

Curso: Arduino y ApplInventor para mejorar tus clases. Nivel Medio (Edición 1)

MEMORIA

1. Propuesta de la práctica.
2. Materiales utilizados.
3. Explicación del funcionamiento.
4. Programa realizado en el IDE Arduino.
5. Esquema eléctrico.
6. Esquema de conexión.
7. Vídeo del proyecto donde se observe su funcionamiento (enlace).

1. Propuesta de la práctica.

En un principio he de explicar el **contexto** en el que se usará el Arduino.

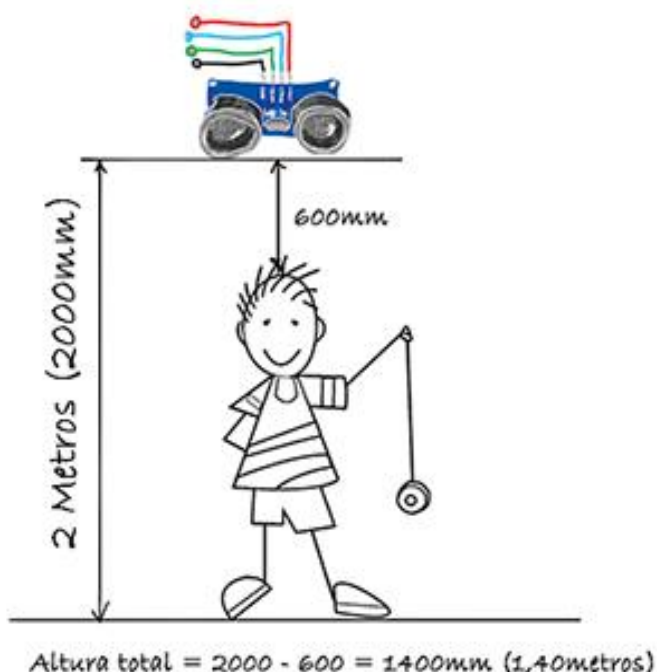
Actualmente trabajo en un colegio de primaria con 20 niños de 4º y 5º en la misma clase. He pensado en el “Arduino” como una herramienta muy útil a la hora de recoger datos, para más tarde ser tratados en asignaturas como matemáticas o ciencias naturales.

Para ello. He pensado hacer una rutina diaria en la que se tomarán datos acerca de la temperatura y la humedad. Para ello colocaremos nuestro dispositivo cerca de una ventana y alejada de cualquier radiador que pudiera falsear los datos.

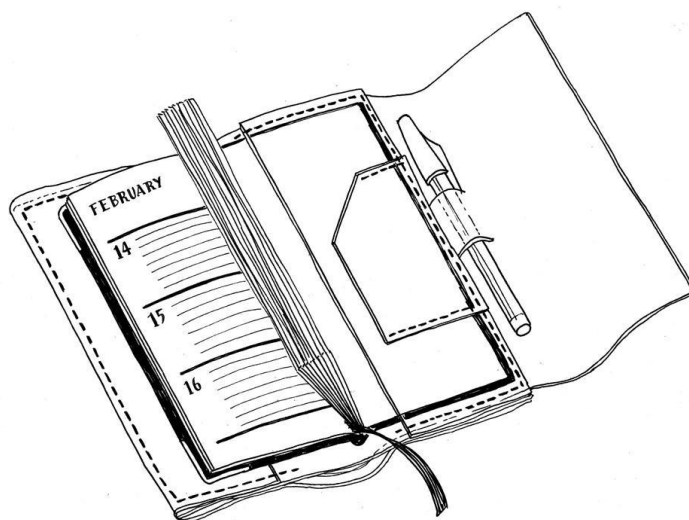
La toma de datos se realizará todos los días en la asamblea por el responsable de la clase, antes de comenzar, y anotarán dichos datos a mano en un diario.

En **Ciencias Naturales** ya en el último trimestre, recopilaremos dichos datos, y los pondremos en unas tablas de registro mensuales. Donde los alumnos podrán recapacitar acerca de la evolución de la temperatura y la humedad desde septiembre hasta mayo (Mayo es la fecha de nuestra programación didáctica de mate donde trataremos el tratamiento e interpretación de tablas y gráficas).

En **Matemáticas** Trabajaremos por trimestres y recopilaremos semanalmente la información relacionada con la altura de los alumnos. Todos los viernes a última hora haremos una medición de nuestros alumnos. Para ello pondremos el sensor de distancia en LA pared mirando hacia abajo a una distancia exacta de 2m (2000 mm), e iremos colocando debajo a nuestros niños. La distancia que mira el sensor se la restaremos a 2000mm, teniendo la altura del niño. Como podemos ver en el siguiente esquema:



Con los datos obtenidos haremos un registro en el diario, y veremos la evolución de la altura de nuestros alumnos. Haciendo medias aritméticas trimestrales por sexo y edad. Como *temas transversales* veremos como se pueden aplicar las diferentes tablas de datos e interpretarlas en términos globales como por ejemplo. ¿Cuánto hemos crecido este trimestre? ¿Y el anterior? ¿Qué tipo de dieta hemos tenido? ¿Cuántas horas he dormido?.



