Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
Domótica con Herramientas de Software Libre

Alonso Montero Fuentes (B34439)
Brian Morera Madriz (B34734)
Isaac Eduardo Gómez Sánchez (B32919)
Grupo 01

8 de mayo de 2016

Índice

Índice de figuras

Índice de tablas

1. Introducción

En la actualidad los sistemas de control automático han tomado gran importancia tanto en el mundo de la ciencia como en el ámbito comercial. Dichos sistemas suponen una gran ventaja, pues permiten realizar tareas durante un largo tiempo sin necesidad de interevención humana, además de la requerida para confeccionar y programarlos. Dentro de tales sistemas los que corresponden a la domótica han ido ganado cada vez más espacio en el mercado, pues suponen eficiencia, confort y seguridad.

Las compañías dedicadas al domótica ofrecen productos que abarcan diferentes servicios de control automático. Algunos de estos servicios son: automatización para el control de iluminacion, escenarios, audio distribuido, control de persianas; la Seguridad, dando control de acceso, alarma contra robo, sistema contra fuego; Comunicación, entre los mismos residentes dentro del hogar, entre la casa y el usuario por medio del celular; Ambiente, como el riego de jardín o aire acondicionado. A pesar del gran beneficio y eficiencia que suponen estos sistemas, no cualquiera puede tener acceso a ellos, ya que el precio de estos productos los convierten más bien en un lujo para las personas de hogares humildes.

Ya que la domótica supone tremendo beneficio para todo aquel que tenga un sistema de esta clase, es el propósito de este proyecto el de desarrollar un sistema de bajo costo y a la vez bajo la modalidad DIY(hágalo usted mismo), utilizando herramientas de software libre. En este proyecto se implementa un sistema de domótica de bajo coste que permite al usuario tener control manual de diversos actuadores a través de un menú único. Para este propósito se emplearon microcontroladores Arduino, los cuales trabajan autónomamente empleando sensores y actuadores para su funcionamiento. Dicha autonomía fue posible gracias a la programación que se le implementó al microcontrolador, empleando el ambiente de desarrollo integrado proporcionado por Arduino. Además se emplea una unidad central de procesamiento, conformada por un Raspberri Pi, la cual permite implementar el control manual.

En esta ocasión se emplean dos Arduinos que tienen adjudicada, cada uno, una funcionalidad específica. En uno de ellos se implementa un sistema que encienda o apague un LED de acuerdo a la lectura de un sensor ultrasónica, mientras que en el otro se implementa un sistema que controla el movimiento de un motor de acuerdo a la lectura de un termistor. Cada uno de estos sistemas de control se conecta al Raspberri Pi, el cual se programa para presentar un menú que permita al usuario imponer su orden sobre la acción que ya hubiera decidido el Arduino. Dicho menú presenta la posibilidad de encender o apagar cada actuador, así como de varia la sensibilidad previamente programada de cada sensor.

2. Nota Histórica

Para llevar a cabo este proyecto se siguieron las actividades planteadas en el cronograma que se presenta en la tabla 1. En él se muestran las actividades realizadas, con respecto a su semana e integrante del equipo.

En esta ocasión se divieron las actividades en semanas, específicamente 5, con el objetivo de realizar las actividades en un determinado tiempo. Dichas semanas abarcan desde el día 4 de abril hasta el 4 de mayo, día de la presentación oral.

Actividades	Semana 1	Semana 2	Semana 3	Semana 4	Semana 5
Elegir componentes que serán obtenidos	Alonso				
Programar la iluminación.(Arduino)	Brian	Brian			
Programar la ventilación.(Arduino)	Isaac	Isaac			
Programa vigilancia(R.Pi y Python)	Alonso	Alonso			
Programa de control manual (Diagrama de flujos)			Brian		
Programa de control manual (Menú)			Alonso		
Programa de control manual (Conexión a Arduinos)			Isaac		
Acceso remoto(openSSH)				Brian	
Diseño de casa modelo				Isaac	
Construccion de casa modelo				Alonso	
Pruebas					Brian, Isaac, Alonso

Tabla 1: Cronograma

3. Desarrollo Teórico y funcionamiento

3.1. Domótica

Se define domótica al conjunto de sistemas capaces de automatizar un inmueble (aportando servicios de gestión energética, seguridad, bienestar, comunicación y accesibilidad), los cuales, deben estar integrados por medio de redes interiores y/o exteriores de comunicación, cableadas o inalámbriacas, cuyo control goza de cierta ubicuidad desde dentro y fuera del recinto.

