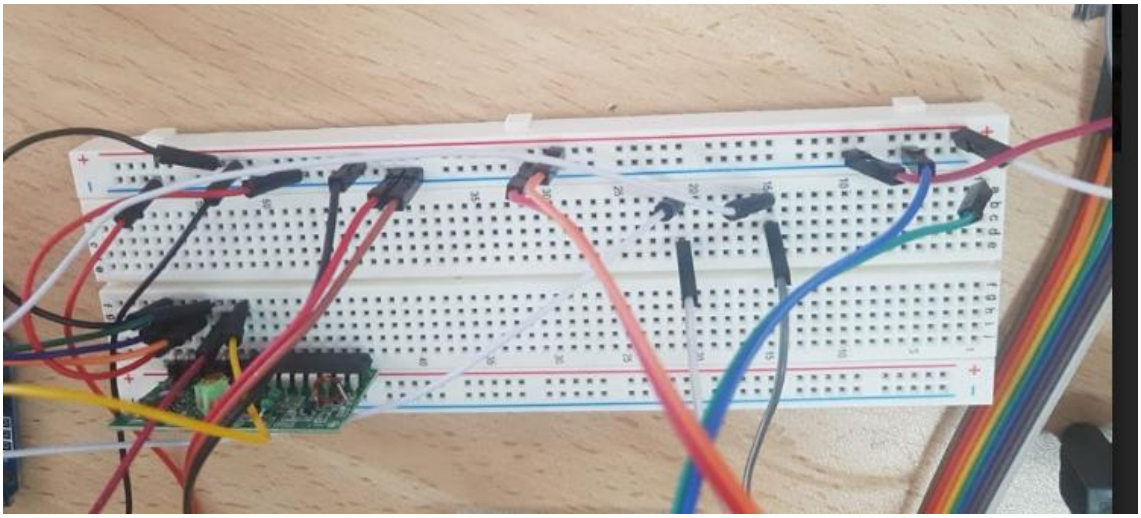
Conexiones de Arduino



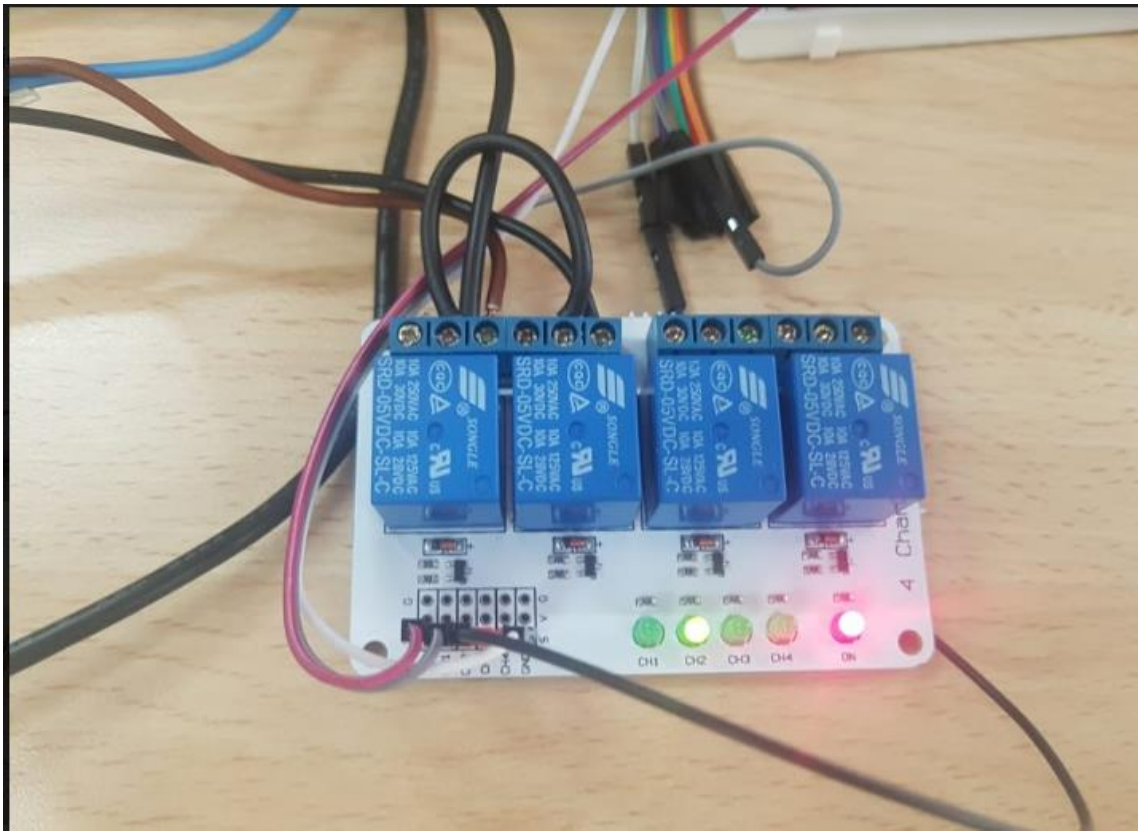
Utilizando el mando de radiofrecuencia con LED



