
UNIVERSIDAD
POLITÉCNICA
DE MADRID


**Trabajo de Informática**

# Maqueta de habitación domótica mediante el uso de Arduino y Dev-C++

Grupo A109
Curso 2018/2019

Pablo Núñez Hernández
pablo.nhernandez@alumnos.upm.es
Nº Mat.: 54773
Jaime Palomino Vaquero
jaime.palomino.vaquero@alumnos.upm.es
Nº Mat.: 54785

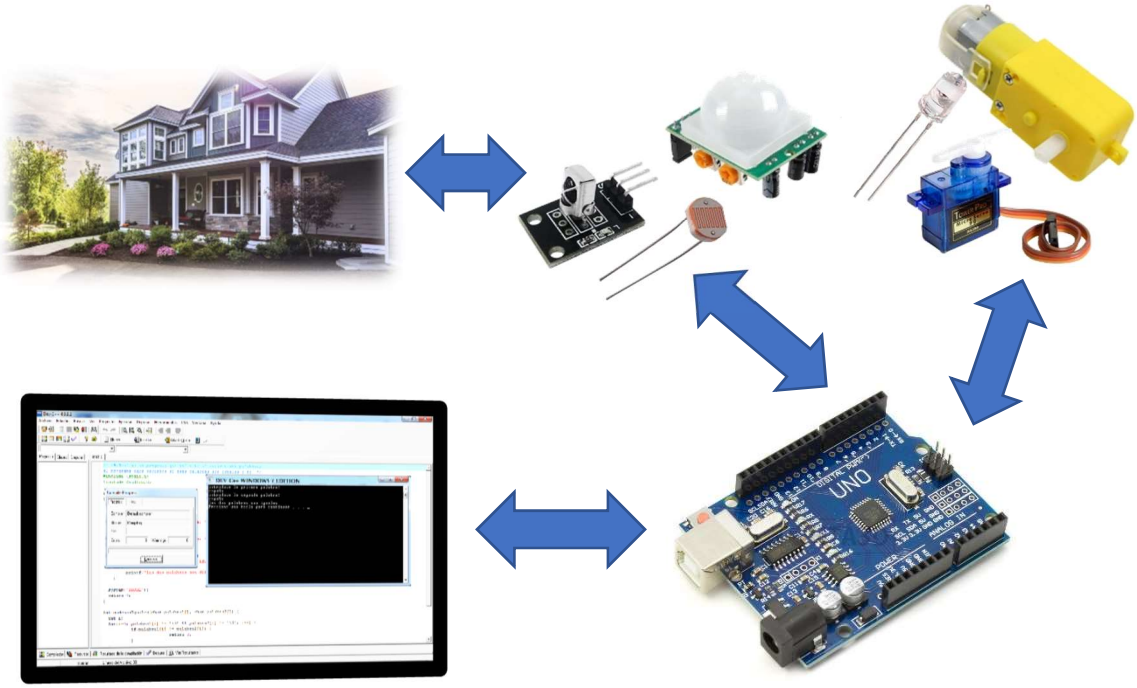
# Índice

# 1. RESUMEN Y REQUISITOS FUNDAMENTALES.

## 1.1. Resumen

Sistema electrónico que a través de un microcontrolador y en función de sensores conectados a él activa los actuadores necesarios según los diferentes condicionantes. Esta aplicación permite encender y apagar la placa desde el ordenador y realiza un informe en un fichero sobre las veces que se ha producido cada actuación y con detalle de fecha y hora.

Para, ello hacemos uso de un PIR para detectar movimiento, un LDR para la cantidad de luz y un receptor IR para recibir la señal de un mando que acciona el servo como sensores. También un motorreductor, un servo y un led como simulación de una persiana, una apertura de puerta y una lampara, respectivamente.



