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

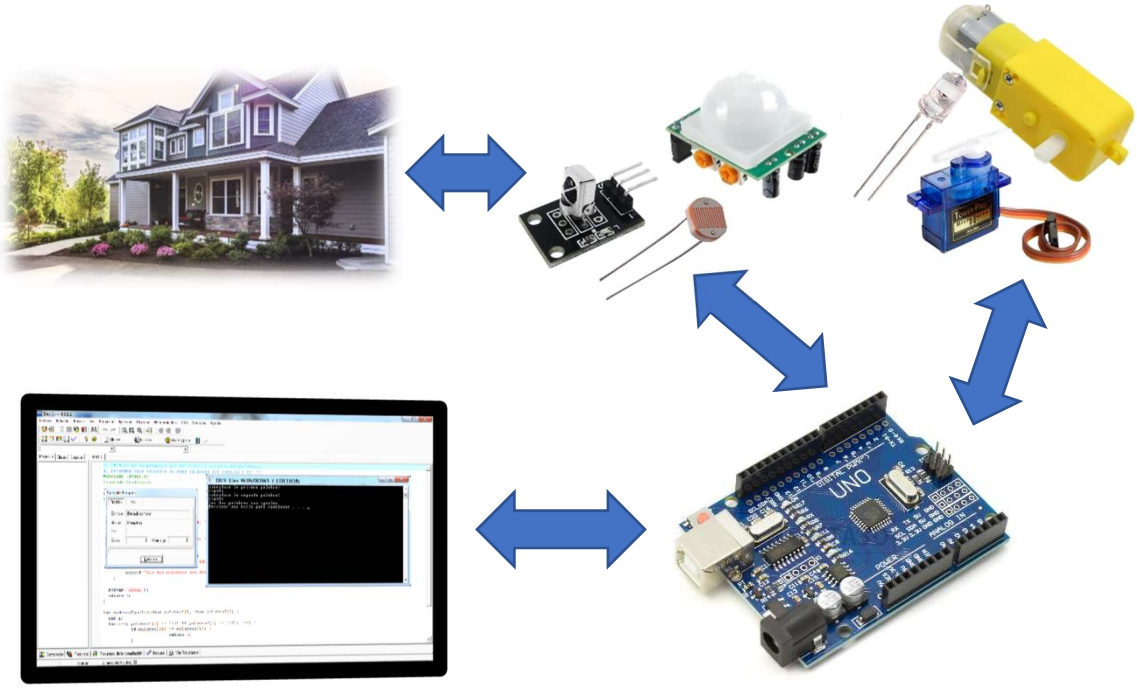
Índice

1. INTRODUCCIÓN.

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/

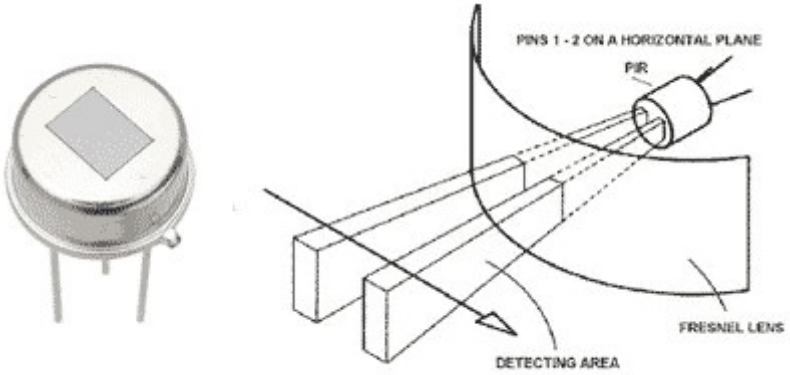
Los sensores infrarrojos pasivos (PIR) son dispositivos para la detección de movimiento. Son frecuentemente usados en juguetes, aplicaciones domóticas o sistemas de seguridad.

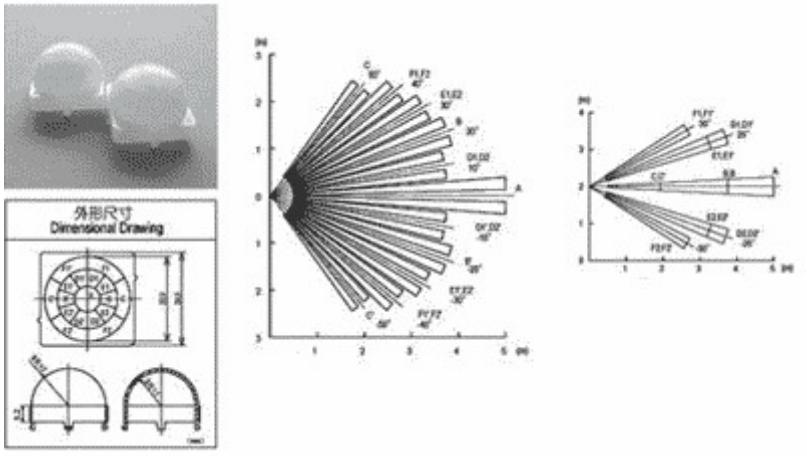
Estos se basan en la medición de la radiación infrarroja. Todos los cuerpos (vivos o no) emiten una cierta cantidad de energía infrarroja, mayor cuanto mayor es su temperatura. Los dispositivos PIR disponen de un sensor piezo eléctrico capaz de captar esta radiación y convertirla en una señal eléctrica.

