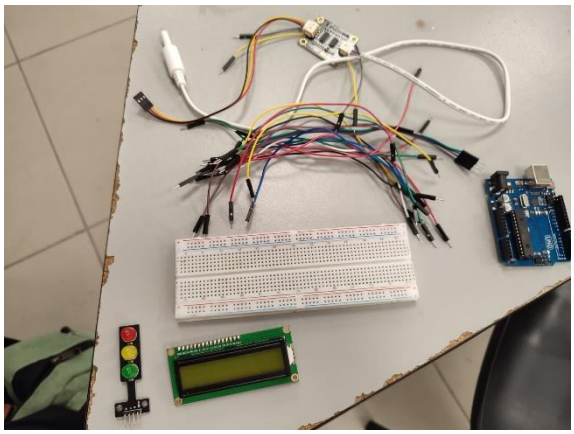
REPORTE ARDUINO MEDIDOR DE TSD

PRUEBA DE FUNCIONAMIENTO

PROYECTO FINAL

MATERIALES



Jumpers

Una pantalla LCD

Un semáforo de LEDS

Un potenciómetro de 10K

Un Arduino UNO

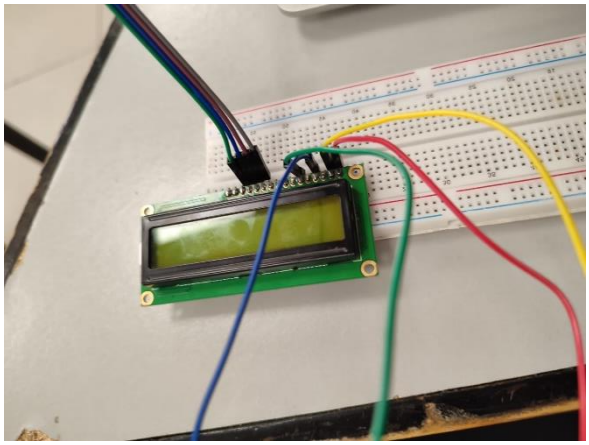
Una protoboard

Medidor de TSD

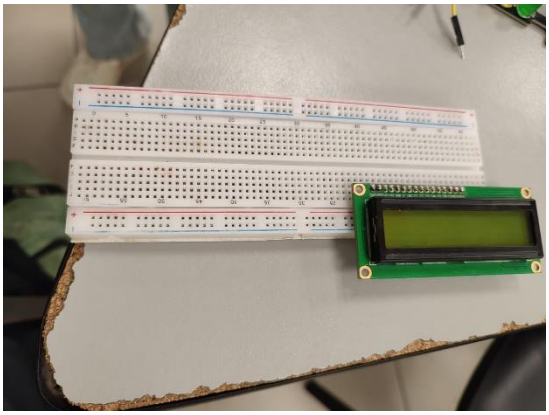
(El precio del proyecto es estimado de 1200 ya que el kit de Arduino corto 900 pesos y lo que es el semáforo y el medidor de TSD salió en 300.)

DESARROLLO

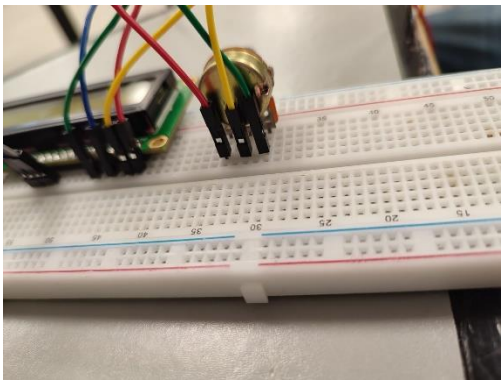
1. Primero vamos a conecta la pantalla LCD ya que es un tanto complicada, recomendación soldé pines para que sea mucho más fácil las conexiones ya que así todos los hoyos que existen en el protoboard se convertirán en ese pin al conectarse.



