Desarrolladores:

Miguel Angel Uribe Zuluaga
Juan Felipe Higuita Perez
David Alejandro Suárez Varón

2. Use el sensor de humedad de suelo y el Sketch realizado en el Laboratorio 2, para determinar la presencia o ausencia de humedad (defina su nivel adecuado de humedad).

Componentes:

-Arduino UNO
-Pantalla LCD 16x2
-Breadboard
-Sensor Soil Moisture
-Potenciómetro
-Resistencia

En este reto nos decían que usemos el sketch del laboratorio 2 para que en vez de detectar presencia o ausencia de lluvia ahora detecte el nivel de humedad del suelo para ver si hay o no hay humedad, para determinar que un suelo este húmedo o no decidimos que el umbral va ser de 500 siendo que si el valor llegado desde el sensor de humedad Soil Moisture es mayor se detectara *presencia de humedad* pero en caso contrario de que este sea menor, se detectará y mostrará en la pantalla LCD *Ausencia de humedad*, ahora vamos hablar del código paso por paso:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd_1(12, 11, 5, 4, 3, 2);
const int sensorPin = A0;
const int sensorDigitalPin = 7;
const int umbralHumedad = 500;

void setup() {
  pinMode(sensorDigitalPin, INPUT);
  lcd_1.begin(16, 2);
  lcd_1.print("Deteccion de humedad");
}

void loop() {
  int sensorValue = analogRead(sensorPin);
  int digitalValue = digitalRead(sensorDigitalPin);

  lcd_1.setCursor(0, 1);
  lcd_1.print("Valor ADC: ");
  lcd_1.print(sensorValue);

  lcd_1.setCursor(0, 0);
  if (sensorValue > umbralHumedad) {
    lcd_1.print("Presencia de humedad");
  } else {
    lcd_1.print("Ausencia de humedad");
  }
  lcd_1.setCursor(0, 1);
  lcd_1.blink();
  lcd_1.print("Introduccion Ing. Electronica");
  delay(2000);
  lcd_1.noBlink();
  delay(2000);
```

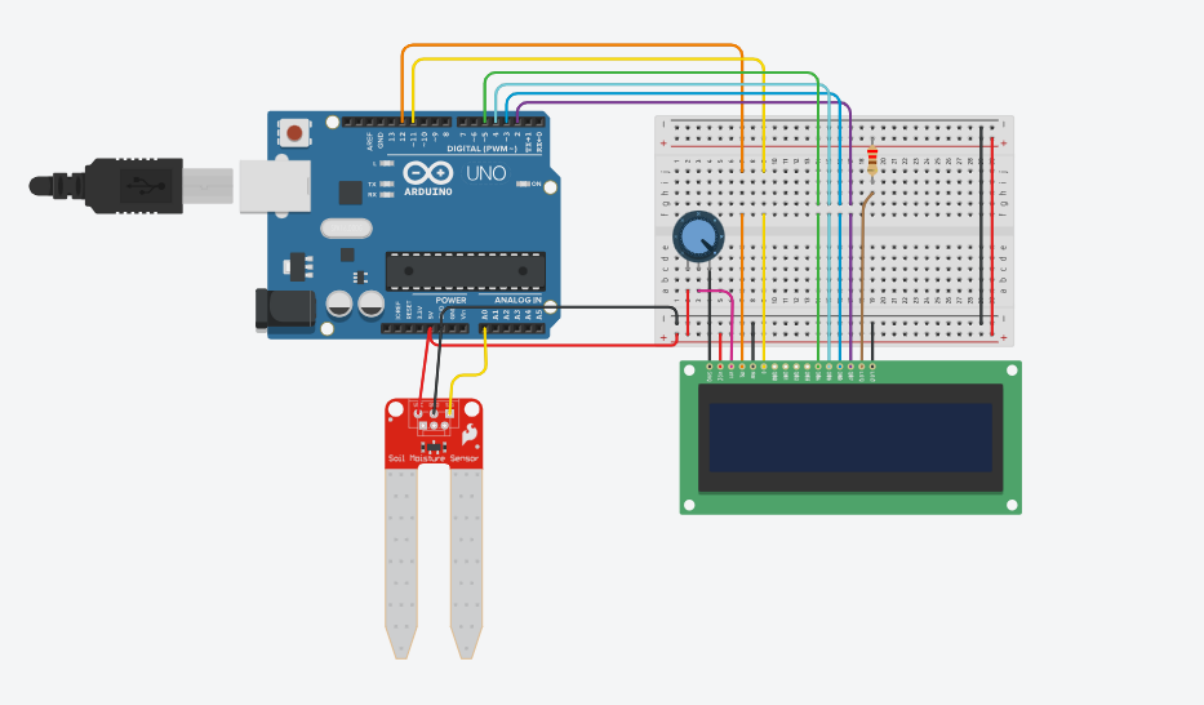
en la primera parte que es setup() definiremos el pin del sensor ya que de este saldrá la información que podremos usar para definir nuestro objetivo principal el cual es determinar si hay presencia o ausencia de humedad, luego pasamos directamente a la parte de loop() donde primero definiremos nuevas variables para que tomen los valores de la lectura análoga del sensor A0 y del pin digital del sensor, ahora haremos que el cursor de la pantalla se posiciona en la posición original para que dependiendo si el valor que es sacado del sensor es mayor o menor que el umbral imprima ausencia o presencia de humedad, luego con setCursor lo posicionamos en la línea de abajo al poner (0,1) para que ahora imprima ***"Introducción a la ingeniera electrónica"*** aunque abreviado para que no quede por fuera el mensaje, se muestra esto por 2 segundos para que el cursor este fijo otro 2 segundos.

