Implementación de una Estación de Monitoreo y Adquisición de Datos

Arduino & Visual Basic

Proyecto:

Desarrollo de una Estación de Monitoreo y Adquisición de datos a distancia con Arduino-VisualBasic.

Equipo:

Alanís Serrato Melanie Guadalupe.
López Negrete Cristina
Pérez Vázquez Roberto Jafet
Pérez Vázquez Bruno Daniel
Salazar Parada Carlos Arturo
Sánchez Ramírez Tristán Azael

Asesor:
Dr. Juan Antonio Sánchez Márquez

INDICE:

1 Generalidades de Arduino y VisualBasic.NET 6
1.1 Microcontrolador Arduino 6
1.2 Sensores usados con Arduino 7
1.2.1 Sensor de Temperatura LM35. 7
1.2.2 Sensor Ultrasónico HC-SR04. 8
1.2.3 Sensor de Fuerza Resistivo Fsr402. 9
1.2.4 Fotorresistencia-LDR. 10
1.2.5 Sensor de inclinación Sw-520d. 11
1.3 Autodesk Tinkercad Circuits. 12
1.4 Visual Basic.NET 13
2 Arreglos, Montaje y Calibración de Sensores. 14
2.2 Sensor de Temperatura LM35 14
2.2.1 Arreglo Electrónico. 14
2.2.2 Sketch (Código Arduino) 15
2.2.3 Funcionamiento del Sensor. 17
2.3 Sensor Ultrasónico Hc-sr04. 18
2.4 Sensor de Fuerza Resistivo Fsr402. 21
2.5 Fotorresistencia Ldr. 26
3 Montaje de la Estación de Monitoreo. 33
4 Registro y Respaldo de Datos (Modulo SD). 46
5 Sistema de Monitoreo y Adquisición a través del Módulo Bluetooth HC-05. 49
6 Bibliografía/Referencias 53

INDICE DE FIGURAS:

Figura 1. Sensor de Temperatura LM35 8
Figura 2. Sensor Ultrasónico Hc-sr04 9
Figura 3 Sensor de Fuerza Resistivo Fsr402 10
Figura 4 Fotorresistencia-LDR(Light Depending Resistor) 11
Figura 5. Sensor de Inclinación Sw-520d de doble esfera 12
Figura 6. Estructura Interna del Sensor de Inclinación Sw-530d de doble esfera 12
Figura 7. Autodesk Thinkercad Circuits. 12
Figura 8. Arreglo del Sensor de Temperatura LM35 desarrollando en Autodesk 14
Figura 9.Funcionamiento del Sensor de Temperatura LM35 en el arreglo electrónico montado en Autodesk Tinkercad Circuits 17
Figura 10.Funcionamiento del Sensor de Temperatura LM35 en el arreglo electrónico montado en Autodesk Tinkercad Circuits 17
Figura 11. Arreglo Electrónico del Sensor Ultrasónico Hc-sr04 desarrollado en Autodesk Tinkercad Circuits. 18
Figura 12. Valores medidos por el Sensor Ultrasónico Hc-sr04 mostrados en la pantalla LCD. 20
Figura 13.Medición de los valores de distancia obtenidos por arreglo Electrónico del Sensor Ultrasónico Hc-sr04 desarrollado en Autodesk Tinkercad Circuits. 21
Figura 14.Arreglo Electrónico del Sensor de Fuerza Resistivo Fsr402 desarrollado en Autodesk Tinkercad Circuits. 21
Figura 15. Valores medidos por el sensor de fuerza Resistivo Fsr402 mostrados en la pantalla LCD. 25
Figura 16.Indicadores de presión visual (LED Amarillo) y digital (Pantalla LCD-Presión Moderada) 25
Figura 17.Arreglo Electrónico del Sensor LDR desarrollado en Autodesk Tinkercad Circuits. 26
Figura 18. Valores medidos por el sensor LDR mostrados en la pantalla LCD. 28
Figura 19.Intensidad de luz moderada medida por el sensor LDR mostrada en la pantalla LCD. 29
Figura 20.Intensidad de luz alta medida por el sensor LDR mostrada en la pantalla LCD. 29
Figura 21.Arreglo electrónico del sensor de inclinación Sw-520d desarrollado en Autodesk Tinkercad Circuits. 30
Figura 22.Sensor de inclinación Sw-520d en posición desarrollado en Autodesk Tinkercad Circuits. 32
Figura 23. Cambio de posición del Sensor de inclinación Sw-520d a un esquema inclinado desarrollado en Autodesk Tinkercad Circuits 33
Figura 24.Consola de Comunicación del IDE de Arduino 34
Figura 25.Consola de Comunicación serial simulada en Tinkercad. 35
Figura 26. Incorporación del control SerialPort al Formulario. 36
Figura 27. Incorporación de botones en el formulario. 36

Figura 28. Abrir la comunicación desde la aplicación por medio del comando SerialPort.Open..... 36
Figura 29.Programación de los botones 1 y 2...... 37
Figura 30. Encendiendo y Apagando el LED de la Tarjeta Arduino con Visual Basic.NET. 37
Figura 31. Arreglo Electrónico de la Estación de Monitoreo..... 38
Figura 32. Conexión de Sensores en la Estación de Monitoreo. 39
Figura 33. Sketch de Arduino de la estación de monitoreo en funcionamiento (Valores reportados en el monitor serial). 41
Figura 34. Valores reportados en la pantalla LCD. 42
Figura 35. Cambios realizados en el Sketch para mostrar los valores obtenidos por los sensores en una sola línea. 42
Figura 36. Valores mostrados en el monitor serial en formato de una sola línea. 43
Figura 37. Control SerialPort. 44
Figura 38. Controles incluidos en la aplicación de la Estación de Monitoreo..... 44
Figura 39. Estación de Monitoreo desarrollada con Arduino y Visual Basic.NET..... 45
Figura 40. Modulo Micro SD de la Tarjeta de Arduino agregado a la Estación de Monitoreo. 46
Figura 41. Funcionamiento Sketch desarrollado para la incorporación del Módulo MicroSD. 48
Figura 42. Arreglo Electrónico de la Estación de Monitoreo y Adquisición de Datos con un módulo Bluetooth HC-05..... 50
Figura 44. Información del sistema de la Configuración de los puertos del módulo BT 52
Figura 43. Vinculación del Módulo Bluetooth con la PC..... 52
Figura 45. Ajuste del control SerialPort para establecer conexión con el módulo BT. 53

INDICE DE CODIGOS:

Sketch (Codigo) 1. Sensor de Temperatura 15
Sketch (Codigo) 2. Sensor Ultrasonic Distance 19
Sketch (Codigo) 3. Sensor de Fuerza Resistivo Fsr402. 22
Sketch (Codigo) 4.Sensor LDR 27
Sketch (Codigo) 5.Sensor de inclinación Sw-520d 31
Sketch (Codigo) 6. Prueba de Comunicación de Visual Basic.NET 33
Sketch (Codigo) 7. Sketch de Arduino desarrollado para la Estación de Monitoreo. . 40
Sketch (Codigo) 8. Código del formulario y del Control Timer. 45
Sketch (Codigo) 9. se presenta el Sketch desarrollado en el IDE de Arduino a fin de realizar el respaldo local de los datos 47
Sketch (Codigo) 10. Sketch desarrollado para la incorporación del Módulo MicroSD. 47
Sketch (Codigo) 11. Funcionamiento Sketch desarrollado para la incorporación del Módulo MicroSD. 49
Sketch (Codigo) 12. Sketch desarrollado en el IDE para establecer comunicación entre la PC y el módulo Bluetooth a través de un microcontrolador Arduino 50
Sketch (Codigo) 13. Sketch desarrollado en el IDE para establecer comunicación entre la PC y el módulo Bluetooth a través de un microcontrolador Arduino 51
Sketch (Codigo) 14. Configuración del Módulo HC-05 como esclavo. 52

INDICE DE TABLAS:

Tablas 1. Componente Principal del Arreglo 14
Tablas 2.Arreglo Electrónico del Sensor Ultrasónico Hc-sr04 desarrollado en Autodesk Tinkercad Circuits. 18
Tablas 3. Componentes Principales del Arreglo. 22
Tablas 4. Sensor LDR 26
Tablas 5. Sensor de inclinación Sw-520d 30

Desarrollo de una Estación de Monitoreo y Adquisición de datos a distancia con Arduino-VisualBasic.

Alanís Serrato Melanie Guadalupe (LICENCIATURA EN INGENIERÍA EN AGRONOMÍA)
López Negrete Cristina (LICENCIATURA EN INGENIERÍA QUÍMICA)
Pérez Vázquez Roberto Jafet (LICENCIATURA EN INGENIERÍA EN COMUNICACIONES Y ELECTRÓNICA)
Pérez Vázquez Bruno Daniel (LICENCIATURA EN INGENIERÍA EN SISTEMAS COMPUTACIONALES)
Salazar Parada Carlos Arturo (LICENCIATURA EN INGENIERÍA CIVIL)
Sánchez Ramírez Tristán Azael (Escuela del Nivel Medio Superior de Guanajuato)

Asesor: Dr. Juan Antonio Sánchez Márquez

1 Generalidades de Arduino y VisualBasic.NET

1.1 Microcontrolador Arduino

Arduino es una plataforma electrónica de código abierto basada en hardware y software gratuitos y fáciles de usar. Las placas Arduino pueden leer entradas (luz en un sensor, un dedo en un botón o un mensaje de Twitter) y convertirlo en una salida, activando un motor, encendiendo un LED, publicando algo en línea, etc. Para llevar a cabo estas acciones el microcontrolador Arduino, utiliza un lenguaje de programación basado en Wiring y un Software denominado Arduino (IDE), basado en Processing[1].

A lo largo de los años, Arduino ha sido el cerebro de miles de proyectos, desde objetos cotidianos hasta complejos instrumentos científicos. Una comunidad mundial de creadores (estudiantes, aficionados, artistas, programadores y profesionales) se ha reunido en torno a esta plataforma de código abierto; sus contribuciones se han sumado a una increíble cantidad de conocimiento accesible que puede ser de gran ayuda tanto para principiantes como para expertos[1].

Arduino nació en el Ivrea Interaction Design Institute como una herramienta fácil para la creación rápida de prototipos, dirigida a estudiantes sin experiencia en electrónica y programación. Tan pronto como llegó a una comunidad más amplia, la placa Arduino comenzó a cambiar para adaptarse a las nuevas necesidades y desafíos, diferenciando su oferta desde placas simples de 8 bits hasta productos para aplicaciones de IoT, wearables, impresión 3D y entornos integrados. Todas las placas Arduino son completamente de código abierto, lo que permite a los usuarios construirlas de forma independiente y eventualmente adaptarlas a sus necesidades particulares. El software también es de código abierto y está creciendo gracias a las contribuciones de los usuarios de todo el mundo[1].

Arduino también simplifica el proceso de trabajar con microcontroladores, pero ofrece algunas ventajas para profesores, estudiantes y aficionados interesados sobre otros sistemas[1]:

a) **Es Económico**: las placas Arduino son relativamente económicas en comparación con otras plataformas de microcontroladores. La versión menos costosa del módulo Arduino se puede ensamblar a mano.
b) **Es Multiplataforma:** el software Arduino (IDE) se ejecuta en los sistemas operativos Windows, Macintosh OSX y Linux. La mayoría de los sistemas de microcontroladores están limitados a Windows.
c) **Tiene un entorno de programación simple y claro:** el software Arduino (IDE) es fácil de usar para principiantes, pero lo suficientemente flexible para que los usuarios avanzados también lo aprovechen. Para los profesores, se basa convenientemente en el entorno de programación de procesamiento, por lo que los estudiantes que aprenden a programar en ese entorno estarán familiarizados con el funcionamiento del IDE de Arduino.
d) **Maneja software de código abierto y extensible**: el software Arduino se publica como herramientas de código abierto, disponibles para su extensión por programadores experimentados. El lenguaje se puede expandir a través de bibliotecas C++, y las personas que quieran comprender los detalles técnicos pueden dar el salto de Arduino al lenguaje de programación AVR-C en el que se basa. Del mismo modo, puede agregar código AVR-C directamente en sus programas Arduino si lo desea.
e) **Maneja hardware de código abierto y extensible**: los planos de las placas Arduino se publican bajo una licencia de Creative Commons, por lo que los diseñadores de circuitos experimentados pueden crear su propia versión del módulo, ampliarlo y mejorarlo. Incluso los usuarios relativamente inexpertos pueden construir la versión de tablero del módulo para comprender cómo funciona y ahorrar dinero.

1.2 Sensores usados con Arduino

1.2.1 Sensor de Temperatura LM35.

El sensor LM35 (Figura 1) es un circuito electrónico sensor que puede medir temperatura. Su salida es analógica, es decir, te proporciona un voltaje proporcional a la temperatura. El sensor tiene un rango desde −55°C a 150°C. Su popularidad se debe a la facilidad con la que se puede medir la temperatura. Incluso no es necesario de un microprocesador o microcontrolador para medir la temperatura. Dado que el sensor LM35 es analógico, basta con medir con un multímetro, el voltaje a salida del sensor. Para convertir el voltaje a la

temperatura, el LM35 proporciona 10mV por cada grado centígrado. También cabe señalar que ese sensor se puede usar sin offset, es decir que, si medimos 20mV a la salida, estaremos midiendo 2°C.

Sus Características Técnicas son las siguientes[2]:

Resolución: 10mV por cada grado centígrado.
Voltaje de alimentación: Esté sensor se puede alimentar desde 4V hasta 20V DC.
Tipo de medición: Salida analógica.
Numero de pines: 3 pines, GND, VCC y V_{Salida}.
Calibración: No requiere calibración.
Precisión: Tiene una precisión de ±¼°C.
Unidades de medición: Esta calibrado para medir °C.
Consumo de corriente: 60 µA
Empaquetados comunes: TO-CAN, TO-220, TO-92 y SOIC8.

***Figura 1**. Sensor de Temperatura LM35*

1.2.2 Sensor Ultrasónico HC-SR04.

El sensor HC-SR04 (Figura 2) es un sensor de distancia de bajo costo que utiliza ultrasonido para determinar la distancia de un objeto, esto lo hace por medio de dos transductores un emisor y un receptor piezoeléctrico, el emisor genera 8 pulsos de ultrasonido a 40kHz, el sonido choca con el objeto del cual se requiere medir la distancia a la que se encuentra, y nuestro receptor lo detecta.

Sus características técnicas son las siguientes[3]:

- **Voltaje de Operación:** 5V DC
- **Corriente de reposo:** < 2mA
- **Corriente de trabajo:** 15mA
- **Ángulo de apertura:** 15°
- **Frecuencia de ultrasonido:** 40KHz
- **Duración mínima del pulso de disparo TRIG (nivel TTL)**: 10 µS
- **Duración del pulso ECO de salida (nivel TTL):** 100-25000 µS

- **Dimensiones:** 45mm x 20mm x 15mm

Este sensor tiene un rango de medición de 2cm a 450cm y tiene una precisión de +/- 3mm.

***Figura 2**. Sensor Ultrasónico Hc-sr04*

1.2.3 Sensor de Fuerza Resistivo Fsr402.

El sensor Fsr402 (Figura 3) es un componente pasivo que exhibe una disminución de la resistencia cuando se produce un aumento en la fuerza aplicada a la superficie activa de, lo que permite crear un sensor que es capaz de detectar la fuerza o presión[4].

Sus características técnicas son las siguientes:

- **Voltaje de Operación:** 2.6-5.5V
- **Resistencia sin actuación:** > 10 MΩ
- **Corriente de típica:** 1.7 mA
- **Dimensiones:** 1,7 cm * 2,9 cm * 0,4 cm
- **Comunicación:** Se comunica con el microcontrolador por medio de un protocolo de tipo serial mediante 2 pines (Clock y Data).
- **Peso:** 0.01 kg
- **Medición:** Rango de medición de 0.2 N a 20 N (20.4 gf a 2.039 kgf).

Figura 3 *Sensor de Fuerza Resistivo Fsr402*

1.2.4 Fotorresistencia-LDR.

Una fotorresistencia o LDR (Light Depending Resistor, o resistencia dependiente de la luz, Figura 4) es un componente fotoelectrónico cuya resistencia varía en función de la luz que incide en él. Esta resistencia es muy baja, de unos pocos Ωs con una luz intensa incide en él y va creciendo fuertemente a medida que esa luz decrece. Se les suele utilizar como sensores de luz, para arrancar luces automáticamente cuando la oscuridad sobrepasa un cierto umbral, o como detectores de movimiento próximo, cuando algo se interpone.

Para poder utilizarlo solo debes agregar una resistencia de 10K en serie y conectarlo a una de las entradas analógicas de tu Arduino (pines del A0-A5) como un divisor de tensión.

Las especificaciones técnicas del sensor son las siguientes[5]:

- **Modelo:** LDR GL5528
- **Resistencia en luz (10 lux):** 8K-20K Ohm
- **Resistencia en oscuridad:** 1M Ohm
- **Voltaje máx:** 150V
- **Potencia máx:** 100mW
- **Material fotosensible:** CdS (Sulfato de Sodio)
- **Frecuencia de luz pico:** 540 nm
- **Tamaño:** 5mm.