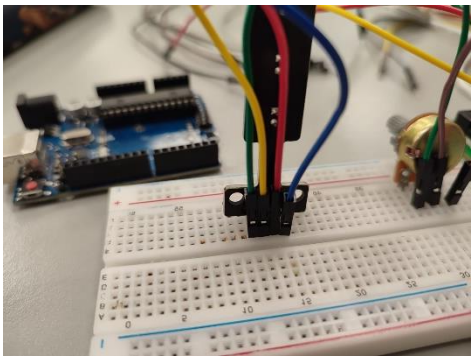
2. Ahora vamos a conectar los siguientes pines que están señalados en la pantalla: GND,VDD, RS, RW, E, D4, D5, D6, D7, BLA, BLK estos pines representan el voltaje, la tierra, el pin análogo, y los pines que irán conectados al Arduino. (Los primeros dos y los dos últimos dos van conectados al positivo y negativo de la protoboard respectivamente.



3. Después conectaremos el potenciómetro, el jumper verde es el de el voltaje, el amarillo es el análogo que ira conectada a la LCD y el rojo es el de la tierra, el jumper amarillo va a ir conectado a lo que el VO de la pantalla LCD.

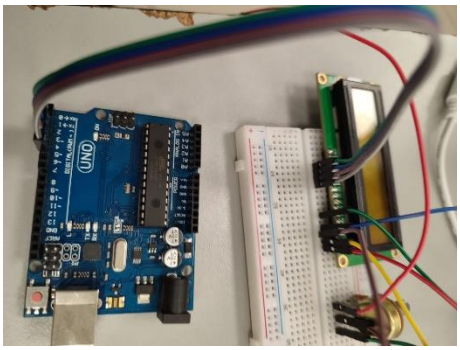


4. Ahora se conectará el semáforo LED el cual solo se usaran4 jumpers tres para conectar los colores y 1 para la tierra.



Con estos componentes ya conectados, se puede empezar la conexión del Arduino la cual consiste en conectar la pantalla y los LEDS al Arduino.

1. Primero conectaremos la pantalla LCD al Arduino, conectaremos los pines digitales, D4, D5, D6, D7, estos pines iran conectados a las entradas 5 a 2. (D7-2, D6-3, D5-4, D4-5), y por últimos el RS al 12 y la E al 11.