## 1.2. Requisitos fundamentales.

1º - La aplicación dispone de un control en pantalla para iniciar y finalizar la placa.
2º - Al iniciarla, se pone en marcha los sensores.
- El LDR mide la cantidad de luz y así ofrece una resistencia.
- El sistema de fecha y hora nos indica la hora en la que nos encontramos.
- El PIR detecta si hay movimiento.
- El IR recibe la señal del mando.

3º - Se realizan o no realizan acción los actuadores.
- El LED se enciende si hay poca cantidad de luz y el PIR detecta presencia.
- El motor se activa hacia un sentido u otro según la hora.
- El servo gira los determinados grados para poder abrir la puerta si el botón pulsado es el correcto.

4º - Manda información a la aplicación de lo que se ha ejecutado y lo guarda en un fichero.
5º - Podemos abrir el fichero para ver y estudiar las veces que se realizan las diferentes acciones y realizar ajustes para el futuro.

# 2. HARDWARE. FUNDAMENTOS TÉCNICOS.

## 2.1. Detección de movimiento. PIR HC-SR501

*https://www.luisllamas.es/detector-de-movimiento-con-arduino-y-sensor-pir/*

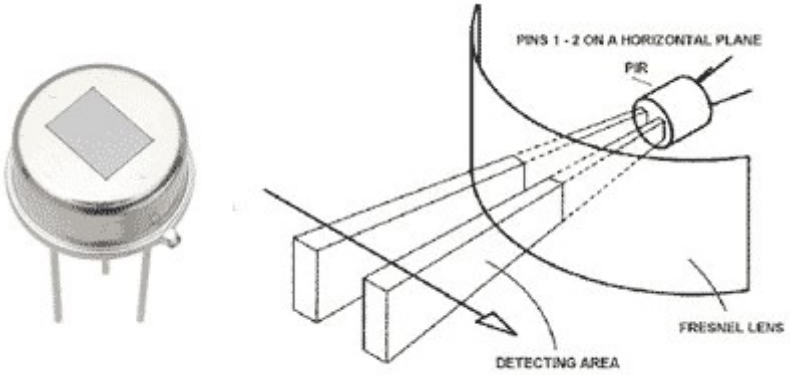
Los sensores infrarrojos pasivos (PIR) son dispositivos para la detección de movimiento. Son frecuentemente usados en juguetes, aplicaciones domóticas o sistemas de seguridad.

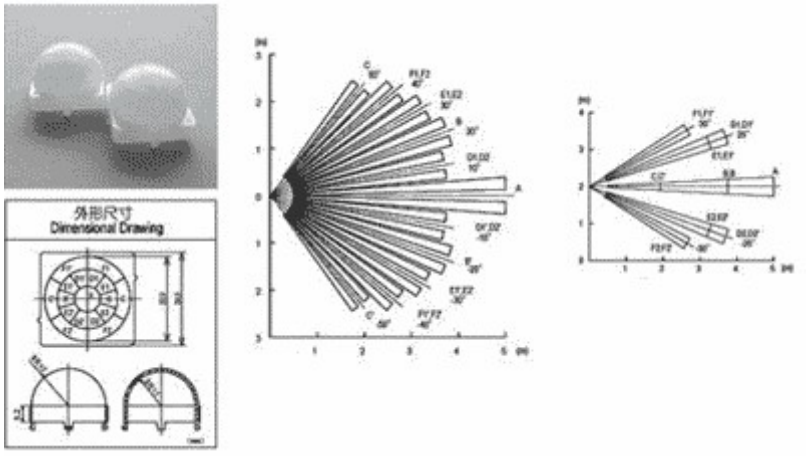
Estos se basan en la medición de la radiación infrarroja. Todos los cuerpos (vivos o no) emiten una cierta cantidad de energía infrarroja, mayor cuanto mayor es su temperatura. Los dispositivos PIR disponen de un sensor piezo eléctrico capaz de captar esta radiación y convertirla en una señal eléctrica.