```
lcd_1.clear();

lcd_1.print("Valor del sensor");
lcd_1.setCursor(0, 1);
lcd_1.print(analogRead(sensorPin));
delay(3000);

lcd_1.clear();
lcd_1.noBlink();

for (int i = 0; i < 10; i++) {
  digitalValue = digitalRead(sensorValue);
  lcd_1.setCursor(i, 0);
  lcd_1.print(digitalValue == HIGH ? "Presencia" : "Ausencia");
  lcd_1.setCursor(i, 1);
  lcd_1.print(digitalValue == HIGH ? "de humedad  " : "de humec
  delay(100);
}

lcd_1.clear();
lcd_1.setCursor(0, 0);
```

ahora con clear() borramos la pantalla para que imprima el siguiente mensaje en la primera línea siendo "***valor del sensor"*** que es la lectura que va sacar el sensor de humedad, a lo que pasara nuevamente de línea para ver cuanto es el valor de humedad que hay por 3 segundos para luego ser borrado y fijar el cursor nuevamente, para luego entrar en un bucle para mantener durante 10 segundos el mensaje que dependerá si se muestra en HIGH o LOW el valor del sensor, donde HIGH será mayor que 500 y LOW menor que esto así imprimiendo esto en la pantalla hasta reiniciar nuevamente todo desde Introducción a la Ingeniera Electrónica hasta que paremos la simulación.

Link de tinkercad

https://www.tinkercad.com/things/g8XAFq0zhBl



3) Use la pantalla LCD del Laboratorio 2 y realice un Sketch para visualizar la información del sensor de lluvia y del ultrasonido.

Componentes:

-Arduino UNO
-Pantalla lcd 16x2
-Sensor Soil Moisture
-Breadboard
-Potenciómetro
-Resistencia
-Sensor Ultra Sonido HC-SR04

En este reto nos dicen que hagamos un trabajo algo parecido al que hicimos en la actividad pasada, solo que agregando el sensor de ultrasonido a esta, entonces solo tenemos cablear este para cambiar el código de este para que en vez de detectar la presencia de lluvia solo nos muestre los valores que sacan el sensor de ultrasonido y sensor de humedad, debe ser sensor de humedad ya que en tinkercad no existe entonces el Soil Moisture podrá reemplazar la función de un sensor de lluvia, así que debemos hacer que la pantalla LCD muestre sus valores haciendo que cambie cada cierto tiempo, a continuación voy a mostrar el código de este Arduino:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd_1(12, 11, 5, 4, 3, 2);
const int sensorPin = A0;
const int triggerPin = 7;
const int echoPin = 8;
unsigned long ultrasoundDisplayTime = 0;
const unsigned long displayInterval = 7000;

void setup() {
  pinMode(triggerPin, OUTPUT);
  pinMode(echoPin, INPUT);
  lcd_1.begin(16, 2);
  lcd_1.print("Humedad y Sonido");
}
```