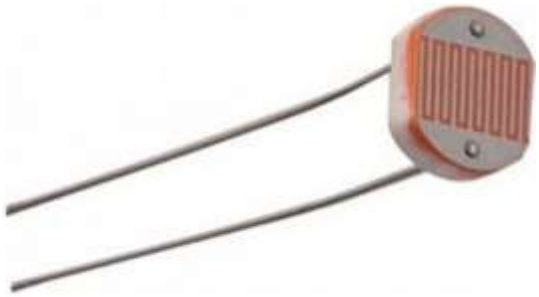


Figura 4 *Fotorresistencia-LDR(Light Depending Resistor)*

1.2.5 Sensor de inclinación Sw-520d.

Un sensor de inclinación Sw-520d (Figura 5) es un dispositivo que proporciona una señal digital en caso de que su inclinación supere un umbral. Este tipo de sensor no permite saber el grado de inclinación del dispositivo, simplemente actúa como un sensor que se cierra a partir de una cierta inclinación.

Antiguamente estos sensores se constituían ubicando una gota de mercurio en el interior de una ampolla de vidrio, en cuyo interior se alojaban dos conductores. A partir de cierta inclinación la gota de mercurio se desplazaba, cerrando el contacto entre ambos conductores. En la actualidad, por motivos medioambientales, casi todos los sensores de mercurio han sido desplazados por sensores Tilt de doble esfera (Figura 6). Se dispone de un cilindro cuya pared constituye un contacto eléctrico, mientras que el otro contacto está localizado en el centro de la base. Al inclinar lo suficiente el dispositivo ambas esferas constituyen un puente entre ambos contactos, cerrando el circuito. Debido a su principio de funcionamiento, estos sensores resultan sensibles a movimientos bruscos y vibraciones[6].

Las características técnicas del sensor son las siguientes:

- **Referencia:** SW-520D
- **Tensión:** <12 voltios
- **Corriente:** <25 mA
- **Temperatura de funcionamiento máxima:** 70 °C
- **Máxima inclinación:** +/- 90 °
- **Tipo de sensor:** bola
- **Separación de terminales:** 0.1

***Figura 5**. Sensor de Inclinación Sw-520d de doble esfera*

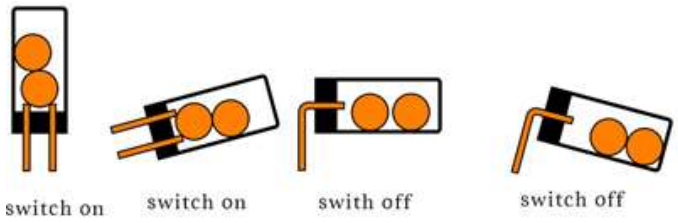


***Figura 6**. Estructura Interna del Sensor de Inclinación Sw-530d de doble esfera*

1.3 Autodesk Tinkercad Circuits.

TinkerCAD es una colección online que incluye herramientas de software de Autodesk que permite a los usuarios crear modelos 3D, además de circuitos electrónicos. Este software CAD se basa en una geometría sólida constructiva (CSG), que permite crear modelos complejos mediante la combinación de objetos más simples. Además, el software hace posible la incorporación de circuitos electrónicos a los diseños 3D para crear objetos con luz y movimiento. Thinkercad circuits es una herramienta que dispone de los elementos necesarios para crear y simular sistemas de control basados en Arduino. Además, permite la

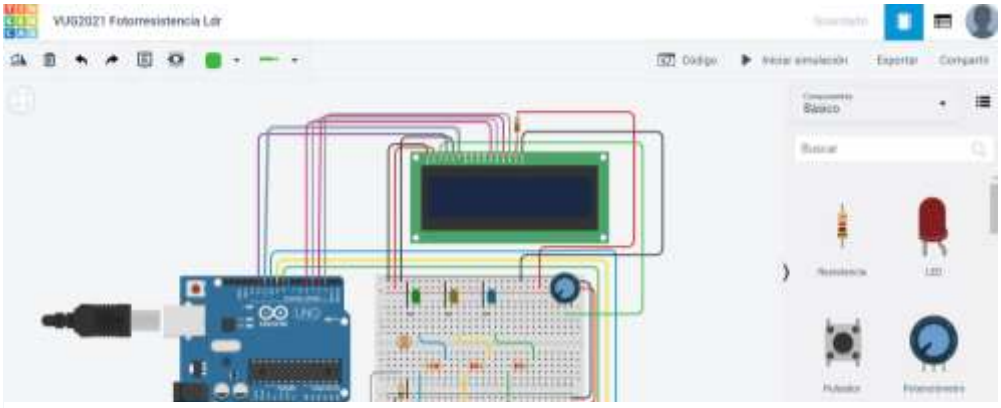


Figura 7. Autodesk Thinkercad Circuits.

programación online de las placas Arduino del simulador. El resultado final incluso se puede simular y con ello se optimizan los resultados obtenidos. Una herramienta muy interesante que ofrece Tinkercad Circuits es el debugger, con ella podemos parar la ejecución de un programa y ver los valores de las variables, algo que con Arduino no podemos hacer. En resumen, con Tinkercad puedes acceder a aplicaciones para diseñar en 3D, crear y simular circuitos eléctricos y electrónicos, programar, etc. Todo ello en un entorno muy sencillo de manejar[7].

1.4 Visual Basic.NET

Visual Basic es un lenguaje de programación orientado a objetos desarrollado por Microsoft. El uso de Visual Basic agiliza y simplifica la creación de aplicaciones .NET con seguridad de tipos. NET Framework y Visual Studio permiten desarrollar aplicaciones de línea de negocio (LOB) modernas con un conjunto completo de características como controles, enlace de datos, animación, estilos, plantillas, etc., basadas en datos para Windows. Con estas herramientas es posible crear experiencias de usuario visualmente increíbles con WPF o usar los productivos diseñadores WYSIWYG de WinForms para incorporar la interfaz de usuario, elementos multimedia y modelos empresariales complejos.

2 Arreglos, Montaje y Calibración de Sensores.

2.2 Sensor de Temperatura LM35

2.2.1 Arreglo Electrónico.

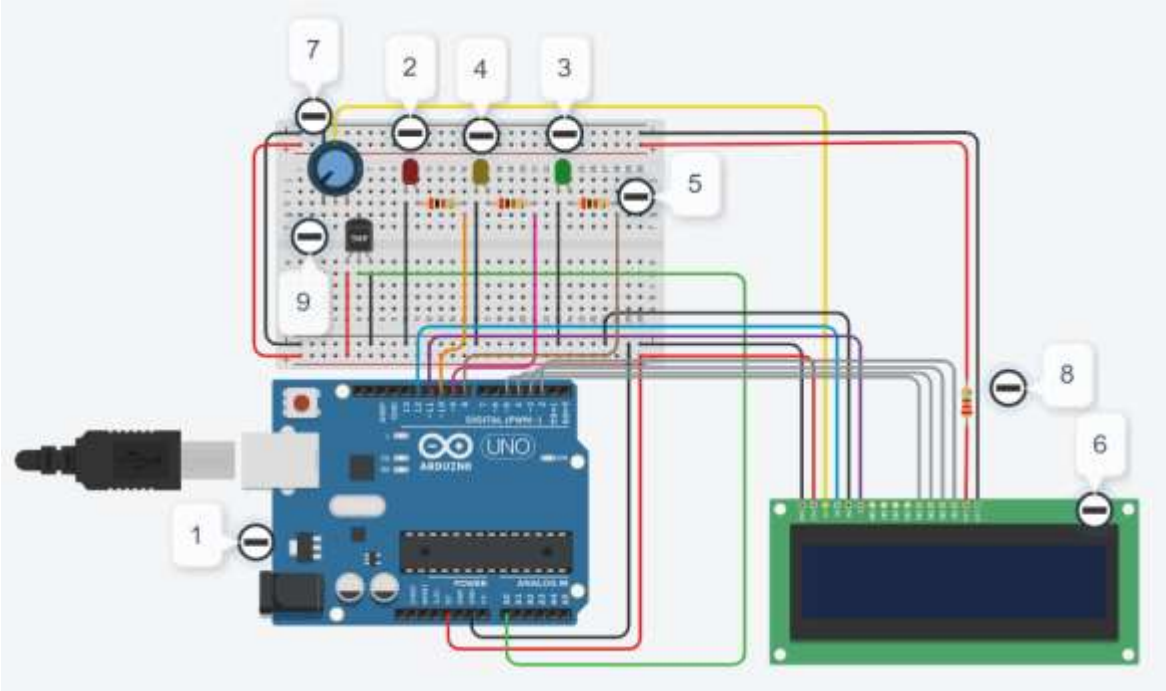


Figura 8. *Arreglo del Sensor de Temperatura LM35 desarrollando en Autodesk*

Tablas 1. Componente Principal del Arreglo

Tabla 1. Componentes Principales del Arreglo.			
No	Nombre	Cantidad	Componente
1	U1	1	Arduino Uno R3
2	D1	1	Rojo LED
3	D3	1	Verde LED
4	D2	1	Amarillo LED
5	R2, R3, R4	3	200 Ω Resistencia
6	U3	1	LCD 16 x 2
7	Rpot1	1	250 kΩ Potenciómetro
8	R5	1	220 Ω Resistencia
9	U2	1	Sensor de temperatura [TMP36]

2.2.2 Sketch (Código Arduino)

Sketch (Codigo) 1. Sensor de Temperatura

```
#include <LiquidCrystal.h>

const int LM35 = A0;
const int ATRASO = 500;
const float BASE_CELSIUS = 0.488758553274682306940371456 5;

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int LED1=8;
int LED2=9;
int LED3=10;

//-------------------------------------
//Funcion principal
//-------------------------------------
void setup() // Se ejecuta cada vez que el Arduino se inicia
{
Serial.begin(9600);
pinMode (LED1, OUTPUT);
pinMode (LED2, OUTPUT);
pinMode (LED3, OUTPUT);
pinMode(12, OUTPUT);
pinMode(11, OUTPUT);
lcd.begin(16, 2);
}

//-------------------------------------
// Función cíclica
//-------------------------------------
void loop()
  {
  lcd.clear();
    lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(tempCelsius());
  lcd.print("C");
    lcd.setCursor(0, 1);
    lcd.print(tempKelvin());
    lcd.print("K/");
    lcd.print(tempFahreinheit());
    lcd.print("F");
  delay(ATRASO);

if (tempCelsius()<=50)
{
digitalWrite(LED1,HIGH);
digitalWrite(LED2,LOW);
digitalWrite(LED3,LOW);
}
```

```
if ((tempCelsius()>50)&&(tempCelsius()<=100))
{
digitalWrite(LED2,HIGH);
digitalWrite(LED1,LOW);
digitalWrite(LED3,LOW);
}

if (tempCelsius()>100)
{
digitalWrite(LED3,HIGH);
digitalWrite(LED2,LOW);
digitalWrite(LED1,LOW);
}
delay(2000);
}


float tempCelsius()
{
return (analogRead(LM35) * BASE_CELSIUS) - 50;
}

float tempKelvin()
{
return tempCelsius() + 273.15;
}

float tempFahreinheit()
{
return (tempCelsius() * (9/5)) + 32;
}
// Fin programa
```

2.2.3 Funcionamiento del Sensor.

Una vez que el sketch de Arduino entra en funcionamiento podemos ver como la señal permite medir el valor de la temperatura en grados Celsius, grados Fahrenheit y en unidades Kelvin (Figura 9). Es posible ajustar el valor de la temperatura a la que se expone el sensor mediante una barra de desplazamiento, como se muestra en la Figura 10. Al desplazar la barra en ambas direcciones podemos accionar los LEDs, los cuáles actúan como indicadores visuales de los niveles de temperatura. Esto permite evaluar el funcionamiento adecuado del sensor y del sketch desarrollado.

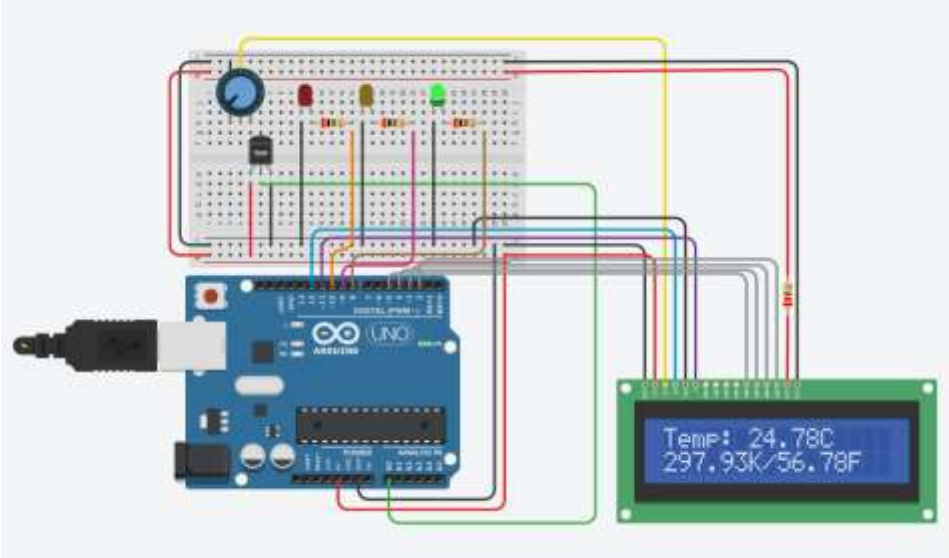


***Figura 9**.Funcionamiento del Sensor de Temperatura LM35 en el arreglo electrónico montado en Autodesk Tinkercad Circuits*

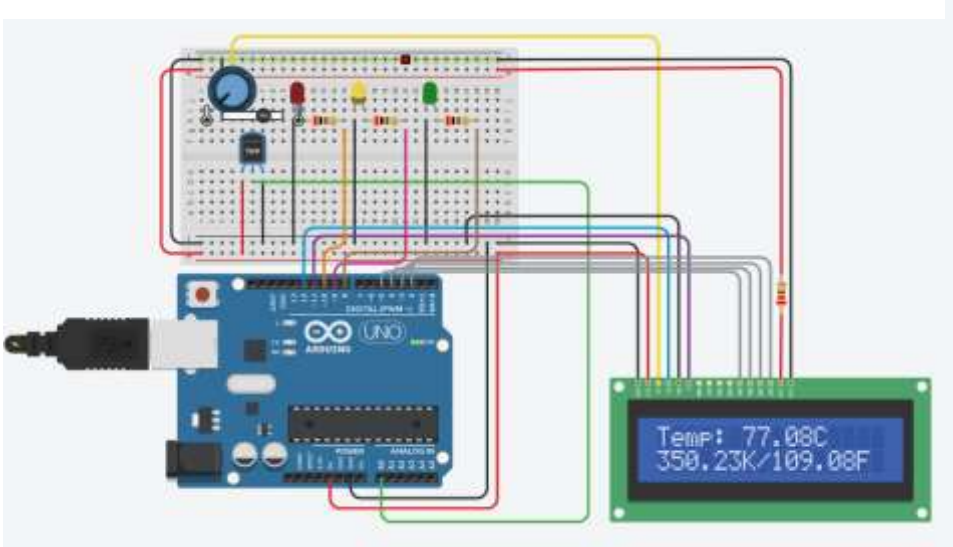


***Figura 10**.Funcionamiento del Sensor de Temperatura LM35 en el arreglo electrónico montado en Autodesk Tinkercad Circuits*

2.3 Sensor Ultrasónico Hc-sr04.

2.3.1 Arreglo Electrónico.

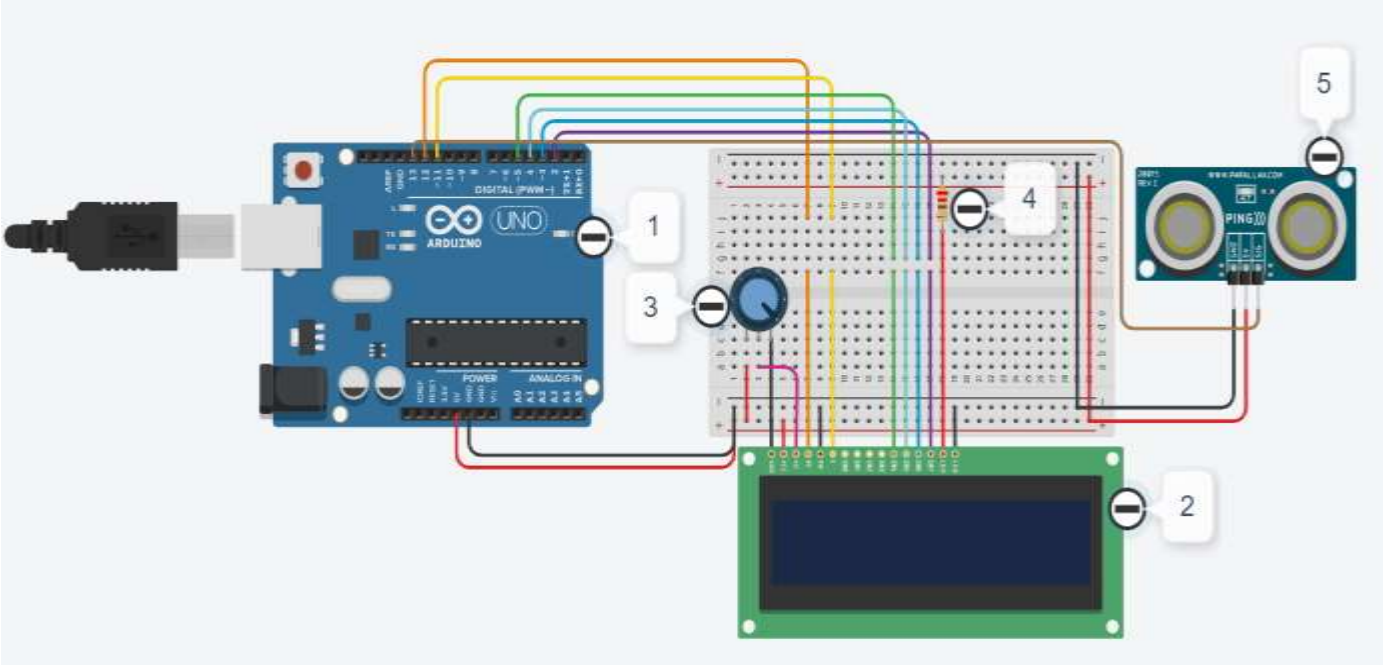


Figura 11. *Arreglo Electrónico del Sensor Ultrasónico Hc-sr04 desarrollado en Autodesk Tinkercad Circuits.*

