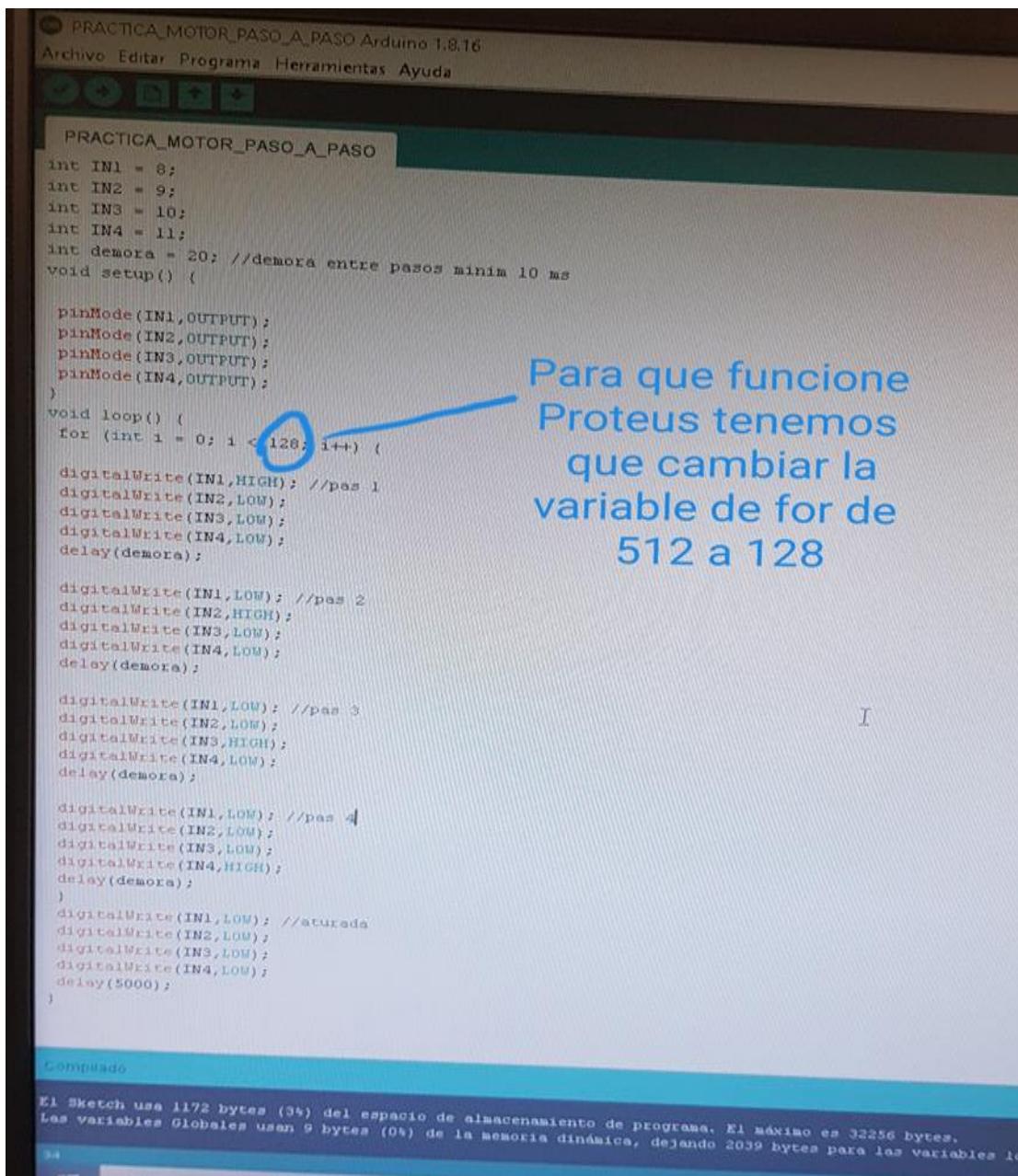


SIMULADOR STEPPER CON PROTEUS



PRACTICA_MOTOR_PASO_A_PASO Arduino 1.8.16

Archivo Editar Programa Herramientas Ayuda

PRACTICA_MOTOR_PASO_A_PASO

```
int IN1 = 8;
int IN2 = 9;
int IN3 = 10;
int IN4 = 11;
int demora = 20; //demora entre pasos minim 10 ms

void setup() {
    pinMode(IN1,OUTPUT);
    pinMode(IN2,OUTPUT);
    pinMode(IN3,OUTPUT);
    pinMode(IN4,OUTPUT);
}

void loop() {
    for (int i = 0; i < 128; i++) {
        digitalWrite(IN1,HIGH); //pas 1
        digitalWrite(IN2,LOW);
        digitalWrite(IN3,LOW);
        digitalWrite(IN4,LOW);
        delay(demora);

        digitalWrite(IN1,LOW); //pas 2
        digitalWrite(IN2,HIGH);
        digitalWrite(IN3,LOW);
        digitalWrite(IN4,LOW);
        delay(demora);

        digitalWrite(IN1,LOW); //pas 3
        digitalWrite(IN2,LOW);
        digitalWrite(IN3,HIGH);
        digitalWrite(IN4,LOW);
        delay(demora);

        digitalWrite(IN1,LOW); //pas 4
        digitalWrite(IN2,LOW);
        digitalWrite(IN3,LOW);
        digitalWrite(IN4,HIGH);
        delay(demora);
    }
    digitalWrite(IN1,LOW); //aturada
    digitalWrite(IN2,LOW);
    digitalWrite(IN3,LOW);
    digitalWrite(IN4,LOW);
    delay(5000);
}
```

Para que funcione Proteus tenemos que cambiar la variable de for de 512 a 128

Compilado

El Sketch usa 1172 bytes (3%) del espacio de almacenamiento de programa. El máximo es 32256 bytes.
Las variables Globales usan 9 bytes (0%) de la memoria dinámica, dejando 2039 bytes para las Variables Locales.

Uno de los pasos que debemos tener en cuenta para que funcione el simulador en

PROTEUS es cambiar la variable de for de 512 a 128.

A continuación hacemos lo mismo en las propiedades del motor según imágenes

Para modificar tomamos en cuenta la siguiente operación matemática, multiplicamos

128 (variable de for)

x 4 (pasos que hemos programado en ARDUINO)

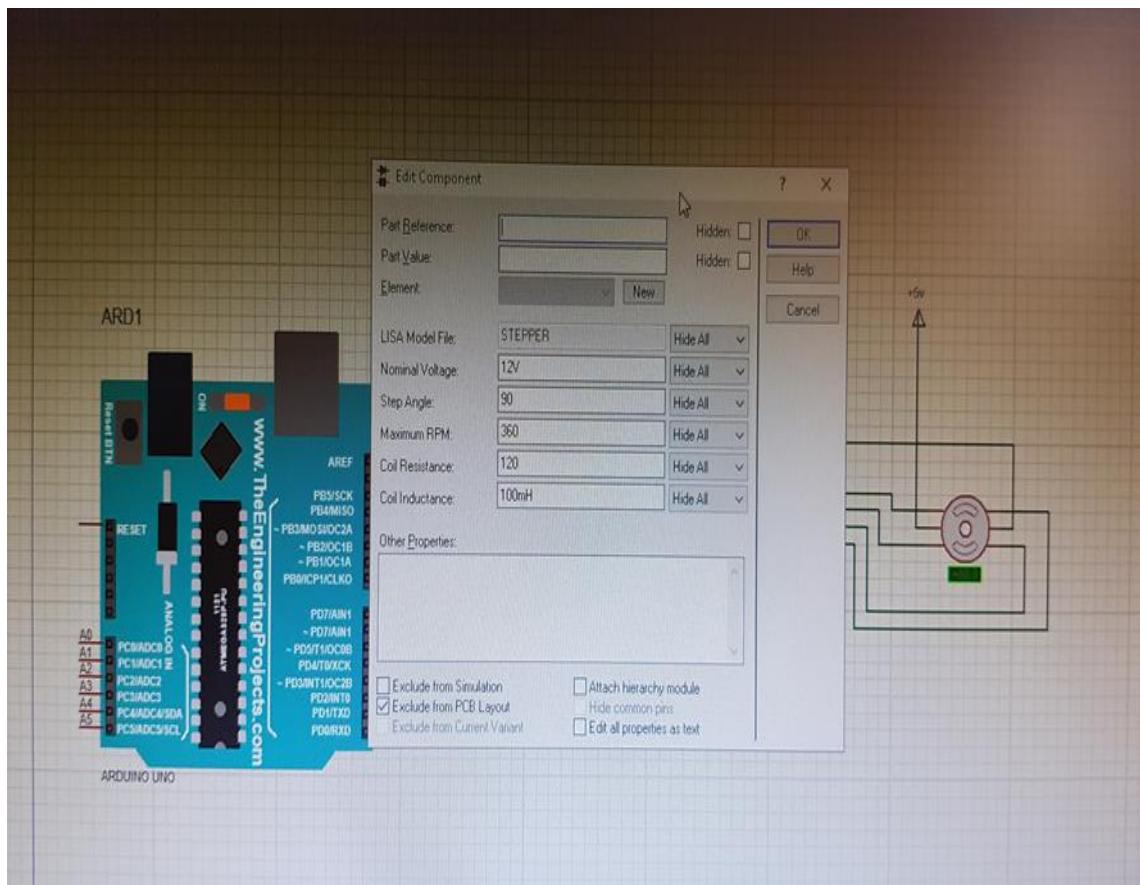
512 (pasos que cumplira en dar una vuelta de 360º)

El valor de Step Angle nos resulta de dividir los 360º que tarda en dar una vuelta por los 512

pasos que tarda en dar una vuelta el motor.

$$360^\circ / 512 = 0,703125.$$

Aquí lo podemos apreciar en la siguiente imagen



 Edit Component

Part Reference:

Part Value:

Element:

Hidden:

Hidden:

LISA Model File: STEPPER Hide All ▾

Nominal Voltage: 12V +5v Hide All ▾

Step Angle: 90 0,703125 Hide All ▾

Maximum RPM: 360 360000 Hide All ▾

Coil Resistance: 120 Hide All ▾

Coil Inductance: 100mH Hide All ▾

Other Properties:

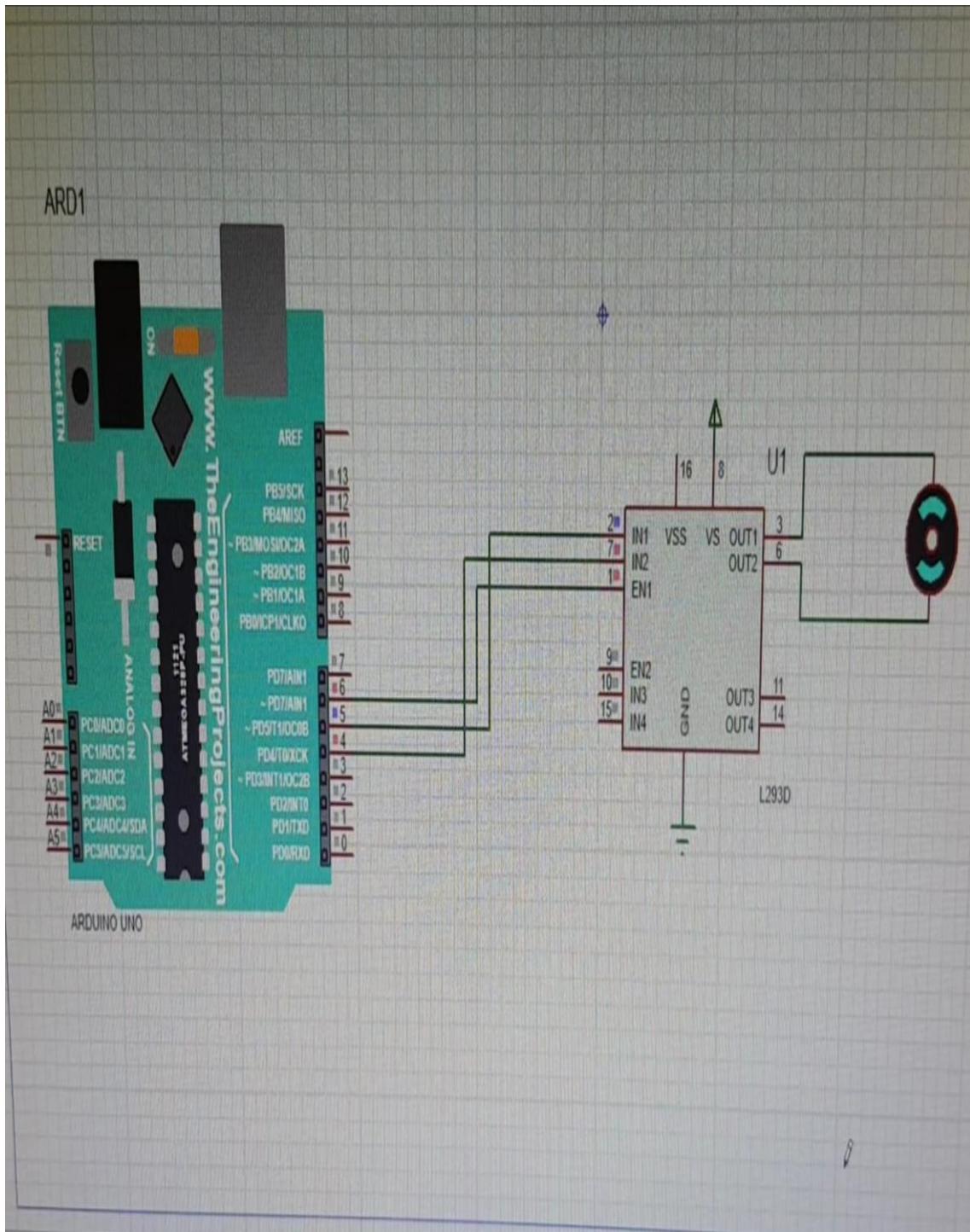
Exclude from Simulation Attach hierarchy module

Exclude from PCB Layout Hide common pins

Exclude from Current Variant Edit all properties as text

Por último el RPM lo multiplicamos por mil para poder visualizar lo que hemos programado.

PRACTICA MOTOR CONTINUA CON CAMBIO DE SENTIDO



Para realizar esta práctica necesitaremos la placa de ARDUINO UNO, la placa L293D y un motor simple ACTIVA, la conexión podemos mirar en el gráfico.

Conectamos desde el **L293D** ala **ARDUINO** :

- El **EN1** a un PIN **PWM** en este caso al **PIN6** (esta conexión nos da la dirección del motor).
- El **IN1** lo conectaremos al PIN5 (esta conexión nos dará la dirección del motor).
- El **AUT1** va conectado a **un polo** del motor. (esta conexión nos dará el giro izq o der).
- El **OV** lo conectamos con un **GND** .
- El **AUT2** lo conectamos al **otro polo** del motor. (esta conexión nos dará el giro izq o der).
- El **INT2** va conectado al **PIN4**. (esta conexión nos dará la dirección del motor).
- Finalmente el **+V** conectaremos para dar alimentación de corriente

Nota:(cabe indicar que no es recomendable conectar directamente del **ARDUINO** porque si no está bien conectado podríamos quemar el **PIN** o la **PLACA completa L293D**)

practica_de_motor_continua_con_cambio_de_sentido_2 Arduino 1.8.16

Archivo Editar Programa Herramientas Ayuda



practica_de_motor_continua_con_cambio_de_sentido_2

```
#define ENABLE 6
#define DIRA 5
#define DIRB 4
void setup() {
    pinMode(ENABLE,OUTPUT);
    pinMode(DIRA,OUTPUT);
    pinMode(DIRB,OUTPUT);
}
void antihorari(int temps) {
    digitalWrite(DIRA,HIGH);
    digitalWrite(DIRB,LOW);
    delay(temp);
}
void horari(int temps) {
    digitalWrite(DIRA,LOW);
    digitalWrite(DIRB,HIGH);
    delay(temp);
}
void loop() {
    digitalWrite(ENABLE,HIGH); // apunt per moure 5 segons en un
    antihorari(5000);
    horari(5000);
    digitalWrite(ENABLE,LOW); // aturada de 1 segon
    delay(1000);
}
```

En este **gráfico** podemos mirar que en **void loop** hemos definido un

digitalWrite (ENABLE,HIGT) que nos hara un giro de **5 segundos**

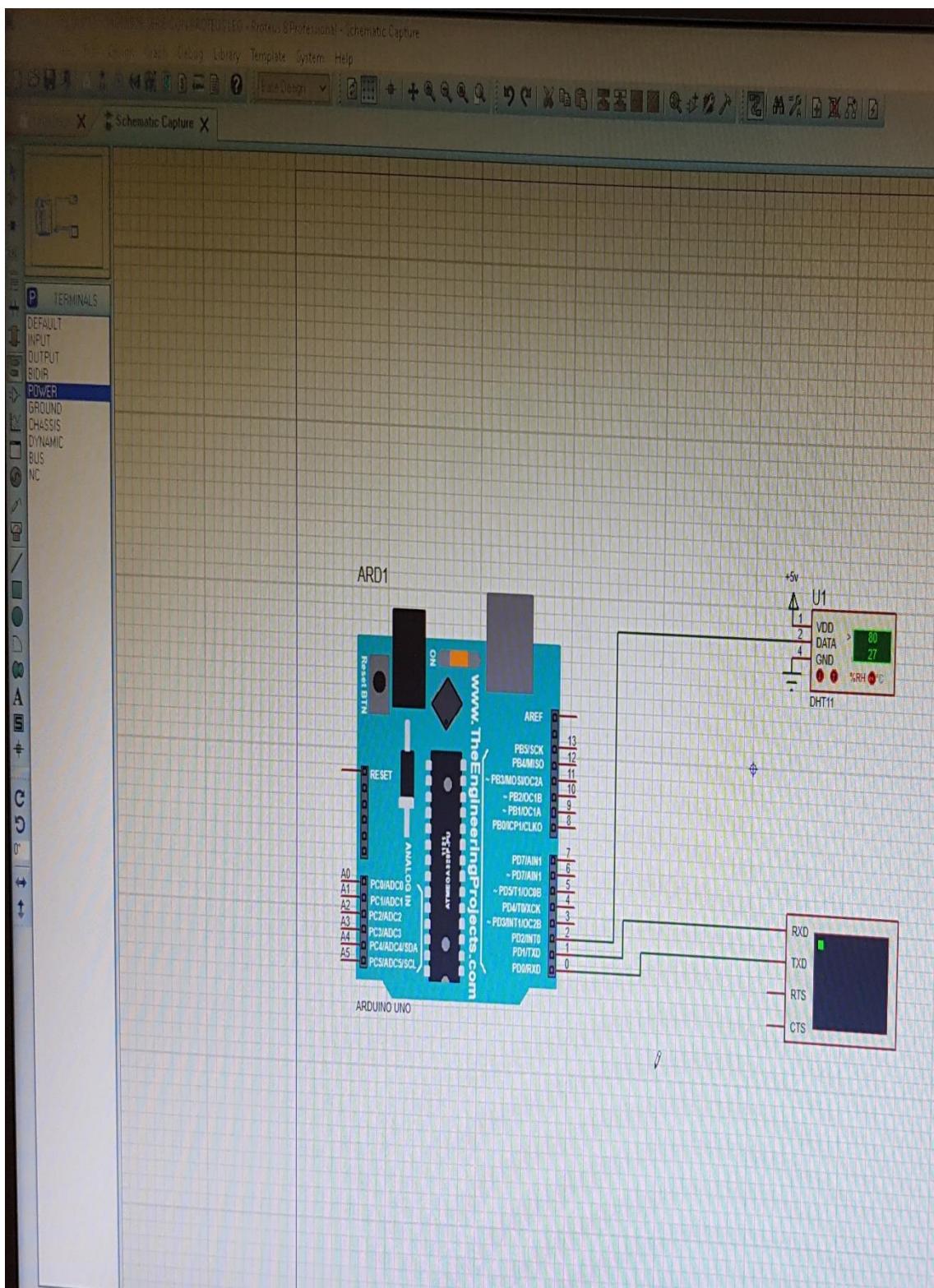
digitalWrite(ENABLE;LOW) nos hace una parada de **1 segundo**

horari(5000) nos hará girar en **sentido de las manecillas del reloj**

antihorari (5000) nos hará girar en **sentido contrario a las manecillas del reloj**

