PROYECTO DE FIN DE CURSO DE PLACAS ROBOTICAS

SISTEMA DE REFRIGERACION

Descripción breve

Se trata de implementar un sistema de ventilación controlado por temperatura cuyo uso es variado, como refrigerar placas o equipos electrónicos y otros de uso general.

Ramiro Calderón R.
Juandemarco_1@yahoo.com.ar

INDICE

MEMORIA

INTRODUCCION:

Como vimos en la descripción del proyecto, se trata de que un ventilador funcione de forma automática conforme aumente o disminuya la temperatura, es decir, a mayor temperatura mayor velocidad de giro y si disminuye menor velocidad de giro. También la idea es que a temperaturas menores de 20ºC no funcione y mientras que a mayores que 30ºC se mantenga a la velocidad máxima de forma constante.

SIMULACION:

Mediante el uso del programa Proteus se realizó la creación del circuito y su respectiva simulación usando código de Arduino y sus respectivas librerías.

En esta primera etapa en Proteus usé elementos que simulaban los que usaré en la práctica, que son los que nos brinda el kit Elegoo y que hasta ahora no me falta nada.

ELEMENTOS:

-Arduino UNO.

-LCD2 LM016L, display.

-DTH11, sensor de temperatura.

-Driver L293D, IC que controla velocidad y sentido de giro del motor.

-Motor de cc de 5V.

-Potenciómetro 10K ohm para el display.

-Fuente externa.

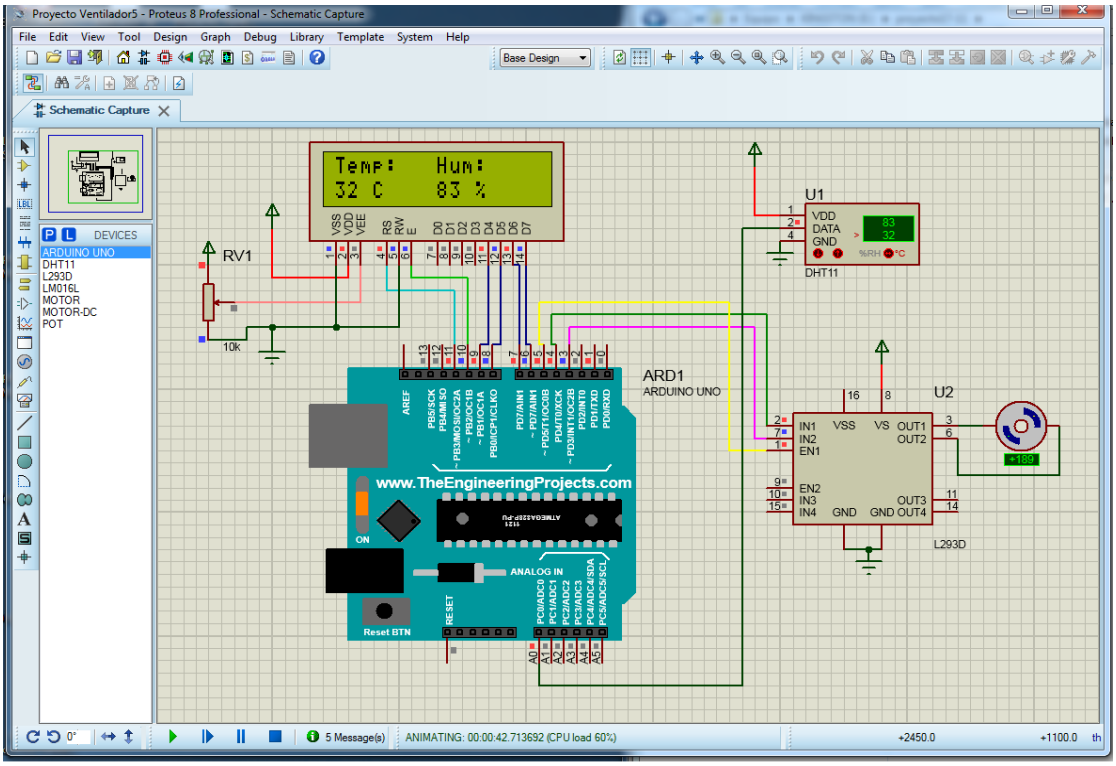
-Relé.