El sensor está dividido en dos campos y se dispone de un circuito eléctrico que compensa ambas mediciones. Si ambos campos reciben la misma cantidad de infrarrojos la señal eléctrica resultante es nula. Por el contrario, si los dos campos realizan una medición diferente, se genera una señal eléctrica.

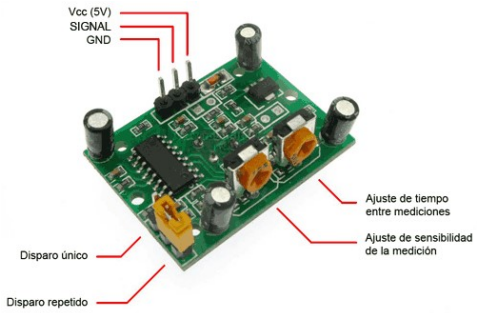
El otro elemento es la óptica del sensor, una cúpula de plástico formada por lentes de fresnel, que divide el espacio en zonas, y enfoca la radiación infrarroja a cada uno de los campos del PIR y así capta un promedio de la radiación infrarroja del entorno.

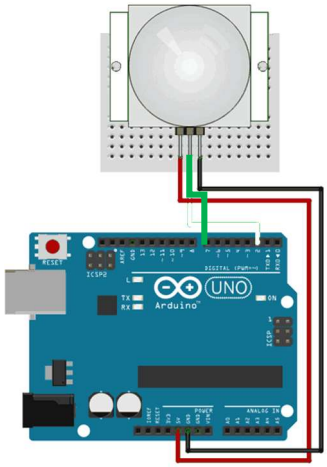
De esta forma, si un objeto atraviesa uno de los campos se genera una señal eléctrica diferencial, que es captada por el sensor, y se emite una señal digital.
Por ello este sensor lo usamos para detectar presencia en la habitación.



**Esquema y código de la función:**



```
const int pir= 7;                                   //PIR en pin 7

void setup()
{
 pinMode(pir, INPUT);                 //El pir es un dispositivo de entrada
}

void loop()
{
  int detector_presencia (void)
  {
  int presencia;
  int valor;
  valor= digitalRead(pir);                    //Leer pir
    if (valor == HIGH )                      //Si detecta presencia
       presencia=1;
    else
      presencia=0;
    return presencia;
  }
}
```

## 2.2. Verificar acceso a la habitación. Detección de la señal infrarroja del mando a distancia.

*https://www.luisllamas.es/arduino-mando-a-distancia-infrarrojo/*

Un mando a distancia es un dispositivo de control que emplea un LED infrarrojo para enviar una señal al receptor. Un mando a distancia emplea un emisor de luz en el infrarrojo cercano, invisible para el ojo humano, pero que puede ser captado con facilidad por un receptor infrarrojo.

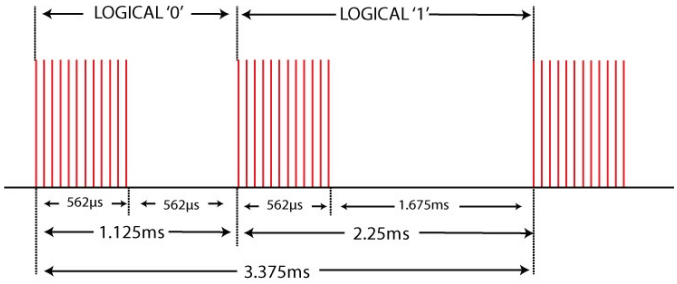
El alcance es limitado, típicamente inferior a 3m. La distancia depende fuertemente del ángulo de emisión, disminuyendo rápidamente a medida que nos desviamos de la dirección frontal.

Un mando a distancia transmita un cierto mensaje al receptor empleando luz infrarroja como sistema de transmisión. La luz empleada típicamente está en el rango de 940 nm.El mensaje nunca se envía directamente como un pulso, si no que se envía modulada sobre una onda portadora. Esto se hace para mejorar el rechazo al ruido y a la luz ambiental. La frecuencia, en general, varía entre 36-50 kHz, siendo el más habitual en torno a los 38 kHz.