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.

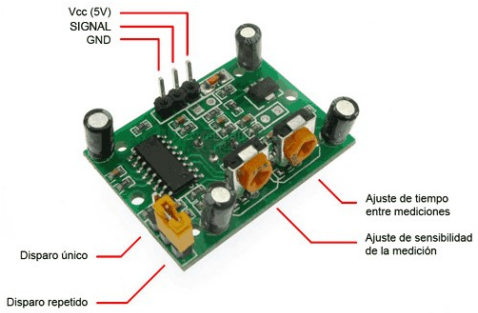
El otro elemento es la óptica del sensor, una cúpula de plástico formada por lentes de fresnel, que divide el espacio en zonas, y enfoca la radiación infrarroja a cada uno de los campos del PIR y así capta un promedio de la radiación infrarroja del entorno.

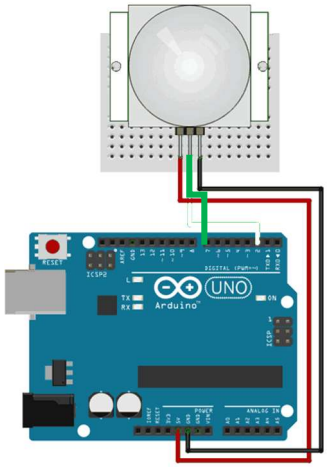
De esta forma, si un objeto atraviesa uno de los campos se genera una señal eléctrica diferencial, que es captada por el sensor, y se emite una señal digital.
Por ello este sensor lo usamos para detectar presencia en la habitación.





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.

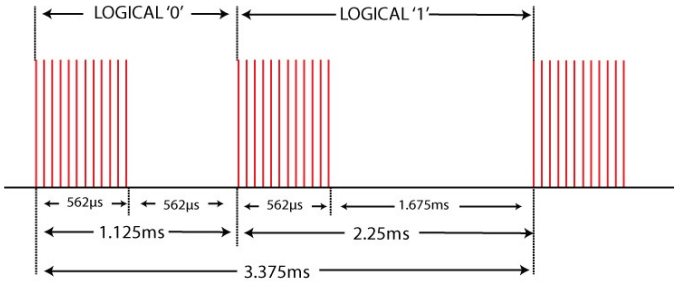
El alcance es limitado, típicamente inferior a 3m. La distancia depende fuertemente del ángulo de emisión, disminuyendo rápidamente a medida que nos desviamos de la dirección frontal.

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.

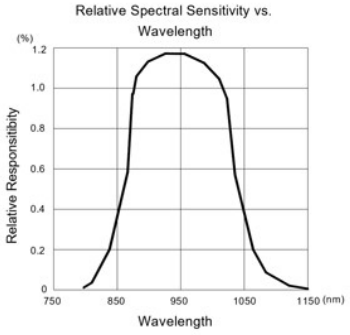
Uno de los protocolos más habituales, que es el que emplearemos con Arduino, es el protocolo NEC, que emplea una onda portadora de 38 kHz y modulación por distancia de pulsos (PDM Pulse Distance Modulation). La transmisión comienza con una señal de 9ms, seguido de un espacio de 4.5ms.

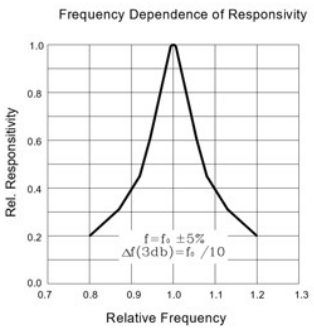
La onda portadora tiene un periodo de 26µs, y la señal transmitida distingue entre 0 y 1 por la duración de los pulsos, siendo:

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.

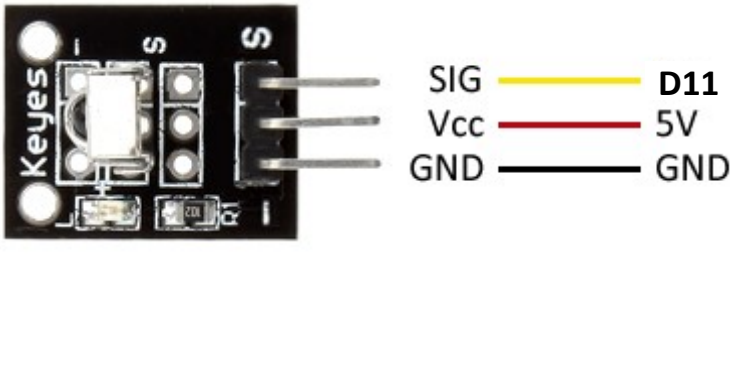


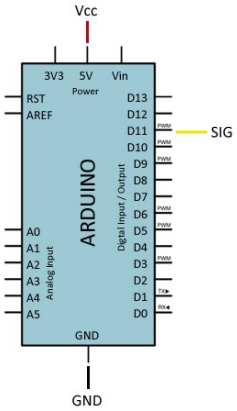
Finalmente para recibir la señal de control se emplea un receptor específico de tres de terminales, que consisten en un sensor infrarrojo que incluyen un demodulador en la banda de 36-38kHz, un filtro PCM (Pulse Code Modulation) y preamplificación rechazo de luz ambiental.