-Receptor infrarojo.

-Mando a distancia.

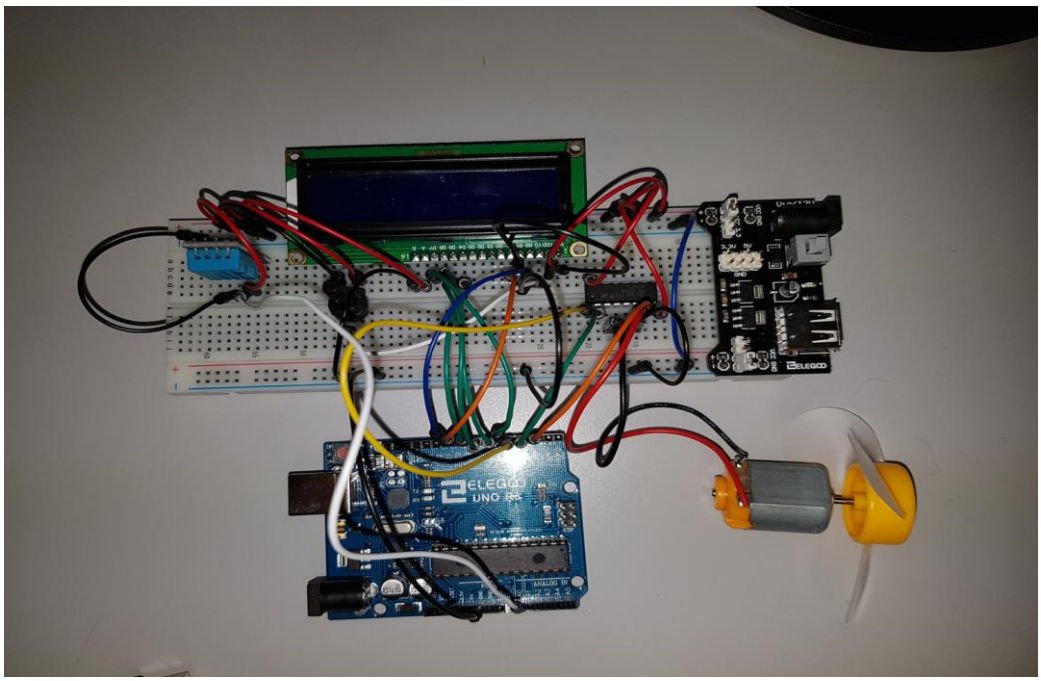
CIRCUITO-PROTEUS:

Apoyándome en este programa diseñé el circuito que de momento me interesa simular, mediante esta imagen se puede observar todo el contenido.



CONEXION:

Luego del diseño y simulación en Proteus pasamos a la simulación en la práctica, es decir, con componentes y dispositivos reales interconectados según esquema ponerlo en marcha.



Una vez realizada la conexión del circuito se pone en marcha alimentándolo y cargando el programa que controla su funcionamiento, finalmente se observó que funciona salvo que a los 20ºC no arranca ya que es ahí donde debe empezar a girar el ventilador, pero a poca velocidad, esto se debe a cuestiones físicas como la corriente mínima que necesita para empezar a girar. Esto es solucionable mediante el programa. Se observa que a los 23ºC empieza a girar, yo quiero que sea a partir de los 20ºC.

Se solucionará ese pequeño inconveniente, luego añadiré un encendido y apagado por mando a distancia, espero buenos resultados.

CODIGO INICIAL:

Este código es el ideado originalmente.

```
#include "DHT.h"
#include <LiquidCrystal.h>
#define DHTPIN A0
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal lcd(11, 10, 9, 8, 7, 6);

#define ENABLE 5
#define IN1 3
#define IN2 4

void setup() {
Serial.begin(9600);
lcd.begin(16, 2);
dht.begin();

pinMode(ENABLE,OUTPUT);
pinMode(IN1,OUTPUT);
pinMode(IN2,OUTPUT);

digitalWrite(IN1, LOW);
digitalWrite(IN2, LOW);
}
void loop() {
 int h = dht.readHumidity();// Lee la humedad
int t= dht.readTemperature();//Lee la temperatura
```

```
lcd.setCursor(0,0);
lcd.print("Temp: ");
lcd.setCursor(0,1);
lcd.print(t);//Escribe la temperatura
lcd.setCursor(3,1);
lcd.print("C");
lcd.setCursor(8,0);
lcd.print("Hum: ");
lcd.setCursor(8,1);
lcd.print(h);
lcd.setCursor(11,1);
lcd.print("%");

if ((t >= 20) and (t<=30)) {
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,HIGH);
  float pweme = 25.5*t -510;
  analogWrite(ENABLE,pweme);
}
}
```