Tablas 2*.Arreglo Electrónico del Sensor Ultrasónico Hc-sr04 desarrollado en Autodesk Tinkercad Circuits.*

Sensor ultrasónico HC-SR04			
No	Nombre	Cantidad	Componente
1	U1	1	Arduino Uno R3
2	U2	1	LCD 16 x 2
3	Rpot2	1	250kΩ Potenciómetro
4	R2	1	220Ω Resistencia
5	PING1	1	Sensor de distancia ultrasónico

2.3.2 Sketch (Código).

***Sketch (Codigo) 2**. Sensor Ultrasonic Distance*

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int inches = 0;
int cm = 0;

// Aqui creamos la funcion donde apagamos
// encendemos lo volvemos apagar
// y luego tomamos la lectura del pin

long readUltrasonicDistance(int pin)
{
  pinMode(pin, OUTPUT);
  digitalWrite(pin, LOW);
  delayMicroseconds(2);
  digitalWrite(pin, HIGH);
  delayMicroseconds(10);
  digitalWrite(pin, LOW);
  pinMode(pin, INPUT);
  return pulseIn(pin, HIGH);
}

void setup()
{
  pinMode(13, INPUT);
  Serial.begin(9600);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  lcd.begin(16, 2);

}
```

```
void loop()
{
  // Se lee el valor y se convierte en cm
  cm = 0.01723 * readUltrasonicDistance(13);
  // para convertirlos en pulgadas se divide cm entre 2.54
  inches = (cm / 2.54);
  Serial.print("Distancia en Pulgadas - ");
  Serial.println(inches);
  Serial.print("Distancia en Centimetros - ");
  Serial.println(cm);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(0.01723 * readUltrasonicDistance(13));
  lcd.print(" cm");
  lcd.setCursor(0, 1);
  lcd.print((0.01723/2.54) * readUltrasonicDistance(13));
  lcd.print(" inches");
  delay(500);
}
```

2.3.3 Funcionamiento del Sensor.

El programa utiliza la librería LiquidCrystal para hacer uso de la pantalla LCD, y mandar comandos hacia el dispositivo de una manera más sencilla (Figura 12). La función "*ReadUltrasonicDistance*" se utiliza para obtener los datos del sensor necesarios para calcular la distancia y la función setup, para declarar puertos como digitales o analógicos, tamaño del LCD, y saber la velocidad de la comunicación de nuestra computadora con el Arduino. Por último, en el main se realizan cálculos para saber la distancia de nuestro sensor hacia el obstáculo detectado, para calcular conversiones y para imprimir datos en la pantalla LCD (Figura 13).

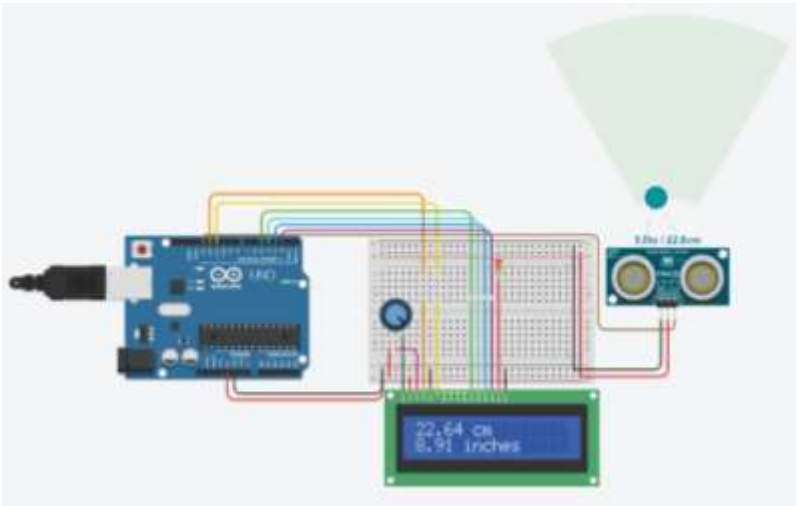


Figura 12.*Valores medidos por el Sensor Ultrasónico Hc-sr04 mostrados en la pantalla LCD.*

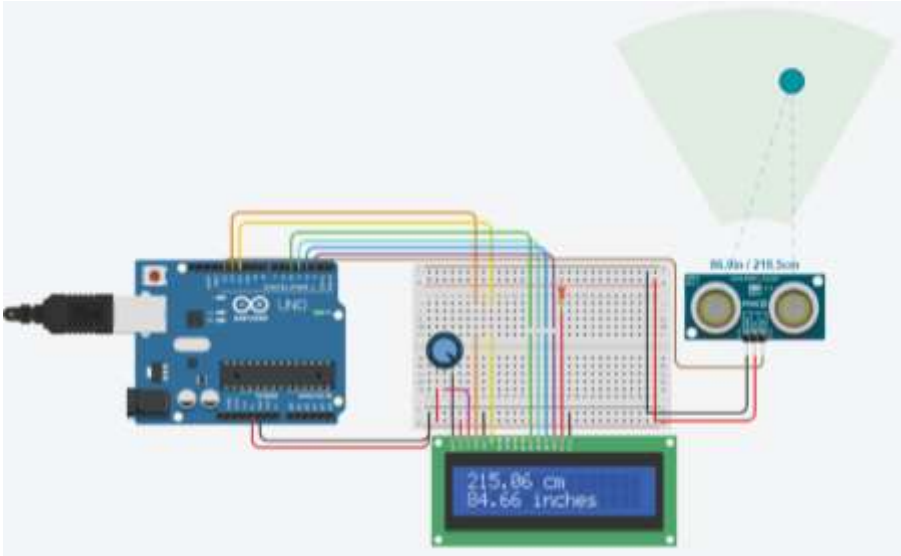


***Figura 13**.Medición de los valores de distancia obtenidos por arreglo Electrónico del Sensor Ultrasónico Hc-sr04 desarrollado en Autodesk Tinkercad Circuits.*

2.4 Sensor de Fuerza Resistivo Fsr402.

2.4.1 Arreglo Electrónico.

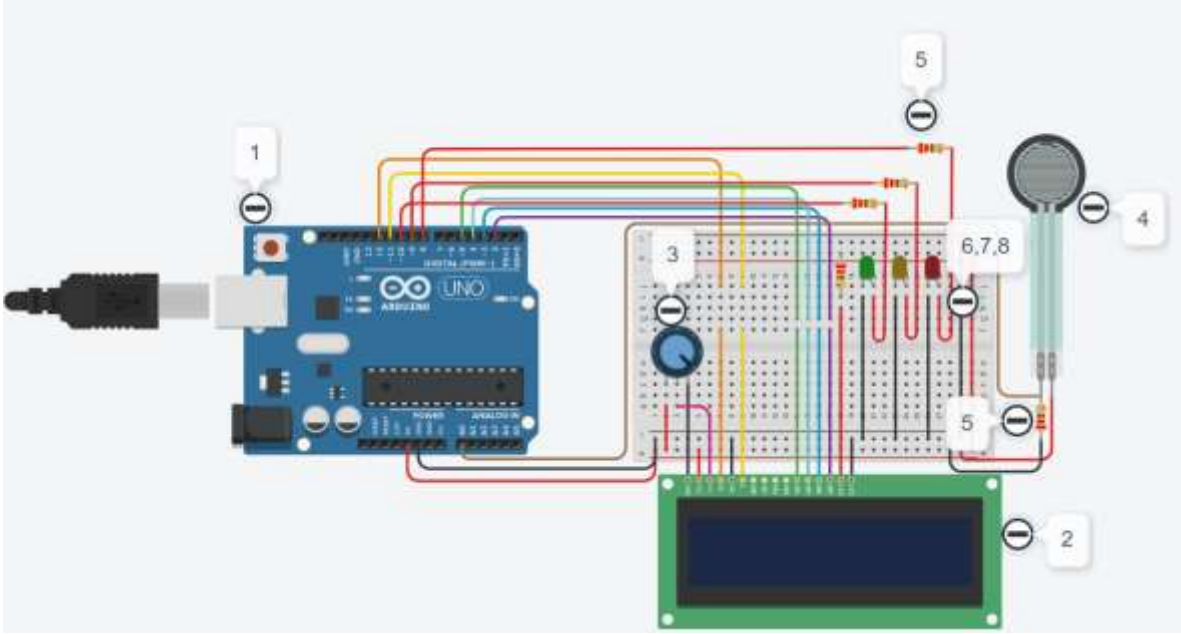


***Figura 14**.Arreglo Electrónico del Sensor de Fuerza Resistivo Fsr402 desarrollado en Autodesk Tinkercad Circuits.*

Tablas 3. *Componentes Principales del Arreglo.*

Componentes Principales del Arreglo.			
No	Nombre	Cantidad	Componente
1	**U1**	**1**	**Arduino Uno R3**
2	**U2**	**1**	**LCD 16 x 2**
3	**Rpot1**	**1**	**250 kΩ Potenciómetro**
4	**R2**	**1**	**Force Sensor**
5	**R1, R3, R5, R6, R7**	**4**	**220 Ω Resistencia**
6	**D1**	**1**	**Verde LED**
7	**D2**	**1**	**Amarillo LED**
8	**D3**	**1**	**Rojo LED**

2.4.2 Sketch (Código).

Sketch (Codigo) 3. *Sensor de Fuerza Resistivo Fsr402.*

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int led1=10;
int led2=9;
int led3=8;
int control;

void setup()
{
   Serial.begin(9600);
   pinMode(12, OUTPUT);
   pinMode(11, OUTPUT);
   lcd.begin(16, 2);
}
```

```
void loop()
{
  control=analogRead(A0)/10;
  analogWrite(led1,control);
  analogWrite(led2,control);
  analogWrite(led3,control);

  float fuerza= (control*500.0/1000.0);

  if(fuerza >= 0.0 && fuerza <= 2.5)
  {
    digitalWrite(led1,HIGH);
    digitalWrite(led2,LOW);
    digitalWrite(led3,LOW);
    Serial.print("Presion baja: ");
    Serial.println(fuerza);

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Presion Baja: ");
    lcd.setCursor(0, 1);
    lcd.print(fuerza);
    lcd.print(" N");
  }


  if(fuerza > 2.5 && fuerza <= 5.0)
  {
    digitalWrite(led1,LOW);
    digitalWrite(led2,HIGH);
    digitalWrite(led3,LOW);
    Serial.print("Presion media: ");
    Serial.println(fuerza);

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Presion Media: ");
    lcd.setCursor(0, 1);
    lcd.print(fuerza);
    lcd.print(" N");
  }
```

```
  if(fuerza > 5.0 && fuerza <= 7.6)
  {
    digitalWrite(led1,LOW);
    digitalWrite(led2,LOW);
    digitalWrite(led3,HIGH);
    Serial.print("Presion alta: ");
    Serial.println(fuerza);

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Presion Alta: ");
    lcd.setCursor(0, 1);
    lcd.print(fuerza);
    lcd.print(" N");
  }
  delay(1000);
}
```

2.4.3 Funcionamiento del Sensor.

El programa utiliza la librería LiquidCrystal para mostrar los valores medidos por el sensor en la pantalla LCD (Figura 15). Durante el desarrollo del código debemos establecer los pines digitales que le corresponderán a los LEDs, los pines que actuarán como analógicos, y aquellos en los que se conectará la pantalla LCD. Además, es necesario configurar la velocidad de la comunicación serial que tendrá lugar entre nuestra computadora y el microcontrolador Arduino. Una vez que la simulación sea iniciada podremos ver como en la pantalla LCD se verán reflejados los valores de fuerza medidos por el sensor y a medida que rebasemos los umbrales definidos irán encendiendo los LEDs (indicadores de nivel) y se mostrarán en la pantalla los niveles bajo, moderado y alto de presión ejercida (Figura 16).

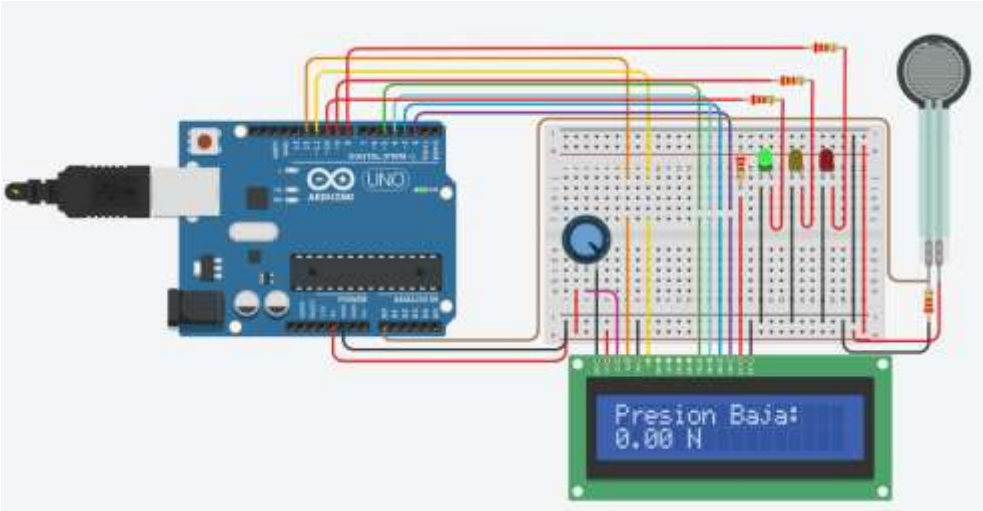


Figura 15. *Valores medidos por el sensor de fuerza Resistivo Fsr402 mostrados en la pantalla LCD.*

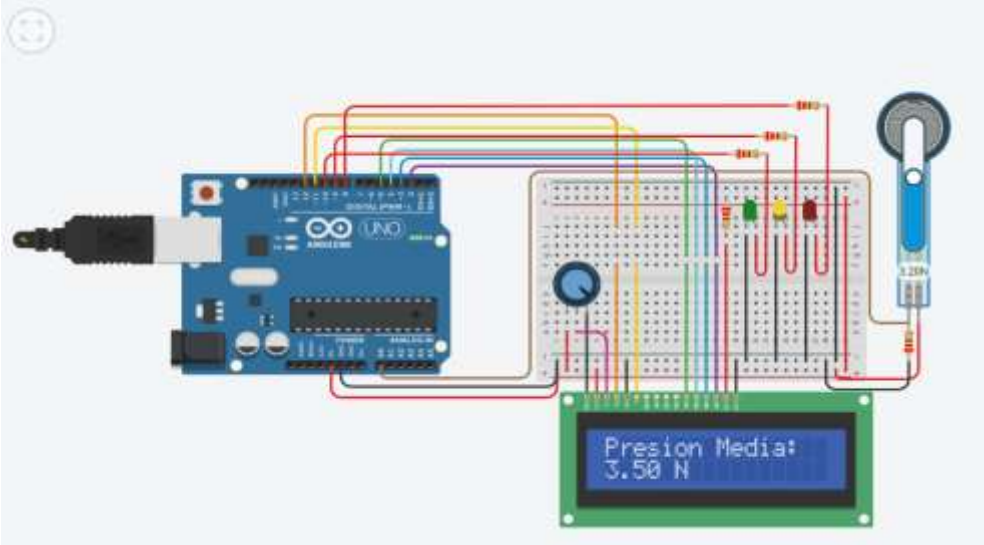


Figura 16*.Indicadores de presión visual (LED Amarillo) y digital (Pantalla LCD- Presión Moderada).*