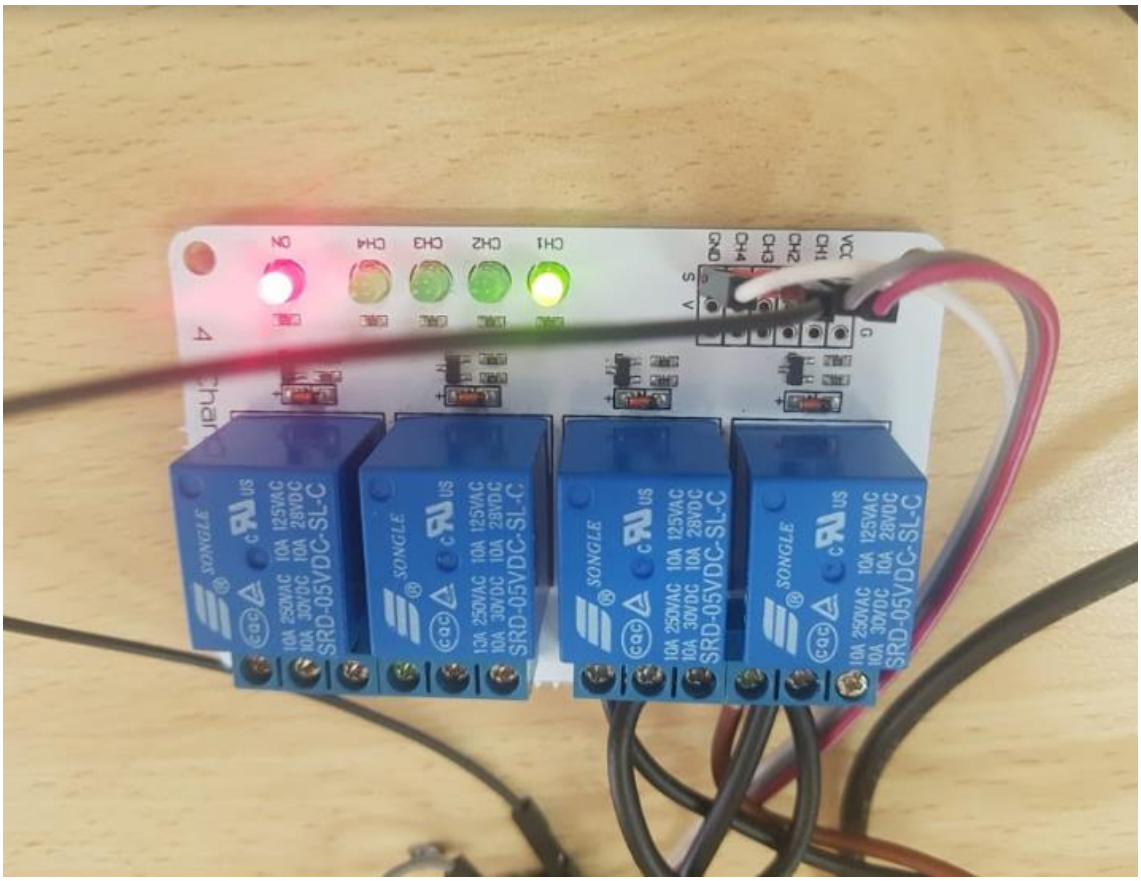
Conexiones de la placa board



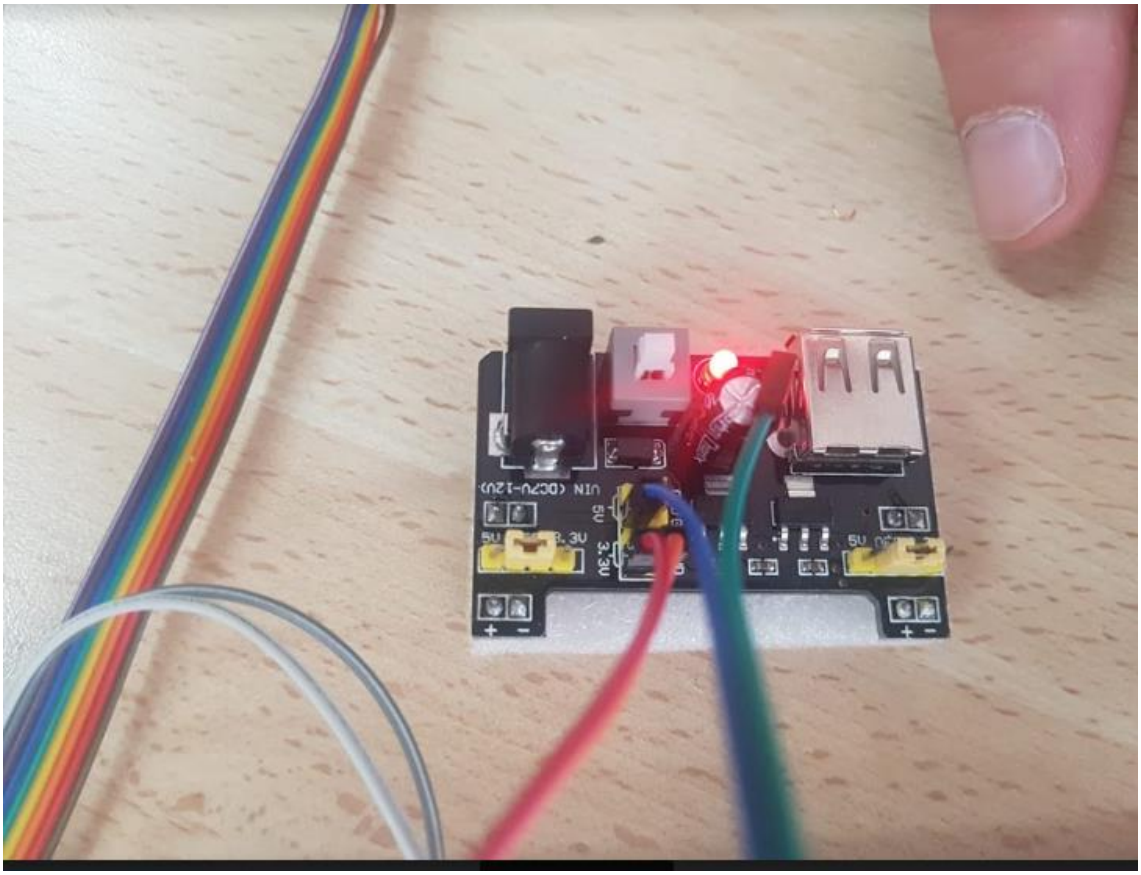
Comprobacion con el relé 1



Comprobación con el relé 2



Conexiones de la fuente de alimentación



Motor de persiana que hemos utilizado

8. CONCLUSIONES E INCONVENIENTES.

Las conclusiones que hemos sacado es la comodidad de subir y bajar una persiana mediante un mando de radiofrecuencia y controlándolo también a través del wifi en comparación a hacerlo accionando un pulsador o a través de una cuerda.
Algunos inconvenientes que nos hemos encontrado han sido que el botón de subir y bajar podían estar encendidos los dos a la vez tanto con el mando de radiofrecuencia como a través del wifi lo que conllevaba a un recalentamiento del motor pudiendo hacer que se estropeara al quemarse el motor.