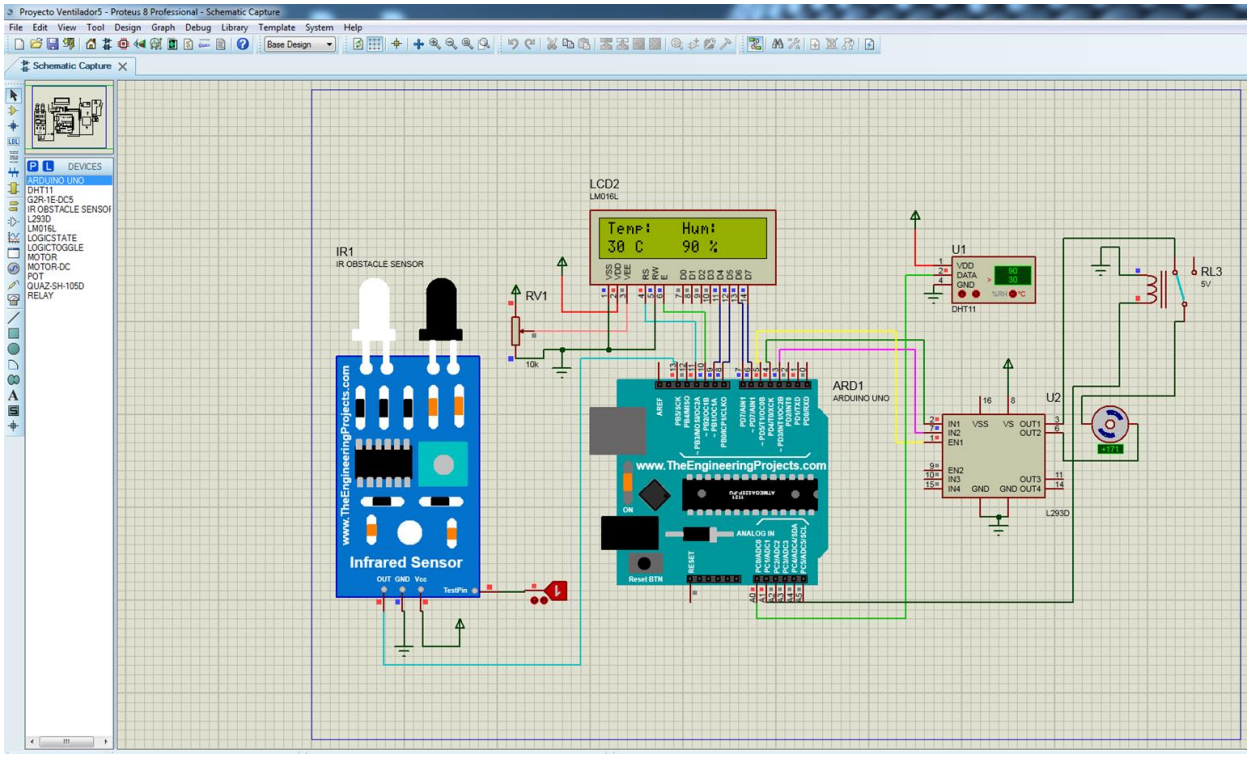
Posteriormente, decidí implementar un relé que controlase el encendido y apagado del motor, esta función se realizará a través de un mando a distancia de tal forma que el consumo sea el mínimo, digamos que quede en standby puesto que no siempre se va a necesitar que esté en automático por razones obvias.

Para esto primero y como siempre me apoyé en la simulación del Proteus, tuve problemas en elegir el tipo de relé pues no respondía. Una solución fue el “rele animate” que si me permitió simular mediante un pulso lógico abrir o cerrar.