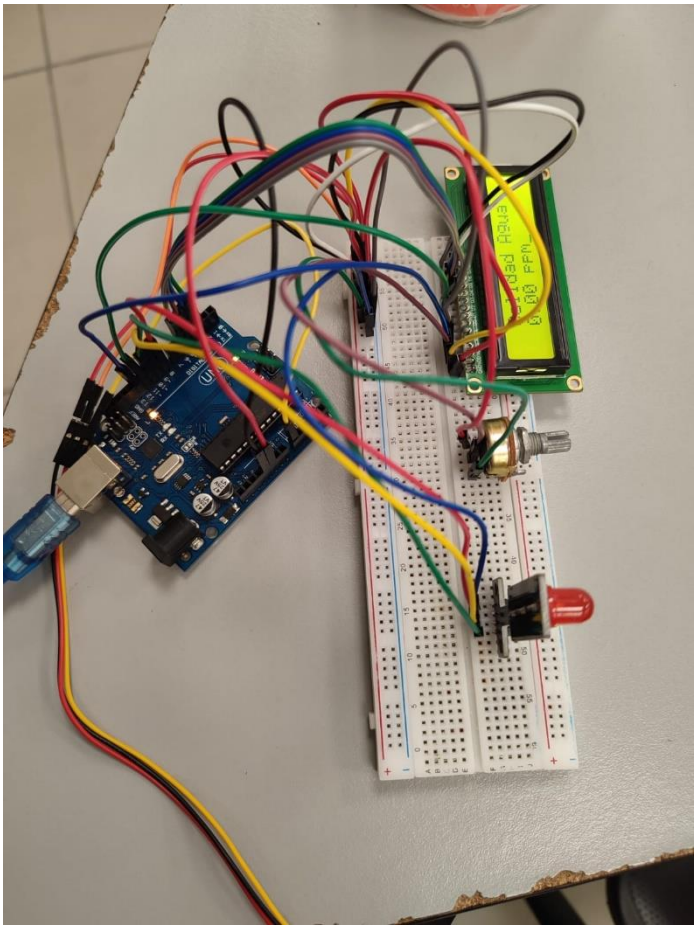


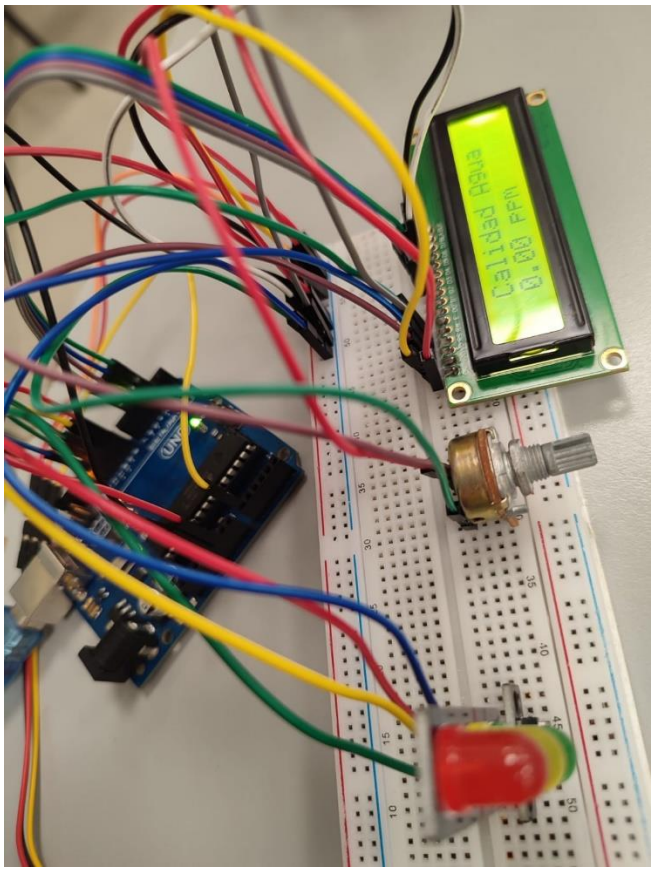
2. Después conectaremos el semáforo de LEDS, el verde ira al 10, el amarillo al 9 y el rojo al 8, recordemos que la tierra va al negativo de la tierra.
3. Luego, se conectará el medidor de TSD el cual tiene sus jumpers los cuales van conectados los primeros dos al negativo y al positivo y la A va a ir conectada al A0 del Arduino.
4. Por último, conectaremos el GND y los 5V del Arduino a la protoboard.

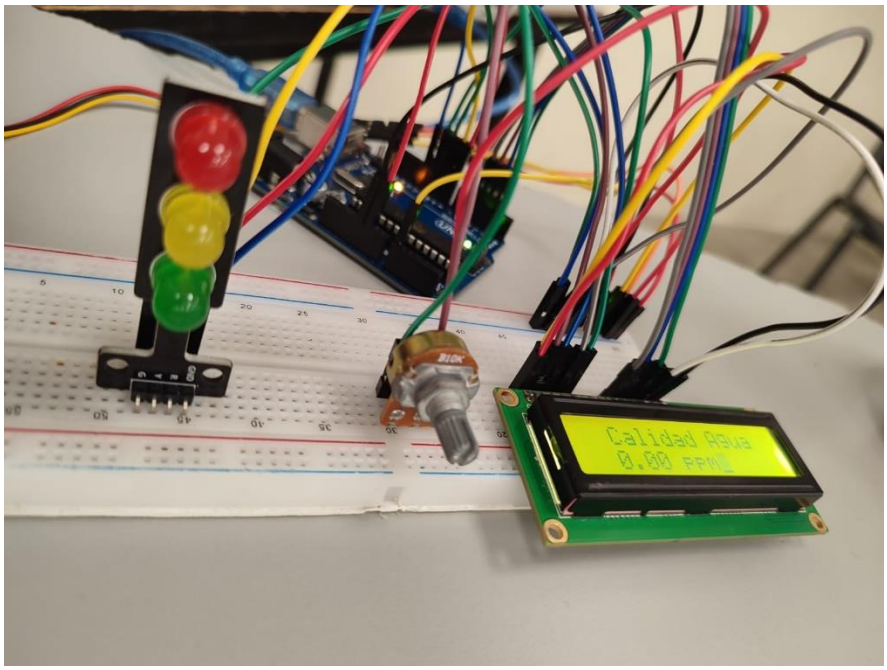
TESTEO

Cuando se inicio la conexión se tuvo algunos problemas ya que la pantalla no mostraba el valor o algún carácter, pero lo demás si funcionaba, se estuvo intentando, pero después se cambió de posición algunas conexiones y ya funciono.