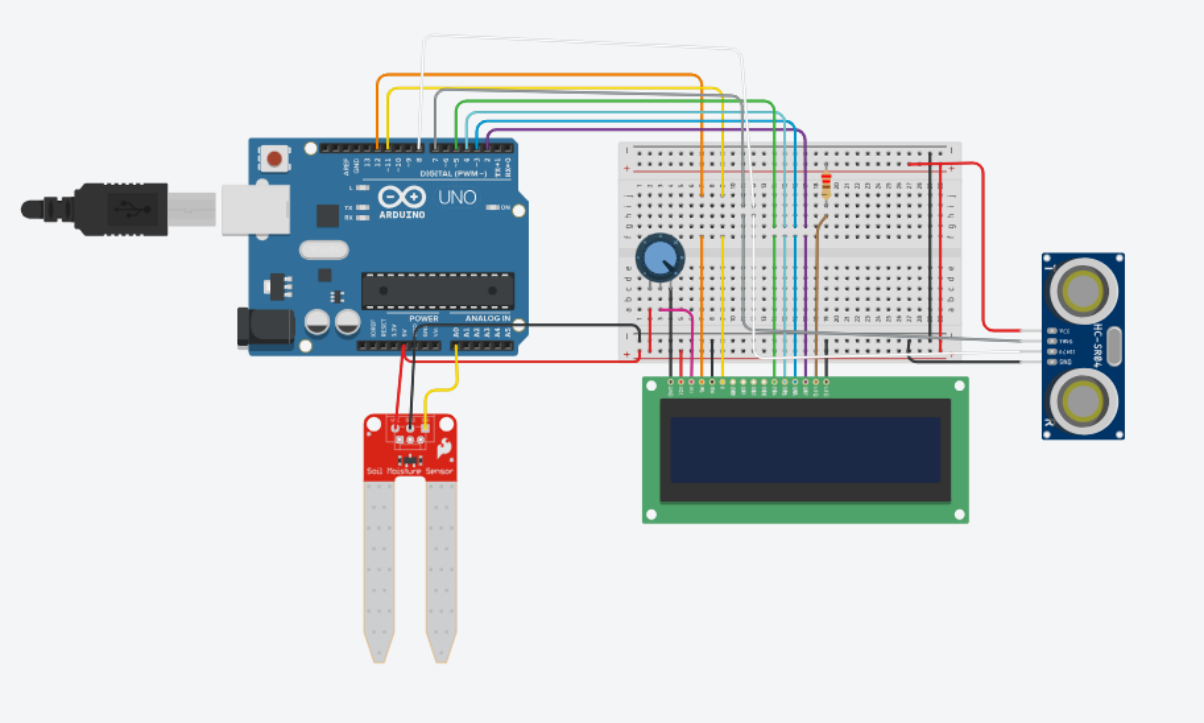
primero inicializamos las variables que estarán determinadas por los pines digitales de echo y trigger del sensor de ultrasonido además de la entrada analógica del sensor Soil Moisture, una vez hecho eso también definimos variables de tiempo de visualización y el intervalo de visualización los cuales son **ultrasounDisplayTime** que se utiliza para realizar un seguimiento del tiempo de la última visualización de datos que será 0 y **displayInterval** que será el tiempo en milisegundos para las actualizaciones de la pantalla LCD que será en 7000 o 7 segundos mejor dicho.