Como elemento extra de recogida de datos, podríamos utilizar una báscula de baño, en la que además mediremos el peso de nuestros alumnos. Todo esto entra dentro de lo que denominamos un estilo de vida saludable. Recogidos en los estándares evaluativos de estos cursos.

Cabe destacar el aspecto interdisciplinar que estamos trabajando. Ya que tanto en matemáticas como en naturales, la recogida de datos es muy significativa. Además que la adquisición de hábitos saludables es algo que no podemos pasar por alto.

En **matemáticas** también podemos enfocarlo en dirección a la resolución de problemas. Tal es el caso de la medida por parte del sensor de la distancia hasta a cabeza. ¿Cómo podemos saber la altura del alumno? Los alumnos podrán ver “en la realidad” dicho problema y manipularlo, para así poder llegar a la conclusión de que la resta de esa medida al total (2000mm), es la solución problema.

En **Lengua** valoraremos la limpieza y el orden a la hora de la recogida de datos. Los alumnos encargados de hacerlos rotarán de tal manera que sea una responsabilidad común y sea necesario que los datos estén bien escrito con limpieza y orden.

2. Materiales utilizados.



Arduino Uno



Breadboard



Cables Dupont



Pantalla LCD 1602



Potenciómetro 10k



Cable usb



Sensor de humedad y temperatura

DHT11



Sensor de Proximidad

HCSR04

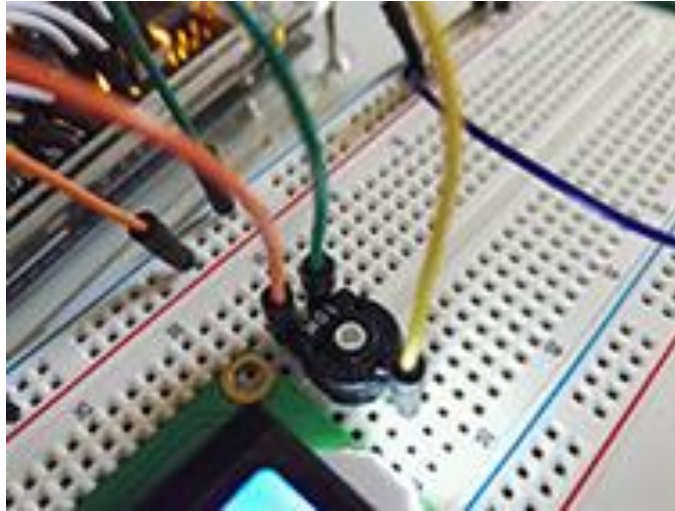


Módulo Bluetooth DH06

3. Explicación del funcionamiento.

“Uso de un móvil para mostrar en una pantalla LCD: distancia, humedad y temperatura”

Hay que recordar que hemos usado una pantalla LCD 1602 que no tiene i2c, por lo que hemos tenido que usar un potenciómetro de 10k para controlar el brillo.



Para poder conectar por Bluetooth hemos utilizado un modulo BT HC06.

Los sensores DHT11 y HCSR04 están conectados en el miso Arduino.

Nada más iniciarse el Arduino, la pantalla mostrará el siguiente mensaje:

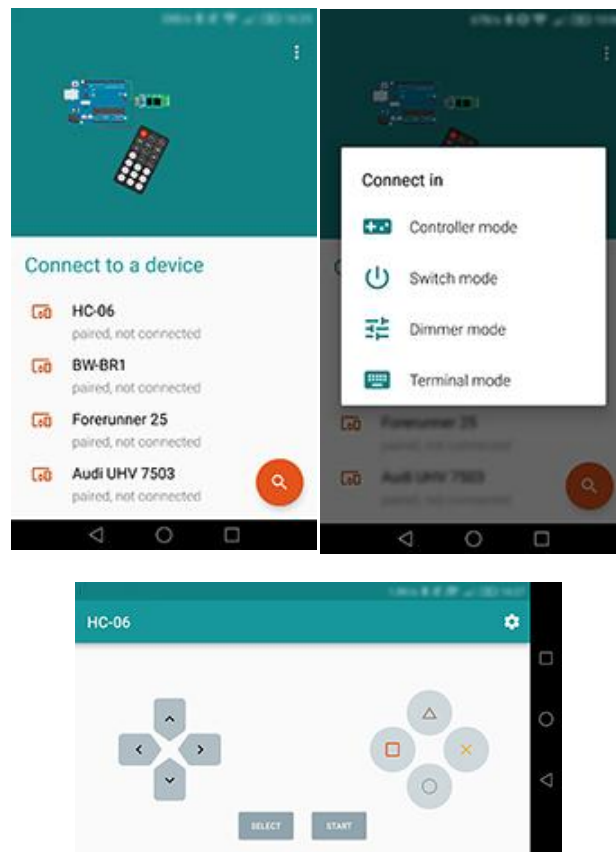
“X Distancia”

“O Humed.Temp.”



Gracias a esta pantalla damos al usuario las opciones disponibles a la hora de pulsar botones en la app.