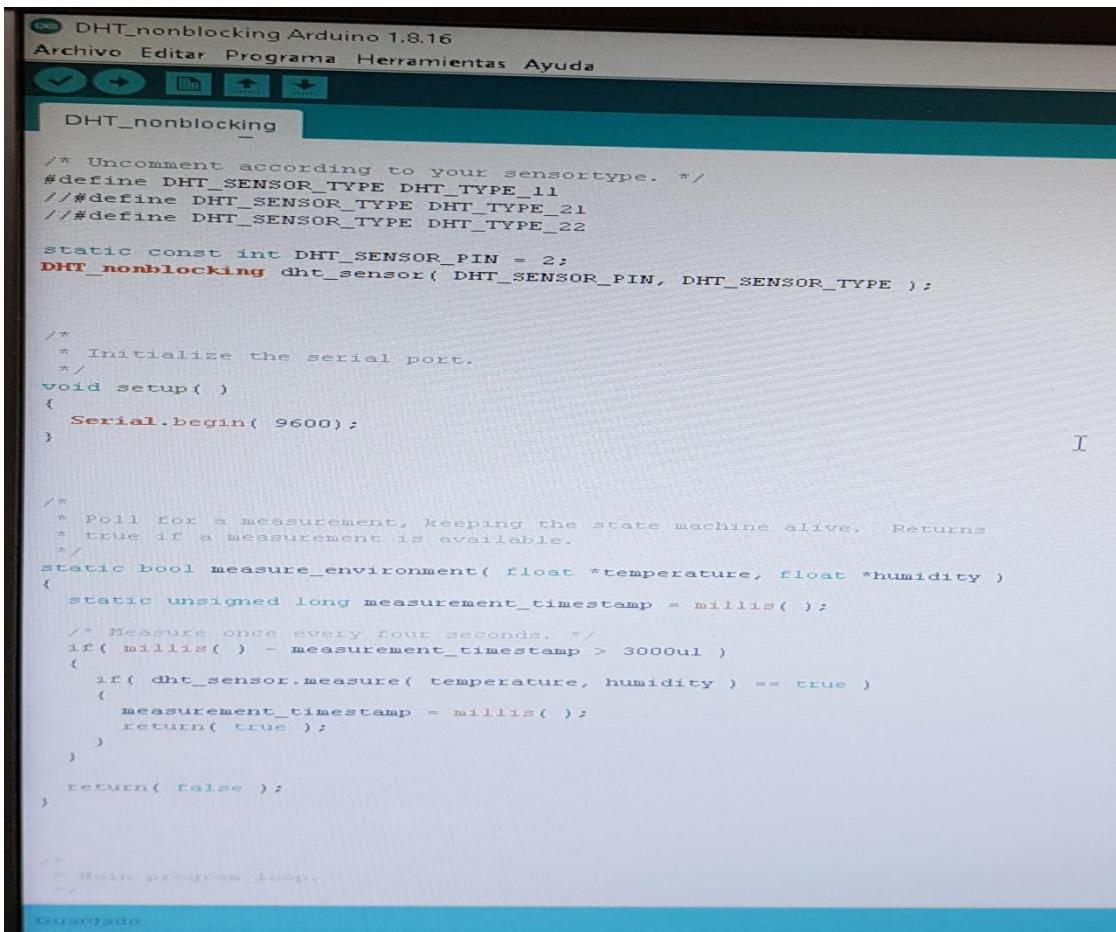
el **delay (1000)** nos permite repetir indefinidamente cada segundo el giro en uno y otro sentido

SIMULADOR DTH11 Y MONITOR SERIE CON PROTEUS



Para esta simulación necesitaremos la placa de **ARDUINO** , el sensor **DTH11** , tomaremos también un **VIRTUAL TERMINAL** , lo que vendría a ser el monitor de **ARDUINO** , ahora realizaremos la conexión.

- El **RXD** del **VIRTUAL TERMINAL** lo conectamos al **PIN 1 de ARDUINO**
- El **TXD** del **VIRTUAL TERMINAL** lo conectamos al **PIN 0 de ARDUINO**
- En el **VDD** conectaremos un **POWER +5v**
- En el **GND** conectaremos un **GROUND**
- El **DATA** lo conectaremos al **PIN 2 de ARDUINO**



The screenshot shows the Arduino IDE interface with the file 'DHT_nonblocking' open. The code implements a non-blocking DHT sensor library. It includes configuration for sensor type (DHT_TYPE_11), pin assignment (DHT_SENSOR_PIN = 2), and serial port initialization (Serial.begin(9600)). The 'measure_environment' function polls the sensor every four seconds, returning true if a measurement is available. The main loop calls this function.

```
/* Uncomment according to your sensortype. */
#define DHT_SENSOR_TYPE DHT_TYPE_11
//#define DHT_SENSOR_TYPE DHT_TYPE_21
//#define DHT_SENSOR_TYPE DHT_TYPE_22

static const int DHT_SENSOR_PIN = 2;
DHT_nonblocking dht_sensor( DHT_SENSOR_PIN, DHT_SENSOR_TYPE );

/*
 * Initialize the serial port.
 */
void setup()
{
    Serial.begin( 9600 );
}

/*
 * Poll for a measurement, keeping the state machine alive. Returns
 * true if a measurement is available.
 */
static bool measure_environment( float *temperature, float *humidity )
{
    static unsigned long measurement_timestamp = millis();

    /* Measure once every four seconds. */
    if( millis() - measurement_timestamp > 3000UL )
    {
        if( dht_sensor.measure( temperature, humidity ) == true )
        {
            measurement_timestamp = millis();
            return( true );
        }
    }
    return( false );
}

/*
 * Main program loop.
 */

```

```

/*
 * Poll for a measurement, keeping the state machine alive. Returns
 * true if a measurement is available.
 */
static bool measure_environment( float *temperature, float *humidity )
{
    static unsigned long measurement_timestamp = millis( );
    /* Measure once every four seconds. */
    if( millis( ) - measurement_timestamp > 3000ul )
    {
        if( dht_sensor.measure( temperature, humidity ) == true )
        {
            measurement_timestamp = millis( );
            return( true );
        }
    }
    return( false );
}

/*
 * Main program loop.
 */
void loop( )
{
    float temperature;
    float humidity;

    /* Measure temperature and humidity. If the functions returns
     * true, then a measurement is available. */
    if( measure_environment( &temperature, &humidity ) == true )
    {
        Serial.print( "T = " );
        Serial.print( temperature, 1 );
        Serial.print( " deg. C, H = " );
        Serial.print( humidity, 1 );
        Serial.println( "%" );
    }
}

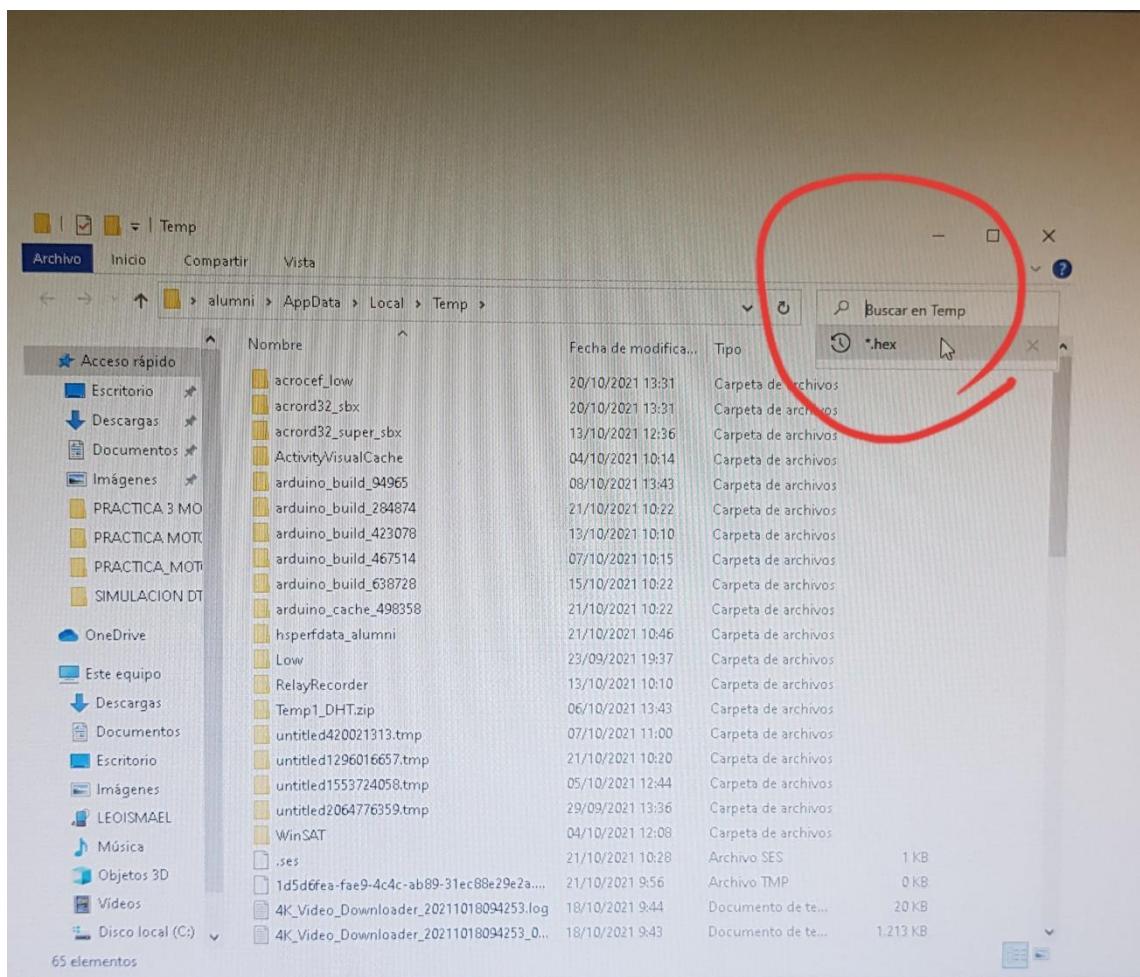
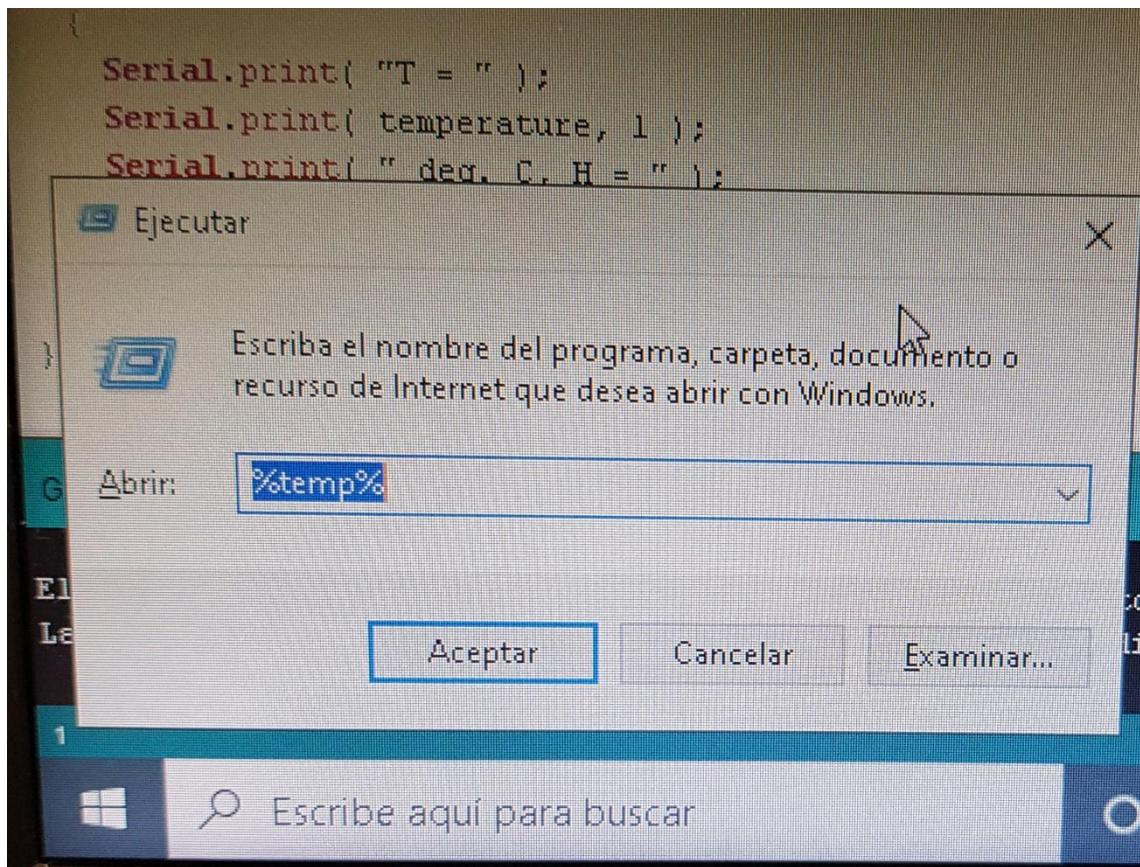
```