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;
}
}
```

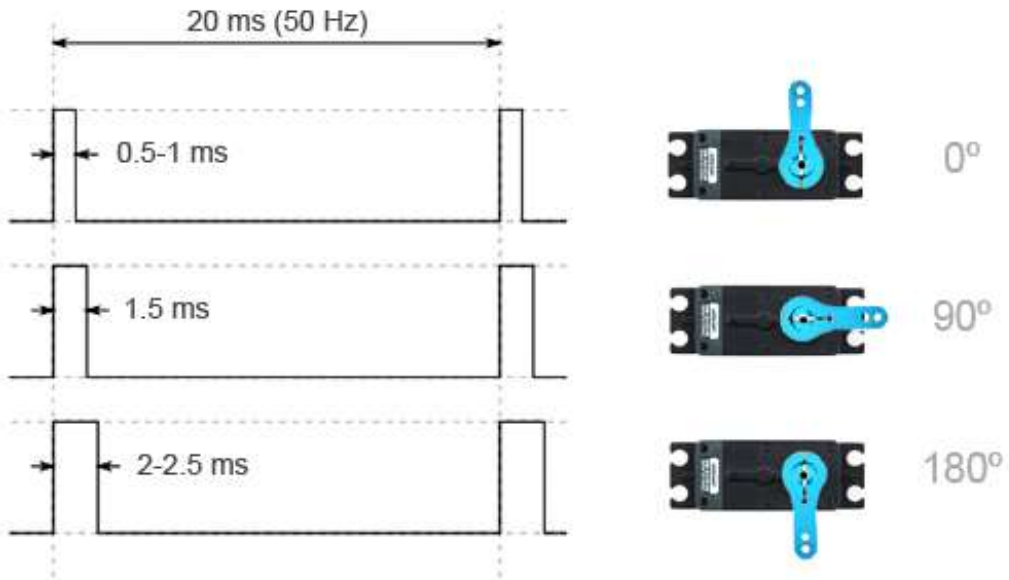
2.3. Desbloqueo de la puerta. Servomotor SG90.

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

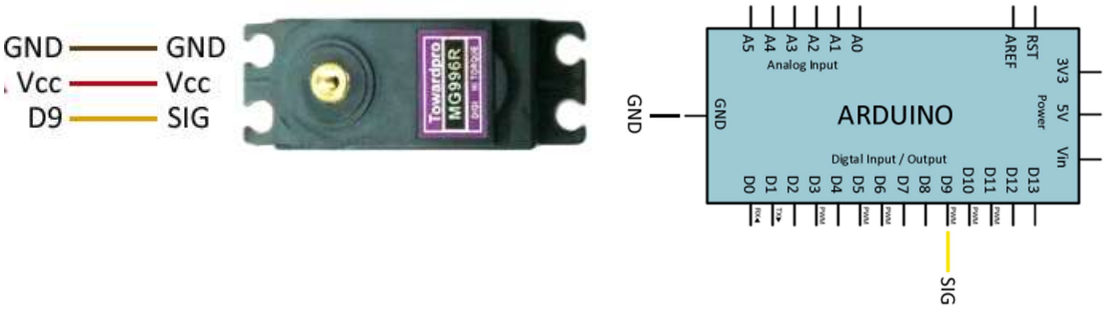
Un servomotor es un accionador ampliamente empleado en electrónica. A diferencia de otros tipos de motores en los que se controla la velocidad de giro, en un servo se indica el ángulo deseado y el servo se encarga de posicionarse en este ángulo.

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. Receptor de cantidad de luz. LDR.

https://www.luisllamas.es/medir-nivel-luz-con-arduino-y-fotoresistencia-ldr/

Un fotoresistor (LDR) es un elemento el cual varia su resistencia en función de la luz que recibe. Suele estar formado por sulfuro de cadmio, un material semiconductor, que absorbe los fotones y los lleva a la banda de conducción reduciendo la resistencia (por lo que un LDR disminuye su resistencia a medida que aumenta la luz que incide en él).

El tiempo de reacción ante los cambios de luz varía dependiendo del modelo, aunque por lo general es relativamente lento (entre 20 y 100 ms). Esto hace que el sensor no sea capaz de registrar variaciones rápidas como las causadas por la corriente alterna en una luz artificial y dota al sensor de una gran estabilidad.

La relación entre la luminosidad y la resistencia del componente se representa con esta ecuación: $$\frac{I}{I_0} = \left(\frac{R}{R_0}\right)^{-gamma}$$

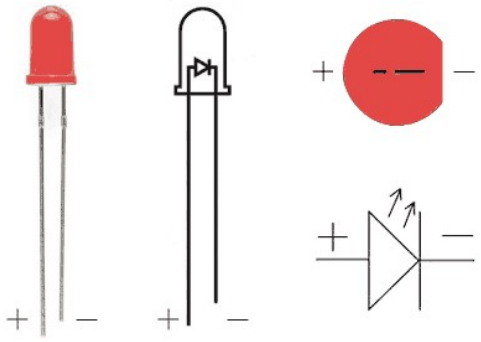
Donde R0 es la resistencia conocida a una intensidad también conocida (I0), y gamma es una constante que representa la pendiente de la gráfica con un valor habitual entre 0.5 y 0.8. El código es el siguiente:

```
#define PIN_ANALOGICO A0 // El pin del LDR es el A0
#define ESPERA_LECTURAS 1000 // tiempo en milisegundos entre lecturas de la
intensidad de la luz

long cronometro_lecturas=0; //Ponemos el cronometro a 0
long tiempo_transcurrido; //Tiempo que pasa
unsigned int luminosidad; //Luminosidad
float coeficiente_porcentaje=100.0/1023.0;

void setup()
{
}

int luz (void)   //Funcion de la cantidad de luz
  {
    int porcentaje;
    int dia=0;
    tiempo_transcurrido=millis()-cronometro_lecturas;
    if(tiempo_transcurrido>ESPERA_LECTURAS)
    {
    cronometro_lecturas=millis();
    luminosidad=analogRead(PIN_ANALOGICO);
    porcentaje= luminosidad*coeficiente_porcentaje;
      if (porcentaje>25) //Si el porcentaje es mayor al 25%, se supone que es
de día
        dia=1;  //Es de día
    }
     return dia;
   }

void loop()
{
 luz (void);
}
```

2.5. Iluminación de la habitación. LED.

https://www.luisllamas.es/encender-un-led-con-arduino/

Un LED es un diodo (unión de dos materiales semiconductores) que emite luz al ser atravesado por una corriente. La diferencia de dopado de los materiales semiconductores del diodo, hace que el paso de la corriente solo sea posible en una dirección, es decir, los LED tienen polaridad.



Los LED poseen una tensión de polarización directa a partir de la cual la corriente puede circular por él y se dice que está polarizado. En el momento en el que se supera este valor se genera una gran corriente causada por la baja resistencia del diodo, que podría destruirlo si no se usa una resistencia adecuada para limitar el paso de la corriente.

Para conocer el valor de la resistencia que se debe utilizar se usara esta fórmula obtenida a partir de la ley de Ohm.

$$R = \frac{V_{cc} - V_d}{I_{nominal}}$$

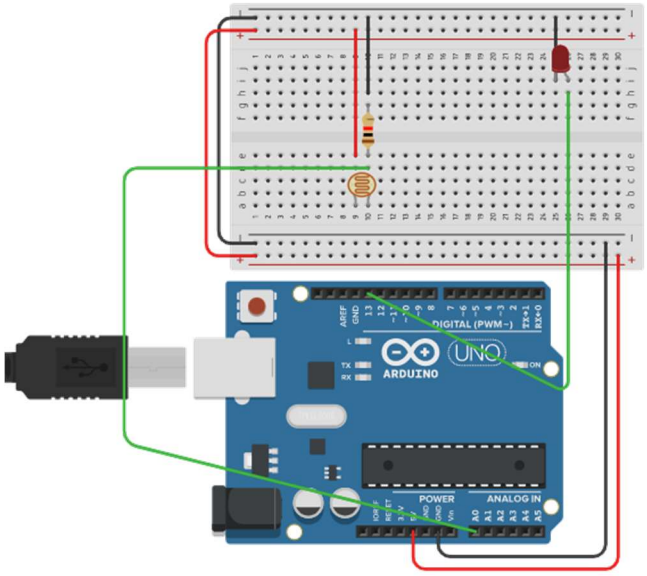
- Vcc: tensión de alimentación, la cual podemos saber dependiendo del modelo de arduino (5V o 3.3V).
- Vd: tensión de polarización directa.
- In: corriente nominal del LED.

Estas dos últimas dependen de los materiales, el color, la luminosidad…

En cuanto a la parte de programación, este es el código de uno de los proyectos más sencillos con un LED.

Código de la función de LED y Esquema de LDR y LED:

```
const int ledPIN = 9;

void setup() {
  Serial.begin(9600);    //iniciar puerto serie
  pinMode(ledPIN , OUTPUT);  //definir pin como salida
}

void loop(){
  digitalWrite(ledPIN , HIGH);   // poner el Pin en HIGH
  delay(1000);                   // esperar un segundo
  digitalWrite(ledPIN , LOW);    // poner el Pin en LOW
  delay(1000);                   // esperar un segundo
}
```



2.6. Controlador de tiempo.

Para hacer un reloj con arduino se necesita el uso de la librería Time.h la cual introduce las funciones necesarias. Este tipo de librería establecen por defecto una fecha y hora concreta, por lo que para ponerlo en hora se necesita usar la función "setTime()" e indicar los parámetros que se deseen establecer como referencia para el inicio del programa (hora, minutos, día, mes…).

El mayor inconveniente de hacer un reloj en arduino mediante software es que al reiniciar la placa la hora vuelve a ser la establecida como fecha de referencia (ya sea por defecto o mediante la función "setTime()").

Si se quiere obtener la fecha y hora actual se debe usar la función "now()", que devuelve un dato de tipo time_t con el que se puede trabajar en otras funciones. En este ejemplo, se puede ver cómo funciona esta librería.

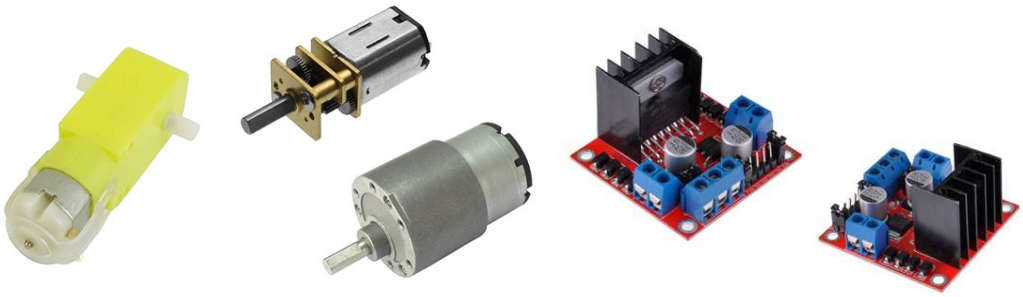
```
int hora (void)
{
  dia=0;
  fecha = now();
  if ((hour(fecha))>=8 && (hour(fecha))<20)
  dia=1;
  else
  dia=0;
  return dia;
}
```

Se le da el nombre "fecha" a la función "now()" y se comprueba si la hora está comprendida entre las 8 y las 20 horas para dar el valor correspondiente a la variable "dia".

2.7. Movimiento de persiana. Motorreductor(Geared Down)

https://www.luisllamas.es/tipos-motores-rotativos-proyectos-arduino/

Motor de corriente continua con un reductor que aumenta el par y reduce la velocidad. Este tipo de motores son habituales para controlar ruedas de vehículos, robots…



Arduino no es lo suficientemente potente como para mover motores, por lo que necesita de drivers para que estos se encarguen del proceso. Un ejemplo de estos drivers es el L298N que puede variar la dirección y la velocidad de giro de motores de corriente alterna.

```
#define dir1PinL  2    //Motor direction
#define dir2PinL  4    //Motor direction
#define speedPinL 6    // Needs to be a PWM pin to be able to control motor
speed
#define speedPinR 5

/*motor control*/
void go_Advance(void)  //Forward
{
  digitalWrite(dir1PinL, HIGH);
  digitalWrite(dir2PinL,LOW);

}

void go_Back(void)  //Reverse
{
  digitalWrite(dir1PinL, LOW);
  digitalWrite(dir2PinL,HIGH);

}
void stop_Stop()    //Stop
{
  digitalWrite(dir1PinL, LOW);
  digitalWrite(dir2PinL,LOW);
}

/*set motor speed */
void set_Motorspeed(int speed_L,int speed_R)
{
  analogWrite(speedPinL,speed_L);
  analogWrite(speedPinR,speed_R);
}
```

```
//Pins initialize
void init_GPIO()
{
      pinMode(dir1PinL, OUTPUT);
      pinMode(dir2PinL, OUTPUT);
      pinMode(speedPinL, OUTPUT);
  pinMode(speedPinR, OUTPUT);
      stop_Stop();
}

void setup()
{
      init_GPIO();
      go_Advance();//Forward
  set_Motorspeed(255,255);
  delay(5000);

  go_Back();//Reverse
  set_Motorspeed(255,255);
  delay(5000);