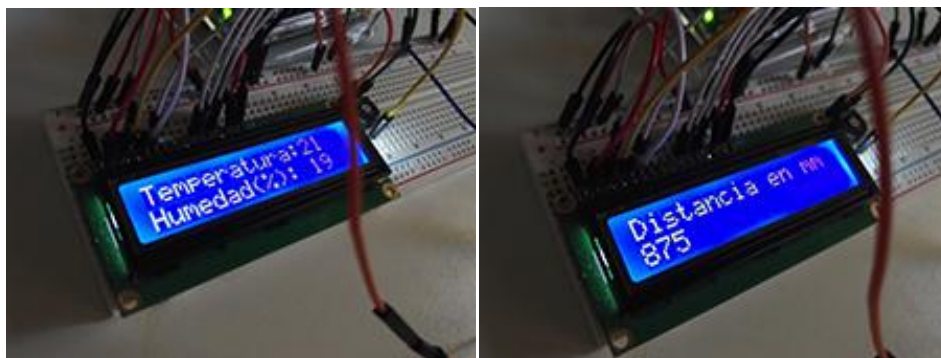
Se usará un Smartphone con la app “Arduino Bluetooth Controller” trabajando en el modo “Controller”. Para ello hay que emparejar el Smartphone con el dispositivo HC06. con en la cual se muestran varios botones virtuales (Direcciones, select, start y 4 botones de acción).



De estos pulsadores, nosotros solo necesitaremos usar 2, que configuraremos en la propia app con las letras “X” y “O”. En cada pulsación la respuesta será la siguiente:

El botón “O” (Circulo) se activará el sensor **DHT11** y mostrará en la pantalla LCD los valores de la humedad y la temperatura. Cada pulsación implica una medición

El botón “X” (Equis) el cual pondrá en marcha el sensor **HCSR04** y pondrá en pantalla la distancia que en ese momento exista cada vez que presionemos el botón.



Una posible ampliación de este proyecto sería utilizar el protocolo I2C donde un Arduino Maestro, se comunicaría con otro Arduino “esclavo” (o varios) que nos aportaría las medidas de los sensores. Por ello hemos dejado libres los pines A4 y A5.

Autor: Juan Antonio Laguna Herrezuelo.