Revisamos que nuestro código no tenga fallos como podemos apreciar en el gráfico tanto

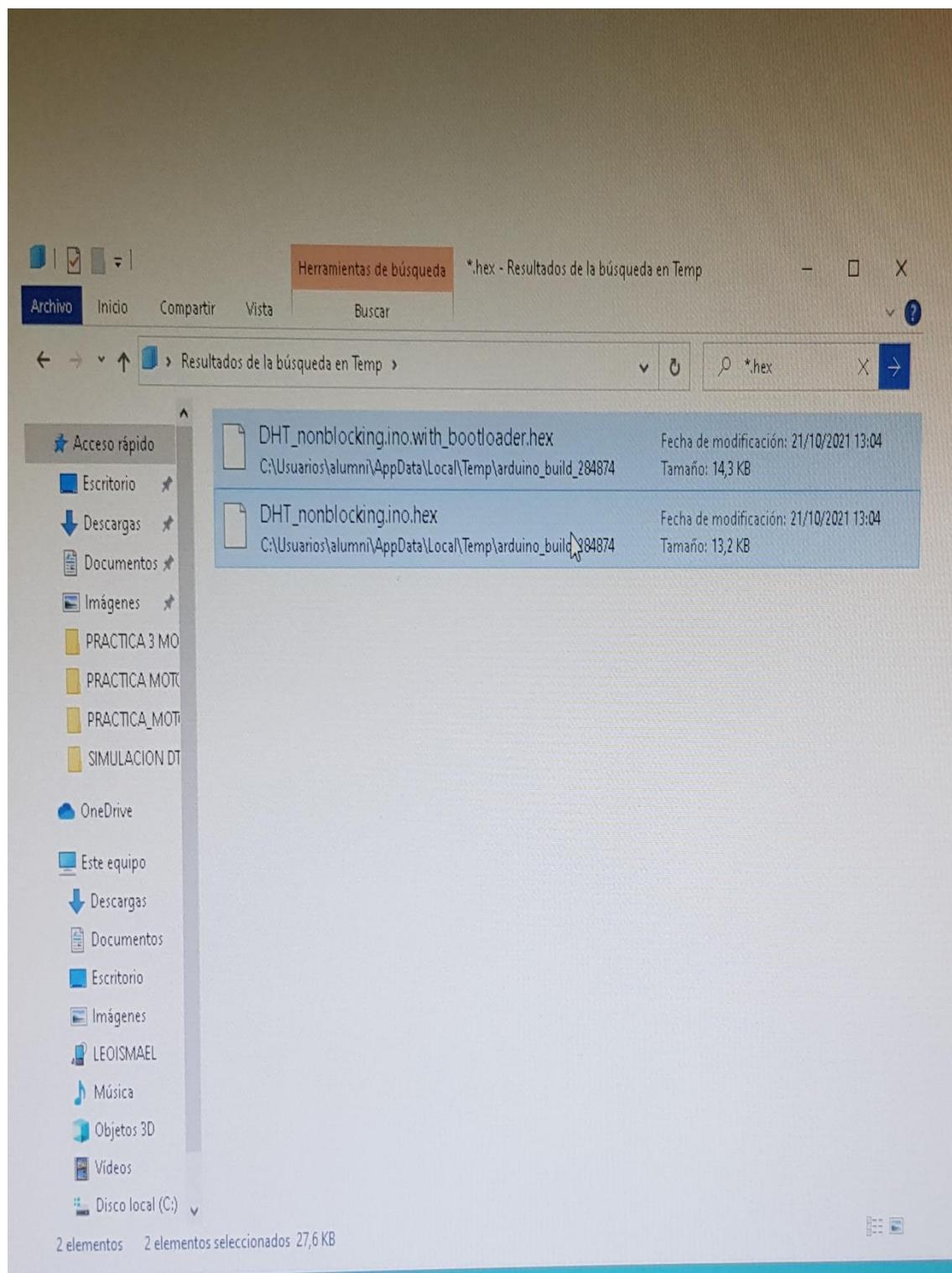
en **setup** como en **loop** y compilamos la información esto genera un archivo hexadecimal el

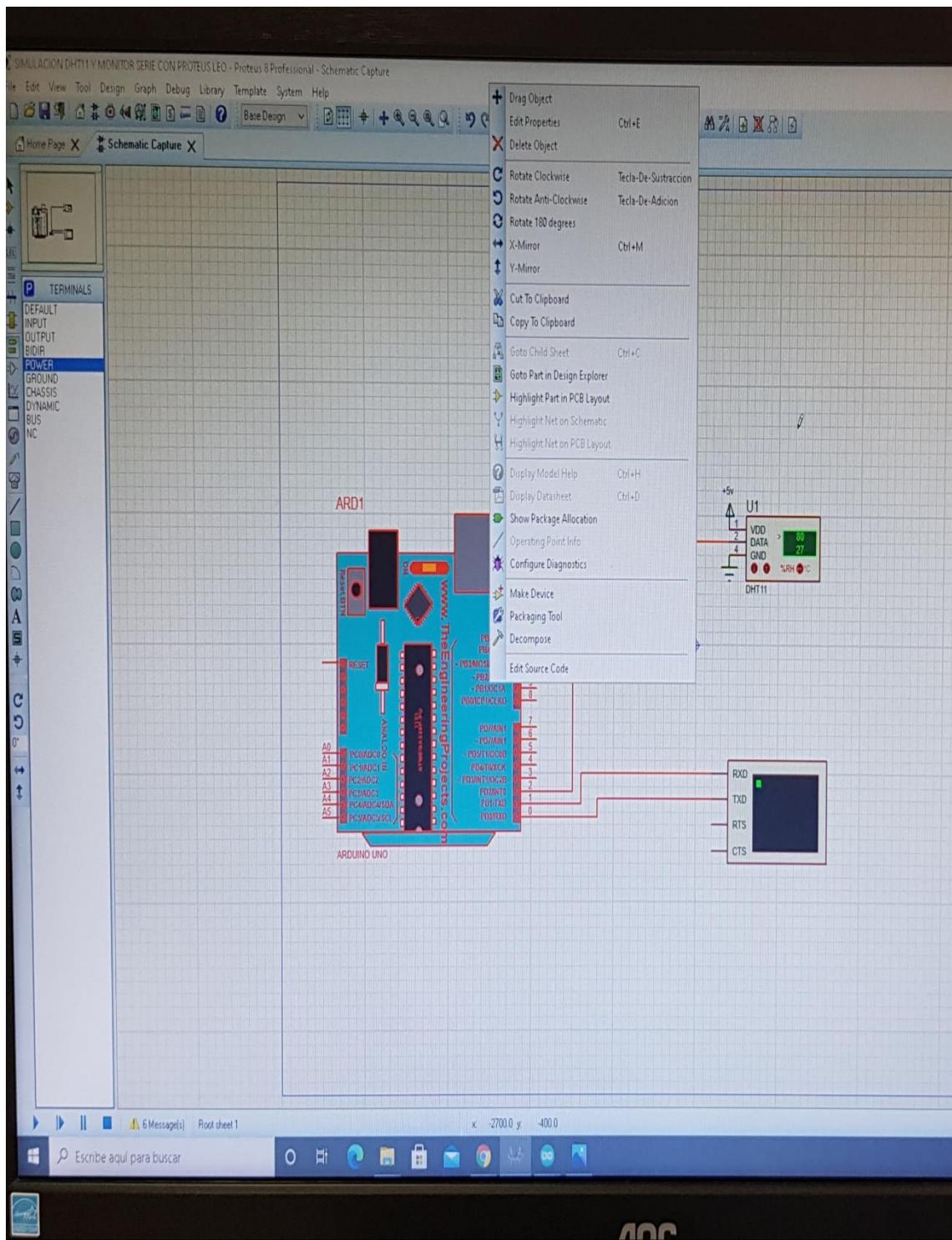
cual subiremos al **ARDUINO** para poder hacer la simulación, para conseguirlo presionamos