2.5 Fotorresistencia Ldr.

2.5.1 Arreglo Electrónico.

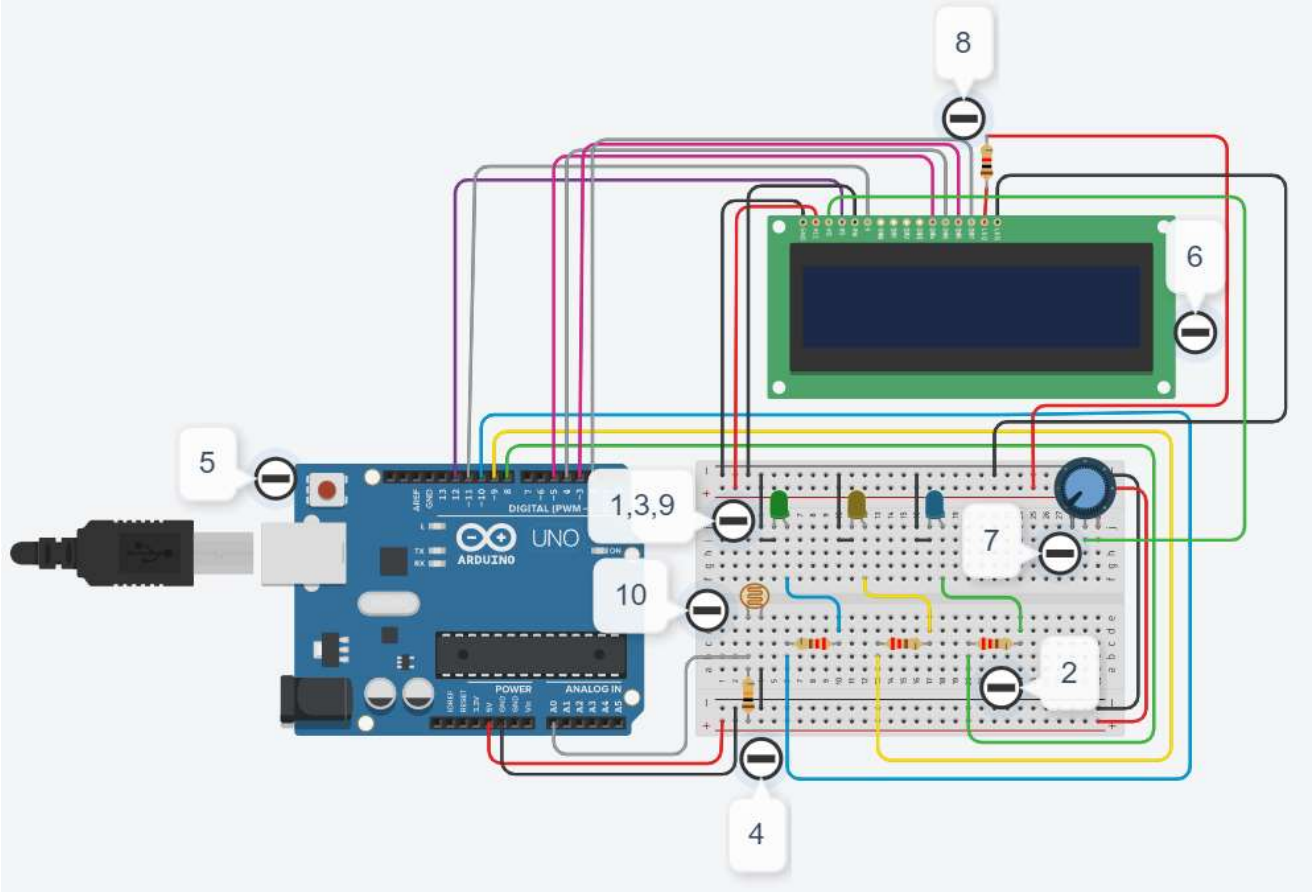


Figura 17*.Arreglo Electrónico del Sensor LDR desarrollado en Autodesk Tinkercad Circuits.*

Tablas 4*. Sensor LDR*

Componentes Principales del Arreglo.			
No	**Nombre**	**Cantidad**	**Componente**
1	D2	1	Verde LED
2	R1, R2, R3	3	220 Ω Resistencia
3	D3	1	Amarillo LED
4	R4	1	10 kΩ Resistencia
5	U1	1	Arduino Uno R3
6	U2	1	LCD 16 x 2
7	Rpot2	1	250 kΩ Potenciómetro
8	R5	1	1 kΩ Resistencia
9	D4	1	Azul LED
10	R6	1	Fotorresistencia

2.5.2 Sketch (Código).

Sketch (Codigo) 4*.Sensor LDR*

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int Brillo=0;

void setup()
{
  pinMode(A0, INPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  lcd.begin(16, 2);
  Serial.begin(9600);
}

void loop()
{
  Brillo = analogRead(A0);
  if (Brillo >= 0 && Brillo < 60)
  {
    digitalWrite(8, HIGH);
    digitalWrite(9, LOW);
    digitalWrite(10, LOW);

  lcd.clear();
  lcd.setCursor(0, 0);
lcd.print("LUZ: ");
lcd.print("100%");
  lcd.setCursor(0, 1);
  lcd.print("Alta");

} else
{
  if (Brillo >= 60 && Brillo < 200) {
    digitalWrite(10, LOW);
    digitalWrite(9, HIGH);
    digitalWrite(8, LOW);

    lcd.clear();
    lcd.setCursor(0, 0);
  lcd.print("LUZ: ");
  lcd.print("50%");
    lcd.setCursor(0, 1);
    lcd.print("Moderada");
```

```
    } else {
      digitalWrite(8, LOW);
      digitalWrite(9, LOW);
      digitalWrite(10,HIGH);

      lcd.clear();
      lcd.setCursor(0, 0);
    lcd.print("LUZ: ");
    lcd.print("0%");
      lcd.setCursor(0, 1);
      lcd.print("Nula");

    }
  }
  delay(10);

}
```

2.5.3 Funcionamiento del Sensor.

Una vez que la simulación se pone en marcha podemos ver como el sensor de luz puede medir la intensidad de la radiación luminosa presente en el medio, empleando una escala de tres niveles (nula, moderada y alta, Figura 18). A medida que desplazamos la banda de intensidad luminosa de izquierda a derecha podemos ver como los niveles de intensidad se van activando gradualmente, encendiendo los LEDs indicadores de nivel y mostrando el nivel de intensidad en la pantalla LCD (Figura 19 y 20).

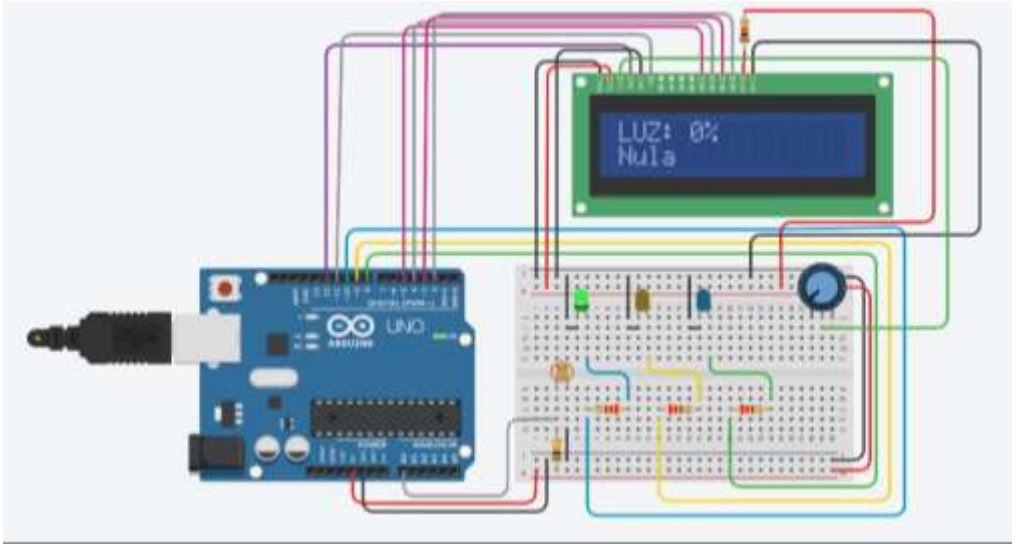


Figura 18. *Valores medidos por el sensor LDR mostrados en la pantalla LCD.*

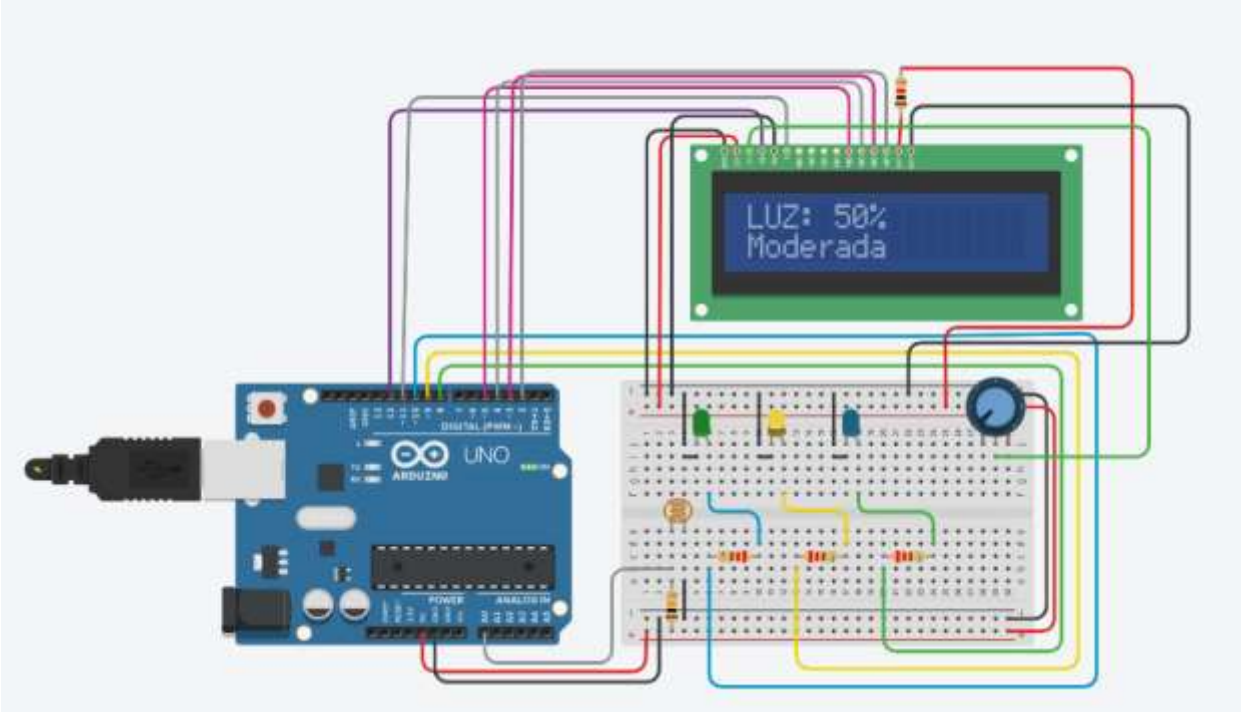


***Figura 19**.Intensidad de luz moderada medida por el sensor LDR mostrada en la pantalla LCD.*

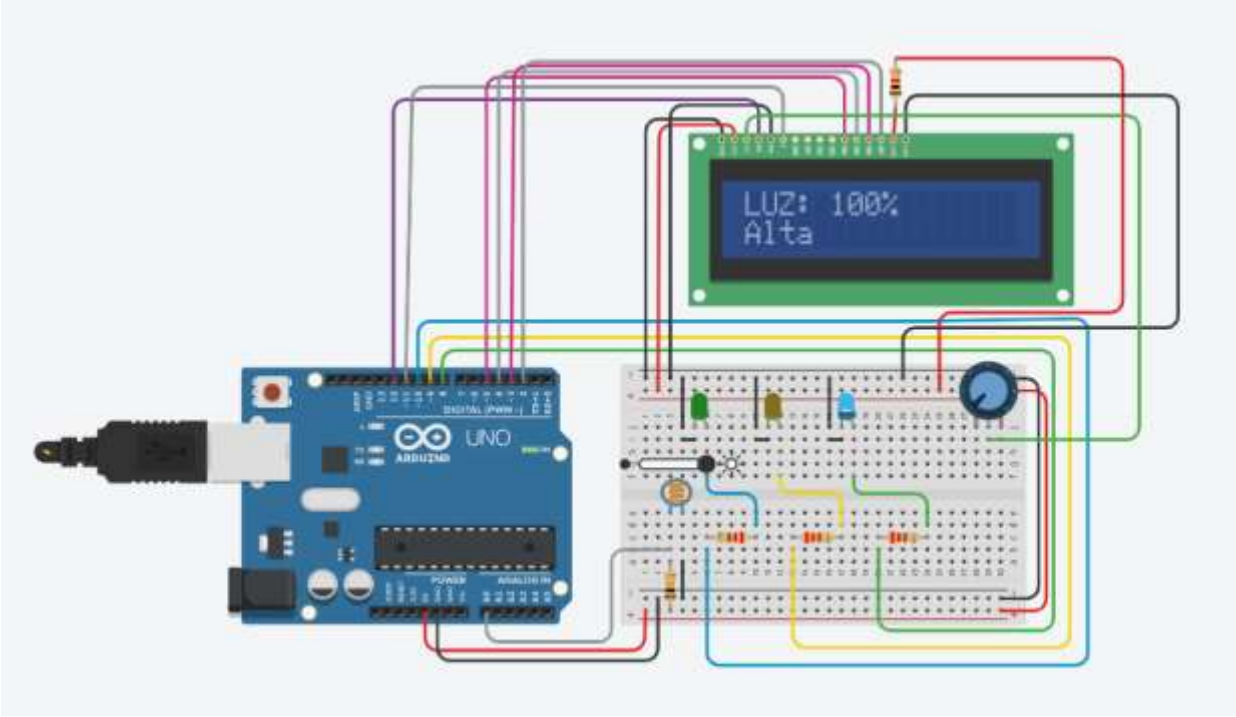


***Figura 20**.Intensidad de luz alta medida por el sensor LDR mostrada en la pantalla LCD.*

2.6 Sensor de inclinación Sw-520d.

2.6.1 Arreglo Electrónico.

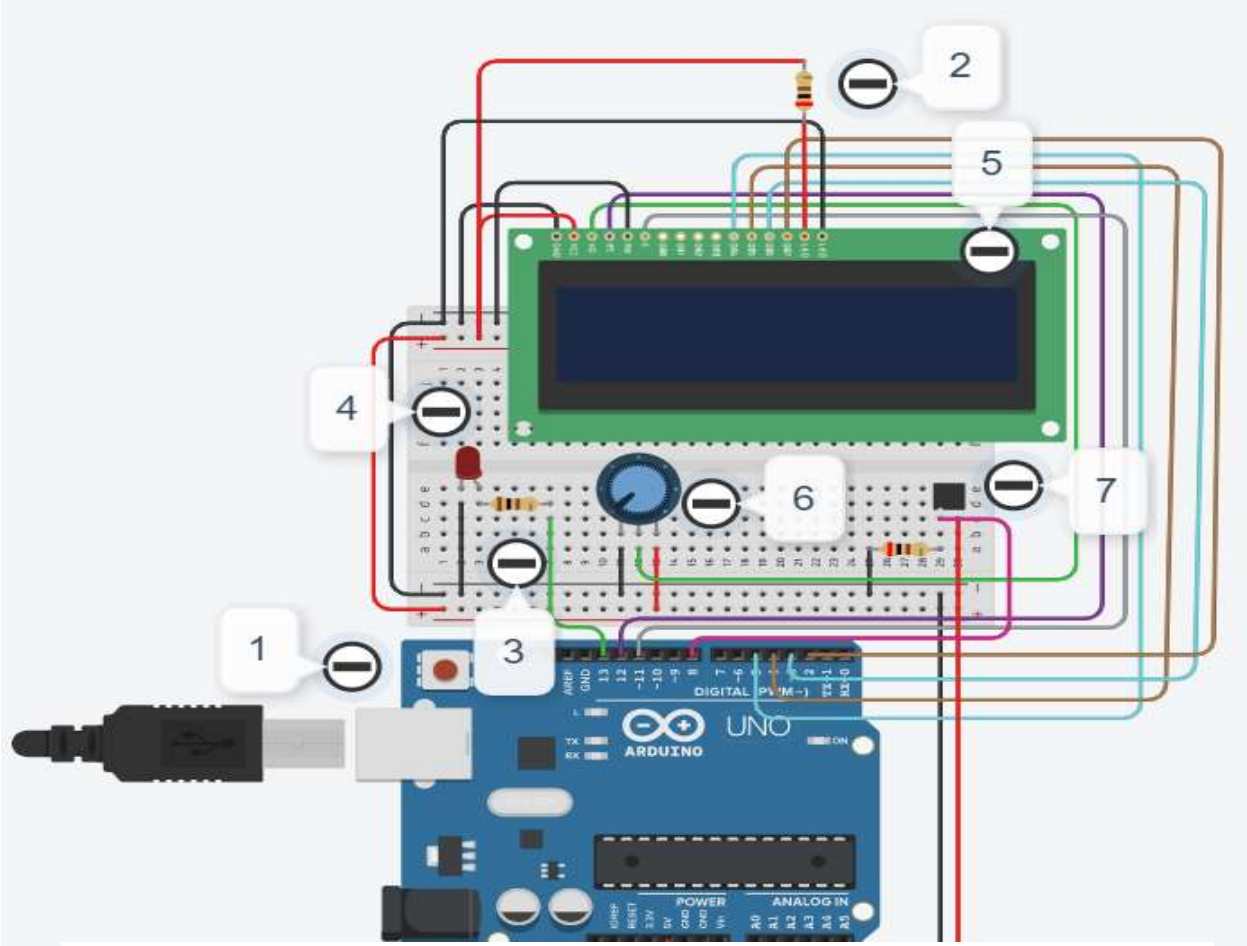


Figura 21*.Arreglo electrónico del sensor de inclinación Sw-520d desarrollado en Autodesk Tinkercad Circuits.*

Tablas 5. *Sensor de inclinación Sw-520d*

No	Nombre	Cantidad	Componente
1	U1	1	Arduino Uno R3
2	R1, R2	2	200 Ω Resistencia
3	R4	1	100 Ω Resistencia
4	D2	1	Rojo LED
5	U3	1	LCD 16 x 2
6	Rpot3	1	250 kΩ Potenciómetro
7	U2	1	Sensor de inclinación de 4 pines

2.6.2 Sketch (Código).

Sketch (Codigo) *5.Sensor de inclinación Sw-520d*

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup()
{
  Serial.begin(9600);
  pinMode(8, INPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  lcd.begin(16, 2);
}
void loop()
{
   if (digitalRead(8) == 1)
   {
    digitalWrite(13, LOW);
    lcd.clear();
    lcd.setCursor(0,0);
  lcd.print("En posicion");
}
  else
  {
    digitalWrite(13, HIGH);
    lcd.clear();
    lcd.setCursor(0,0);
  lcd.print("Inclinado");
  }
  delay(1000);
  }
```

2.6.3 Funcionamiento del Sensor.

El programa utiliza la librería LiquidCrystal para hacer uso de la pantalla LCD, y mostrar los valores medidos por el sensor de una manera más sencilla. Como se ha comentado anteriormente, los sensores de inclinación Tilt disponen de un cilindro en cuyo interior al dos esferas y cuya pared constituye un contacto eléctrico, mientras que el otro contacto está localizado en el centro de la base. Al inclinar lo suficiente el dispositivo ambas esferas constituyen un puente entre ambos contactos, cerrando el circuito. Debido a su principio de funcionamiento, estos sensores resultan sensibles a movimientos bruscos y vibraciones. Cuando nosotros ajustamos la posición del sensor mediante la barra de desplazamiento podemos pasar de un estado “En posición” a un estado “Inclinado” al abrir o cerrar el circuito (valores 1 y 0).