3.2. Especificaciones Generales

Los productos realizados por varios fabricantes pueden ser combinados entre sí, acarreando con esto la traquilidad de tener basto soporte de productos compatibles de cientos de empresa.

Garantizar el mantenimiento y las amplificiones futuras de la instalación con productos de total continuidad en el mercado y en constante evolución.

3.3. Arquitectura

La arquitectura, se define desde el punto de vista de dónde reside la inteligencia del sistema domótico.

Para este caso es centralizada, ya que un controlador central (en este caso el Rasberry Pi) decide el comportamiento de varios periféricos que a su vez procesan (si el controlador central se lo permite) la lectura de los sensores para generar la acción de los actuadores.

Existen otros tipos de arquitectura, sean distribuida y mixta.

3.4. Topología

La topología se refiere a la forma en que está diseñada la red, sea físicamente o lógicamente.

La topología empleada es estrella extendida, ya que del controlador central salen varios dispositivos (arduinos) que a su vez, de estos se despliega los sensores y actuadores. Otras topologías conocidas son estrella, anillo, bus, malla, árbol.

3.5. Enlaces

Se refiere en la manera en que los componenetes están conectados entre sí.

Una forma es por medio de **cableado**, por medio de: línea eléctrica, par trenzado, UTP. Además Inalámbrico, Óptico, Mixto.

3.6. Protocolo

Un protocolo es un conjunto de reglas normalizadas para la representación, señalización, autentificación y detección de errores necesario para enviar información a través de un canal de comunicación.

3.7. Diagrama Lógico

Indica los símbolos de los componentes bus utilizados y la conexión física (cableado) a la línea.

3.8. Diagama Funcional

Muestra la conexión funcional entre cada componente y los efectos que produce sobre los demás.

3.9. Bloques de Parámetros

Representan un resumen de componentes + aplicaciones + objetos de comunicación + parámetros.

3.10. Funcionamiento

El sistema se puede comportar de dos maneras: forma automáticamente o forma manual.

Cuando es de forma manual, el sistema acepta la condición dada sin imporatar las condiciones del ambiente existentes, por ende las medidas de los sensores. Esto es, apagar o encender un actuador un tiempo indefinido hasta que el usuario lo halla especificado o cambie el estado.

De manera automática, es importante conocer qué tan sensible se le dá los valores al sistema para que el actuador realice su acción.

Para la iluminación, esta se encenderá si y sólo si, es de noche y se detecta movimiento dentro de la habitación. Con respecto a la ventilación, cuando la temperatura es mayor a la deseada y se apagará cuando la temperatura descienda. Con respecto a la cámara, esta toma una fotografía cuando el usuario así lo requiera.

4. Instrucciones de Instalación y uso

4.1. Requerimientos

- Python 2.7
- Arduino IDE
- Raspberry Pi
- 2 x Arduino Uno

4.2. Preparación del Raspberry Pi

Para el proyecto se utilizó la distribución Raspbian como el sistema operativo que corre en el Raspberry Pi. Para instalar el sistema operativo, siga las instrucciones oficiales de instalación de https://www.raspberrypi.org/documentation/installation/installing-images/.

4.3. Preparación de los arduinos

Los arduinos deben estar conectados a los sensores y actuadores de la siguiente manera:

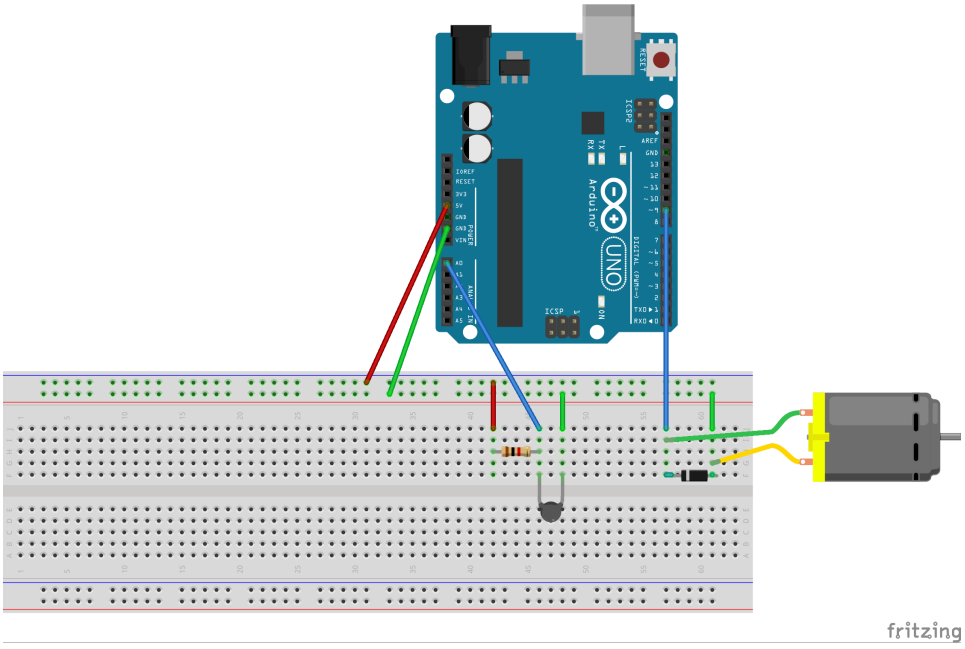

Figura 1: Circuito conexión del ventilador

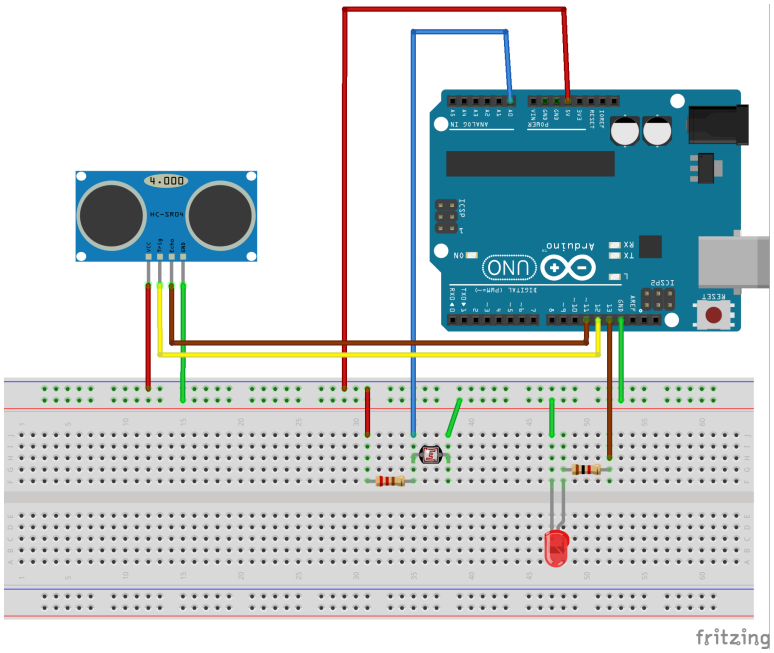

Figura 2: Circuito conexión de la iluminación

4.4. Obtención del código

Vaya al directorio donde quiere instalar el programa

```
pi@raspberrypi:~$ git clone https://github.com/alonsomonterofuentes/dhsl
```

4.5. Instalación del Sistema

Una vez dentro del sistema operativo, instale Arduino IDE. Puede hacer esto desde la terminal con:

```
pi@raspberrypi:~$ sudo apt-get install arduino
```

Cuando Arduino IDE esté instalado, ejecútelo:

```
pi@raspberrypi:~$ arduino
```

Ahora de ahí,abra el archivo iluminacion.ino. Conecte un arduino y presione ctrl+u para subir el programa al arduino. Observe en la esquina derecha inferior a cual puerto está conectado el arduino, por ejemplo

```
Arduino Uno on /dev/tty/ACM0
```

Conecte el segundo arduino y suba el programa ventilador.ino.
Asegurarse de que no se este subiendo al mismo puerto, esto puede ser verificado en la esquina inferior derecha del Arduino IDE

Cuando los arduinos estén conectados, se debe conectar una cámara web al Raspberry Pi

5. Instrucciones de uso (menú)

Una vez que ambos arduinos estén conectados, y con su programa subido, vaya el directorio donde clonó el repositorio y corra el comando

```
pi@raspberrypi:~/dhsl$ python menu.py
```

Esto correrá el menu principal del programa.