Ahora pasamos a setup() donde definimos los pines del sensor de ultrasonido para variables de salida y entrada, ahora pasamos para iniciar la comunicación con la pantalla LCD para que nos muestre el mensaje ***"Humedad y Sonido"***

```
void loop() {
  unsigned long currentTime = millis();
  if (currentTime - ultrasoundDisplayTime < displayInterval) {
    int sensorValue = analogRead(sensorPin);

    lcd_1.setCursor(0, 1);
    lcd_1.print("Humedad: ");
    lcd_1.print(sensorValue);
  } else {

    long duration;
    int distance;
    digitalWrite(triggerPin, LOW);
    delayMicroseconds(2);
    digitalWrite(triggerPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(triggerPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    distance = duration * 0.034 / 2;

    lcd_1.setCursor(0, 1);
    lcd_1.print("Distancia: ");
    lcd_1.print(distance);
    lcd_1.print(" cm");
  }

  if (currentTime - ultrasoundDisplayTime >= displayInterval * 2) {
    ultrasoundDisplayTime = currentTime;
  }

  delay(1000);
}
```

Ahora en la función loop() donde definimos el tiempo actual en milisegundos con currentTime y luego se verifica con esto si ha pasado menos tiempo del intervalo de visualización con displayInterval desde la ultima actualización de la pantalla, si pasa eso entonces se imprimirá la humedad y su valor en la pantalla lcd, luego pasamos a que si pasa el tiempo entonces nos mostrara luego el valor que capta el sensor de ultrasonido para luego operarlo para que pueda sacar la distancia y ponerla en cm para que se imprima en la pantalla lcd cada 1 segundo/1000 milisegundos hasta que detengamos la simulación.

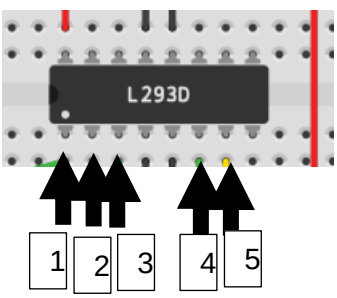
Link de tinkercad:
https://www.tinkercad.com/things/8UAnRhfUWDE



4. Use un módulo L298 y realice un Sketch para controlar la velocidad y el giro de un motor DC,
según los valores enviados a través del puerto serie.

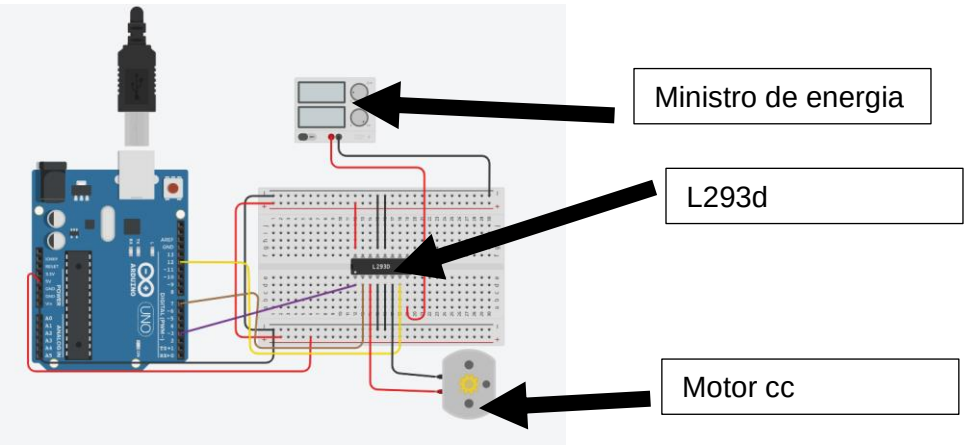
Problemas y consideraciones:

Al momento de realizar el punto, primero tocaba que investigar sobre el uso del modulo 298, que en este caso en tinkercad contábamos con el modulo l293d, el cual nos ayudaría a controlar un motor cc, en este caso el modulo tiene 5 entradas muy importantes.



Podemos ver la enumeración de los pines: el pin 1, es el que controla la velocidad del motor, ósea en este caso que le queremos dar dinamismo al motor ósea no solo tenerlo en alto y bajo, debemos conectarlo a un pin pwm, en el pin2 y el pin 5, son los que le da la dirección de rotación al motor (en el código lo veremos en uso), y el pin 3 y 4, es donde van conectadas las salidas del motor.
Entonces las consideraciones que debíamos tener en cuenta al momento de la potencia del motor, era que el usuario la ingresará, lo plantee de dos formas que el usuario ingresara en el rango 0 a 255, o hacerlos los intervalos del 0 al 10, con la función map, en este caso se utilizó la opción de la función map, a continuación, se verá la estructura y luego el código.
Y para poder encender el motor tenemos que agregarle ese suministro de energía, el cual nos permitirá darle potencia al motor cc.