4. Programa realizado en el IDE Arduino.

En este apartado mostraré el programa con todas las anotaciones pertinentes:

```
#include <LiquidCrystal.h> //Librería pantalla

#include <SimpleDHT.h> // librería que tenemos que incluir

#include <SoftwareSerial.h>

#include <hcsr04.h> //Librería del sensor de distancia

#define TRIG_PIN A0 //Pines del sensor de distancia

#define ECHO_PIN A1

HCSR04 hcsr04(TRIG_PIN, ECHO_PIN, 20, 4000); //Valores del sensor

SoftwareSerial BT1(3,4); // Sensor BT. Pines 3 y 4 OJO: RX, TX recordar que se cruzan

SimpleDHT11 sensor; // Se usa para decir el tipo de sensor, dht11 <-- ÉSTE. o dht22

// Constante del sensor y su pin

int SensorHT=2; // Sensor Humedad y Temperatura "HT" en Pin2

LiquidCrystal lcd(7, 8, 9, 10, 11 , 12); //Pantalla y pines.

int BluetoothData; //Variable para el BT

void setup()

{

  pinMode (SensorHT,INPUT); //pin2 y de entrada.

  Serial.begin(9600); //Iniciamos monitor serie

  lcd.begin(16, 2); //Iniciamos la pantalla

  lcd.print("X Distancia");// Escribe en la primera línea y columna

  lcd.setCursor(0,1);//Segunda línea y primera columna

  lcd.print("O Humed.Temp. "); //Escribe Humedad(%): Segunda línea primera columna

  BT1.begin(9600); //Iniciamos BT

}

void loop()

{
```

```
//Aquí definimos las variables

byte temperature = 0;

byte humidity = 0;

int err = SimpleDHTerrSuccess;

byte data[40] = {0};

//Que hace Arduino si encuentra algún tipo de error

if(sensor.read(SensorHT,&temperature,&humidity, data))

{

    Serial.print("No reading , err="); Serial.println(err);delay(1000);

    return;

}

if (BTI.available()) //Para emparejar los dispositivos BT

{

    BluetoothData=BTI.read();//Leemos valores del BT

    if(BluetoothData=='0') //Al presionar "0"

    {

        lcd.clear();

        lcd.setCursor(0,0);//Primera linea y columna

        lcd.print("Temperatura:");//Escribe Temp:º En la primera linea

        lcd.print((int)temperature);//Escribe el valor de "Temp: "

        lcd.setCursor(0,1);//Segunda linea y primera columna

        lcd.print("Humedad(%):");//Escribe Humedad(%): Segunda linea primera columna

        lcd.print((int)humidity);//Escribe el valor después de Humedad(%): "

        delay(750);//Esperamos un poco.

    }

}
```



```
if(BluetoothData=='X') //Al presionar "X"
{
  lcd.clear();
  Serial.println(hcsr04.distanceInMillimeters()); // Configura distancia en mm
  Serial.println(hcsr04.ToString()); // Información del dispositivo, driver, y distancia en formato JSON
  lcd.setCursor(0,0); //Primera línea y columna
  lcd.print("Distancia en mm "); //Escribe en primera línea.
  lcd.setCursor(0,1); //Cambia de línea.
  lcd.print(hcsr04.distanceInMillimeters()); //Escribe la distancia en mm
  delay(750); //Esperamos un poco
}
}

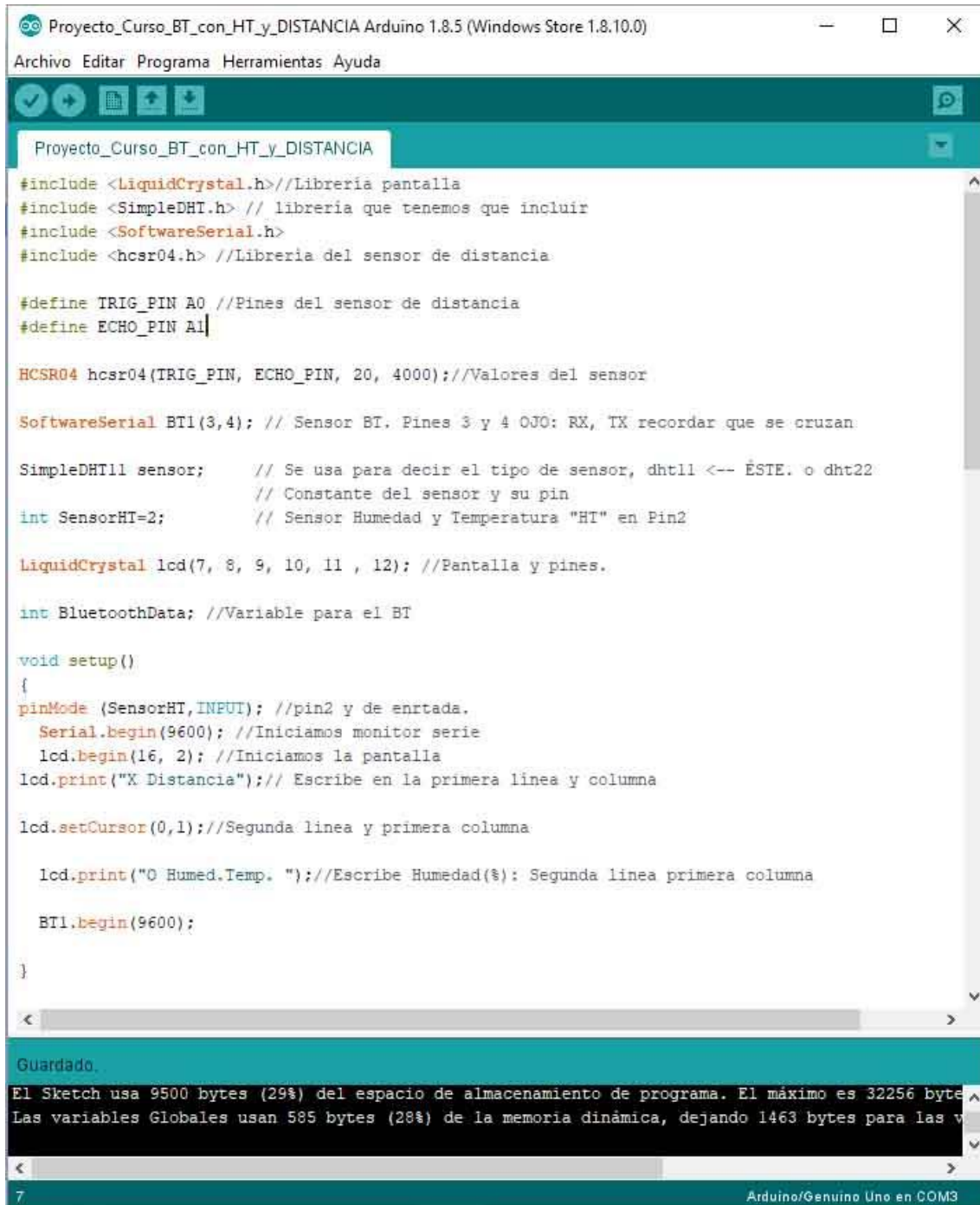
}
```

Debido a que el formato de texto es un poco incomodo. Proporciono enlace y capturas con el código del programa:

Enlace archivo INO:

<https://drive.google.com/file/d/1RNVvqNFYSW1zzDtqHiHoTwCiI3Bt5gYk/view?usp=sharing>

Capturas:



```

Proyecto_Curso_BT_con_HT_y_DISTANCIA Arduino 1.8.5 (Windows Store 1.8.10.0)
Archivo Editar Programa Herramientas Ayuda

Proyecto_Curso_BT_con_HT_y_DISTANCIA

#include <LiquidCrystal.h> //Libreria pantalla
#include <SimpleDHT.h> // libreria que tenemos que incluir
#include <SoftwareSerial.h>
#include <hcsr04.h> //Libreria del sensor de distancia

#define TRIG_PIN A0 //Pines del sensor de distancia
#define ECHO_PIN A1

HCSR04 hcsr04(TRIG_PIN, ECHO_PIN, 20, 4000); //Valores del sensor

SoftwareSerial BT1(3,4); // Sensor BT. Pines 3 y 4 OJO: RX, TX recordar que se cruzan

SimpleDHT11 sensor; // Se usa para decir el tipo de sensor, dht11 <-- ÉSTE. o dht22
// Constante del sensor y su pin
int SensorHT=2; // Sensor Humedad y Temperatura "HT" en Pin2

LiquidCrystal lcd(7, 8, 9, 10, 11, 12); //Pantalla y pines.

int BluetoothData; //Variable para el BT

void setup()
{
  pinMode (SensorHT, INPUT); //pin2 y de entrada.
  Serial.begin(9600); //Iniciamos monitor serie
  lcd.begin(16, 2); //Iniciamos la pantalla
  lcd.print("X Distancia");// Escribe en la primera línea y columna

  lcd.setCursor(0,1);//Segunda línea y primera columna

  lcd.print("O Humed.Temp. "); //Escribe Humedad(%): Segunda línea primera columna

  BT1.begin(9600);
}

Guardado...
El Sketch usa 9500 bytes (29%) del espacio de almacenamiento de programa. El máximo es 32256 bytes
Las variables Globales usan 585 bytes (28%) de la memoria dinámica, dejando 1463 bytes para las v
7 Arduino/Genuino Uno en COM3