  stop_Stop();//Stop

}

void loop(){
}
```

2.8. 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.

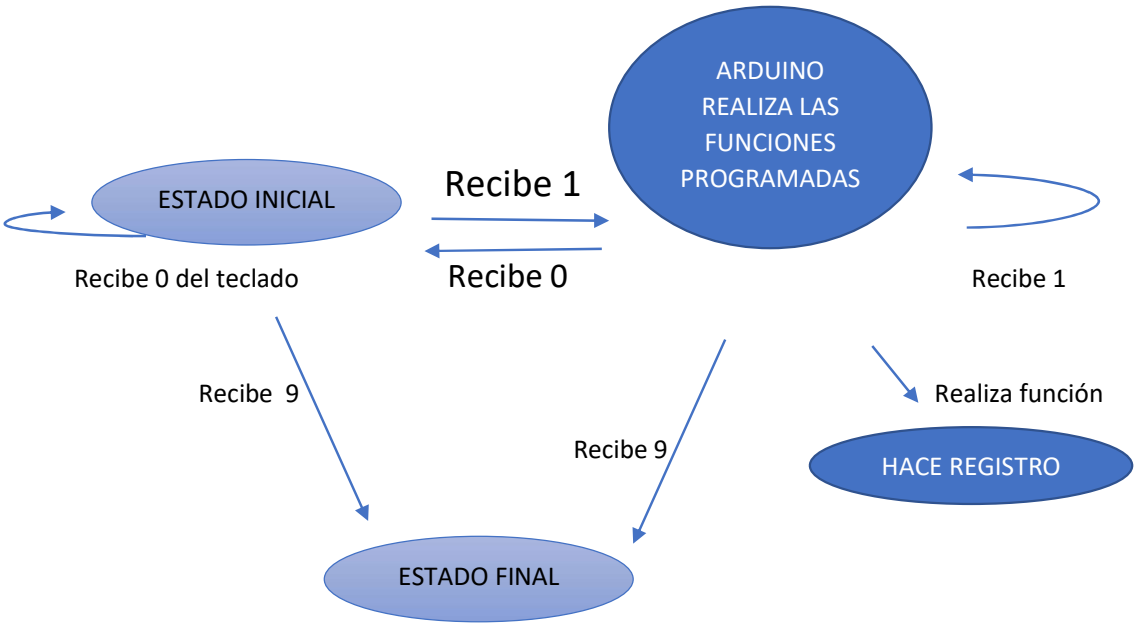
Todo este código de intercambio de información entre Arduino y Dev-C++ será detallado con mayor intensidad en el punto tres dedicado al diseño del software.

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.

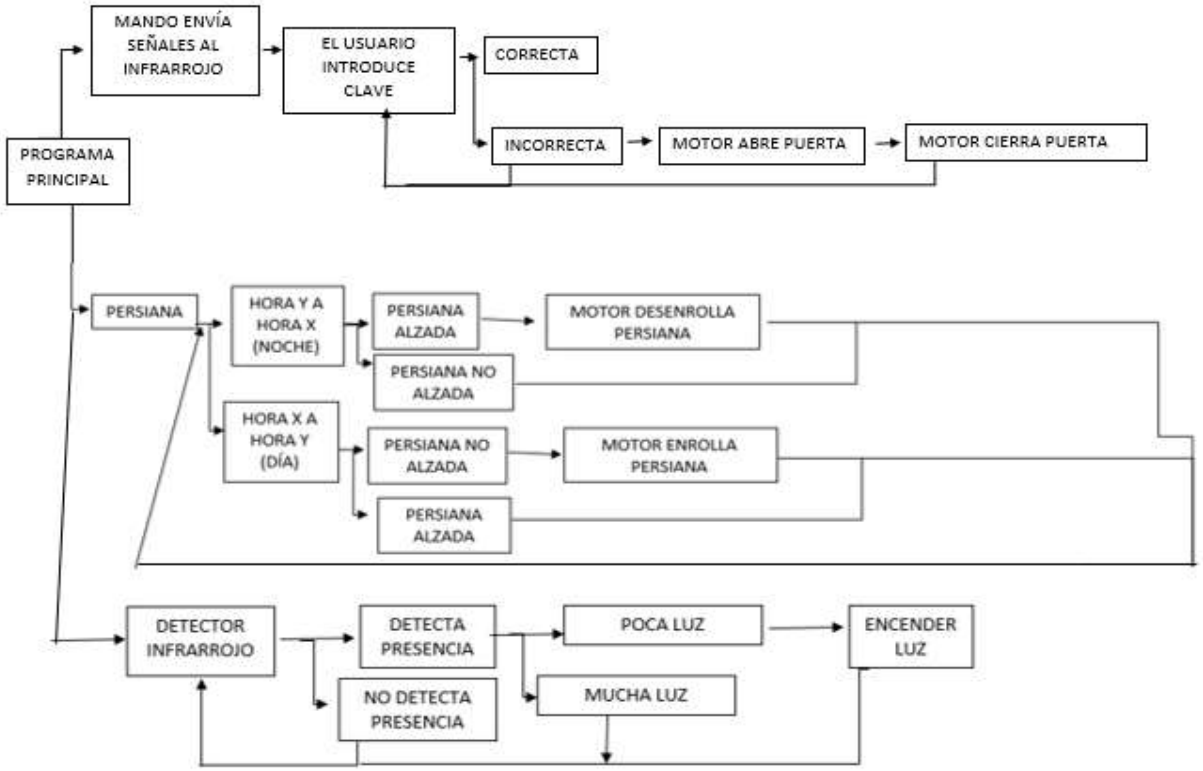
A continuación, pasamos a presentar el código:

```
#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.