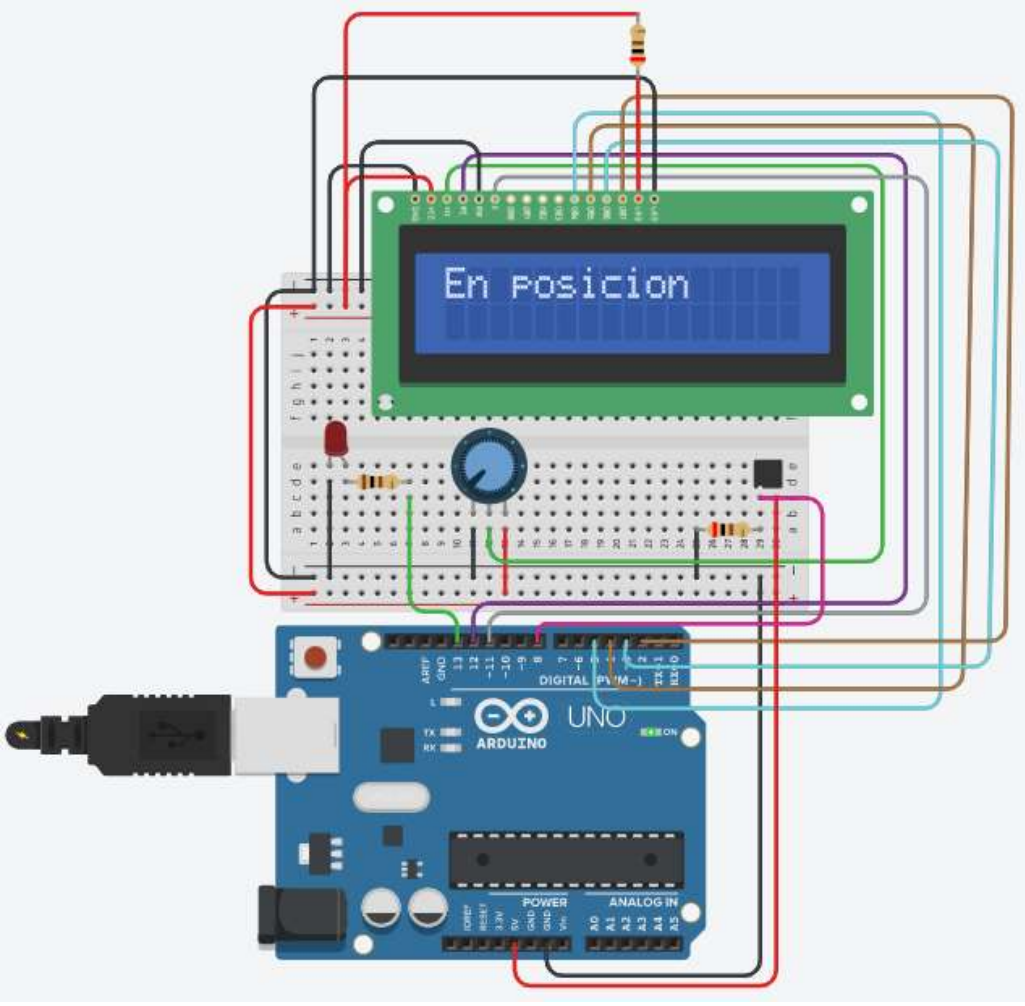


Figura 22*.Sensor de inclinación Sw-520d en posición desarrollado en Autodesk Tinkercad Circuits.*

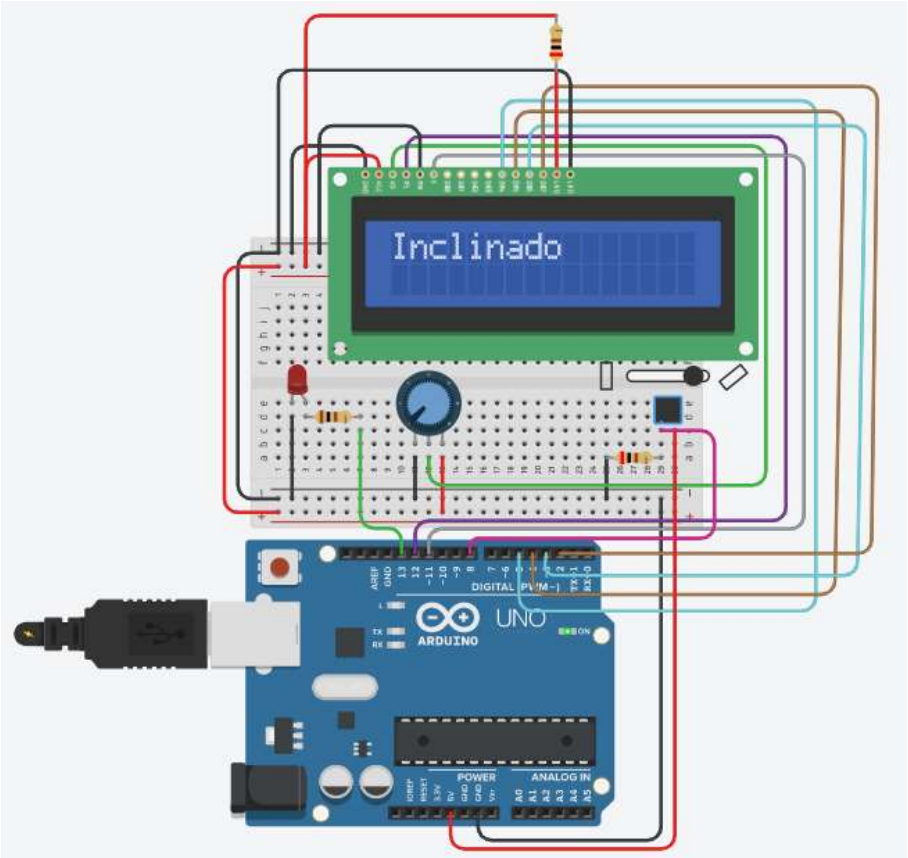


***Figura 23**. Cambio de posición del Sensor de inclinación Sw-520d a un esquema inclinado desarrollado en Autodesk Tinkercad Circuits*

3 Montaje de la Estación de Monitoreo.

3.2 Prueba de Comunicación de Visual Basic.NET con el microcontrolador Arduino.

El primer paso consiste en desarrollar el código (Sketch) en la placa Arduino para recibir el dato desde Visual Basic.NET:

```
int ledpin=13;
int dato;

void setup() {
  pinMode (ledpin, OUTPUT);
  Serial.begin (9600);
}

void loop() {
 if(Serial.available()) {
  dato=Serial.read();
```

***Sketch (Codigo) 6.** Prueba de Comunicación de Visual Basic.NET*

```
    if(dato=='1') {
      digitalWrite(ledpin,HIGH);
    }
    if(dato=='0'){
      digitalWrite(ledpin,LOW);
    }
  }
}
```

Una vez desarrollado el código descargamos el sketch a la tarjeta Arduino verificando el puerto COM al que se ha conectado la tarjeta. Para realizar una prueba de comunicación y verificar que nuestro programa cargado sea el correcto, se utiliza la consola de comunicación serial del IDE de Arduino, Figura 24 y 25. Al escribir 1 o 0 en la consola podemos encender o apagar el LED conectado a la tarjeta.

Figura 24*.Consola de Comunicación del IDE de Arduino*

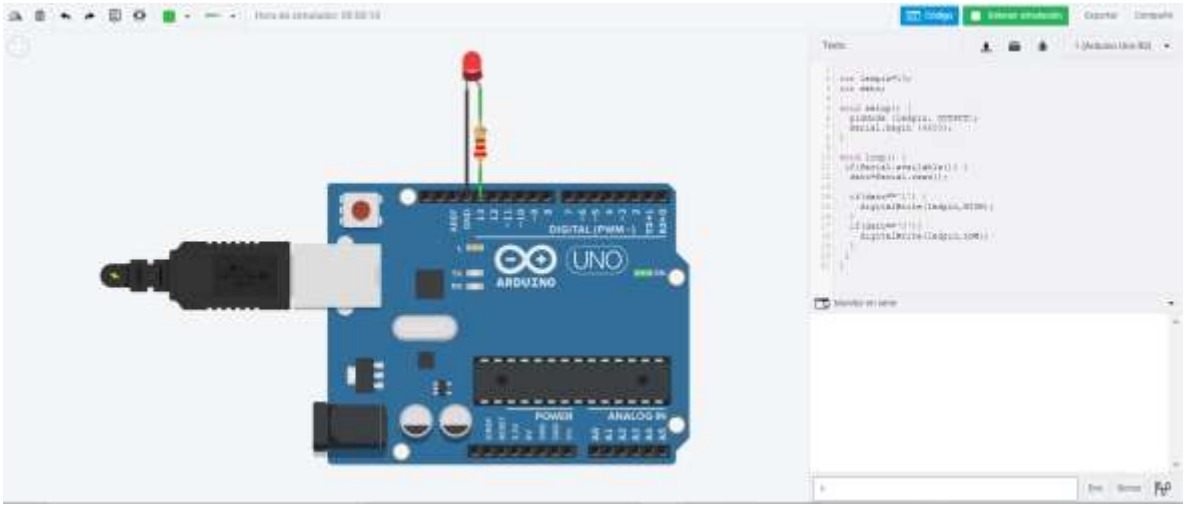


Figura 25*.Consola de Comunicación serial simulada en Tinkercad.*

Una vez que se ha desarrollado el Sketch de Arduino para controlar el LED, el siguiente paso radica en activar un LED conectado a la tarjeta Arduino haciendo un clic en un botón de una interfaz desarrollada en Visual Basic.NET. Primeramente, se debe arrastrar el control SerialPort al formulario. Una vez que el control SerialPort se encuentre en el formulario debemos configurar sus propiedades con los siguientes datos: BaudRate de 9600, Databits de 8, Parity=None y PortName=COM1, tal y como se muestra en la Figura 26. El siguiente paso es insertar dos controles Button a la pantalla, tal como se muestra en la Figura 27. Posteriormente, hay que cambiar la propiedad a cada control Button a texto. Cada control debe nombrase “1” y “0”, respectivamente. Posteriormente, hay que establecer comunicación entre los botones y la tarjeta Arduino con el control SerialPort, Figura 28. Hecho esto, hay que regresar a la forma de diseño para continuar programando los botones, Figura 29. Una vez configurados los botones el paso final radica en encender y apagar la tarjeta haciendo uso de Visual Basic.NET, Figura 30.

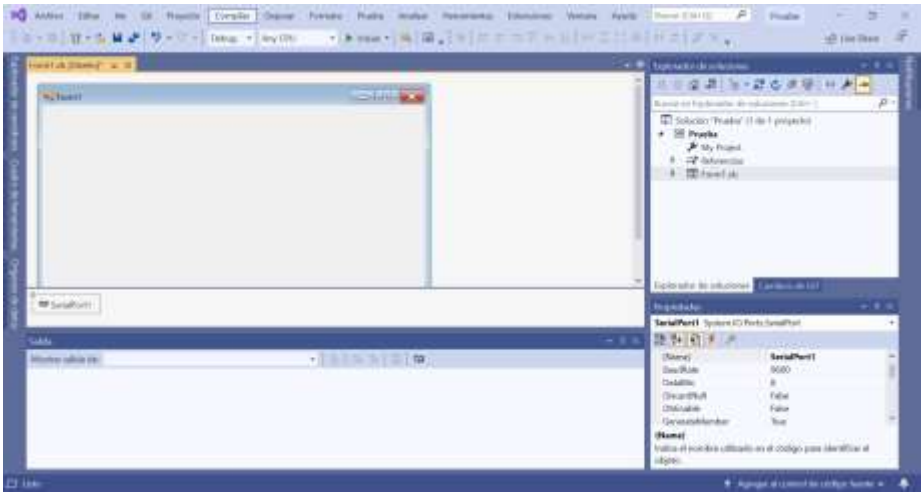


Figura 26. *Incorporación del control SerialPort al Formulario.*

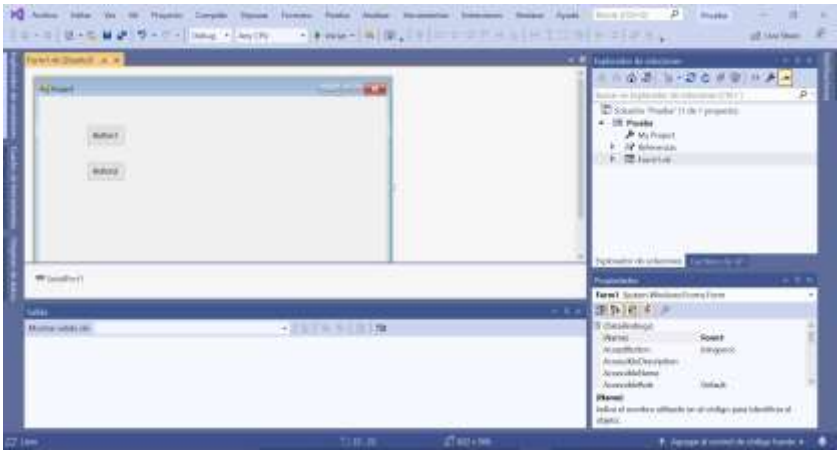


Figura 27*. Incorporación de botones en el formulario.*

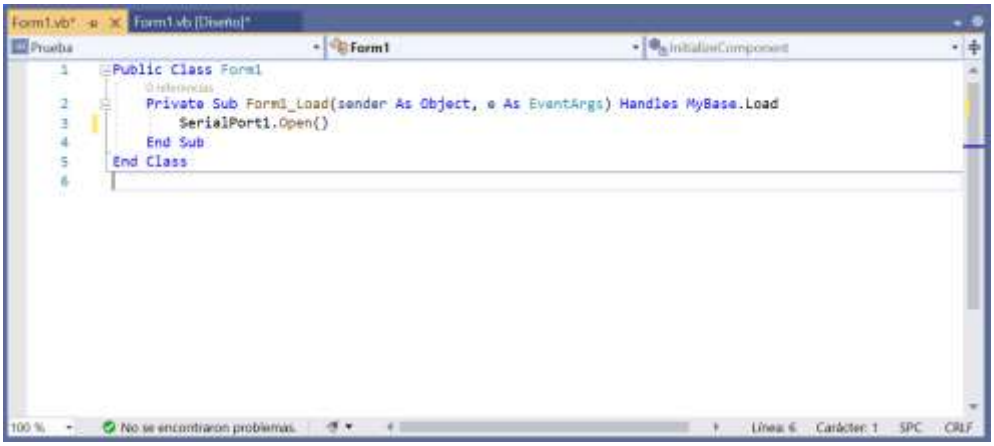


Figura 28. *Abrir la comunicación desde la aplicación por medio del comando SerialPort.Open.*

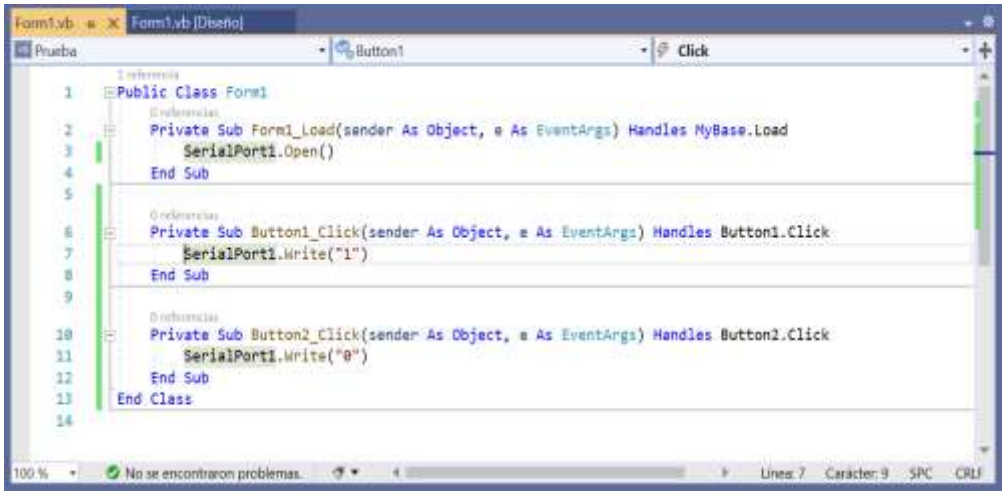


Figura 29.Programación de los botones 1 y 2.