```

```

Proyecto_Curso_BT_con_HT_y_DISTANCIA Arduino 1.8.5 (Windows Store 1.8.10.0)
Archivo Editar Programa Herramientas Ayuda

Proyecto_Curso_BT_con_HT_y_DISTANCIA

void loop()
{
    //Aquí definimos las variables

    byte temperature = 0;
    byte humidity = 0;
    int err = SimpleDHTErrSuccess;
    byte data[40] = {0};

    //Que hace Arduino si encuentra algún tipo de error

    if(sensor.read(SensorHT,&temperature,&humidity, data))

    {
        Serial.print("No reading , err="); Serial.println(err);delay(1000);
        return;
    }
    if (BT1.available()) //Para emparejar los dispositivos BT
    {
        BluetoothData=BT1.read();//Leemos valores del BT

        if(BluetoothData=='0') //Al presionar "0"
        {
            lcd.clear();

            lcd.setCursor(0,0);//Primera linea y columna

            lcd.print("Temperatura:");//Escribe Temp:° En la primera linea

            lcd.print((int)temperature);//Escribe el valor de "Temp: "

            lcd.setCursor(0,1);//Segunda linea y primera columna
        }
    }
}

```

Guardado.

El Sketch usa 9500 bytes (29%) del espacio de almacenamiento de programa. El máximo es 32256 bytes.
 Las variables Globales usan 585 bytes (28%) de la memoria dinámica, dejando 1463 bytes para las variables locales.

7 Arduino/Genuino Uno en COM3

```

Proyecto_Curso_BT_con_HT_y_DISTANCIA

lcd.print((int)temperature);//Escribe el valor de "Temp: "

lcd.setCursor(0,1);//Segunda linea y primera columna

lcd.print("Humedad(%): ");//Escribe Humedad(%): Segunda linea primera columna

lcd.print((int)humidity);//TEscribe el valor después de Humedad(%): "
delay(750);

}
if(BluetoothData=='X')
{
  lcd.clear();

  Serial.println(hcsr04.distanceInMillimeters());// Configura distancia en mm

  Serial.println(hcsr04.ToString());// Información del dispositivo, driver, y distancia en forma de string

  lcd.setCursor(0,0);//Primera linea y columna

  lcd.print("Distancia en mm ");//Escribe en primera linea.

  lcd.setCursor(0,1); //Cambia de linea.
  lcd.print(hcsr04.distanceInMillimeters()); //Escribe la distancia en mm

  delay(750);
}
}

}

```

Guardado.