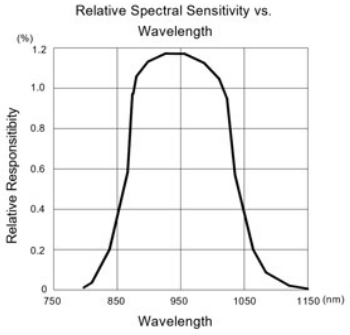
Uno de los protocolos más habituales, que es el que emplearemos con Arduino, es el protocolo NEC, que emplea una onda portadora de 38 kHz y modulación por distancia de pulsos (PDM Pulse Distance Modulation). La transmisión comienza con una señal de 9ms, seguido de un espacio de 4.5ms.

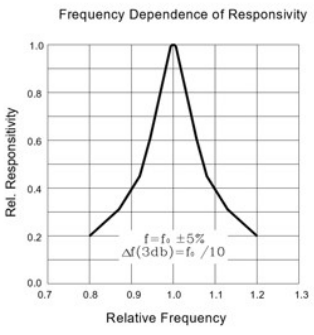
La onda portadora tiene un periodo de 26µs, y la señal transmitida distingue entre 0 y 1 por la duración de los pulsos, siendo:

Logical 0 – Un pulso de 562.5µs seguido por un espacio de 562.5µs.
Logical 1 – Un pulso de 562.5µs seguido por un espacio de 1.675µs.

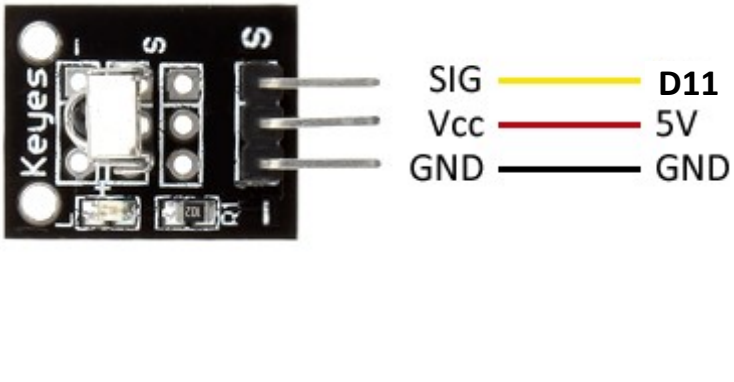


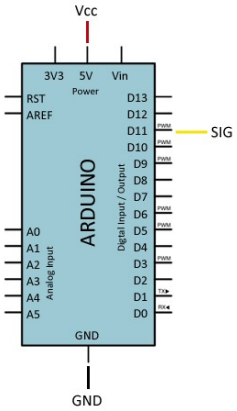

Finalmente para recibir la señal de control se emplea un receptor específico de tres de terminales, que consisten en un sensor infrarrojo que incluyen un demodulador en la banda de 36-38kHz, un filtro PCM (Pulse Code Modulation) y preamplificación rechazo de luz ambiental.



**Esquema y código de la función:**




```
#include <IRremote.h>                    //Libreria del receptor IR
int RECV_PIN = 11;                                          //IR en pin 11
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup()
{
  Serial.begin(9600);
   irrecv.enableIRIn();
}

void loop()
{
  int dump(decode_results *results)
{
 dato=(results->value);
 return dato;
}

int clave (void)
{
 error=1;
 if (irrecv.decode(&results))
  {
   numero=dump(&results);
   if (numero==765) //5
    error=0;
   irrecv.resume();
  }
  delay(300);
  return error;
}
}
```