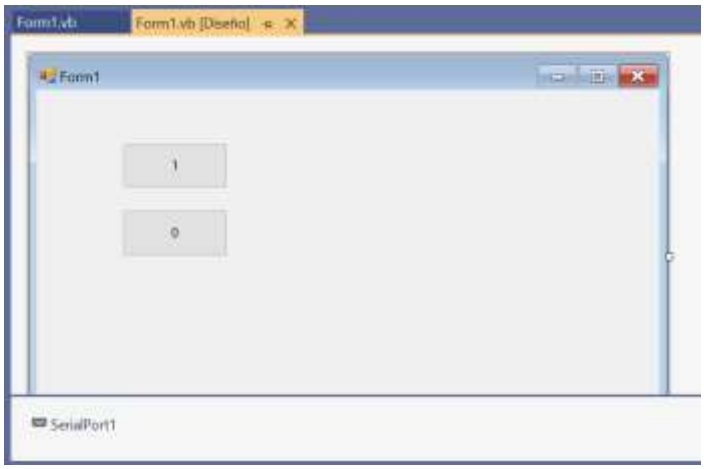


Figura 30. Encendiendo y Apagando el LED de la Tarjeta Arduino con Visual Basic.NET.

3.3 Desarrollo de la Estación Meteorológica de Monitoreo con Arduino y Visual Basic.NET.

Para la implementación de la Estación de Monitoreo se requiere armar el arreglo electrónico del sensor de Medición de Humedad y Temperatura (DHT11) y de la fotorresistencia incluida en la estación, Figura 31-32.

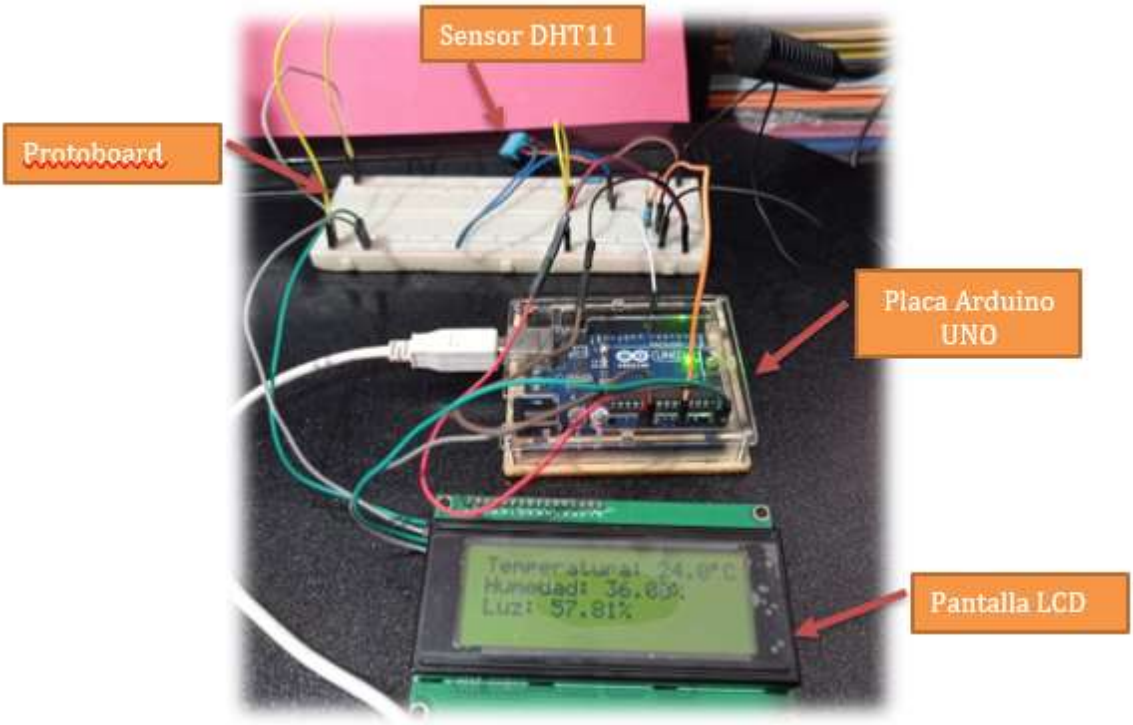


Figura 31. *Arreglo Electrónico de la Estación de Monitoreo.*

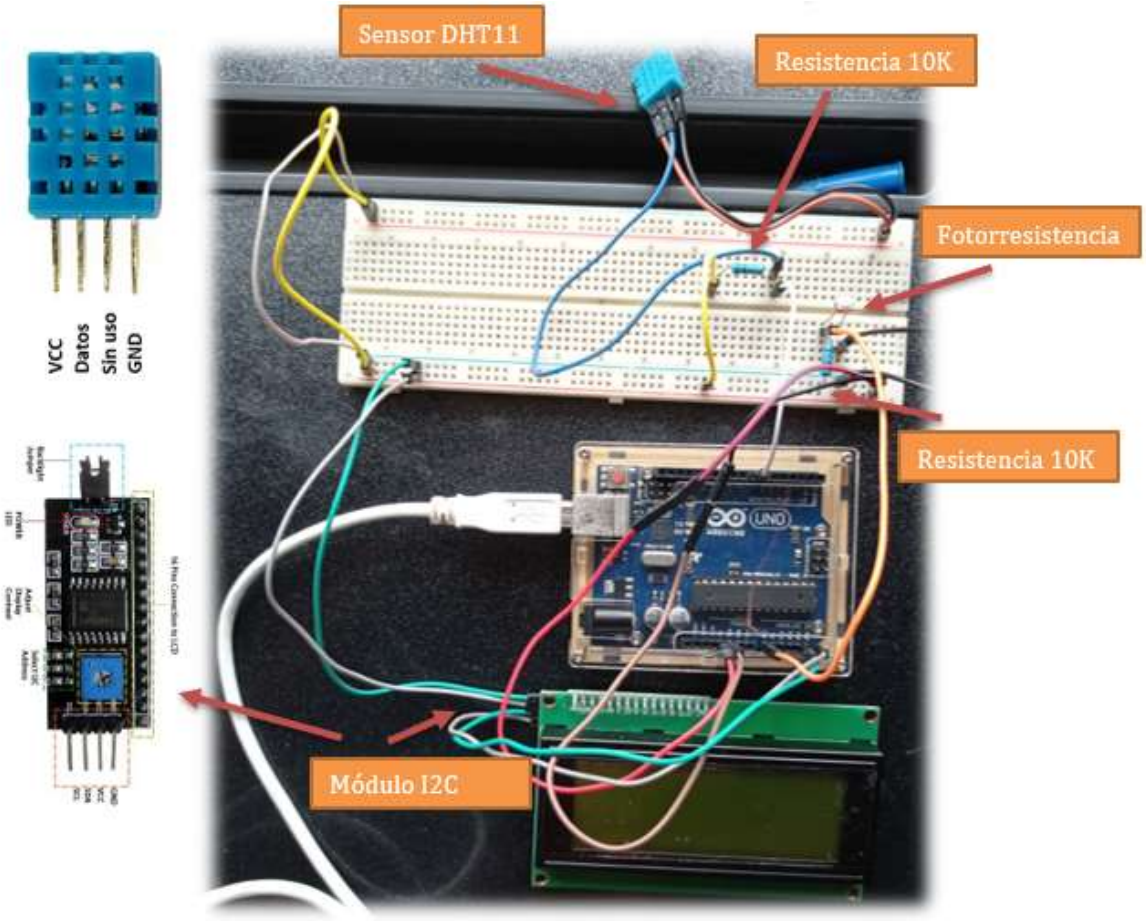


***Figura 32**. Conexión de Sensores en la Estación de Monitoreo.*

Una vez que el hardware del proyecto está totalmente montado, hay que probar los diferentes sensores mediante la construcción del sketch que permita realizar mediciones y que muestre los valores de las variables definidas en la pantalla LCD 20x4. Para ello, se escribe un sketch en el IDE de Arduino que incluya las siguientes librerías: **Wire.h, DHT.h** y **LiquidCrystal_I2C.h**. Para instalar las librerías es necesario descargar las carpetas de algún sitio web y copiar éstas en la carpeta Arduino/libraries de nuestro ordenador. El código desarrollado para el arreglo correspondiente a la Estación de Monitoreo se detalla en la Figura 33. El código debe compilarse y descargarse a la placa de Arduino a fin de ponerlo a prueba. También debe abrirse el comunicador serial para visualizar los datos enviados. Es importante verificar que la velocidad de transmisión del puerto serial esté a 9600. En las Figuras 34 y 25 se muestra el funcionamiento del código.

***Sketch (Codigo) 7**. Sketch de Arduino desarrollado para la Estación de Monitoreo.*

```
#include <DHT.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

//Sensor DHT11
#define DHTPIN 7
#define DHTTYPE DHT11

LiquidCrystal_I2C lcd(0x27,20,4);

// Instancia del Sensor
DHT dht(DHTPIN,DHTTYPE);

void setup() {
  lcd.begin();
  lcd.backlight();
  lcd.setCursor(1,0);
  lcd.print("Hola!!!");
  lcd.setCursor(1,1);
  lcd.print("Inicializando...");
  Serial.begin(9600);
  dht.begin();
  delay(2000);
  lcd.clear();
}

void loop() {
  float tem =dht.readTemperature();
  float hum =dht.readHumidity();
  float sensor =analogRead(0);
  float luz =sensor/1024*100;

  Serial.print("Temperature: ");
  Serial.print (tem);
  Serial.println(" C");

  Serial.print("Humedad: ");
  Serial.print(hum);
  Serial.println("%");
```

```
    Serial.print("Luz: ");
    Serial.println(luz);
    Serial.println("%");
    Serial.println("");

    lcd.setCursor(1,0);
    lcd.print("Temperatura: ");
    lcd.print(tem,1);
    lcd.print((char)223);
    lcd.print("C");

    lcd.setCursor(1,1);
    lcd.print("Humedad: ");
    lcd.print(hum);
    lcd.print("%");

    lcd.setCursor(1,2);
    lcd.print("Luz: ");
    lcd.print(luz);
    lcd.print("%");
    delay(700);
}
```

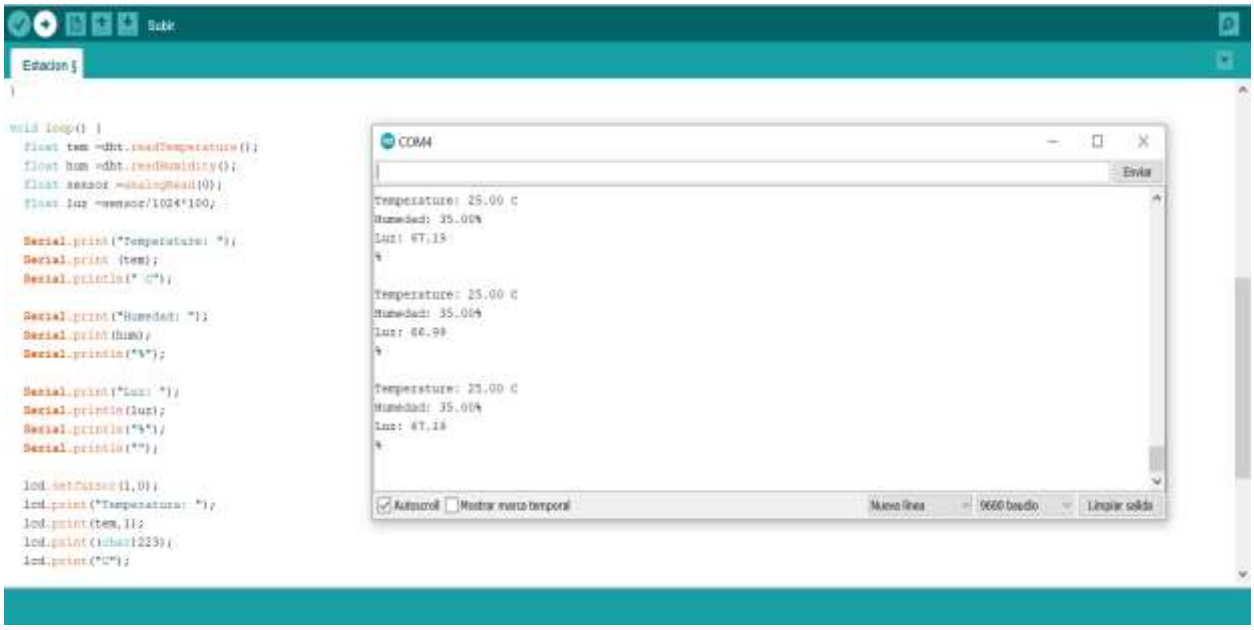


Figura 33. Sketch *de Arduino de la estación de monitoreo en funcionamiento (Valores reportados en el monitor serial).*

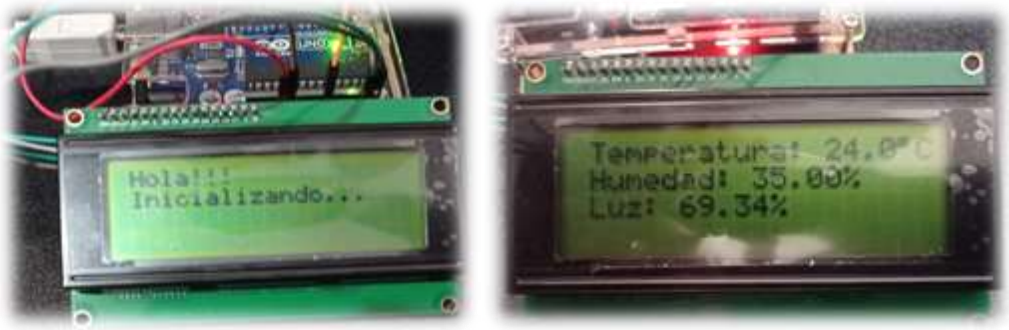


***Figura 34**. Valores reportados en la pantalla LCD.*

Antes de proceder a crear la interfaz, es necesario adecuar el código del arreglo electrónico a fin de que los datos que se envíen al monitor serial se muestren en una sola línea. La interfaz que se desarrollará en Visual Basic.NET requiere que los datos que arroje el controlador se presenten en este formato. Para que los datos se muestren en una sola línea es necesario es necesario modificar unas líneas del código presentado en la Figura 33. A continuación se detallan las líneas que deben cambiarse en el sketch, Figura 36. Los valores reportados por los sensores en el formato de una sola línea se muestran en la Figura 37.

```
Serial.print("Temperature: ");
Serial.print (tem);
Serial.println(" C");

Serial.print("Humedad: ");
Serial.print(hum);
Serial.println("%");

Serial.print("Luz: ");
Serial.println(luz);
Serial.println("%");
Serial.println("");
```

ANTES

```
Serial.print (tem);
Serial.print(",");
Serial.print(hum);
Serial.print(",");
Serial.println(luz);
```

DESPUÉS

***Figura 35.** Cambios realizados en el Sketch para mostrar los valores obtenidos por los sensores en una sola línea.*

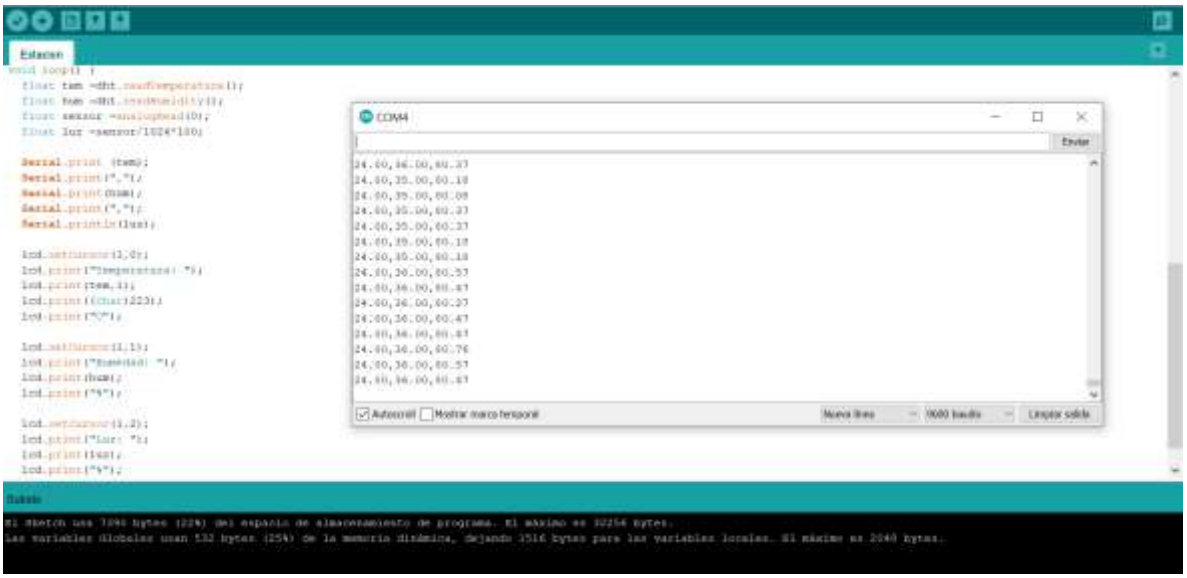


***Figura 36**. Valores mostrados en el monitor serial en formato de una sola línea.*

3.3.1 Desarrollo de la Interfaz de Monitoreo con Visual Basic.NET.

El primer paso para crear la interfase de monitoreo es crear una aplicación en Visual Basic.NET 2019. Para ello, hay que abrir el programa de Visual Basic.NET. Una vez abierto, debemos hacer clic en crear un proyecto, elegir Aplicación de Windows Forms (.NET Framework) y darle Nombre. Es importante elegir un Framework del 4 en adelante. Una vez creado el proyecto procedemos a incluir el control SerialPort del entorno de Visual Basic.Net el cual permite establecer una comunicación serial con cualquier dispositivo, Figura 36. Una vez que el control SerialPort está en el formulario, e posible ver las propiedades del control. Se requiere que dichas propiedades queden configuradas de la siguiente manera: **BaudRate= 9600, Databits= 8, Parity= None, PortName= COM4** (debe coincidir con el puerto al que está conectado el Arduino).