Cuando se le demostró al profesor se usaron 3 vasos para en uno testear uno con sal, uno con tierra y uno siendo agua potable, se puedo demostrar que el prototipo trabajaba de manera correcta.







El inicio de este reporte explicará cada una de las líneas de código de este proyecto, este fue sacado del canal de youtube de Vadechips: https://www.youtube.com/watch?v=Ysq6PQtlKZw&t=148s&ab_channel=vadechips.

Librerías:

Al inicio del código tenemos la inicialización de librerías:

```
// Incluimos las librerías
#include <EEPROM.h>
#include "GravityTDS.h"
#include <LiquidCrystal.h>
```

Estas constan de 3, en donde solo “GravityTDS.h” debe de instalarse localmente, sus funciones son:
<EEPROM.h>: Sirve para guardar información en la memoria EEPROM, que ya se encuentra ubicada en el arduino, curiosamente no se encuentra en la línea del código ninguna funcionalidad de esta librería, así que se podría quitar sin ningún problema.

“GravityTDS.h”: Sirve para poder registrar los inputs del medidor TPS que nos ayuda a calcular el nivel de ppm(partes por millón) del agua.

<LiquidCrystal.h>: Una librería utilizada para poder manejar las pantallas LCD.

Definiendo los pines:

```
#define TdsSensorPin A0
#define GreenLED 10
#define YellowLED 9
#define RedLED 8
```

El único pin analógico colocado va a ser el TdsSensorPin, las demás líneas, que ejemplifican a las luces LED, se encuentran ubicados en los pines digitales.

Instanciar librerías:

```
// Instanciamos el objeto GravityTDS y LiquidCrystal para poder usar los sensores
GravityTDS gravityTds;
// LiquidCrystal(rs, e, d4, d5, d6, d7)
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

La primera línea: “GravityTDS gravityTds;” nos sirve para colocar una variable de tipo GravityTDS con nombre gravityTds, este último se podría cambiar por el que queramos, modificando todos las demás líneas con el mismo nombre.

La segunda línea es LiquidCrystal “lcd(12,11,5,4,3,2);”, esta sirve para dar ubicaciones donde los pines digitales estén conectados en el arduino, se suele conectar los primeros 2 para el rs y e de la pantalla lcd respectivamente, y los últimos 4 van de las conexiones DB4-DB7 de la pantalla.

Otorgar valores.

```
float temperature = 15, tdsValue = 0;
```

Se crean dos variables de tipo float: temperature = 15 y tdsValue =0;
La variable: “temperature” tiene un valor definido debido a que no contamos con un medidor de este parámetro, así que simplemente colocamos el valor usual, en grados, que tiene el agua en condiciones normales, y “tdsValue” es donde vamos a guardar los datos que obtengamos con el medidor TPS.

Función void(setup):

```
// Configuración inicial
void setup()
{
  Serial.begin(115200);

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

  // Configuramos el pin donde irá conectado el sensor TDS
  gravityTds.setPin(TdsSensorPin);
  // Definimos voltaje de referencia del ADC (5V en Arduino UNO)
  gravityTds.setAref(5.0);
  // Definimos el rango del ADC (1024 en Arduino UNO)
  gravityTds.setAdcRange(1024);
  // Inicializamos el sensor
  gravityTds.begin();