la tecla **Start R** y le damos al **ACEPTAR**

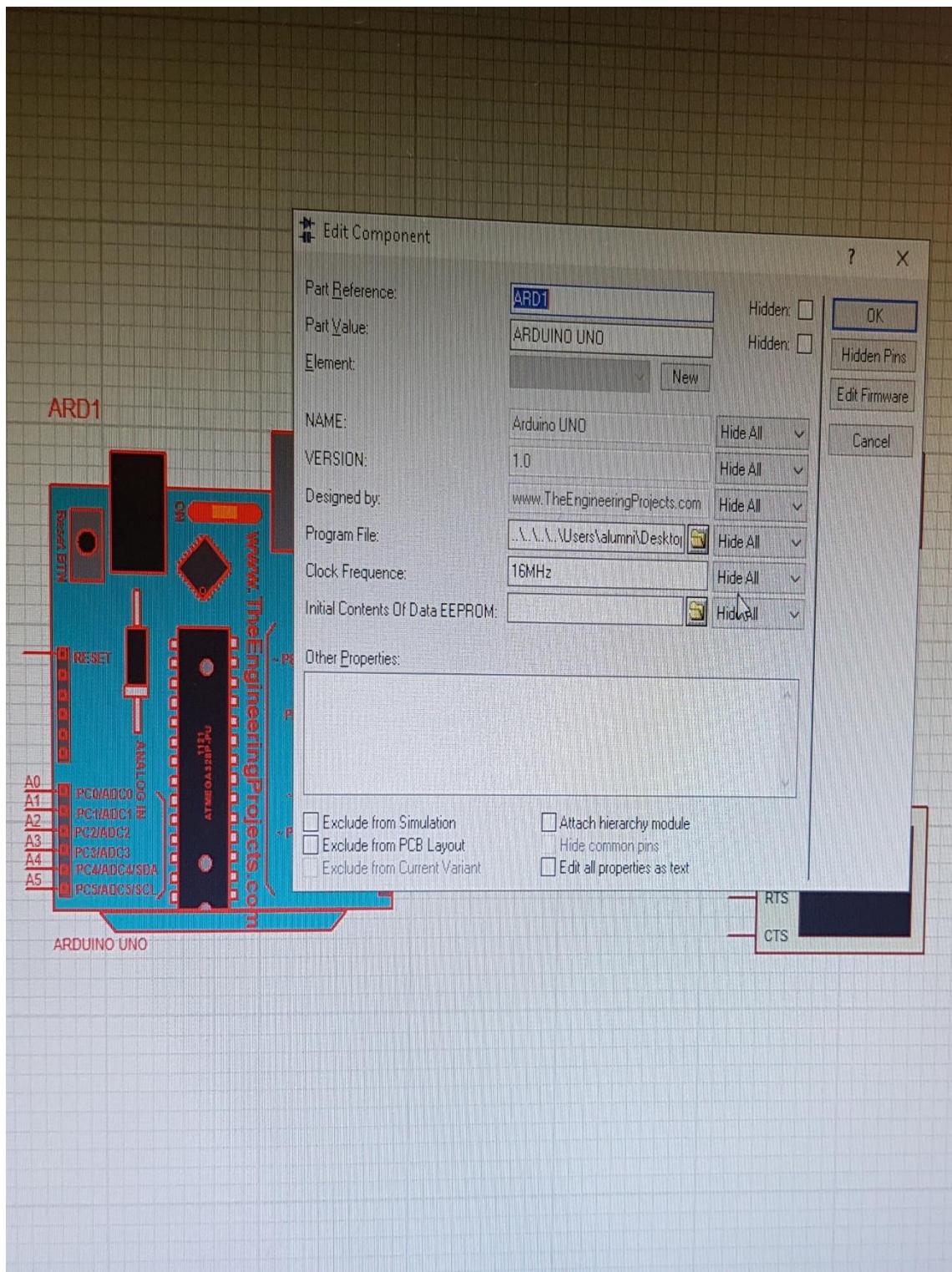


Y nos enviara a una carpeta de archivo temporal nos vamos al buscador y buscamos los archivos ***.hex** lo copiaremos en el escritorio

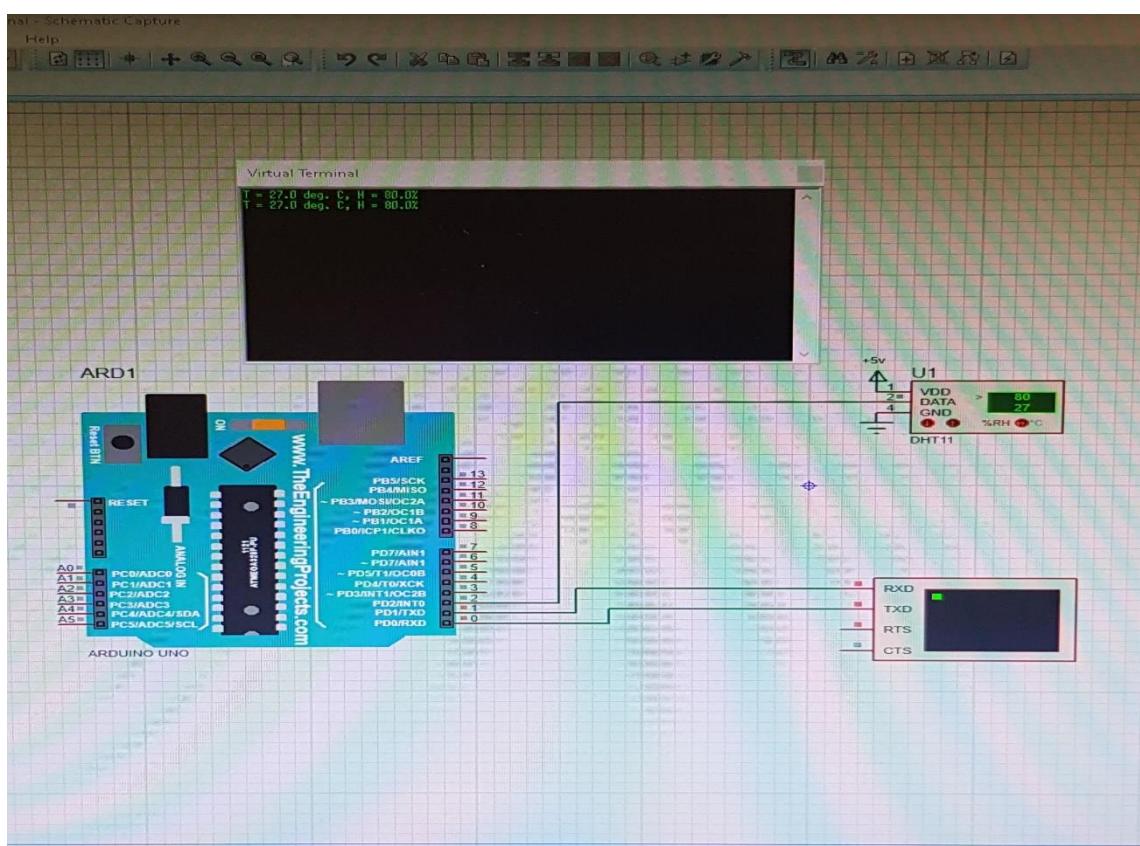
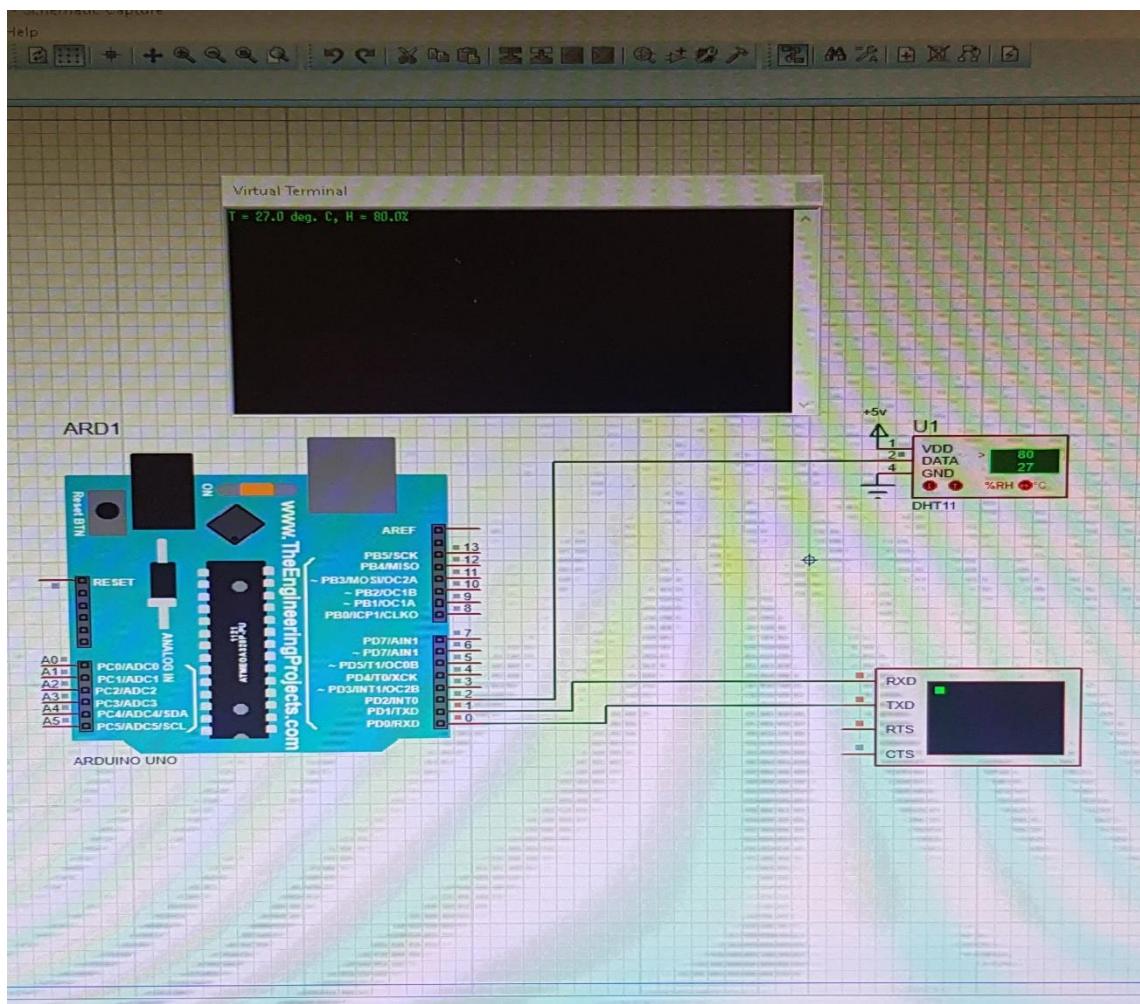