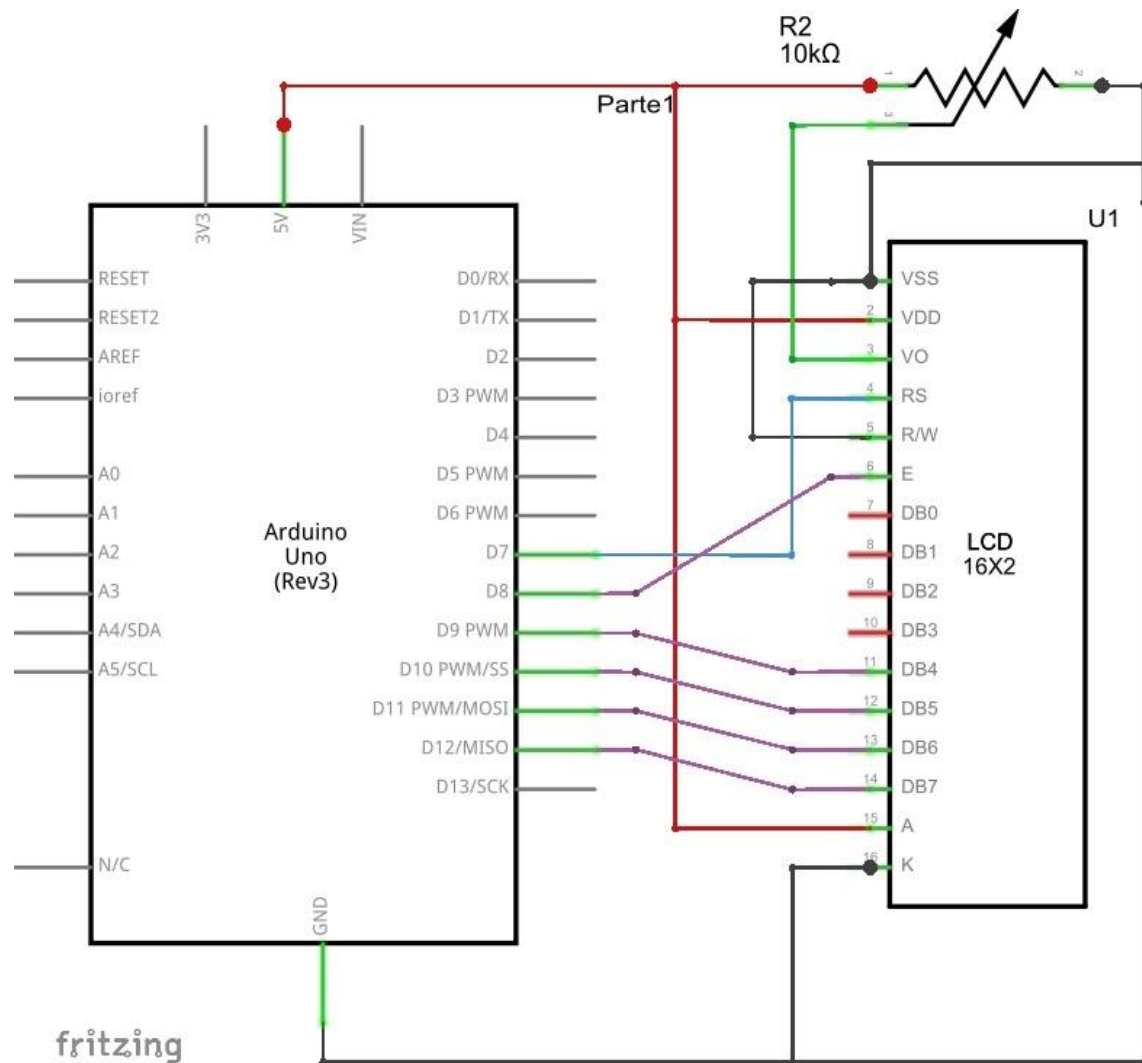
El Sketch usa 9500 bytes (29%) del espacio de almacenamiento de programa. El máximo es 32256 bytes.
 Las variables Globales usan 585 bytes (28%) de la memoria dinámica, dejando 1463 bytes para las variables locales.
 Las variables Locales usan 0 bytes (0%) de la memoria dinámica, dejando 0 bytes para las variables locales.
 Las variables Estáticas usan 0 bytes (0%) de la memoria dinámica, dejando 0 bytes para las variables estáticas.

7 Arduíno/Genuíno Uno en COM3

5. Esquema eléctrico:

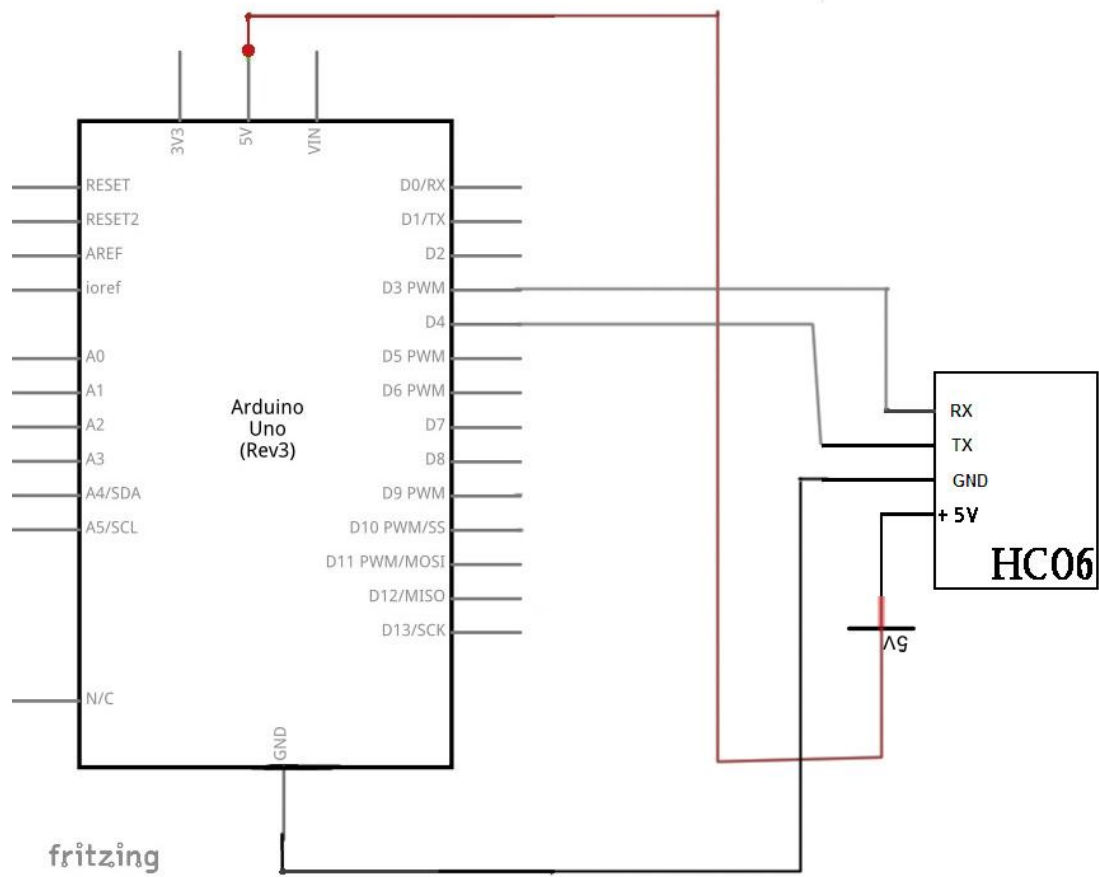
Los siguientes esquemas han sido hechos con Fritz o Photoshop.