Estructura



(podemos observar el Arduino, el módulo y el motor)

Código

El código se presentará por partes.

```
int pin2=7; // entrada 2 del l293D
int pin7=12; // Entrada 7 del l293D
int pin1=3; // activamos el motor

int potencia = 0;
int sentido = 0;
int velocidad = 0;
void menu();
void setup() {
  // Iniciamos los pines de salida
  pinMode(pin2,OUTPUT);
  pinMode(pin7,OUTPUT);
  pinMode(pin1,OUTPUT);
  Serial.begin(9600);
  Serial.println("Inicio de simulacion \n");

}
void clearScreen(){
  /* Esta funcion da la sensacion de limpiar el monitor serial. */
  for(int i=0; i<20; ++i){
    Serial.println();
  }
}
```

Al principio tendremos la inicialización de los pines, y de algunas variables que iremos a utilizar a lo largo del desarrollo del código, en este caso se puede observar que los nombres de las variables van muy acorde a lo que harán en el funcionamiento, después tendremos la declaración de una función menú, la cual se explicará más adelante, tenemos el setup, en el cual se configuran los pines de esta forma, ya que de ellos saldrá información, también esta la inicialización del monitor serial, y la impresión de un texto en el monitor serial y por ultimo tenemos la función clearscreen la cual da la ilusión del que en el monitor serial se haya borrado.

```
}
void menu(){
 Serial.println("Ingrese la potencia a usar en el motor, va desde un rango del 1 al 10");
    Serial.print("potencia: ");Serial.read();
    while(Serial.available() == 0);
    potencia = Serial.parseInt();
    clearScreen();
    if (potencia>=0 && potencia<=10){
    velocidad = map(potencia,0,10,0,255);
    Serial.println("Ingrese el sentido: ingrese 0 si negativo y 1 si es poitivo");
    Serial.print("Sentido: ");Serial.read();
    while(Serial.available() == 0);
    sentido = Serial.parseInt();
    clearScreen();

      }
    else{
      Serial.print("Velocidad invalida:");
      Serial.println(potencia);
    }
```

Tenemos el menú, en el cual se pedirá la potencia con la cual el usuario quiere que tenga el motor, y podemos ver un if el cual, como les hable anteriormente, verificara el intervalo de potencia, y este se hallará gracias a la función mapa, la cual divide un intervalo grande, en intervalos más pequeños, después de tener la potencia, se le pedirá el usuario el sentido de rotación, y por ultimo ese “else”, se tiene por si el usuario comete errores de digitación.

```
void loop() {
 menu();
  if(sentido == 0){ // activamos el pin de activación para el motor
    analogWrite(pin1,velocidad);// cambio de direccion
    digitalWrite(pin2,LOW);
    digitalWrite(pin7,HIGH);

    delay(2000);

    digitalWrite(pin1,LOW); // paro de motor
    delay(1000);
  }
  else if(sentido == 1){

    analogWrite(pin1,velocidad);// cambio de direccion
    digitalWrite(pin2,HIGH); // Marcha
    digitalWrite(pin7,LOW);
    delay(2000);
    digitalWrite(pin1,LOW); // paro de motor
    delay(1000);
  }
  else if(sentido > 1){
    Serial.println("El valor ingresado es incorrecto");
  }
```

Y por último tenemos el void loop, en el cual se ejecuta la función menú vista anteriormente, y la lógica va que si el sentido es igual a 0 (o sea girara hacia la izquierda), escriba en valor analógica sobre el pin explicado anteriormente, el cual dirige la potencia del motor, y luego dos digitalwrite que esa combinación de alto y bajo es lo que le da el sentido de rotación al motor, y luego tenemos si es igual a 1(o sea girará a la derecha) y sigue la misma lógica, y por ultimo tenemos, por si el usuario comete un error de digitación

Para ver la simulación:
https://www.tinkercad.com/things/fshDJcyQ13Y

5. Investigue que es una electroválvula, qué tipos existen y cómo se manejan con un Arduino.
Usando los valores enviados a través del puerto serie, controle una electroválvula con un Arduino.
Una electroválvula es un dispositivo que usa una bobina electromagnética para controlar el flujo de un fluido, como agua, aire o gas. Hay diferentes tipos de electroválvulas según su función, como normalmente abiertas, normalmente cerradas, de dos vías, de tres vías, etc.