La aplicación de adquisición de medidas y control de los actuadores está desarrollada con Arduino y su funcionamiento se puede observar en el siguiente esquema:



El código de Arduino es el siguiente:

```
#include <TimeLib.h>                //Libreria de la hora
#include <IRremote.h>               //Libreria del receptor IR
#include <Servo.h>                  //Librería del servo para la puerta
#define dir1PinL  2                 //Direccion de motor
#define dir2PinL  4                  //Direccion de motor
#define speedPinL 6                  // Para la velocidad
#define speedPinR 5                   //Para la velocidad
#define CINCO 765
 int estado;                          //Para la activación y desactivación de
la placa
 char dato1;                          //Para la activación y desactivación de
la placa



////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////
///Funciones
int descodificar(decode_results *);  //Descodifica la señal IR
int clave (int);                    //Comprueba si la clave es correcta
int ldr (void);                      // Mide la cantidad de luz
int detector_presencia (void);       //Observa si hay alguna perturbación en
la habitación
void bombilla (int, int);            //Enciende el LED
```

```
void puerta (int);                    //Acciona el servo motor que bloquea la
puerta
int hora (void);                      //Informa sobre el momento del día
void avanza (void);                   //Desenrolla el motor
void retrocede(void);                 //Enrolla el motor
void para (void);                     //Para el motor
void set_Motorspeed(int,int);         //Velocidad de los motores
void init_GPIO(void);                 //Inicializa los motores
void motor (int);                     //Acciona el motor hacia delante o atrás
segun quiera la persiana

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////
int movimiento;                                //Hay movimiento
int numero;                                     //Codigo del boton del mando
int dato;                                       //Para activar o desactivar la
placa
int error;                                      //Para accionar o no el
servomotor
int LDR_Pin = A0;                               //LDR en el pin analógico A0
int estadopir;                                  //Detección o no de
presencia
time_t fecha;                                   // Declaramos la variable
del tipo time_t
int luz, tiempo, correcto;                      //Indica si es de día por la
luz, la hora y si el botón pulsado es el correcto
int repetir1=0, repetir2=0;                     //Para que no se mueva el
motor constantemente
int pos;                                        //Posicion en angulos del
servo
int dia;                                       //Indica si es de día o no

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////
//Posiciones de los sensores
const int pir= 7;                               //PIR en pin 7
const int led= 12;                              //LED en pin 12
int RECV_PIN = 11;                                    //IR en pin 11
IRrecv irrecv(RECV_PIN);                        //Recibe del receptor IR
decode_results results;                        //Descodifica resultados
Servo servo;                                   //El servo de la puerta

////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////
//Estado inicial
void setup()
{
  Serial.begin(9600);
  setTime(19, 59, 45, 13, 12, 2016);  // Establecemos la fecha
  pinMode(pir, INPUT);               //El pir es un dispositivo de entrada
  pinMode (led, OUTPUT);             //El led es un dispositivo de salida
  irrecv.enableIRIn();             // Empezamos la recepción  por IR
  servo.attach(9);                 //Pin del servo

}

////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////
int dump(decode_results *results)      //Descodificación de los resultados
```

```
{
 dato=(results->value);
 return dato;                          //Nos devuelve el código del botón
pulsado
}

////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////
//MANDO
int clave (void)
{
 error=1;
 if (irrecv.decode(&results))
  {
   numero=dump(&results);
   if (numero==CINCO) //5             //Si el pulsado es el cinco(en este
caso), no hay error. Si es otro botón da error
    error=0;
   irrecv.resume();                   //Comienza la recepción de nuevo
  }
  delay(300);
  return error;                       //Devuelve error o no error
}

////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////
//LDR
int ldr (void)   //Funcion de la cantidad de luz
  {
dia=0;
int LDRReading = analogRead(LDR_Pin);
if (LDRReading>10)                        //Si supera esa cantidad de luz, es
de día, si no, no.
 dia=1;
delay(250);
return dia;                               //Devuelve si es de día o no
}

////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////
//PIR
int detector_presencia (void)
  {
  int presencia;
  int valor;
  valor= digitalRead(pir);                //Leer pir
    if (valor == HIGH )                   //Si detecta presencia, presencia=1.
       presencia=1;
    else
      presencia=0;                        //Si no detecta presencia,
presencia=0.
    return presencia;                     //Devuelve la presencia
  }

////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////
//HORA
int hora (void)
{
```

```
  dia=0;
  fecha = now();
//Mira la hora que es
  if (((hour(fecha))>=8 && (hour(fecha))<12)|| ((hour(fecha))>=18 &&
(hour(fecha))<20))     //Si está entre las 8 y las 12 ó entre las 18 y las
20, se considera de día para alzar la persiana
  dia=1;
  else
//Si no, se considera de noche
  dia=0;
  return dia;
}

////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////
//LED
void bombilla (int luz, int movimiento)
{
 if (luz==0 && movimiento==1)                        //Si es de noche y
detecta movimiento
  {
  digitalWrite(led,HIGH);         // Enciende el led
  delay (2000);                   //Espera 20 segundos para apagar la luz si
no detecta movimiento
  Serial.println("LED");          //Manda el mensaje para el registro
  }
  else
  digitalWrite(led,LOW);          //Se apaga el led
}

////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////
//Persiana/motor

void go_Advance(void)               //Despliega la persiana
{
  digitalWrite(dir1PinL, HIGH);
  digitalWrite(dir2PinL,LOW);


}

void go_Back(void)                    //Recoge la persiana
{
  digitalWrite(dir1PinL, LOW);
  digitalWrite(dir2PinL,HIGH);

}
void stop_Stop()    //Stop            //Para el motor
{
  digitalWrite(dir1PinL, LOW);
  digitalWrite(dir2PinL,LOW);
}


void set_Motorspeed(int speed_L,int speed_R)    //Velocidad del motor
{
  analogWrite(speedPinL,speed_L);
  analogWrite(speedPinR,speed_R);
```

```
}

//Pins initialize
void init_GPIO()                      //Inicializar la salida de los pines
para los métodos
{
  pinMode(dir1PinL, OUTPUT);
  pinMode(dir2PinL, OUTPUT);
  pinMode(speedPinL, OUTPUT);
  pinMode(speedPinR, OUTPUT);
  stop_Stop();
}

void motor (int tiempo)                         //Función del motor
{
  if (tiempo==1 && repetir2==0)                  //Si es de día y ya no se ha
recogido
  {
  init_GPIO();
  go_Back();                                     //Se enrolla
  set_Motorspeed(255,255);
  delay(5000);
  Serial.println("AVANZA");                      //Manda el mensaje para el
registro
  stop_Stop();
  repetir2=1;
  repetir1=0;
  }
  else
  if (tiempo==0 && repetir1==0)                  //Si es de noche y no se ha
desenrollado
  {
  init_GPIO();
  go_Advance();                                  //Se desenrolla
  set_Motorspeed(255,255);
  delay(5000);
  Serial.println("RETROCEDE");                  //Manda el mensaje para el
registro
  stop_Stop();
  repetir1=1;
  repetir2=0;
  }
}

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////
///Servo puerta
void puerta (int correcto)
{
 if (correcto==0)                          //Si el boton pulsado es el
correcto
 {
 for(pos = 0; pos <= 180; pos += 1)      // Se mueve de 0 a 180 grados
  {
    servo.write(pos);                   //Se lo manda al servo
    delay(20);                          // Espera 15ms
  }
  delay (2000);
    for(pos = 180; pos>=0; pos-=1)     //Va de 180 grados a 0 grados
```

```
  {
    servo.write(pos);                  //Se lo manda al servo
    delay(20);                         // Espera 15ms
  }
  Serial.println("SERVO");             //Manda el mensaje para el registro
 }
}

////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////
void loop()
{

  if (Serial.available() > 0)            //Si recibe por el puerto serie
  {
  dato1=Serial.read();                   //Lo lee
  if (dato1=='1')                        //Si es uno
  estado=1;                              //Estado es uno
  else
  if (dato1=='0')                       //Si es 0
  estado=0;                             //Estado es 0
  }

switch (estado)
{
case 0:                                  //Si el estado es 0, no hace nada
{
  estado=0;
    break;
}

case 1:                                  //Si es estado es 1, activa las
funciones
{
tiempo=hora ();
motor (tiempo);
luz=ldr ();
movimiento=detector_presencia ();
bombilla (luz,movimiento);
correcto= clave();
puerta(correcto);
estado=1;
}
}
}
```