LCD1602

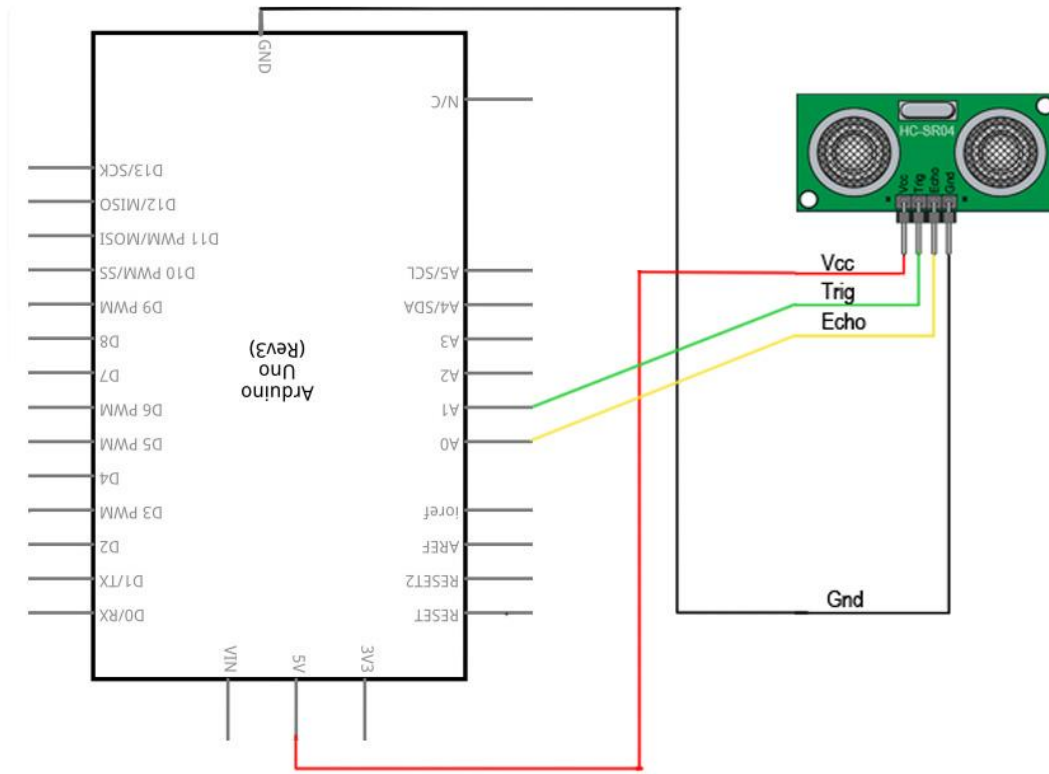


Autor: Juan Antonio Laguna Herrezuelo.