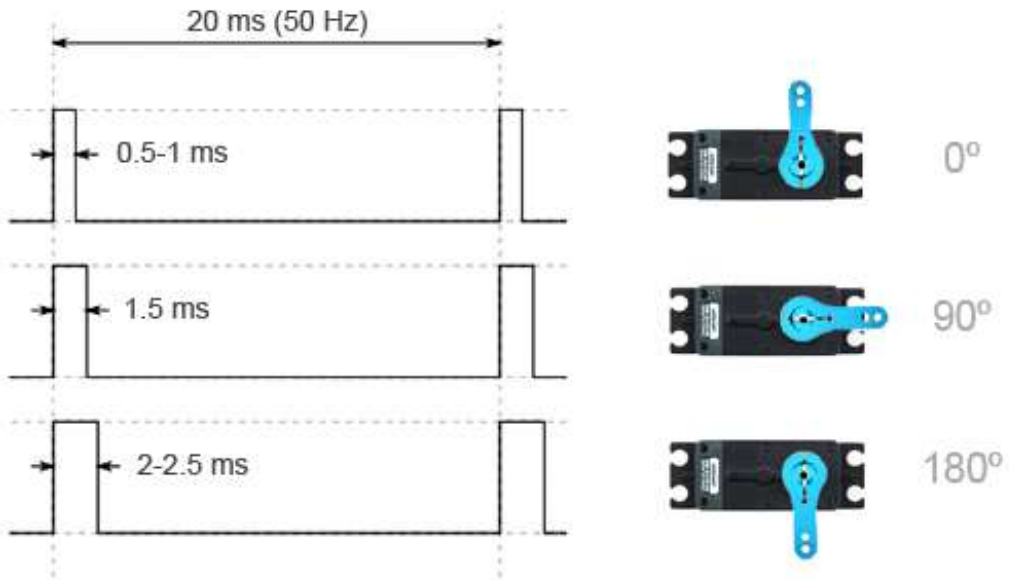
## 2.3. Desbloqueo de la puerta. Servomotor SG90.

*https://www.luisllamas.es/controlar-un-servo-con-arduino/*

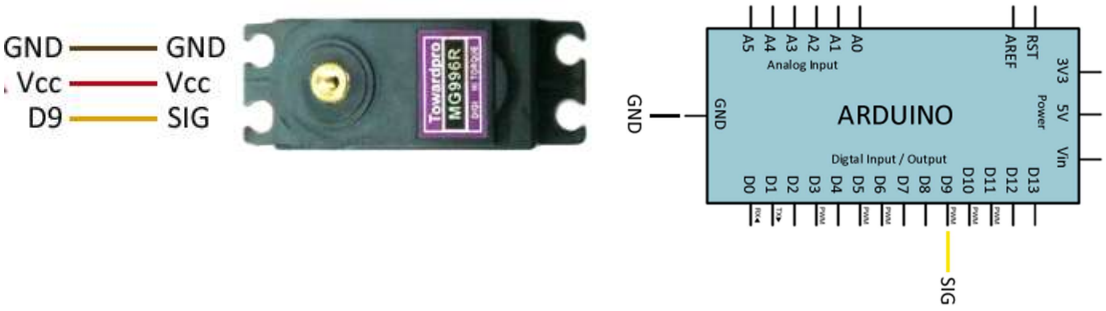
Un servomotor es un accionador ampliamente empleado en electrónica. A diferencia de otros tipos de motores en los que se controla la velocidad de giro, en un servo se indica el ángulo deseado y el servo se encarga de posicionarse en este ángulo.

Típicamente los servos disponen de un rango de movimiento de entre 0 a 180°. Es decir, no son capaces de dar la vuelta por completo.
Internamente un servo está constituido por un motor de corriente continua, acoplado a un reductor para reducir la velocidad de giro, junto con la electrónica necesaria para controlar su posición. Frecuentemente se dispone de un potenciómetro unido al eje del servo que permite al servo conocer la posición del eje. Esta información es tratada por un controlador integrado que se encarga de actuar sobre el motor para alcanzar la posición deseada.