En la opción de Configurar Servidor se encontrán las opciones de arduino1 y arduino2. Estos representan el arduino para iluminación y ventilación respectivamente. Estos deben ser configurados con sus puertos respectivos. Además de estas opciones de encontrarán las opciones para Temperatura:

- max : Representa el valor en grados Centígrados sobre el cual se encederá el ventilador en modo automático.
- controlmanual: 1 representa control manual activado, 0 representa control automático activado.

Si se activa el control manual, debe además establecerse el estado del abanico, esto se encuentra en la opción de abanico:

- estado: Toma valores de 1 para encendido y 0 para apagado

Para controlar la iluminación se selecciona iluminación, sus opciones son:

- max : Representa el valor sobre el cual se encederá el bombillo en modo automático.
- controlmanual : 1 representa control manual activado, 0 representa control automático activado.

Si se activa el control manual, debe además establecerse el estado del bombillo, esto se encuentra en la opción de bombillo:

- estado : Toma valores de 1 para encendido y 0 para apagado

Si se selecciona control automático, es importante darle un valor a la distancia bajo la cual el sensor ultrasónico leerá movimiento, esto se hará en la opción Ultrasonico:

- sensibilidad: Toma valores entre 0 y 500 centímetros.

Al terminar de configurar las opciones se debe seleccionar la opción Escribir a arduino para que las opciones tomen efecto.

Si se desea tomar una foto se debe seleccionar la opción Tomar Foto e ingresarle el nombre que quiere que tenga la foto.

6. Referencias

Referencias

[1] Arduino.c.c. (2016). *Arduino-Introduction*. Recuperado el 5 de abril del 2016 desde: https://www.arduino.cc/en/Guide/Introduction

[2] Chris Liechti. *Python. pyserial (versión 3.0.1)*. [software]. Disponible en: https://pypi.python.org/pypi/pyserial

[3] Colegio de Ingenieros Especialistas de Córdoba. (2011). *Guía de contenidos mínimos para la elaboración de un proyecto de domótica*. Recuperado de: http://web.archive.org/web/20120710203547/http://ingenieria.org.ar/archivos/Domotica-CIEC.pdf

[4] OpenSSH. *Openssh Features*. Recuperado el 4 de abril del 2016 desde :http://www.openssh.com/features.html

[5] Raspbian. *Frontpage-Raspbian*. Recuperado el 4 de abril del 2016 desde: https://www.raspbian.org/

[6] Wiring. *Frontpage-Wiring*. Recuperado el 3 de abril del 2016 desde: http://wiring.org.co/