HC06

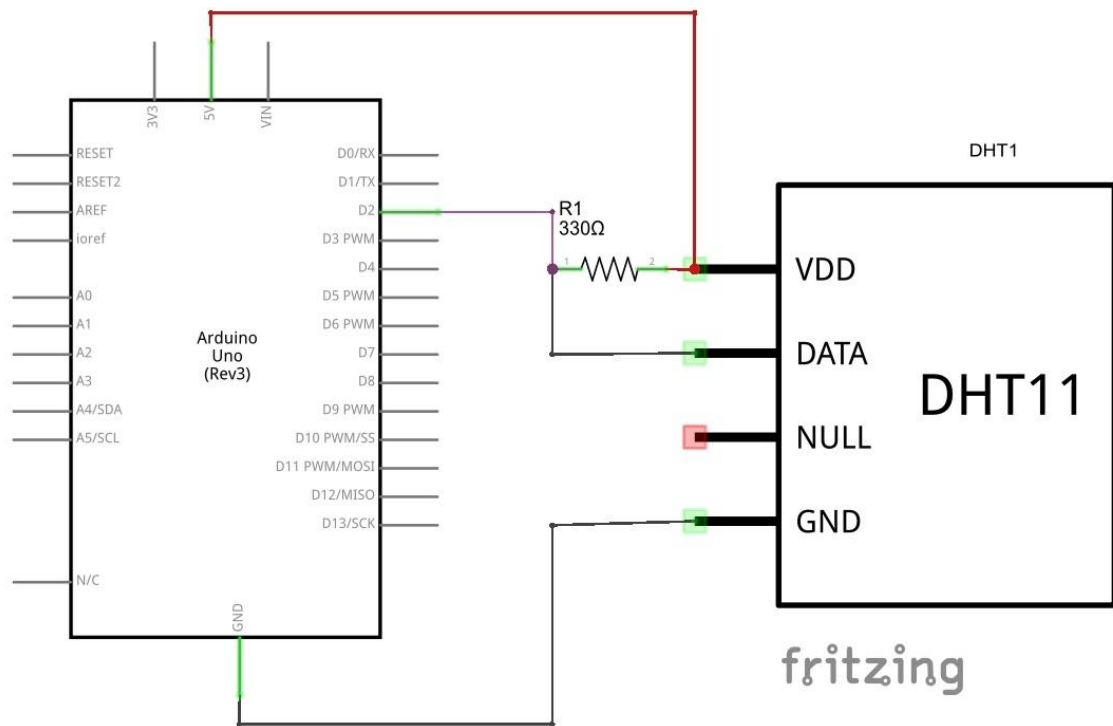


HCSR04



fritzing

DHT11

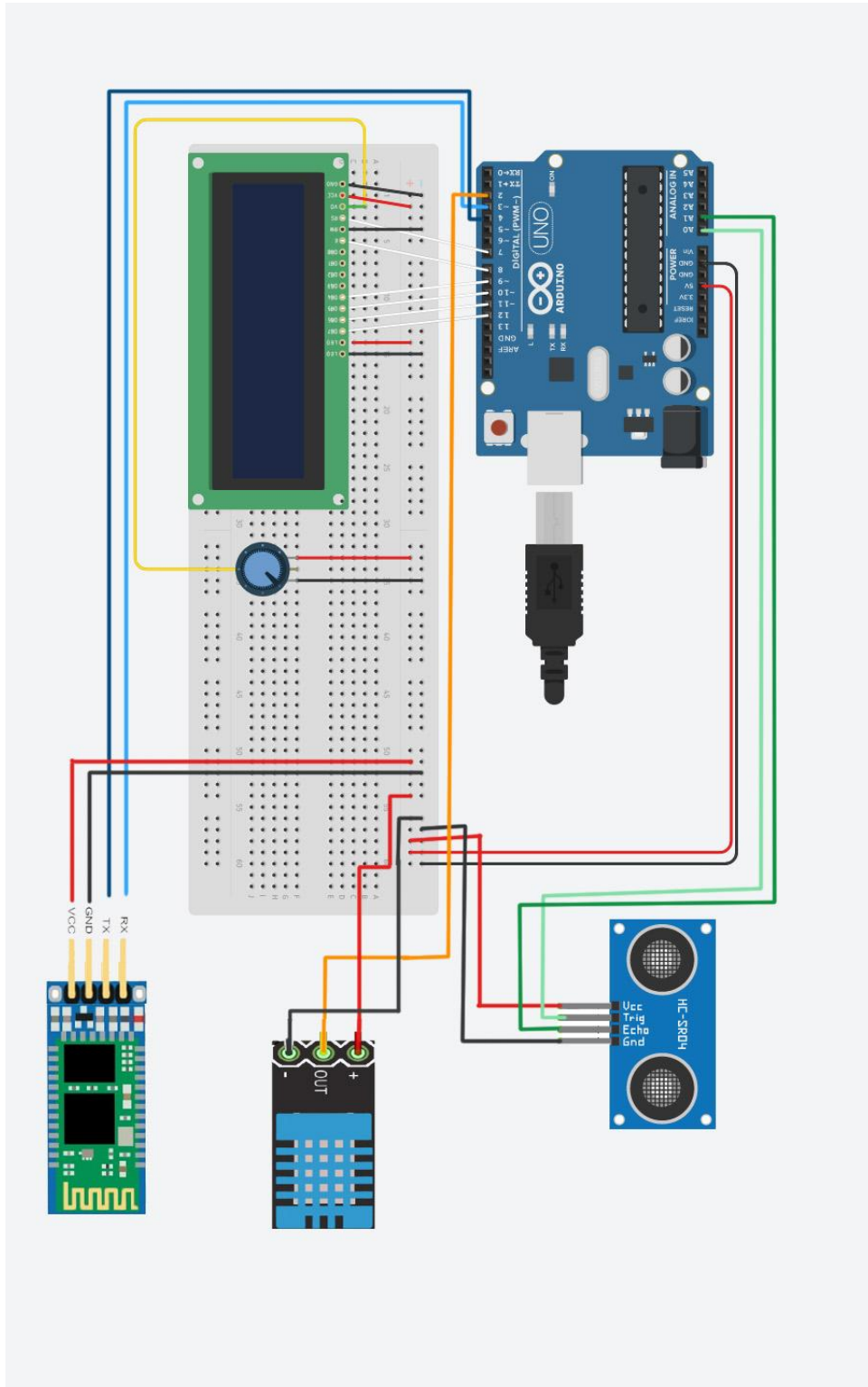


6. Esquema de conexión:

El esquema de conexión ha sido realizado gracias a Tinkercad, y las piezas que no salen en dicha web (DHT11, HCRS04 Y HC06), han sido aparte editado con Photoshop. Se nota perfectamente por los ángulos creados, que son diferentes:

Link PSD:

<https://drive.google.com/file/d/1iUd8yo1QlIxxhRAXQNvdM7axEplJkHRBU/view?usp=sharing>



Autor: Juan Antonio Laguna Herrezuelo.

7. Vídeo del proyecto donde se observe su funcionamiento (enlace).

<https://drive.google.com/file/d/1z2cJU7d8y7fFE4sR7HDgJ7-sKGdYYvYr/view>