A continuación, implementé en el Proteus un receptor infrarojo para el cual también se tuvo que descargar una librería (IRremote) para poder utilizarlo.

Finalmente, todo el esquema realizado en Proteus funciona, ahora se realizará el montaje físicamente de estos nuevos dispositivos.

Abajo se observa el circuito con los elementos añadidos.



CODIGO PARA PROTEUS:

En este caso añadí el relé y el sensor de infrarojos. Este es el código para simular en Proteus.

```
#include <IRremote.h>

#include "DHT.h"

#include <LiquidCrystal.h>

#define DHTPIN A0

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

LiquidCrystal lcd(11, 10, 9, 8, 7, 6);


#define ENABLE 5

#define IN1 4

#define IN2 3

 int rele = A1;

 int IR = 13;    // sensor KY-022 a pin digital

IRrecv irrecv(IR);      // establece al 13 para objeto irrecv

decode_results codigo;      // crea objeto codigo de la clase decode_results


void setup() {

Serial.begin(9600);

lcd.begin(16, 2);
```

```
dht.begin();
pinMode(rele, OUTPUT);
pinMode (IR, INPUT);
irrecv.enableIRIn();      // inicializa recepcion de datos

pinMode(ENABLE,OUTPUT);   //activa L239D
pinMode(IN1,OUTPUT);      //entrada IN 1de l239D
pinMode(IN2,OUTPUT);      //entrada IN2 de l239D
// Inicio de motor, no gira.
digitalWrite(IN1, LOW);
digitalWrite(IN2, LOW);

}

void loop() {
 int h = dht.readHumidity();    // Lee la humedad
int t= dht.readTemperature();  //Lee la temperatura
lcd.setCursor(0,0);

lcd.print("Temp: ");
lcd.setCursor(0,1);
lcd.print(t);                  //Escribe la temperatura
lcd.setCursor(3,1);
lcd.print("C");
lcd.setCursor(8,0);
lcd.print("Hum: ");
lcd.setCursor(8,1);
lcd.print(h);                   //Escribe la humedad
lcd.setCursor(11,1);
lcd.print("%");
if ((t >= 20) and (t<=30)) {
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,HIGH);
  float pweme = 25.5*t -510;
  analogWrite(ENABLE,pweme);
}
if(irrecv.decode(&codigo)){
  Serial.println( codigo.value,HEX);
  irrecv.resume();
}
 int x = digitalRead(IR);
```

```
 if ( x == HIGH ){
  digitalWrite(rele, HIGH);
 }
if ( x == LOW ){
digitalWrite(rele, LOW);
}
}
```

CODIGO PARA CIRCUITO FISICO:

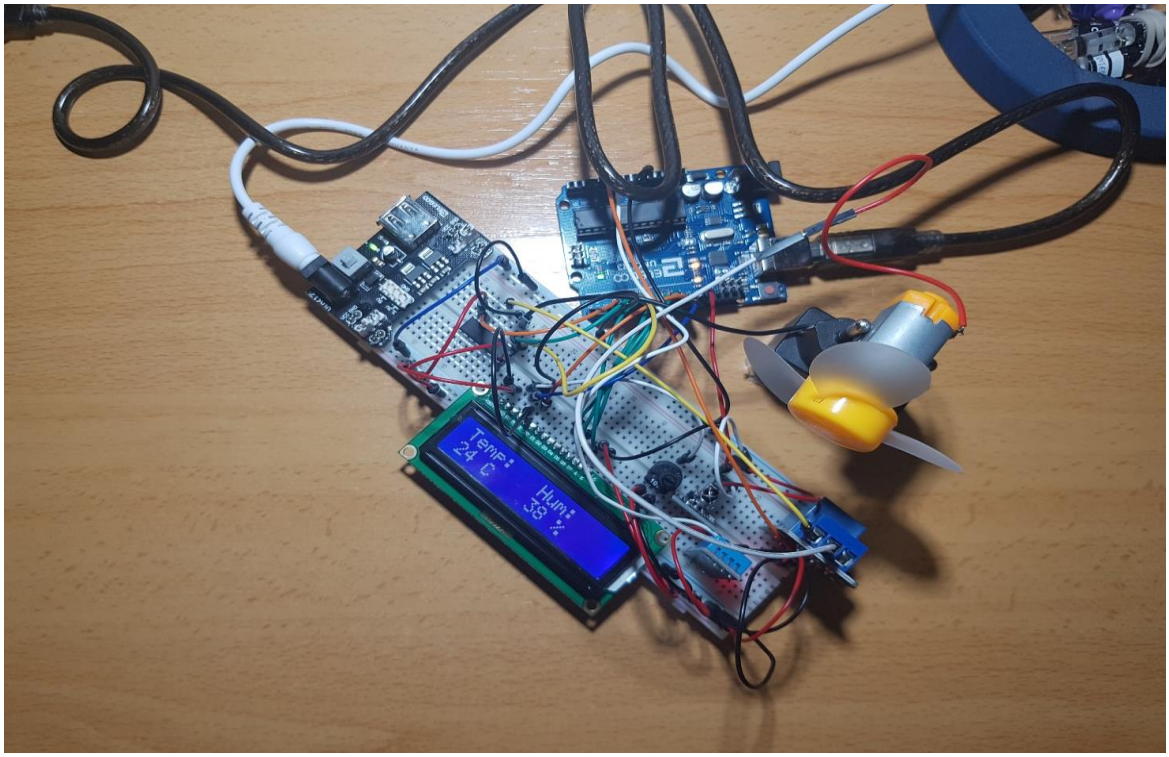
Luego, en el código del circuito físico hay una variación, es debido a que utilizamos el mando a distancia. En el Proteus solo simulamos enviando dos estados, cero y uno, para ver que el relé respondiera como lo pensé.

Esa variación lo vemos a continuación en el nuevo código para el prototipo:

```
#include <IRremote.h>
#include "DHT.h"
#include <LiquidCrystal.h>
#define DHTPIN A0
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal lcd(11, 10, 9, 8, 7, 6);

#define ENABLE 5
#define IN1 4
#define IN2 3
 int rele = A1;  // Salida analógica que también funciona como digital, características de Arduino
 int IR = 13;    // sensor KY-022 a pin digital
IRrecv irrecv(IR);      // establece al 13 para objeto irrecv
decode_results codigo;      // crea objeto codigo de la clase decode_results

void setup() {
Serial.begin(9600);
lcd.begin(16, 2);
dht.begin();
pinMode(rele, OUTPUT);
pinMode (IR, INPUT);
irrecv.enableIRIn();      // inicializa recepcion de datos

pinMode(ENABLE,OUTPUT);   //activa L239D
pinMode(IN1,OUTPUT);      //entrada IN 1de l239D
```

```
pinMode(IN2,OUTPUT);      //entrada IN2 de l239D

digitalWrite(IN1, LOW);
digitalWrite(IN2, LOW);

}

void loop() {
 int h = dht.readHumidity();    // Lee la humedad
int t= dht.readTemperature();  //Lee la temperatura
lcd.setCursor(0,0);
lcd.print("Temp: ");
lcd.setCursor(0,1);
lcd.print(t);                  //Escribe la temperatura
lcd.setCursor(3,1);
lcd.print("C");
lcd.setCursor(8,0);
lcd.print("Hum: ");
lcd.setCursor(8,1);
lcd.print(h);                  //Escribe la humedad
lcd.setCursor(11,1);
lcd.print("%");

if (irrecv.decode(&codigo))
{
switch(codigo.value)
{
case 0x00FF30CF:
funcion1();
break;
case 0x00FF18E7:
funcion2();
break;
}
irrecv.resume();
}
delay(300);
if((t>=20) and (t<=30)) {
  digitalWrite(IN1,HIGH);
  digitalWrite(IN2,LOW);
  float pweme = 15.5*t -210;     //función lineal que calcula el PWME en base a la temperatura.
```

```
  analogWrite(ENABLE,pweme);
  Serial.println(pweme);
}
}
void funcion1(){
  Serial.println("Enciende Ventilador");
  digitalWrite(rele, HIGH);
  }
void funcion2() {
 Serial.println("Apaga ventilador");
 digitalWrite(rele, LOW);
}
```



DESDRIPCION DE DISPOSITIVOS:

-Señal PWM:

Quiero resaltar el funcionamiento de la salida PWM, es una salida del Arduino, señal cuadrada de 5V, donde se varía el ancho de pulso generando una tensión promedio que va de 0V a 5V, esta salida consta de un byte, lo que implica que tiene de o a 255 posibilidades de tensión, esto porque se calcula por 2^8=256 posibilidades.