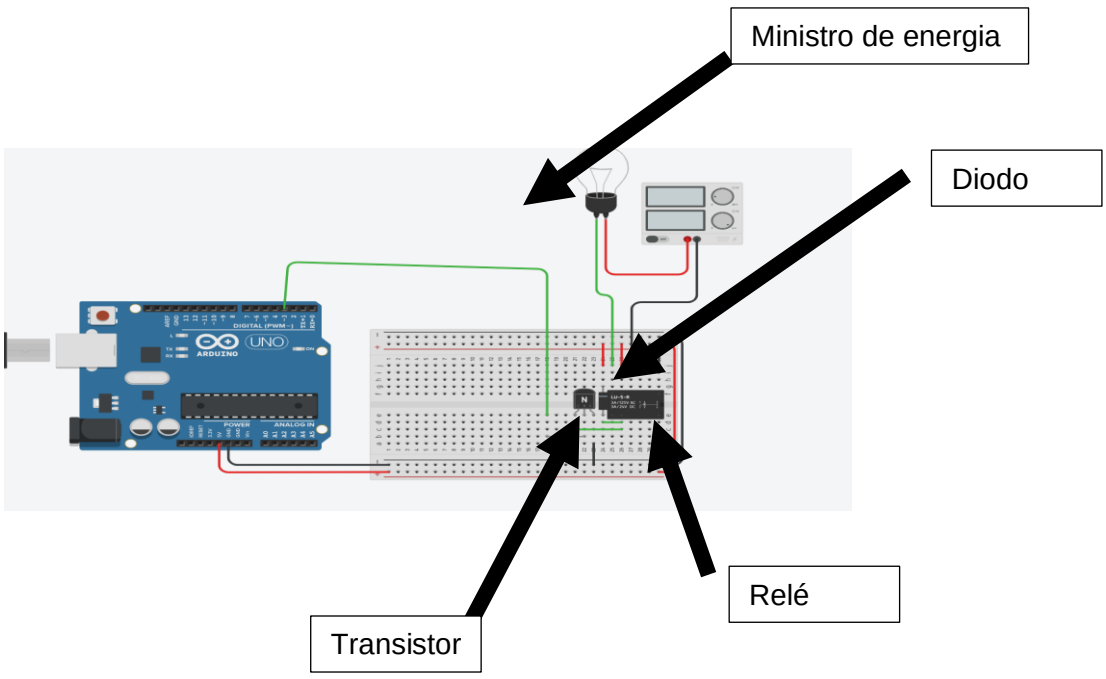
Para manejar una electroválvula con un Arduino, necesitas los siguientes componentes:

- Un Arduino Uno o compatible
- Una electroválvula de 12 V
- Un transistor TIP120
- Un diodo 1N4001
- Una resistencia de 220 ohmios
- Un cable USB
- Una fuente de alimentación de 12 V
- Un protoboard y cables

Ejemplo de solución:

En este caso no se puede simular la electroválvula, pero si se puede dar una demostración parecida, al funcionamiento de esta con el uso de un bombillo y un relé.

Estructura:



Problemas y consideraciones:
Hay que tener en cuenta para que estos objetos funcionen tienen que exportarse la librería servo.h, y además el mayor problema al momento de simular esto fue el hecho de que no existiera una electroválvula en el tinkercad para poder simularla.