La comunicación de la posición deseada se realiza mediante la transmisión de una señal pulsada con periodo de 20ms. El ancho del pulso determina la posición del servo. La relación entre el ancho del pulso y el ángulo depende del modelo del motor. Por ejemplo, algunos modelos responden con 0° a un pulso de 500 ms, y otros a un pulso de 1000 ms. En general, en todos los modelos un pulso entre 500-1000µs corresponde con 0°, un pulso de 1500µs corresponde con 90° (punto neutro) y un pulso entre 2000-2500µs corresponde con 180°. Por tanto, variando la señal en microsegundos podemos disponer de una precisión teórica de 0.18°-0.36°, siempre que la mecánica del servo acompañe.



**Esquema y código de la función:**




```
#include <Servo.h>                    //Librería del servo para la puerta
int pos;                                        //Posicion en angulos del servo
Servo servo;

void setup()
{
    Serial.begin(9600);
    servo.attach(9);
}

void loop()
{
void puerta (int correcto)
{
 if (correcto==0)
 {
 for(pos = 0; pos <= 180; pos += 1) // goes from 0 degrees to 180 degrees
  {                                  // in steps of 1 degree
    servo.write(pos);               // tell servo to go to position in variable 'pos'
    delay(20);                       // waits 15ms for the servo to reach the position
  }
  delay (2000);
    for(pos = 180; pos>=0; pos-=1)     // goes from 180 degrees to 0 degrees
  {
    servo.write(pos);               // tell servo to go to position in variable 'pos'
    delay(20);                       // waits 15ms for the servo to reach the position
  }
  Serial.println("SERVO");
 }
}
}
```

### 2.4. Comunicación serie. Arduino y Ordenador

La comunicación entre Arduino y ordenador se puede realizar de diferentes maneras: inalámbricas a través de conexión WiFi o Bluetooth y de forma cableada a través del protocolo Serie por medio de un puerto USB integrado.

Para ello, hemos desarrollado una aplicación que nos permite inicializar y finalizar el uso de la placa, además de realizar registros a través de un fichero de texto.
Esta aplicación se ha desarrollado a través de Dev-C++, que permite realizar programación en C entre muchos otros tipos de programación.

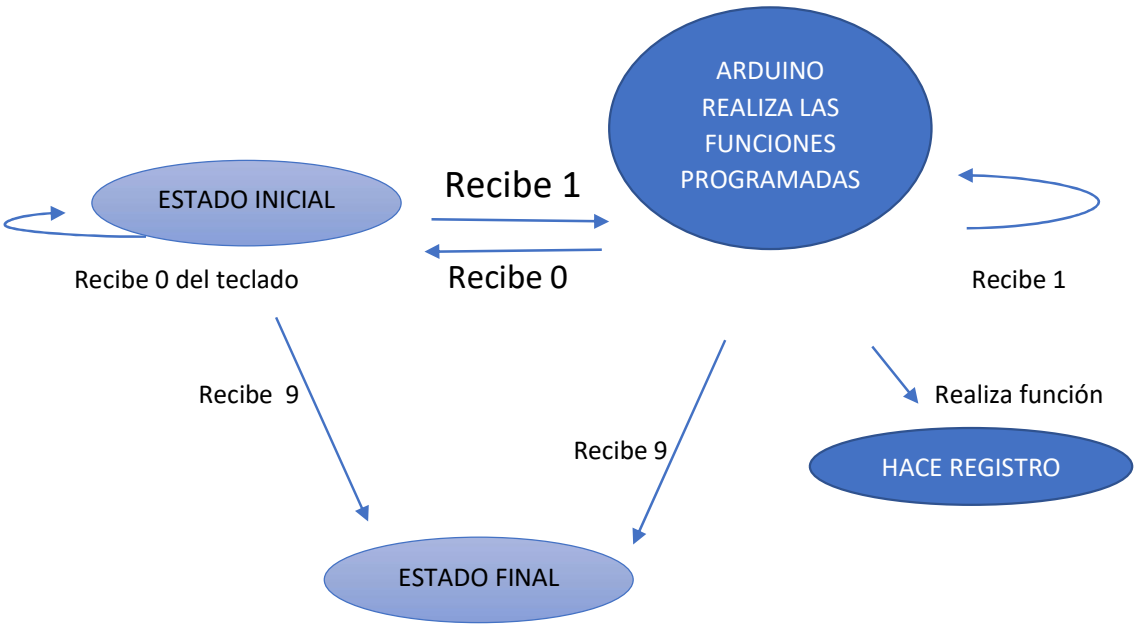
Todo este código de intercambio de información entre Arduino y Dev-C++ será detallado con mayor intensidad en el punto tres dedicado al diseño del software.