-Fuente externa:

Utilizo una fuente externa para no exigir a la placa, ya que cada pin de I/O puede entregar una corriente máxima de 40mA y de 0V a 5V.

Esta fuente entrega hasta 700 mA según especificaciones con salidas de 3,3V y 5V. Lo suficiente para alimentar todo lo que utilizaré para mi proyecto. Aquí copio las características según Elegoo.

Especificaciones de la fuente externa:

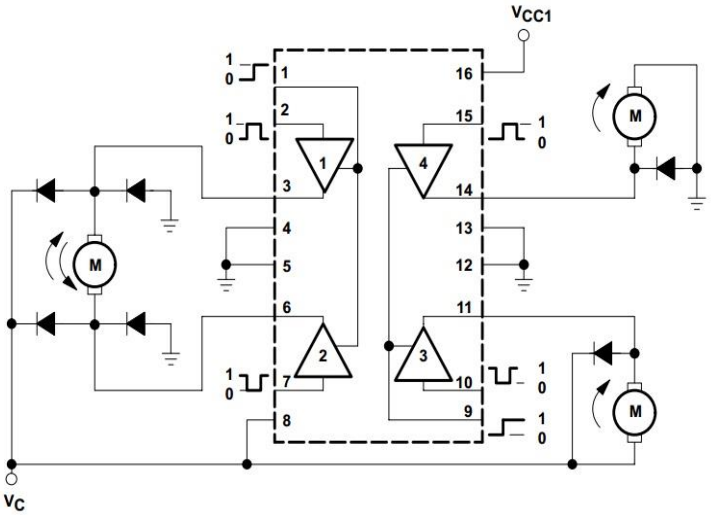
• Bloqueo Encendido interruptor
• LED Power indicador
• Entrada voltaje: 6.5-9v (CC) través 5.5mm x 2,1 mm enchufe
• Salida voltaje: 3.3V / 5v
• Máximo salida: 700 mA
• Salida. 0v, 3.3v, 5v a protoboard
• Salida pins principal para usos externos
• Tamaño: 2.1 en x 1.4 en
•Conector USB

-Driver L293D:

También utilizo el IC L293D, es un "driver" que permite el manejo del motor tanto para el control de velocidad como el sentido de giro. Este integrado también entrega una corriente suficiente para que el motor de cc funcione sin problemas. Copio datos del IC de Elegoo:

Especificaciones del L293D:

• Amplio rango de tensión de alimentación: 4,5 V a 36 V
• Alimentación de entrada lógica separada
• Protección interna ESD
• Apagado térmico
• Alta inmunidad de ruido entradas
• Funcionalmente Similar al L293 SGS y SGS L293D
• Máxima salida de corriente 1 A por canal (600 mA para el L293D)
• Máxima salida de corriente 2 A por canal (1.2 A para L293D)



Arriba muestro un esquema en bloques del IC, vemos como funcionaría el motor de la izquierda en uno u otro sentido.

La velocidad se controla también por la misma entrada que es la pata 1 que es el EN1, a través de esta entrada se envía la señal PWM que es la que variará la velocidad. A través de las salidas 3 y 6 se controla el sentido de giro del motor. En mi caso solo usaré un solo sentido.

-DTH11:

Este es un dispositivo importante en este proyecto, mide la temperatura ambiental y la humedad relativa. Tiene una resolución de 1°C con un rango de trabajo de 0°C a 50°C para temperatura, y una resolución del 1% y un rango de trabajo del 20% al 95% de humead relativa. Se alimenta con una tensión de 3,3V y 5V.

Este sensor se caracteriza por enviar una señal digital calibrada por lo que asegura una alta calidad y una fiabilidad a lo largo del tiempo, ya que contiene un microcontrolador de 8 bits integrado. Está constituido por dos sensores resistivos (NTC y humedad). Tiene una excelente precisión y una respuesta rápida en las medidas.