7. Anexos

7.1. Código Fuente

7.1.1. menu.py

El siguiente programa es del menú del sistema, además del uso de la cámara:

```
#!/usr/bin/env python
from escribirSerial import *
from conf import *
import os
from time import sleep
N = None
conf = conf(None,None,None)

#pide un input al usuario
def setN(n,mensaje):
                global N
                if n == None:
                #manejo de errores(si no es un numero, lop pide de nuevo)
                                try:
                                                N = int(raw_input(mensaje))
                                except ValueError:
                                                setN(None,"no es un numero, intentelo
                                                    de nuevo.")
                else:
                                N = n

#devuelve el input del usuario
def getN():
                global N
                return N

#crea un "banner" donde se pone un mensaje de manera que llame la atencion del
    usuario
def banner(opcion):
                os.system("clear")
                opcion = str(opcion)
                l = len(opcion)
                #print "\n"
                print (l+4)*"*"
                print 2*"*" +  str(opcion) + 2*"*"
                print (l+4)*"*"

#Hace una lista con opciones numeradas para cada entrada de la lista
def opciones_menu(lista):
                for item in lista:
                                print str(lista.index(item)+1)+"."+item
                setN(None,"\nSeleccione una opcion:")

#Define que es lo que se hace al seleccionar algo en el menu principal
def seleccion_principal(opcion):
                if(opcion == 1 ):
                        configurar_servidor()
                elif(opcion == 2):
```

```
                        tomarFotos()
                elif(opcion ==3):
                        es = escribirSerial(conf.Config.get("ArduinoUno","
                            puerto"),conf.Config.get("ArduinoDos","puerto"))
                elif(opcion ==4):
                        exit()

#menu de configurar servidor
def configurar_servidor():
                banner("Configurar Servidor")
                secciones = conf.Config.sections()
                secciones2 = secciones
                secciones2.append("Salir")
                opciones_menu(secciones2)
                if(getN()!= 9):
                        configurar_sec(secciones,getN()-1)
                else:
                        principal()

#menu de configurar una seccion del config.ini
def configurar_sec(secciones,seccion):
                banner("Configurar "+secciones[seccion])
                opciones_menu(conf.Config.options(secciones[seccion]))
                seccionElegida = conf.Config.options(secciones[seccion])
                opcionElegida =  seccionElegida[getN()-1]
                print("Se eligio: " + str(opcionElegida))
                print ("Valor actural configuracion manual de: " + str(conf.Config.
                    get(secciones[seccion],opcionElegida)))
                setN(None,"Nuevo valor de configuracion manual de" + opcionElegida +
                     " ")
                conf.set(str(secciones[seccion]),str(opcionElegida),getN())
                principal()

#permite al usuario tomar una foto y escoger el nombre de esta
def tomarFotos():
                n = raw_input("Nombre de la imagen")
                os.system("fswebcam -r 1920x1080 " + str(n) +".jpg")
                principal()

#funcion principal
def principal():
                #ensena el banner de bienvenido por medio segundo
                banner("Bienvenido/a a la casa inteligente")
                sleep(0.5)
                #ensena el banner de el menu principal
                banner("Que desea hacer?")
                print "-Si es la primera vez que usa el programa, escoger la opcion
                    1\n"
                #estas son las opciones del menu principal, se dan como un vector a
                    opciones_menu para que las despliegue
                opciones_menu_principal = ["Configurar servidor","Tomar foto","
                    Escribir a arduino", "Salir"]
                opciones_menu(opciones_menu_principal)
                seleccion_principal(getN())

#corre la funcion principal en un ciclo infinito
```

```
while True:
                principal()
```

7.1.2. escribirSerial.py

Este es el programa que escribe las órdenes del usuario a través del puerto serial sobre los diferentes Arduinos.

```
import ConfigParser
import serial
class escribirSerial:
                #el init pide el puerto en que esta conectado el arduino de
                    iluminacion(uno) y el puerto donde esta el arduino de ventilacion
                    (dos)
                #despues escibe los datos del archivo de configuracion por medio del
                    serial
                def __init__(self,uno,dos):
                                                self.Config = ConfigParser.
                                                    ConfigParser()
                                                #inicializa comunicacion al puerto "
                                                    uno" a 9600 bauds
                                                self.ser = serial.Serial("/dev/ttyACM
                                                    "+uno,9600)
                                                #lee del archivo de configuracion
                                                self.Config.read("config.ini")
                                                controlManualIluminacion =self.Config
                                                    .get("Iluminacion","controlManual
                                                    ")
                                                #lee los datos requeridos del archivo
                                                     de configuracion, luego escribe
                                                    al serial las opciones de la
                                                    configuracion
                                                if(controlManualIluminacion == "0"):
                                                                dato  = str(self.
                                                                    Config.get("
                                                                    Iluminacion","
                                                                    controlManual")+
                                                                    self.Config.get("
                                                                    Bombillo","estado
                                                                    ")+self.Config.
                                                                    get("Iluminacion"
                                                                    ,"max")+self.
                                                                    Config.get("
                                                                    Ultrasonico","
                                                                    sensibilidad"))
                                                                self.ser.write(dato)
                                                elif(controlManualIluminacion == "1")
                                                    :
                                                                dato  = str(self.
                                                                    Config.get("
                                                                    Iluminacion","
                                                                    controlManual")+
                                                                    self.Config.get("
                                                                    Bombillo","estado
                                                                    ")+self.Config.
```

```
                                        get("Iluminacion"
                                        ,"max")+self.
                                        Config.get("
                                        Ultrasonico","
                                        sensibilidad"))
                                    self.ser.write(dato)
                #hace lo mismo que la seccion
                    anterior pero en este caso se
                    conecta al puerto "dos"
                self.ser2 = serial.Serial("/dev/
                    ttyACM"+dos,9600)
                controlManualTemperatura = self.
                    Config.get("Temperatura","
                    controlManual")
                if(controlManualTemperatura == "0"):
                                    dato2 = str(self.
                                        Config.get("
                                        Temperatura","
                                        controlManual")+
                                        self.Config.get("
                                        Abanico","estado"
                                        )+self.Config.get
                                        ("Temperatura","
                                        max"))
                                    self.ser2.write(dato2
                                        )
                elif(controlManualTemperatura == "1")
                    :
                                    dato2 = str(self.
                                        Config.get("
                                        Temperatura","
                                        controlManual")+
                                        self.Config.get("
                                        Abanico","estado"
                                        )+self.Config.get
                                        ("Temperatura","
                                        max"))
                                    self.ser2.write(dato2
                                        )
```