## 3. DISEÑO DEL SOFTWARE.

En nuestro caso, hemos desarrollado dos plataformas hardware que se están comunicando mediante un puesto serie con USB.

### 3.1. Aplicación de gobierno y desarrollo de ficheros. Dev-C++.

En primer lugar, la primera plataforma se encarga de desarrollar una aplicación de gobierno y realización de ficheros. Esa aplicación está desarrollada en Dev-C++ y su funcionamiento se describe en el esquema siguiente:




Como se puede observar, al pulsar el 1, se activa la placa y al pulsar 0, se desactiva. Por otra parte para finalizar la aplicación se pulsaría el 9 en el teclado.

Al realizar una acción la aplicación recibe la orden en texto y la agrega al fichero agregando también la fecha y hora en la que se ha realizado.

A continuación, pasamos a presentar el código:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "SerialPort.h"
#include "SerialPort.C"
#define MAX_DATA_LENGTH 255
char accion[10];
// Funciones prototipo
void autoConnect(SerialPort *arduino,char*);
int main(void)
{
//Arduino SerialPort object
SerialPort *arduino;
// Puerto serie en el que está Arduino
char* portName = "\\\\.\\COM3";
// Buffer para datos procedentes de Arduino
char incomingData[MAX_DATA_LENGTH];
// Crear estructura de datos del puerto serie
arduino = (SerialPort *)malloc(sizeof(SerialPort));
// Apertura del puerto serie
Crear_Conexion(arduino,portName);
autoConnect(arduino,incomingData);
return 0;
}
void autoConnect(SerialPort *arduino,char *incomingData)
{
char sendData = 0;
int readResult;
// Espera la conexión con Arduino
while (!isConnected(arduino))
{
Sleep(100);
Crear_Conexion(arduino,arduino->portName);
}
//Comprueba si arduino está connectado
if (isConnected(arduino))
{
printf ("Conectado con Arduino en el puerto %s\n",arduino->portName);
}
// Bucle de la aplicación
printf ("0 - OFF, 1 - ON, 9 - SALIR\n");
while (isConnected(arduino) && sendData!='9')
{
sendData = getch();
writeSerialPort(arduino,&sendData, sizeof(char));
readResult=readSerialPort(arduino,incomingData,MAX_DATA_LENGTH);
if (readResult!=0)
{
 time_t t;
struct tm *tm;
  char fechayhora[100];
```

```
  t=time(NULL);
  tm=localtime(&t);
  strftime(fechayhora, 100, "%d/%m/%Y %H:%M %S, tm);
    accion=incomingData;
                            FILE *registro;
                            registro=fopen ("./registo.txt","at");
                            if (registro==NULL)
                            printf ("No se encuentra el fichero\n");
                            else
                            fprintf (registro,"%s
%s\n",accion,fechayhora);
    fclose(registro);
}
sleep(10);
}
if (!isConnected(arduino))
printf ("Se ha perdido la conexión con Arduino\n");
}
```

### 3.2. Aplicación de microcontrolador. Arduino