Código:

```
#include <Servo.h>
int electroValvula = 3;
int tiempo=5000;
int potencia = 0;


void setup(){
  pinMode(electroValvula, OUTPUT);
  Serial.begin(9600);
    Serial.println("Inicio de simulacion \n");
}

void loop(){
 Serial.println("Ingrese la potencia a usar en la electrovalvula, va desde un rango del 0 al 255");
    Serial.print("potencia: ");Serial.read();
    while(Serial.available() == 0);
    potencia = Serial.parseInt();
    analogWrite(electroValvula,potencia);
    delay(tiempo);
    analogWrite(electroValvula,0);
}
```

Al principio podemos ver la utilización de la librería “servo.h”, después tendremos la declaración de las variables a utilizar, “electroValvula” es la utilización del pin digital pwm a la electroválvula, después el tiempo que durará la electroválvula accionada, y luego la potencia que esta utilizará.
En el “setup”, podemos ver la configuración del pin digital, y la inicalizacion del serial con la función “Serial.begin”.
En el loop, imprimimos lo que le queremos dar a conocer al usuario, recibimos lo ingresado por el usuario, y luego le damos ese valor de potenica, por medio de un analogWrite a la válvula.

Para ver la simulación:
https://www.tinkercad.com/things/biDEkfvWVOO

Realice la implementación mediante software y/o hardware de la siguiente aplicación: Detecte la presencia de poca humedad de suelo con el sensor. En caso de humedad baja, gire el motor DC hacia la izquierda simulando abrir la compuerta de un recipiente precargado, en donde se dejará pasar agua. Con el sensor de ultrasonido detecte el nivel de agua en el recipiente que servirá para el riego del terreno. En la pantalla LCD muestre la información de ambos sensores (humedad del suelo y nivel de agua). Una vez se detecte el nivel mínimo de agua, gire el motor DC hacia la derecha simulando cerrar

la compuerta del recipiente. A continuación, active un LED indicando que el tanque está vacío.

Problemas y consideraciones:
Al momento de desarrollar el ejercicio, me pude encontrar con tres problemas, los cuales son el funcionamiento del sensor de ultrasonido, la lógica a utilizar, y el hecho de que uno mismo debe manipular los datos, debido a que es a una simulación virtual y para que se viera un buen funcionamiento se tuvo que extrapolar unos rangos, estos son a la humedad y el rango de cm del tnaque.
Entonces la mayor consideración a tener en cuenta cuando lo intente simular es que debes de manjar los datos usted mismo, del sensor humedad y del sensor ultrasónico.

Código:

```
#include <LiquidCrystal.h>

const int rs = 2;
const int en = 3;
const int d4 = 4;
const int d5 = 5;
const int d6 = 6;
const int d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

long duracion, cm;
const int pinSensor = A5;
const int valorSensorMaximo = 850;
const int intervaloActualizacion = 25;
const int sensor = 9;
int pin1=12; // entrada 2 del 1293D
int pin2=13; // Entrada 7 del 1293D
int led = 10;
```

en esta parte del código lo único que vemos es al principio la declaración de variables para el uso de la pantalla lcd, y luego otras inicializaciones las cuales son utilizadas mas adelantes incluyendo la de los números que hacen referencia a algunos pines del Arduino.

```
void setup() {
  lcd.begin(16, 2);
  lcd.print("Humedad: 0%");
  lcd.setCursor(0, 1);


  pinMode(pinSensor, INPUT);
  pinMode(pin2,OUTPUT);
  pinMode(pin1,OUTPUT);
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}
```

En el “setup” lo único que hacemos es inicializar el lcd, y la configuración de pines, el “pinSensor” esta hecho input debido a que este recibe la señal(obtenida por el sensor de ultrasonido) y ya los otros pines son los del motor y el led a utilizar para marcar, si el tanque esta vacío.

```
int convertirCM()
{
  int cm =0;
  cm = 0.01723 * ObtenerDatos(sensor);
  Serial.print(cm);
  Serial.println(" cm");
  delay(100); // Wait for 100 millisecond(s)
  return cm;
}

long ObtenerDatos(int pin)
{
  pinMode(pin, OUTPUT);  // Clear the trigger
  digitalWrite(pin, LOW);
  delayMicroseconds(2);
  // Sets the pin on HIGH state for 10 micro seconds
  digitalWrite(pin, HIGH);
  delayMicroseconds(10);
  digitalWrite(pin, LOW);
  pinMode(pin, INPUT);
  // Reads the pin, and returns the sound wave travel time in microseconds
  return pulseIn(pin, HIGH);
}
```