7.1.3. conf.ini

Este es el código que envía datos a config.py

```
[ArduinoUno]
puerto = 0

[ArduinoDos]
puerto = 1

[Temperatura]
max = 100
min = 0

[Iluminacion]
```

```
max = 100
min = 0

[Ultrasonico]
sensibilidad = 10

[Abanico]
estado = 0

[Bombillo]
estado = 0

[Camara]
numerodeimagenes = 5
```

7.1.4. conf.py

Este es el código que permite leer y modificar los datos brindados por archivo de formato config.ini

```
import ConfigParser

class conf:
                #inicializa el archivo config.ini, si tiene argumentos corre la
                    funcion set
                def __init__(self,a,b,c):
                                self.Config = ConfigParser.ConfigParser()
                                self.Config.read("config.ini")
                                if(a != None):
                                                self.set(a,b,c)
                                                file = open("config.ini","w")
                                                self.Config.write(file)
                                                file.close()
                #escribe al archivo config.ini

                def set(self,a,b,c):
                                self.Config.set(str(a),str(b),str(c))
                                file = open("config.ini","w")
                                self.Config.write(file)
                                print "se cambio el estado del " + str(a)
```

7.1.5. iluminacion.ino

Este es el código de control automático de iluminación, a través de un sensor ultrasónico, para uno de los Arduinos.

```
/*
 *Programa en que se entran datos establecidos por el usuario por medio de serial
 *Estos datos son ingresados en un orden especial, que el programa tomara como
 *Parametros bajo los cuales se realizaran las acciones correspondientes
 */

const int led = 13, echo = 11, trig = 12, fotoresistor = A0;
char serverInput[10],serverOutput[8],valorMaxChar[4],valorSensibilidadChar[4];
long duracion, distancia;
```

```
int i,valorMaxInt,valorSensibilidadInt;
String lecturaSerial;

//se inicializan los pines y el serial
void setup(){
                pinMode(led,OUTPUT);
                pinMode(fotoresistor,INPUT);
                Serial.begin(9600);
}

//retorna lo que esta leyendo el fotoresistor
int lecturaFotoresistor(){
                return analogRead(fotoresistor);
}

//retorna lo que esta leyendo el sensor ultrasonico
int lecturaUltrasonico(){
                digitalWrite(trig, LOW);
                delayMicroseconds(2);
                digitalWrite(trig, HIGH);// genera el pulso de triger por 10ms
                delayMicroseconds(10);
                digitalWrite(trig, LOW);
                duracion = pulseIn(echo, HIGH);
                distancia = (duracion/2) / 29;  // calcula la distancia en
                    centimetros
                return distancia;
}

//retorna verdadero si el sensor ultrasonico detecta movimiento(mide una distancia
    menor a la establecida
boolean hayMovimiento(int distanciaLeida, int distanciaEstablecida){
                if(distanciaLeida<distanciaEstablecida){
                                return true;
                }
                else{
                                return false;
                }
}

//enciende el led
void encenderLed(){
                digitalWrite(led, HIGH);
}

//apaga el led
void apagarLed(){
                digitalWrite(led,LOW);
}

//lee los datos recibidos por serial, los convierte a un string para ser analizados
void recibirDatos(){
                while(Serial.available()){
                                lecturaSerial=Serial.readString();
                                lecturaSerial.toCharArray(serverInput,9);
                                //convierte parte del string en un int que representa
                                     el parametro bajo el que se encendera la luz
```

```
                                valorMaxChar[0] = serverInput[2];
                                valorMaxChar[1] = serverInput[3];
                                valorMaxChar[2] = serverInput[4];
                                valorMaxInt = atoi(valorMaxChar);
                                //convierte parte del string en un int que representa
                                    el parametro bajo el que se leera el movimiento
                                valorSensibilidadChar[0] = serverInput[5];
                                valorSensibilidadChar[1] = serverInput[6];
                                valorSensibilidadChar[2] = serverInput[7];
                                valorSensibilidadInt = atoi(valorSensibilidadChar);
                }
}

//retorna verdadero si el usuario activo el control manual, sino retorna falso
boolean controlManualActivado(char inputUsuario){
                if(inputUsuario == '1'){return true;}
                else if(inputUsuario == '0'){return false;}
}

//si el control manual fue activado se revisa el estado que pide el usuario para el
    led
void controlManual(){
                if(controlManualActivado(serverInput[0])==true){
                                if(serverInput[1]=='0'){
                                                apagarLed();
                                }
                                else if(serverInput[1] == '1'){
                                                encenderLed();
                                }
                }
                else if(controlManualActivado(serverInput[0])==false){
                                controlAutomatico();}
}

//revisa si hay movimiento, para realizar el control automatico
//el control automatico compara el nivel de luz medido por el fotoresistor con el
    valor establecido por el usuario, para
//encender o apagar el led
void controlAutomatico(){
                while(hayMovimiento(lecturaUltrasonico(),valorSensibilidadInt) ==
                    true && Serial.available()==false){
                                if(valorMaxInt<lecturaFotoresistor()){
                                                encenderLed();
                                }
                                else {
                                                apagarLed();
                                }
                }
}

//funcion principal que corre en ciclo infinito
void loop(){
                recibirDatos();
                controlManual();
                }
```