Especificaciones:

Model	DHT11
Power supply	3-5.5V DC
Output signal	digital signal via single-bus
Sensing element	Polymer resistor
Measuring range	humidity 20-90%RH; temperature 0-50 Celsius
Accuracy	humidity +-4%RH (Max +-5%RH); temperature +-2.0Celsius
Resolution or sensitivity	humidity 1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-1Celsius
Humidity hysteresis	+-1%RH
Long-term Stability	+-0.5%RH/year
Sensing period	Average: 2s
Interchangeability	fully interchangeable
Dimensions	size 12*15.5*5.5mm

-Potenciómetro:

Tenemos un potenciómetro (resistencia variable) de 10K ohm que nos ayuda a controlar el brillo del display, ajustando este potenciómetro dejaremos que el display muestre mejor los datos.

-Relé:

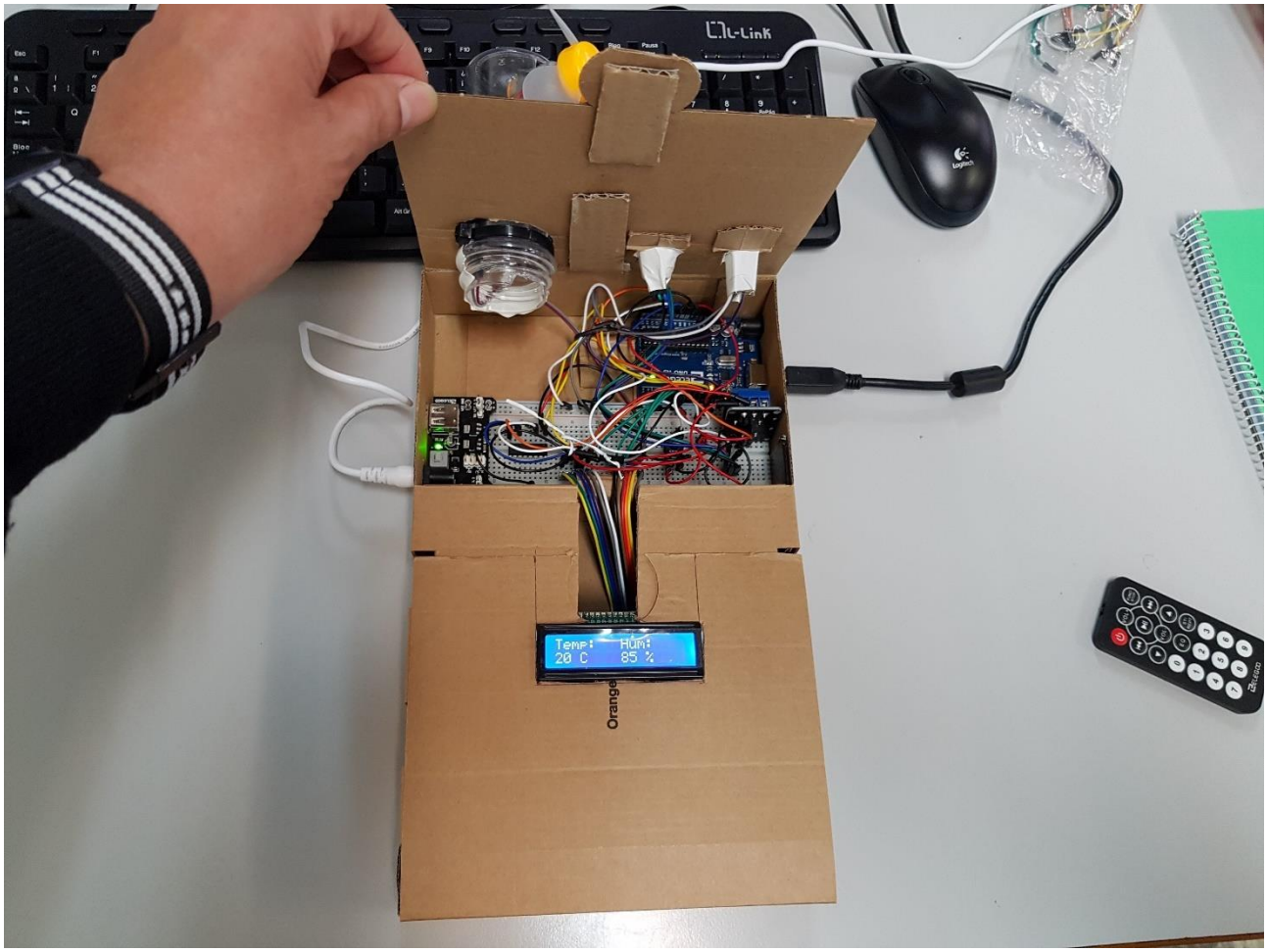
Otro elemento no menos importante que he usado es el relé, es un dispositivo simple que, mediante una alimentación, en este caso de 5V, puedo conectar otro elemento cuya alimentación sea distinta y de mayor consumo en el que ambos circuitos están separados uno del otro. Lo utilizaré para habilitar o deshabilitar solo el motor.