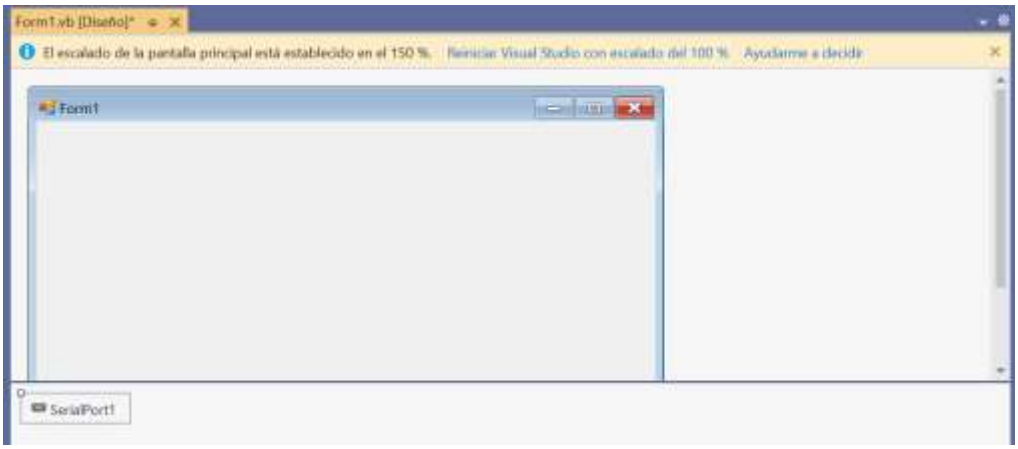


Figura 37. *Control SerialPort.*

Una vez configurado el SerialPort procedemos a incluir el control Timer, el cual, permitirá establecer un tiempo de lectura para que el búfer no se sature y, en consecuencia, el envío y recepción de los datos estén sincronizados. En las propiedades del objeto Timer se deben realizar los siguientes ajustes: Propiedad Interval (especificar el valor de 2000 o, mínimo, 1900 milisegundos). No se recomienda que sea menor, ya que los cambios en los valores serán muy rápidos y no se podrán visualizar correctamente. En la Figura 37 se señalan los controles adiciones que se agregan al formulario, los cuales son esencialmente: Labels, Textbox, SerialPort y Timer, con los cuales se desarrollará la aplicación, Figura 38.

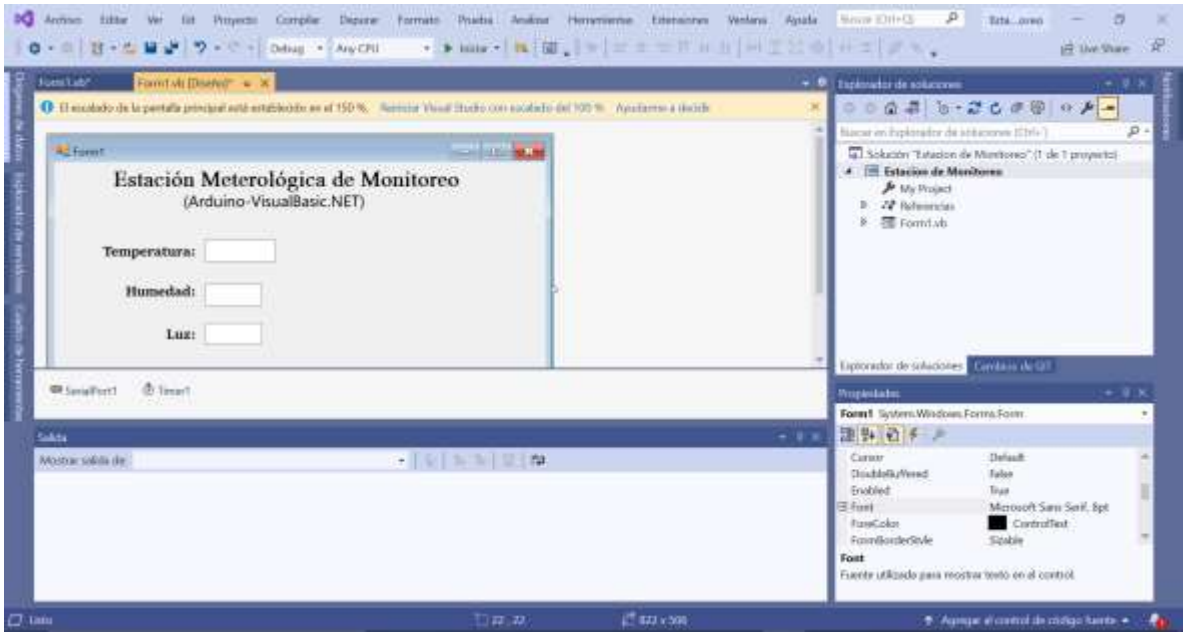


Figura 38. *Controles incluidos en la aplicación de la Estación de Monitoreo.*

Para leer los datos enviados desde la tarjeta Arduino en el tiempo establecido, hay que abrir el puerto y habilitar el control Timer incluyendo las siguientes líneas de código, Figura 39.

```
14
15          cadena1 = Mid(cadena, 1, 5)
16          cadena2 = Mid(cadena, 7, 5)
17          cadena3 = Mid(cadena, 13, 5)
18
19          txttem.Text = cadena1.Trim
20          txthum.Text = cadena2.Trim
21          txtluz.Text = cadena3.Trim
22
23
24      End Sub
25  End Class
26
 9          Dim cadena1 As String
10          Dim cadena2 As String
11          Dim cadena3 As String
12
13          cadena = SerialPort1.ReadExisting()
```

***Sketch (Codigo) 8**. Código del formulario y del Control Timer.*

Una vez escrito el código del formulario y del Timer podremos visualizar los valores reportados por el Arduino en la interfaz desarrollada en Visual Basic.NET. En esta interfaz tendremos los valores de temperatura, humedad y luz en tiempo real tal y como se muestra en la Figura 40. Estos valores coinciden con los mostrados en la pantalla LCD del arreglo electrónico.

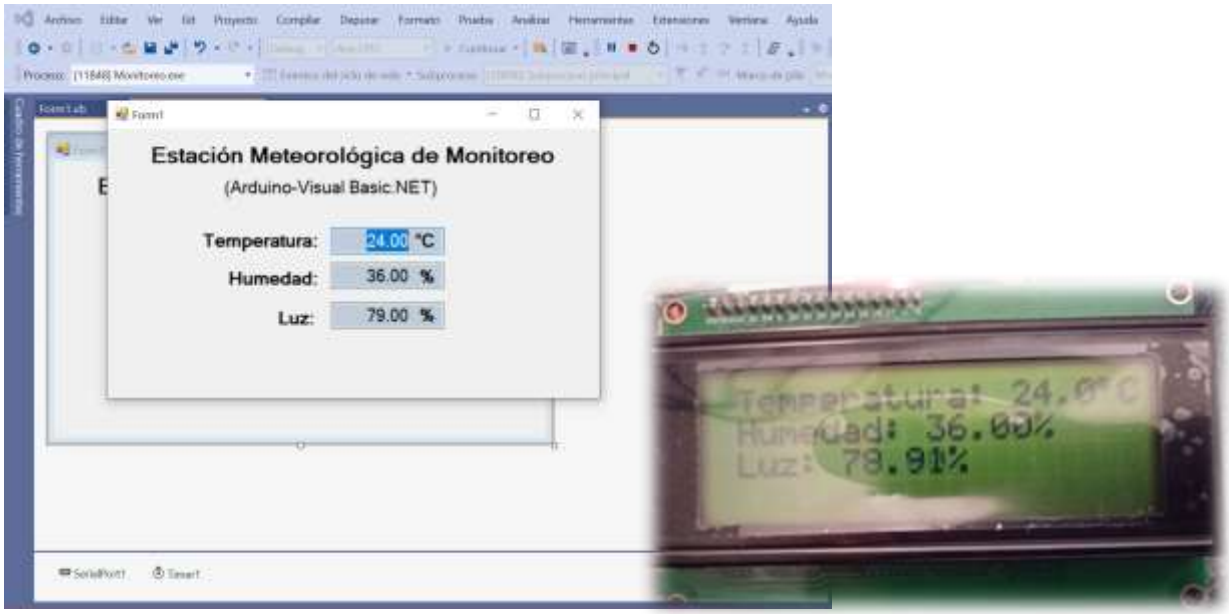


***Figura 39**. Estación de Monitoreo desarrollada con Arduino y Visual Basic.NET*

4 Registro y Respaldo de Datos (Modulo SD).

Al arreglo electrónico anterior se le adicionó un módulo microSD que servirá de respaldo de la estación de registro de datos para guardar los datos localmente, Figura 41. Esto se hace con el fin de tener almacenado el momento en el que ocurrió un suceso, se activó un evento dentro del proceso o si un valor registrado sobrepasa un límite programado. La incorporación del Módulo MicroSD demanda del uso de las librerías **SD.h y SPI.h** incluidas ya en el IDE de Arduino. Por lo que no se requiere su descarga de fuentes externas.

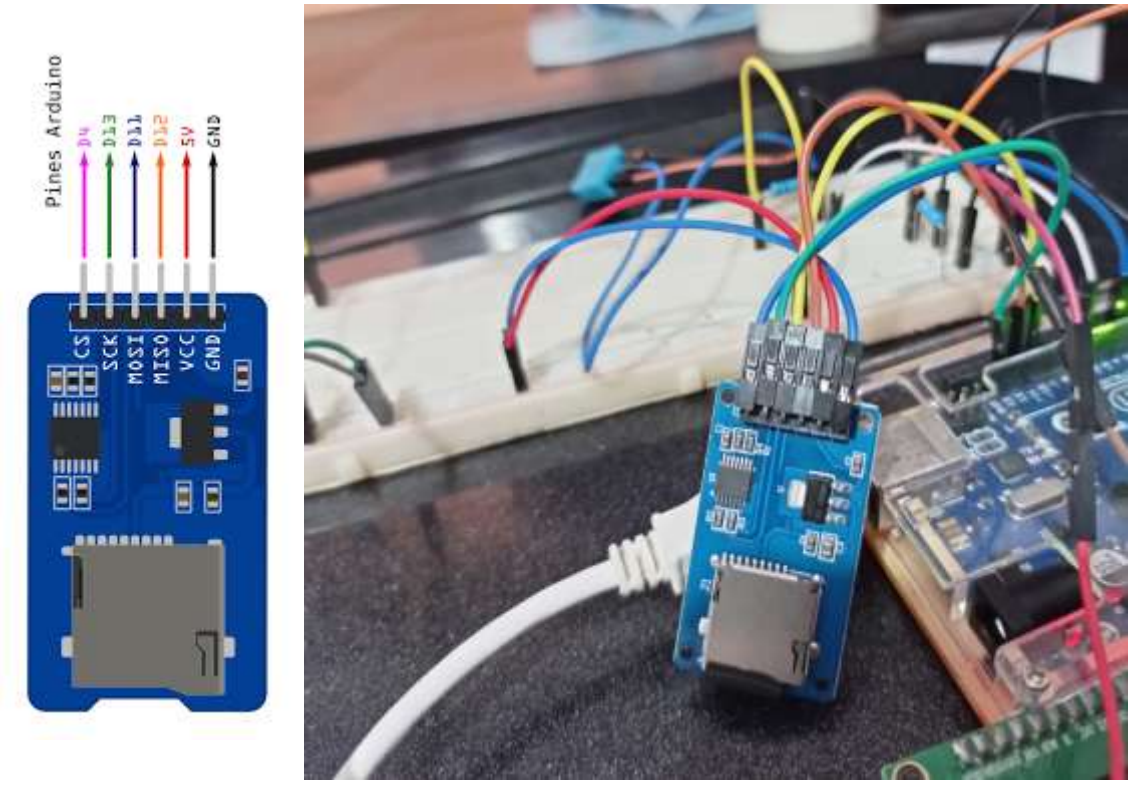


***Figura 40**. Modulo Micro SD de la Tarjeta de Arduino agregado a la Estación de Monitoreo.*

A continuación.

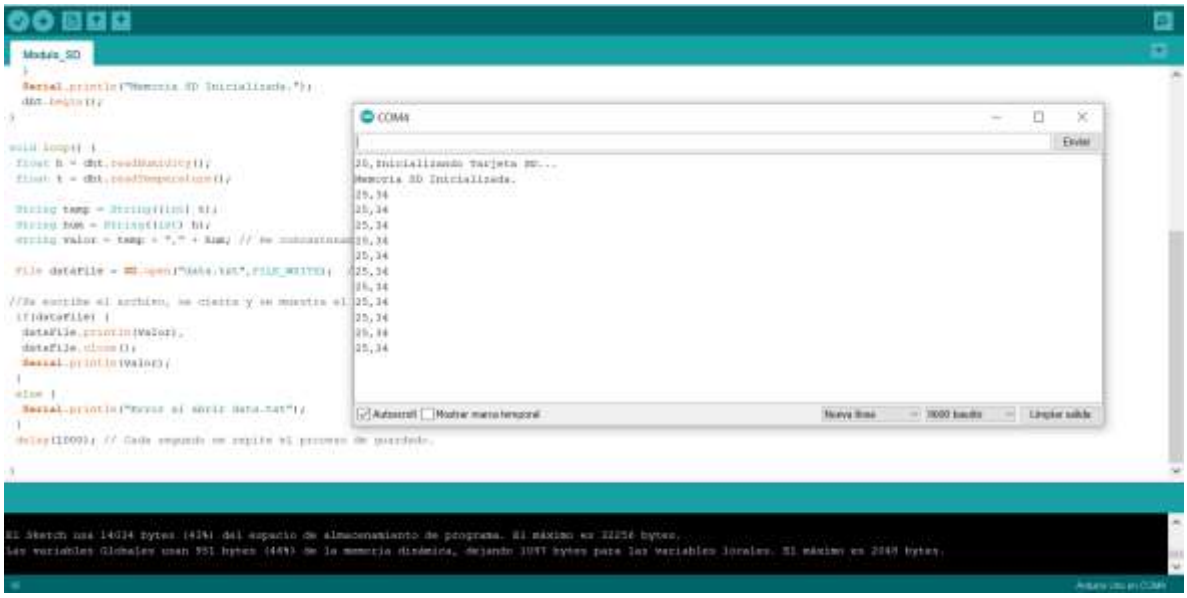
***Sketch (Codigo) 9**. se presenta el Sketch desarrollado en el IDE de Arduino a fin de realizar el respaldo local de los datos*

```
//Librerias del programa
#include <DHT.h>
#include <SD.h>
#include <SPI.h>

#define DHTPIN 7
#define DHTTYPE DHT11

const int chipSelect =4; // Pin de selección de la memoria.
DHT dht(DHTPIN, DHTTYPE);

void setup() {
 Serial.begin(9600);
 // Se verifica que la memoria se haya detectado.
 Serial.println("Inicializando Tarjeta SD...");
  if (!SD.begin(chipSelect)) {
    Serial.println("Tarjeta no presente");
    return;
  }
  Serial.println("Memoria SD Inicializada.");
  dht.begin();
}

void loop() {
 float h = dht.readHumidity();
 float t = dht.readTemperature();

 String temp = String((int) t);
 String hum = String((int) h);
 String valor = temp + "," + hum; // Se concantenan los valores.

 File dataFile = SD.open("data.txt",FILE_WRITE);  // Se crea el archivo.

//Se escribe el archivo, se cierra y se muestra el valor guardado en el monitor serial.
 if(dataFile) {
  dataFile.println(valor),
  dataFile.close();
  Serial.println(valor);
 }
 else {
  Serial.println("Error al abrir data.txt");
 }
 delay(1000); // Cada segundo se repite el proceso de guardado.

}
```

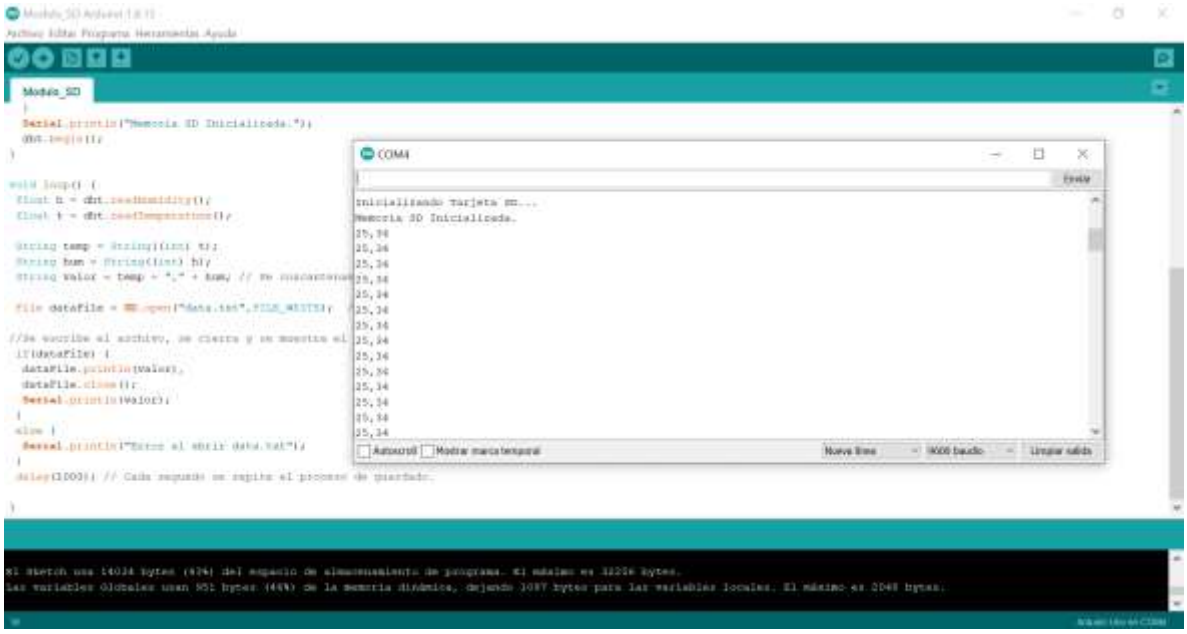
***Sketch (Codigo) 10.** Sketch desarrollado para la incorporación del Módulo MicroSD.*

El funcionamiento del sketch puede ser apreciado en las Figuras 43-44. Podemos ver en estas imágenes como la tarjeta MicroSD se inicializa correctamente y comienza a crear el archivo data.txt, registrando los valores de tempertura y humedad propocionados por el sensor DHT11. Cada segundo se realiza el registro de un nuevo valor en el archivo data.txt.