  // Configuramos como salida los pines donde están conectados loso LEDS
  pinMode(RedLED, OUTPUT);
  pinMode(YellowLED, OUTPUT);
  pinMode(GreenLED, OUTPUT);
}
```

“Serial.begin(115200);” otorga la velocida de datos en la que va a correr el programa, se mide en baudios y el predeterminado a usar si tenemos un arduino UNO es 115200. Si está conectada a una computadora y se cambia el valor causaría errores, debido a que los baudios entre ambos deben de ser iguales.

lcd.begin(16,2); Inicializa la pantalla LCD, los 2 valores ejemplifican el número de filas y el número de columnas. Normalmente es 16,2 pero existen otras pantallas más grandes o más chicas.

gravityTds.setPint(TdsSensorPin); Aquí se coloca el pin donde irá la medición de gravityTds en este caso ya definimos TdsSensorPin como A0. Es un pin analógico ya que va cambiando continuamente de valor.

gravityTds.setAref(5.0); Esta línea se usa porque estamos trabajando con valores continuos (TdsSensorPin) y necesitamos un límite máximo donde estos valores pueden oscilar. Ese límite es el 5 y dejamos el 0, ya que no queremos un mínimo.

gravityTds.setAdcRange(1024); En esta parte estamos colocando tenemos 2 puntos importantes “1024” y “ADC” básicamente lo que quiere decir es que estamos usando toda la capacidad del ADC del microprocesador para nuestras funciones.

gravityTds.begin(); Se inicializa el sensor.

pinMode(RedLED, OUTPUT);
pinMode(YellowLED, OUTPUT);
pinMode(GreenLED, OUTPUT);

Esta serie de instrucciones “pinMode” nos permite colocarle el valor donde están conectados los LED.

Función void(loop)

Parte 1.

```
void loop()
{
  // Ejecutamos la compensación por temperatura indicando la temperatura del agua a testear
  gravityTds.setTemperature(temperature);
  // Ejecutar una lectura
  gravityTds.update();
  // Obtener el valor en PPM
  tdsValue = gravityTds.getTdsValue();

  // Imprimimos por consola el valor leído
  Serial.print(tdsValue,0);
  Serial.println("ppm");

  // Limpiamos la pantalla
  lcd.clear();

  // Ponemos el cursor en la columna y fila deseada y escribimos la información
  lcd.setCursor(4, 0);
  lcd.print("Calidad Agua");
  lcd.setCursor(4, 1);
  lcd.print(tdsValue, 0);
  lcd.print(" ppm");
```

En esta parte vamos a ir mostrando los valores obtenidos por el TDS en la pantalla LCD.

gravityTds.setTemperature(temperature); Esta línea simplemente está especificando un valor constante, en este caso 15, a la función setTemperature, esto se utiliza para poder medir los datos arrojados por el TDS

gravityTds.update(); y **tdsValue =gravityTds.getTdsValue();**
Estas dos líneas están obteniendo valores diferentes por cada iteración del loop, la primera nos otorga un dato y en la segunda función lo guardamos en la variable tdsValue.

Serial.print(tdsValue,0); En esta parte estamos consiguiendo en el programa el valor otorgado por el TDS que obtuvimos desde la función .update y se mostrará con 0 decimales.

serial.println("ppm"); Aquí usamos un mismo print pero con ln ya que queremos un salto de línea después del ppm.

lcd.clear() Limpia la pantalla

lcd.setCursor(4, 0); Aquí vamos a decirle al Arduino donde queremos comenzar a colocar texto
lcd.print("Calidad Agua"); Ponemos el texto que deseamos colocar.
lcd.setCursor(4, 1); Volvemos a colocar el lugar del siguiente dato.
lcd.print(tdsValue, 0); Ponemos el valor otorgado por tdsValue con 0 decimales.
lcd.print(" ppm"); Se coloca "ppm" después del valor.

-Básicamente en esta serie de instrucciones conseguimos un valor dado por el TDS lo vamos a mostrar en la pantalla LCD y luego se eliminará ese dato para volver a arrojar otro valor diferente, dando como resultado la ilusión de que se están actualizando los valores en la pantalla conforme pasa el tiempo.

Parte 2.

```
  if(tdsValue > 0 && tdsValue <= 300){
    digitalWrite(GreenLED, HIGH);
    delay(200);
    digitalWrite(GreenLED, LOW);
    delay(200);
  }else if(tdsValue > 300 && tdsValue <= 600){
    digitalWrite(YellowLED, HIGH);
    delay(200);
    digitalWrite(YellowLED, LOW);
    delay(200);
  }else if(tdsValue > 600){
    digitalWrite(RedLED, HIGH);
    delay(200);
    digitalWrite(RedLED, LOW);
    delay(200);
  }

  delay(100);
}
```

Esta serie de instrucciones nos dice cuál luz led vamos a prender dependiendo del valor arrojado por el TDS, los datos son sacados según la OMS.