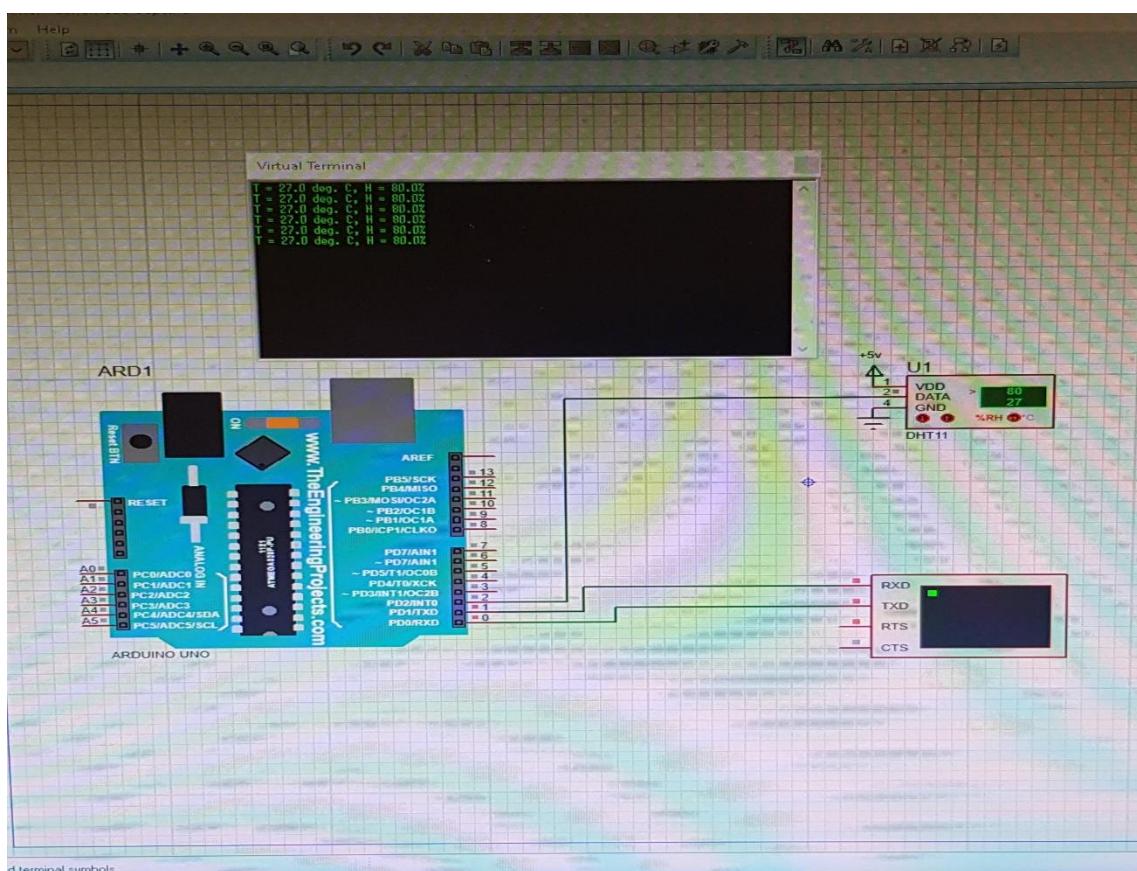
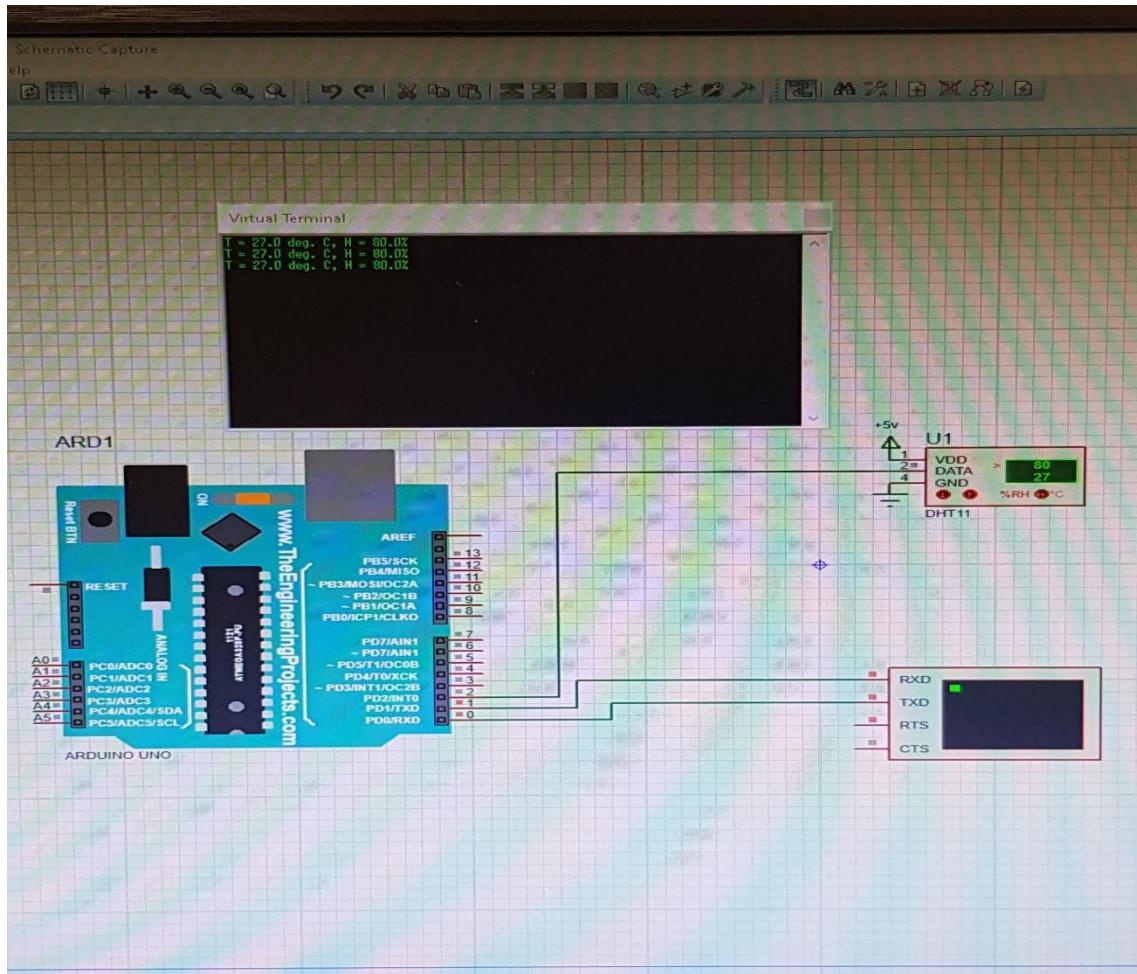