-Sensor infrarojo KY-022:

Uno de los elementos también muy usados en nuestra vida diaria es el mando a distancia. Con este dispositivo y mediante un receptor infrarojo KY-022 haremos que haga algunas cosas como conectar y desconectar el relé que a su vez abre o cierra el circuito de alimentación del motor de cc. Este receptor mediante un cierto protocolo de lectura decodificará las señales que le envíe el mando

a distancia, éste le envía una serie de pulsos de distinto ancho que representan ceros y unos lógicos en código hexadecimal, cada tecla tendrá un valor distinto en hexadecimal que hará lo que nosotros deseemos. Esta es una forma básica de usar el control, en mi caso encendido y apagado del motor controlando el relé. Podría haber utilizado para otras cosas más, pero no era necesario para la idea de funcionamiento de este proyecto. Lo que si valoro es que el uso de este dispositivo me da la idea de cómo se puede aprovechar los distintos dispositivos que tenemos a mano así de cómo es su funcionamiento interno para así sacarle el máximo provecho.

El proyecto terminado en una caja adaptada para una mejor presentación.



Conexión y cableado en el interior de la caja.

CONCLUSIONES:

-Este proyecto es muy sencillo de realizar, pero me ha permitido, en el poco tiempo del curso, nuevos conocimientos aplicados, la conjunción de la parte electrónica con el software nos da unas potentes herramientas para poder realizar lo que nuestra imaginación desee. Esto solo se puede con la constante práctica y con el ingenio de uno mismo.

-Otras herramientas de ayuda son las que hemos usado para simular nuestros trabajos como Proteus (diseño electrónico) o Tinkercad (diseño útil y sencillo pero potente para código o software).

-Yendo al proyecto en sí mismo, en la simulación todo funciona perfecto, pero cuando se monta físicamente todos los componentes y dispositivos me encuentro con ciertas dificultades; la principal para mí fue la que no podía girar el motor cuando a la temperatura de 20 °C entrega una PWM de 0, empezó a girar a partir de los 23 °C y que el valor del PWM era de 125. Esto se debe a problemas físicos propios del motor, como dije uno de esos problemas puede ser la corriente mínima que necesita para girar, otra causa puede ser el torque del motor. Comprobé cuál era el consumo de corriente a los 23 °C y este era de unos 41 mA, es decir a partir de esta corriente empieza a girar bien el motor, como necesitaba que empiece a girar a los 20 °C lo que hice fue variar la ecuación

lineal para que se de esto, para 20 °C un PWM de 100. Mediante prueba y error llegué a este valor y en cuanto va aumentando la temperatura la corriente consumida aumenta hasta mas o menos 62mA donde ya se estabiliza incluso para mayores temperaturas que 30ºC porque aquí llega a su valor máximo de 255.
La ecuación para el cálculo del PWM era: pweme = 25.5*t -510 y ahora es pweme = 15.5*t -210, esta ecuación es parte del código para que funcione el ventilador.

-Finalmente el ventilador funciona como deseaba y también responde con el control remoto o mando a distancia.

REFERENCIAS:

-https://pitonis.moodlecloud.com/
-https://github.com/
-https://www.arduino.cc/
-https://www.youtube.com/c/BitwiseAr/featured
-https://www.tinkercad.com/
-Proteus para diseño
-Otras páginas mas de interés.