Figura 41.*Funcionamiento Sketch desarrollado para la incorporación del Módulo MicroSD.*



***Sketch (Codigo) 11**. Funcionamiento Sketch desarrollado para la incorporación del Módulo MicroSD.*



5 Sistema de Monitoreo y Adquisición a través del Módulo Bluetooth HC-05.

Al arreglo electrónico desarrollado para el monitoreo y adquisición de datos se le adicionó un módulo Bluetooth HC-05 que servirá para realizar la transferencia y lectura de datos a distancia mediante tecnología Bluetooth, Figura 45. Esto se hace con el fin de poder enviar los datos desde Arduino hasta un dispositivo a distancia teniendo como intermediario de la comunicación al sensor HC-05.

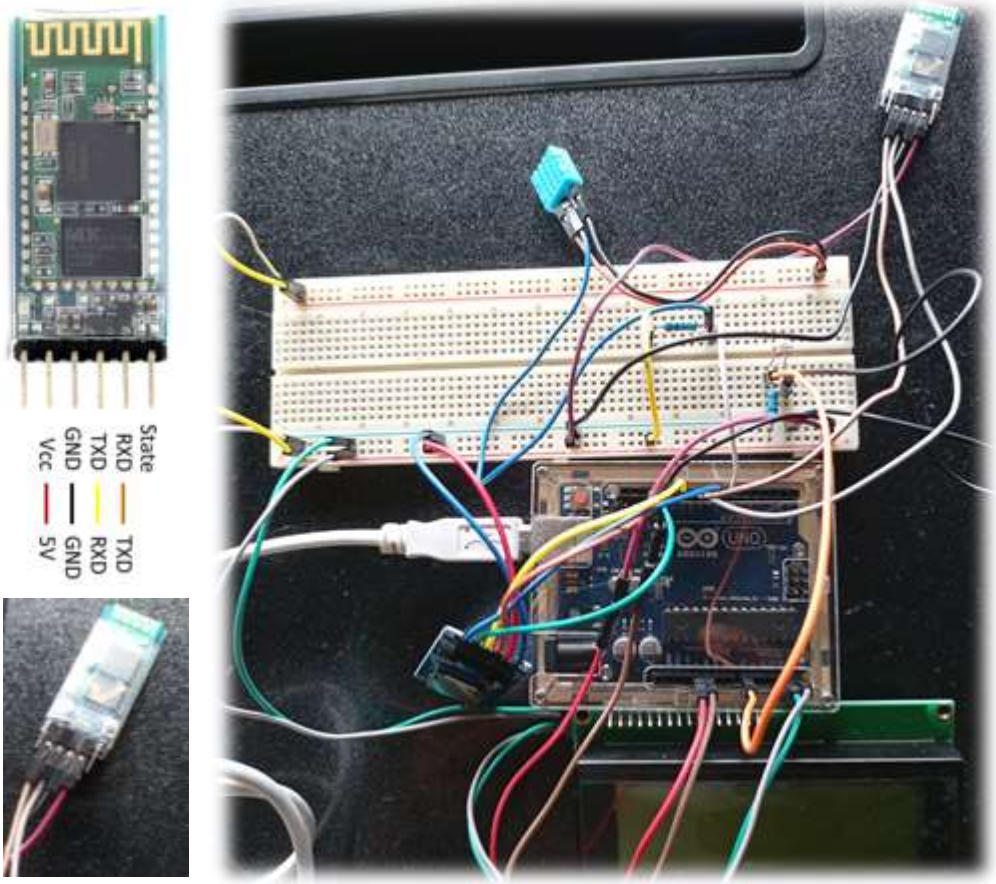


Figura 42. *Arreglo Electrónico de la Estación de Monitoreo y Adquisición de Datos con un módulo Bluetooth HC-05.*

Una vez montado el Módulo Bluetooth HC-05 procedemos a desarrollar el Sketch que nos permita configurar el Módulo, Figura 46.

Sketch (Codigo) 12. Sketch desarrollado en el IDE para establecer comunicación entre la PC y el módulo Bluetooth a través de un microcontrolador Arduino.

```
#include <SoftwareSerial.h>   // Incluimos la libreria  SoftwareSerial
SoftwareSerial BT(9,10);    // Definimos los pines RX y TX del Arduino conectados al Bluetooth

void setup()
{
  BT.begin(9600);       // Inicializamos el puerto serie BT (Para Modo AT 2)
  Serial.begin(9600);   // Inicializamos  el puerto serie
}

void loop()
{
  if(BT.available())    // Si llega un dato por el puerto BT se envia al monitor serial
  {
    Serial.write(BT.read());
  }

  if(Serial.available())  // Si llega un dato por el monitor serial se envia al puerto BT
  {
     BT.write(Serial.read());
  }
}
```

Una vez desarrollado y cargado el Sketch procedemos a configurar nuestro Módulo Bluetooth HC-05 como esclavo, Figura 47. Cuando está configurado de esta forma, se comporta similar a un HC-06, esperando a que un dispositivo bluetooth maestro se conecte a este. Generalmente este modo se utiliza cuando el módulo se conecta con una PC o Celular, pues éstos se comportan como dispositivos maestros.

A continuación, se muestran los pasos que se deben seguir para realizar la configuración:

Sketch (Codigo) 13. *Sketch desarrollado en el IDE para establecer comunicación entre la PC y el módulo Bluetooth a través de un microcontrolador Arduino*

a) Verificar si estamos en modo AT:
   Enviar: AT
   Recibe: OK

b) Establecer el Role como Esclavo:
   Enviar: AT+ROLE=0
   Respuesta: OK

c) Configurar el Nombre del módulo:
   Enviar: AT+NAME=Estacion
   Respuesta: OK

d) Establecer el Pin de vinculación:
   Enviar: AT+PSWD="1212"
   Respuesta: OK

e) Configura la Velocidad:
   Enviar: AT+UART=9600,0,0
   Respuesta: OK

f) Verificar los parámetros cambiados:
   Enviar:
   AT+ROLE?
   AT+PSWD?
   AT+UART?
   Respuesta:
   +ROLE:0
   OK
   +PSWD:1212
   OK
   +UART:9600,0,0
   OK

g) Reiniciar el módulo:
   Enviar: AT+RESET
   Respuesta: OK

Sketch (Codigo) 14. *Configuración del Módulo HC-05 como esclavo.*

Una vez configurado el Módulo Bluetooth vinculamos éste con la PC o con el dispositivo móvil en el que recibiremos los datos suministrados por el sensor. Agregamos el dispositivo con el nombre dado en el paso anterior "Estacion" y con clave previamente definida "1212", Figura 48-49.

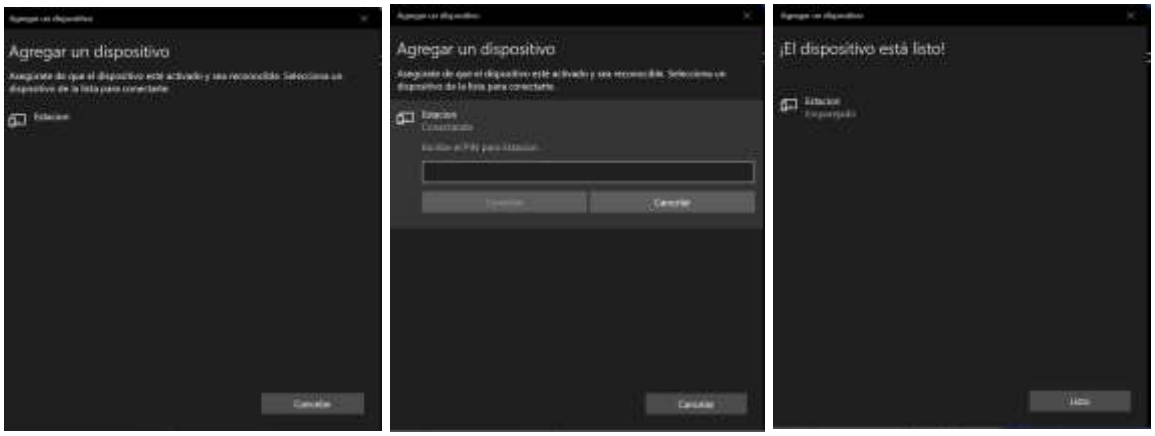


Figura 44. *Vinculación del Módulo Bluetooth con la PC.*

Figura 43. *Información del sistema de la Configuración de los puertos del módulo BT*

Una vez vinculado el Módulo Bluetooth las posibilidades del uso de la estación se monitoreo se amplían enormemente toda vez que la estación puede vincularse con aplicaciones desarrolladas para dispositivos móviles en las versiones de Android o IOS; o bien puede vincularse con directamente con formularios de Visual Studio.NET. Para esto basta con configurar el parámetro PortName para que coincida con el puerto del módulo Bluetooth designado para el módulo BT.

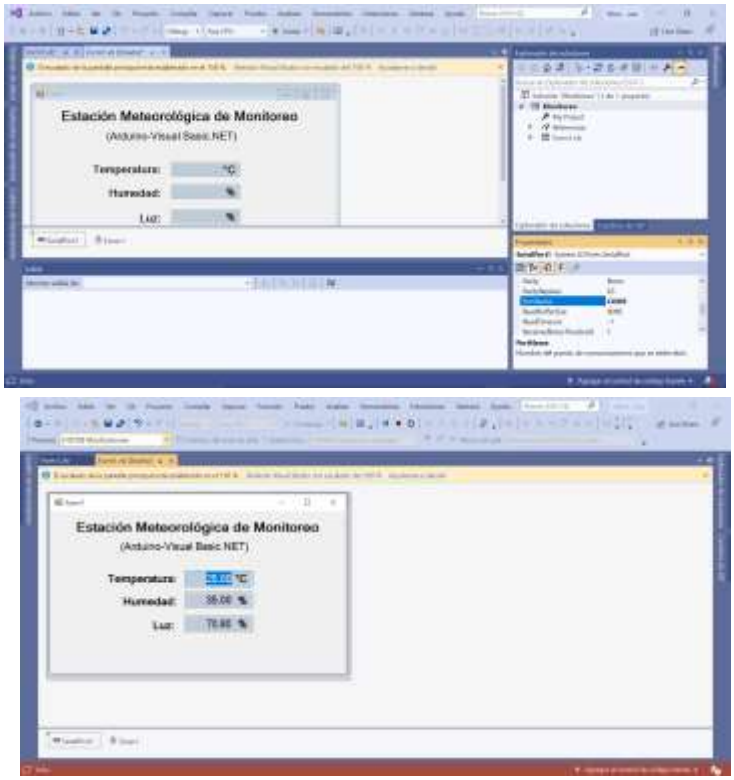


Figura 45. *Ajuste del control SerialPort para establecer conexión con el módulo BT.*

6 Bibliografía/Referencias

(1) Extraído a partir de: https://www.arduino.cc/en/Guide/Introduction

(2) Extraído a partir de: https://hetpro-store.com/TUTORIALES/lm35/

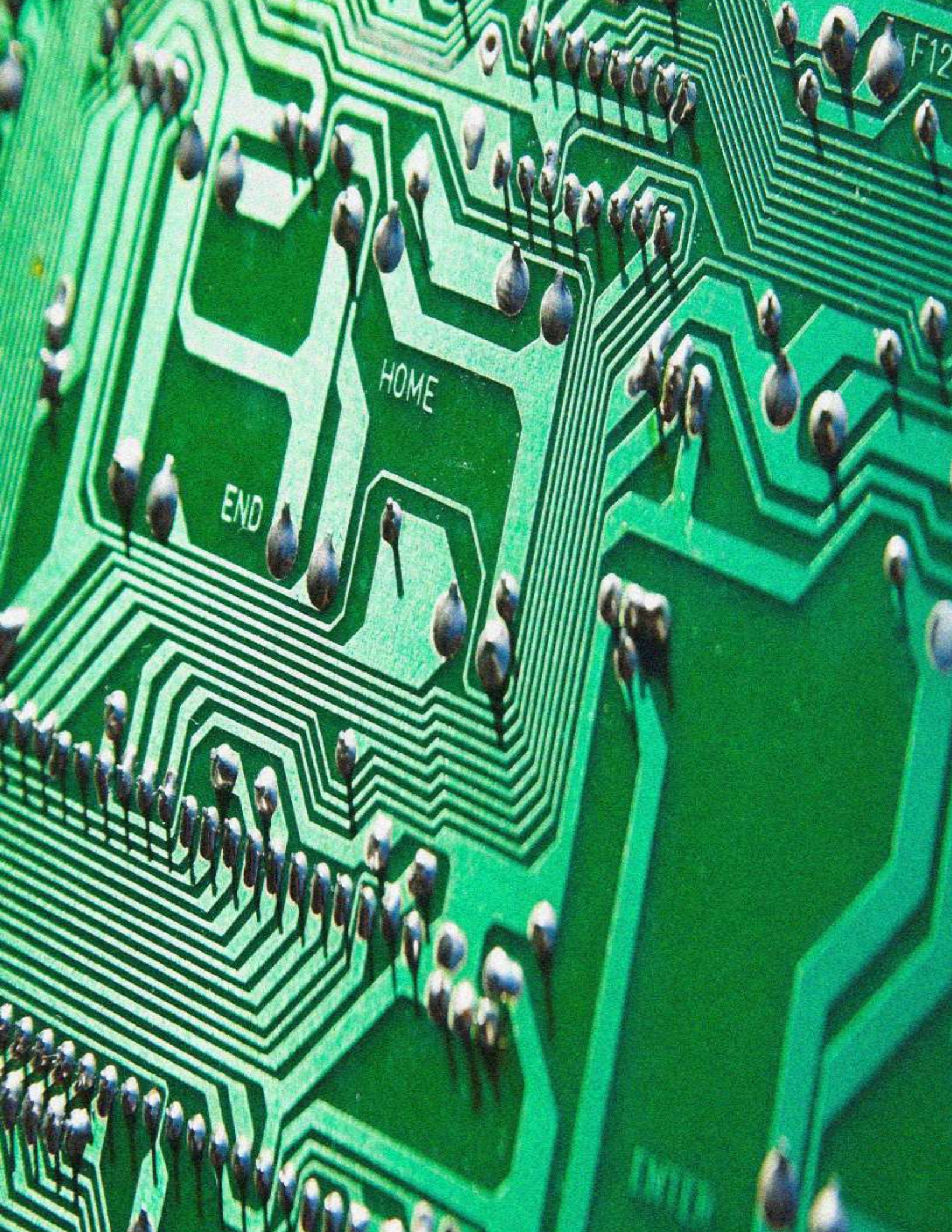
(3) Extraído a partir de: https://naylampmechatronics.com/sensores-proximidad/10-sensor-ultrasonido-hc-sr04.html

(4) Extraído a partir de: https://sandorobotics.com/producto/hs1149/

(5) Extraído a partir de: https://prometec.mx/producto/ldr-sensor-de-luz/

(6) Extraído a partir de: https://www.luisllamas.es/medir-inclinacion-con-arduino-y-sensor-tilt-sw-520d/

(7) Extraído a partir de: https://formacion.intef.es/catalogo/mod/book/view.php?id=69&chapterid=361


F12
HOME
END