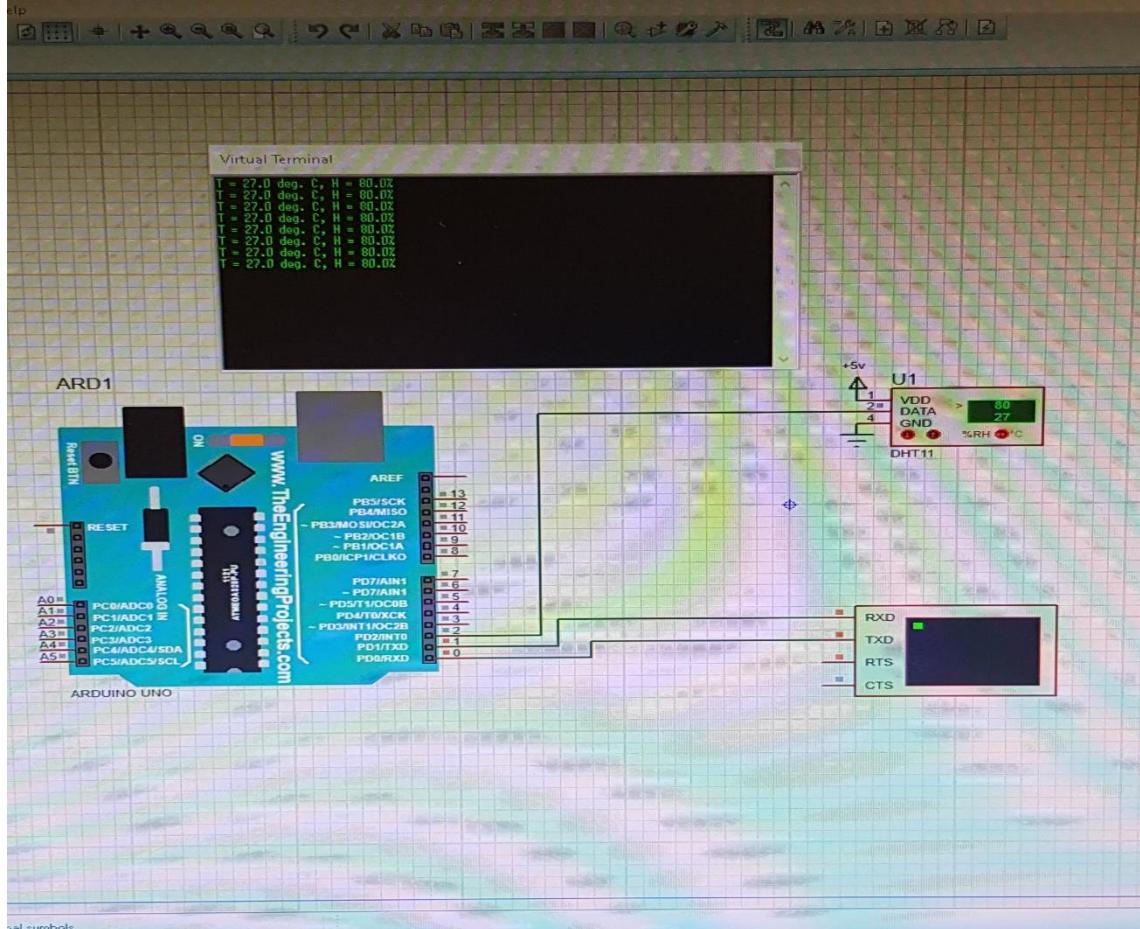
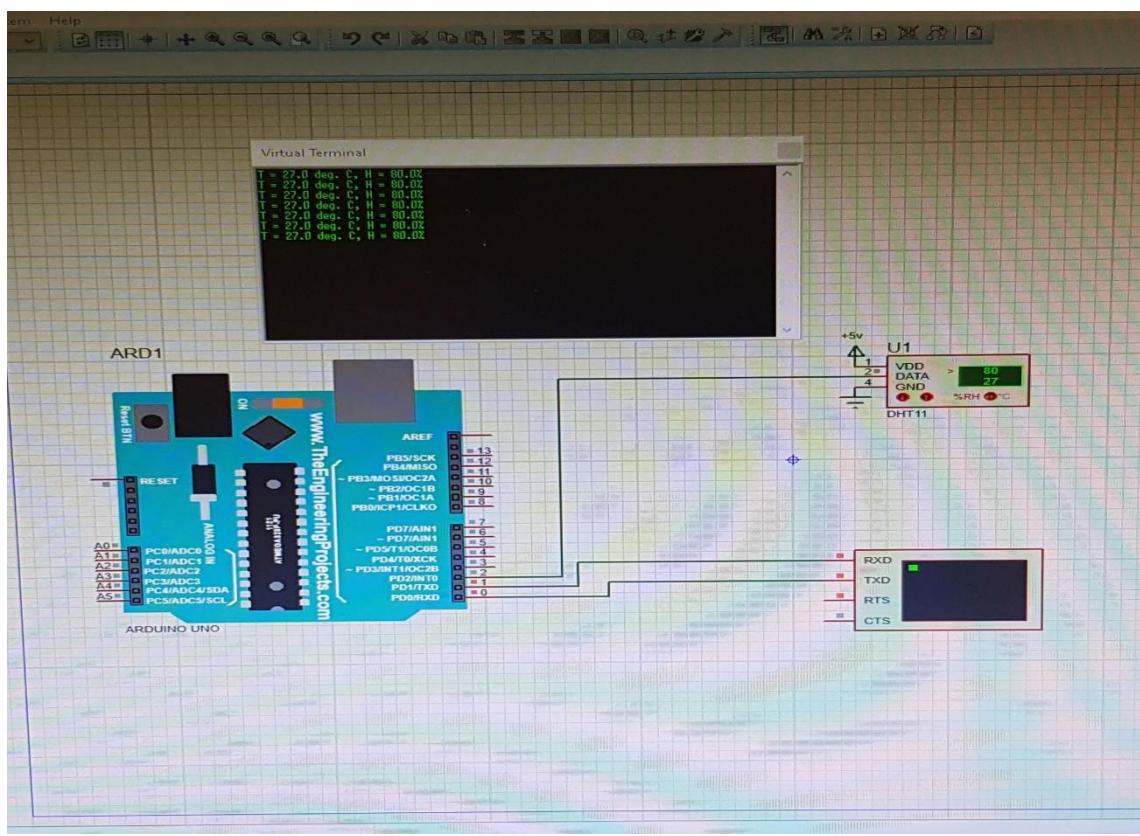
Una vez guardado en escritorio abrimos el **PROTEUS**, vamos al **ARDUINO** edición de propiedades, verificamos los datos completamos y subimos el **archivo hexadecimal** y le damos al simulador

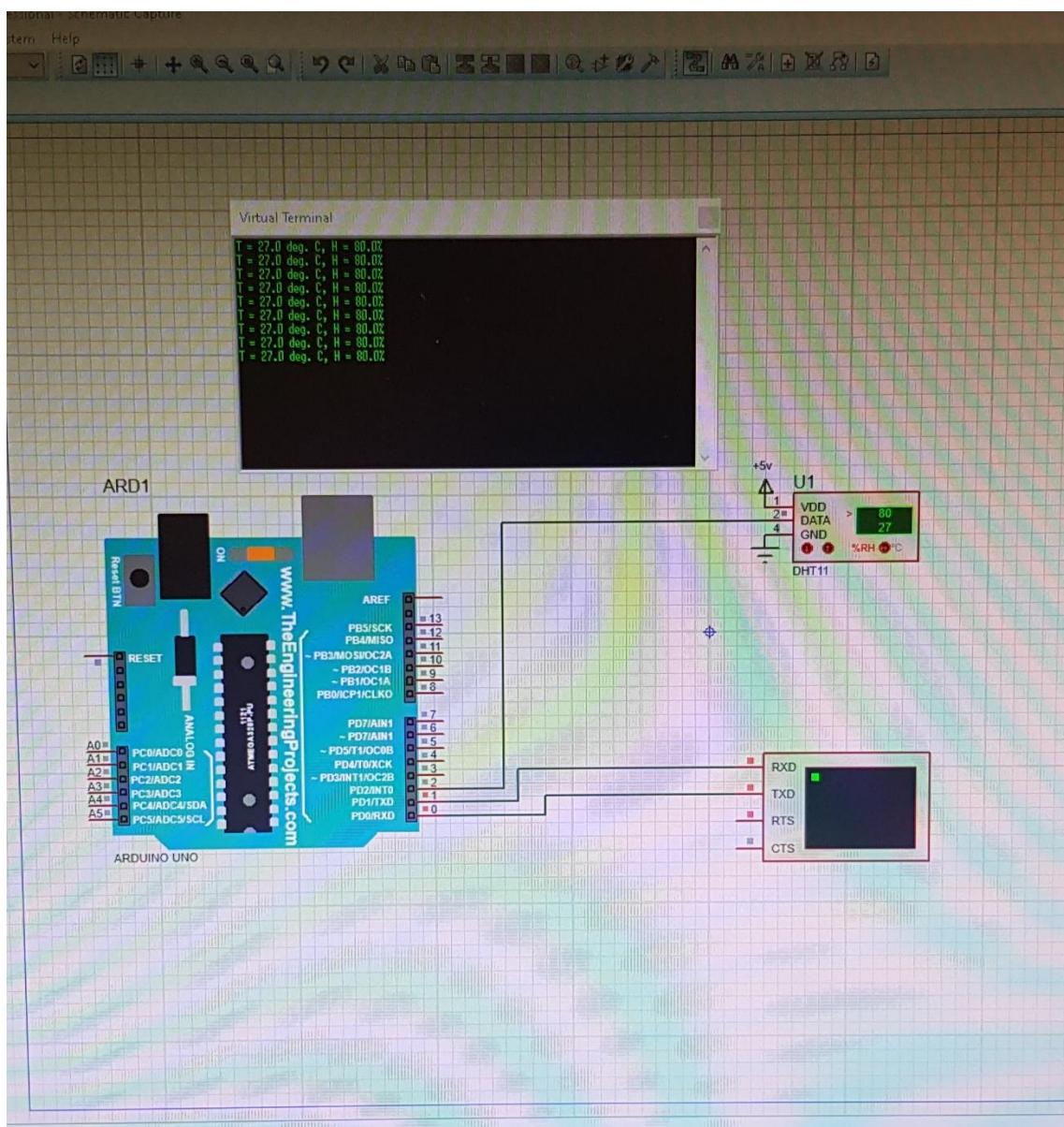


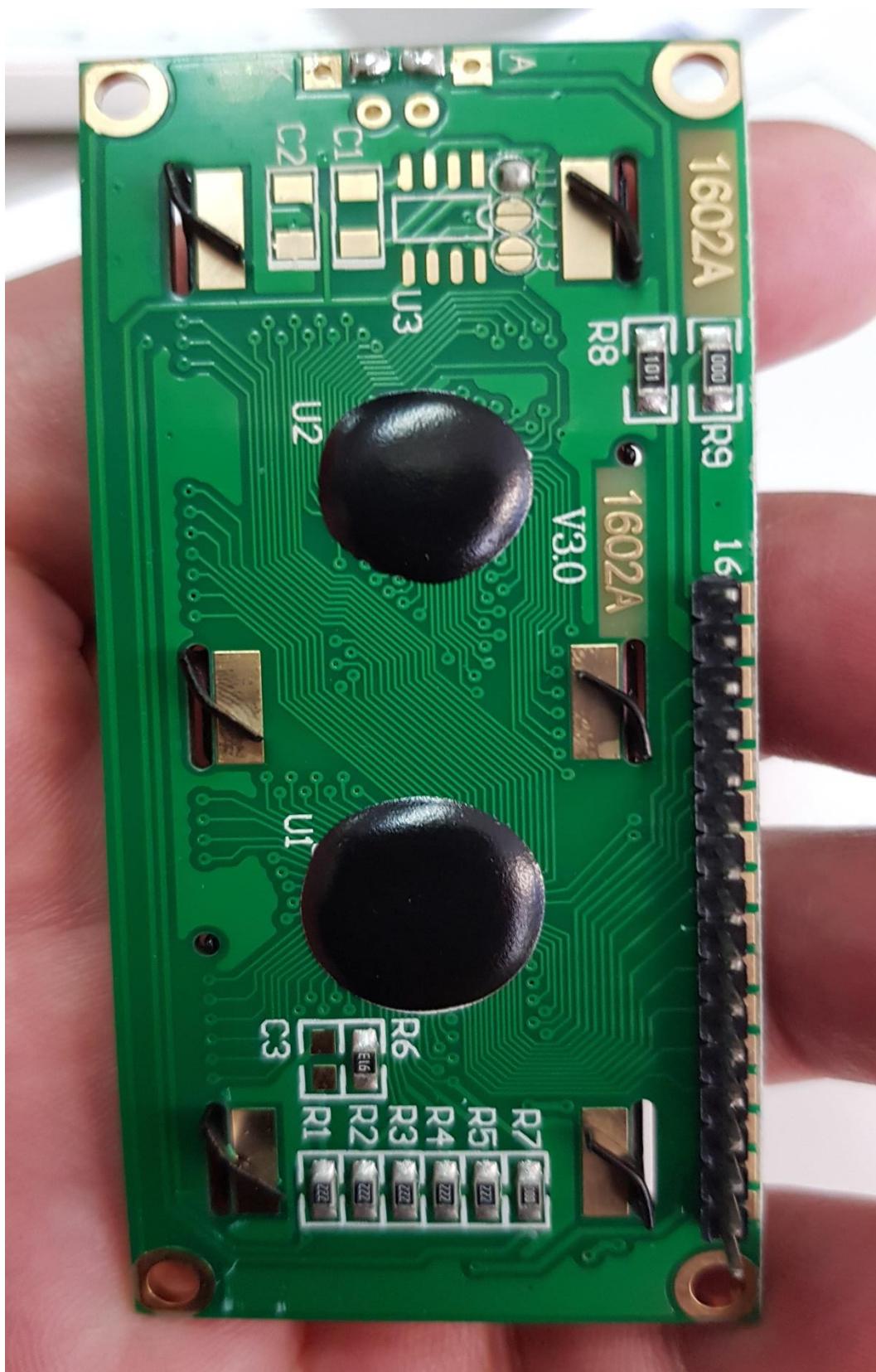
Finalmente podemos mirar en el **monitor** la lectura que nos va recogiendo en **milisegundos**