7.1.6. ventilador.ino

Este es el código de control automático de ventilador, a través de un termistor, para el otro Arduino.

```
#include <math.h>

const int termistor = A0, motor = 9;;
int sensorValue = 0, i, value_max, valorSensibilidadInt, valorMaxInt;
char input_server[6], valorMaxChar, valorSensibilidadChar[5];
String x, lecturaSerial;
double temperaturaC;

//se inicializan los pines y el serial
void setup(){
      Serial.begin(9600);
      pinMode(termistor, INPUT);
      pinMode(motor, OUTPUT);
}

//retorna lo que lee el fotoresistor en grados centigrados
double temperaturaCentigrados(){
                int lecturaTermistor = analogRead(termistor);
                double voltage = (lecturaTermistor * 5.0) / (1023);
                double resistencia = (voltage) / (5 - voltage);
                double temperaturaC = pow(0.003356 + log(resistencia/50)*0.0002379,
                    -1) -273;
                return temperaturaC;
}

//enciende el motor
void encenderMotor(){
                digitalWrite(motor, HIGH);
}

//apaga el motor
void apagarMotor(){
                digitalWrite(motor, LOW);
}

//compara lo que lee el termistor con el valor establecido por el usuario para
    encender o apagar el motor
void controlAutomatico(){
                while(Serial.available()==false){
                                if(temperaturaCentigrados() > valorSensibilidadInt ){
                                                encenderMotor();
                                }
                                else{
                                                apagarMotor();
                                }
    }
}

//lee los datos recibidos por serial, los convierte a un string para ser analizados
void recibirDatos(){
        while(Serial.available()){
```

```
                lecturaSerial=Serial.readString();
                lecturaSerial.toCharArray(input_server,7);
                valorSensibilidadChar[0] = input_server[2];
                valorSensibilidadChar[1] = input_server[3];
                valorSensibilidadChar[2] = input_server[4];
                valorSensibilidadInt = atoi(valorSensibilidadChar);
        }
}

//retorna verdadero si el usuario activo el control manual, sino retorna falso
boolean controlManualActivado(char inputUsuario){
                if(inputUsuario == '1'){return true;}
                else if(inputUsuario == '0'){return false;}
}

//si el control manual fue activado se revisa el estado que pide el usuario para el
    led
void controlManual(){
                if(controlManualActivado(input_server[0])==true){
                                if(input_server[1]=='0'){
                                                apagarMotor();
                                }
                                else if(input_server[1] == '1'){
                                                encenderMotor();
                                }
                }
                else if(controlManualActivado(input_server[0])==false){
                    controlAutomatico();}
}

//funcion principal que corre el ciclo infinito
void loop(){
                recibirDatos();
                controlManual();
}
```