4. FUTURAS APORTACIONES AL TRABAJO.

Al realizar este trabajo como una iniciación a la programación, se podrían añadir nuevas aportaciones como, por ejemplo:

- Que la persiana no solo vaya con la hora del día sino, además, con el LDR.
- Que la puerta no se abra solo al introducir un dígito, aumentando la seguridad con una clave.
- Que el LED no funcionase con el LDR y con el sensor de presencia, sino en función de la persiana, que ya lleva integrado en su función el LDR.
- Incluir nuevos componentes como sensores de temperatura que active un motor como ventilador de techo.
- Pulsadores e interruptores que también permitan el gobierno por parte del usuario en la habitación.

5. CONCLUSIÓN.

Después de haber desarrollado un trabajo como este hemos concluido en que la satisfacción por nuestra para aumenta las expectativas creadas por el escaso tiempo que hemos tenido para desarrollarlo ya que los contenidos necesarios y claves para realizarlo se explican al final del cuatrimestre en el horario de clase.

Por ello, y aunque creemos que hay cosas por mejorar, hemos aprendido como funcionan componentes de hardware, su conexión, etc.

Además nos ha permitido desarrollar nuestra originalidad, nuestra lógica y nuestra capacidad de relacionar ideas y conceptos en el mundo de la programación.

Como complejidades nos hemos encontrado sobre todo con la comunicación serie, ya que no abarcábamos ese espacio y nos ha hecho investigar más acerca de ese asunto.

En definitiva, este trabajo nos ha permitido realizar algo práctico y visual que nos permite ver el uso de esta asignatura para nuestro futuro tanto académico como profesional.