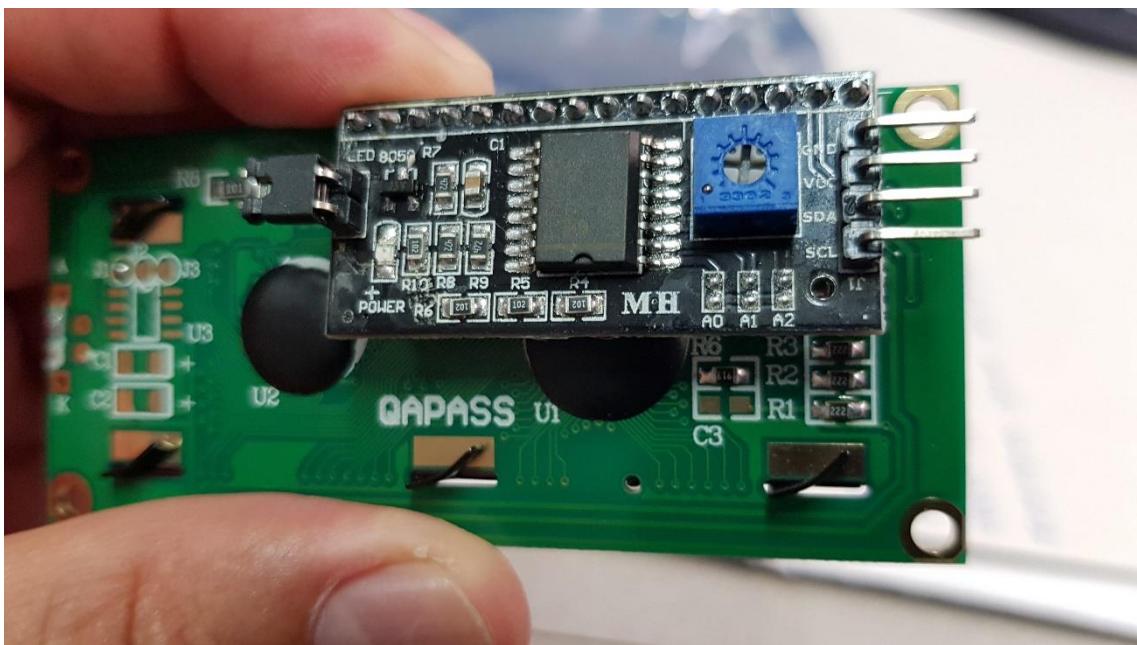


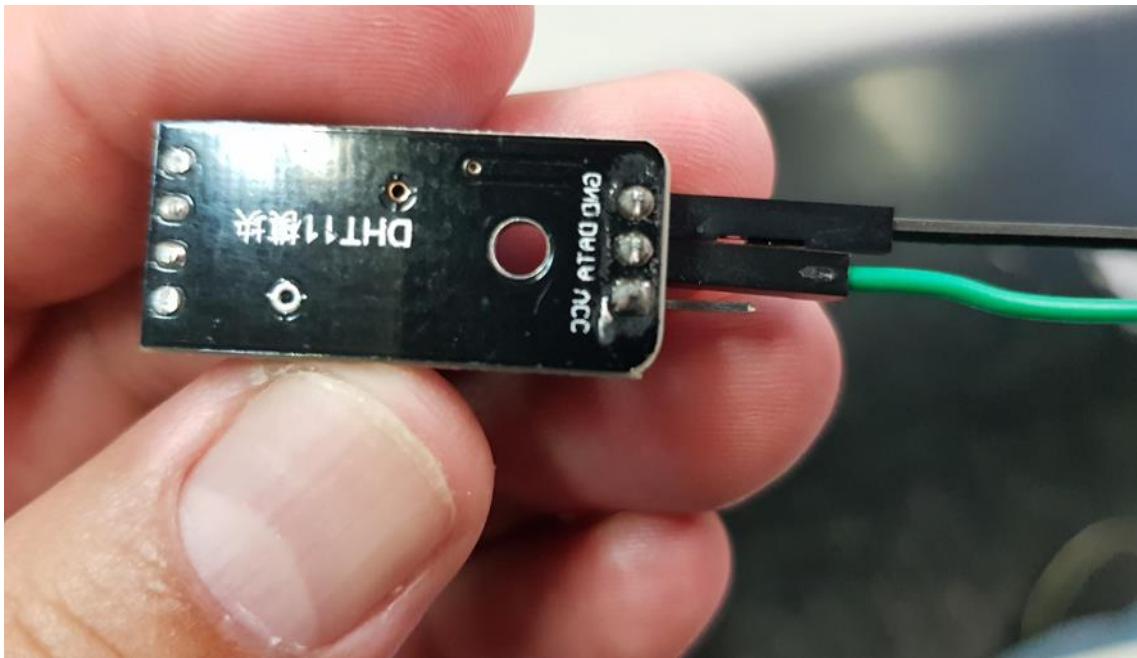


Placa LCD 1602A Cara posterior.

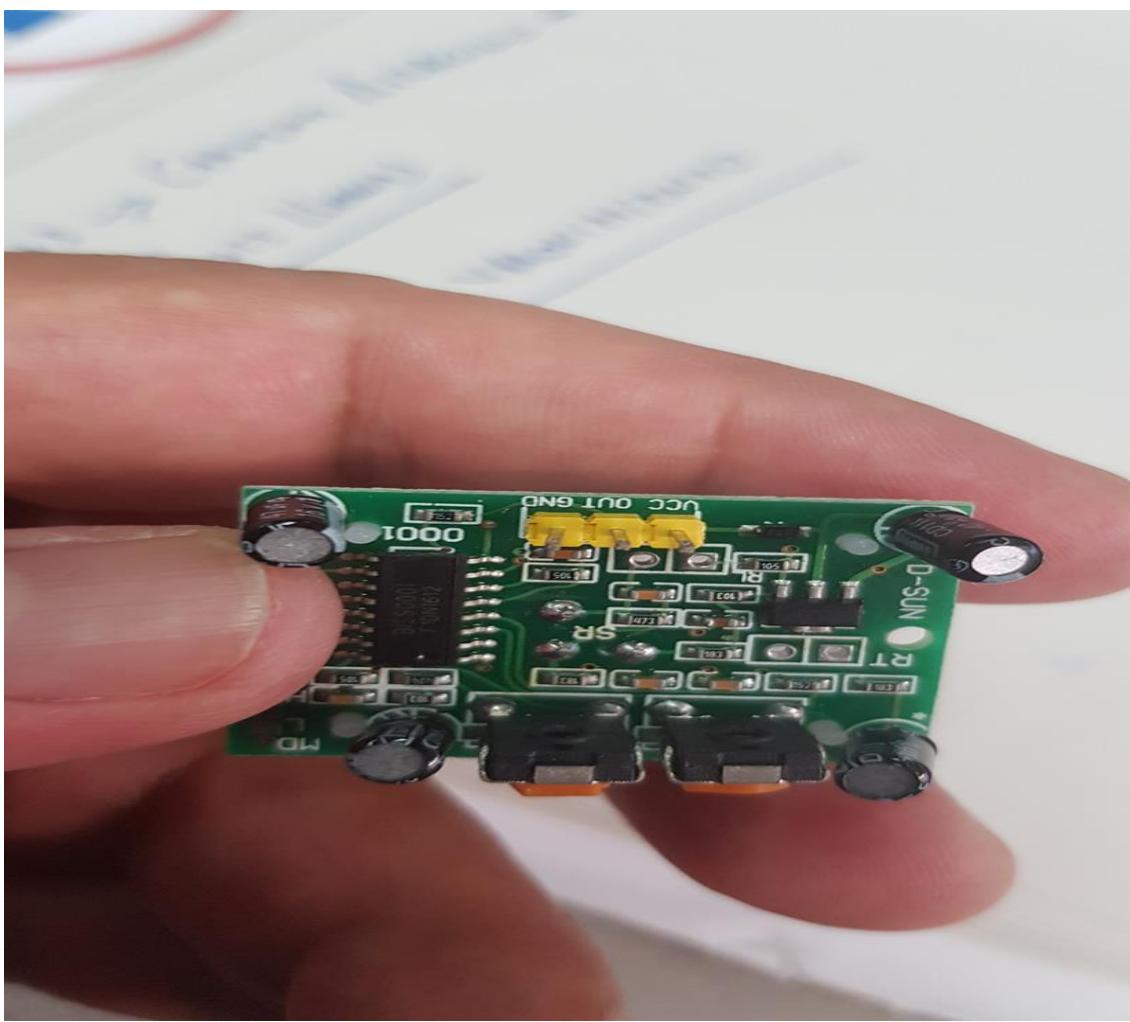


Placa LCD 1602A Cara anterior.





Sensor de temperatura DTH11



Sensor de MOVIMIENTO