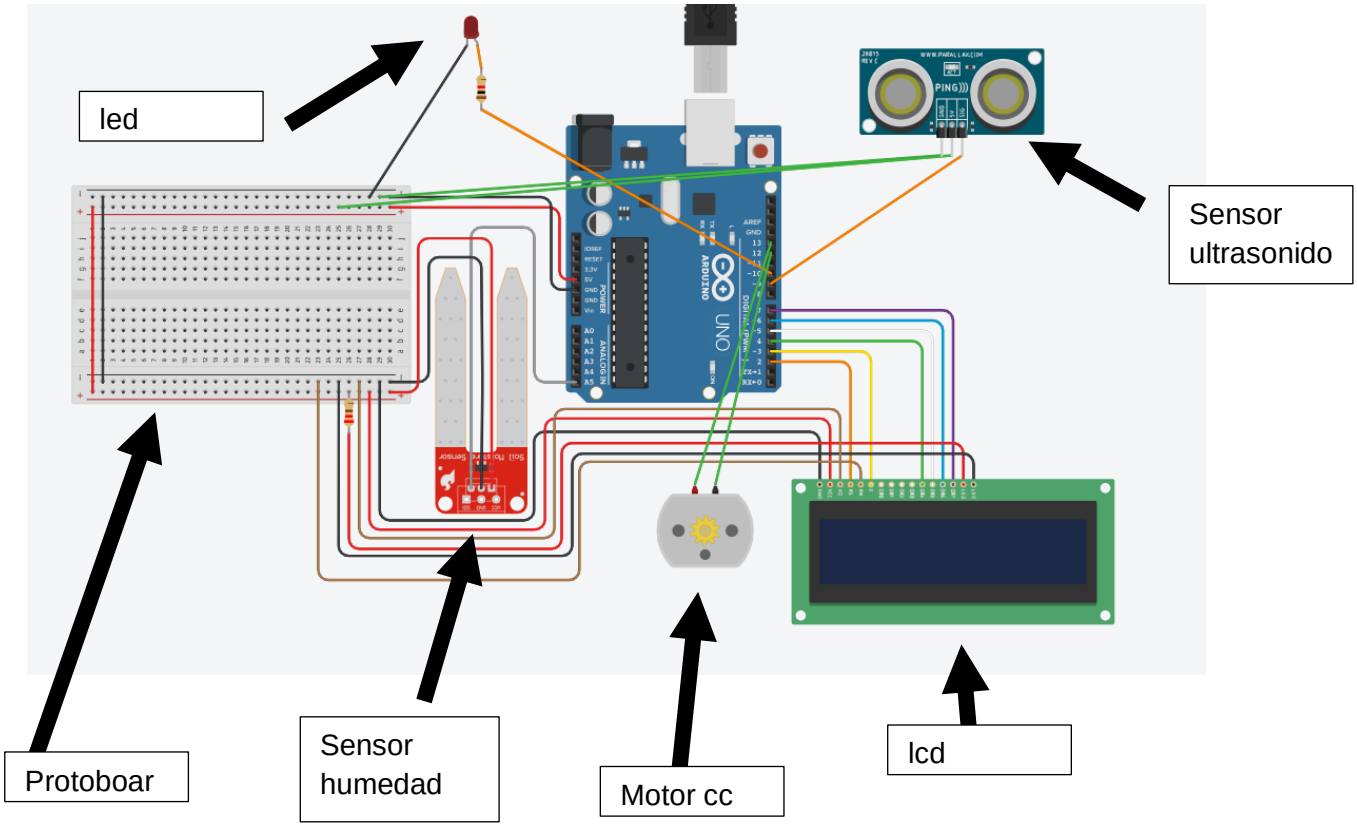
Estas dos funciones son las que nos sirven para la utilización del sensor de ultrasonido, la primera que es “convertiCM” como su mismo nombre lo dice convierte el tiempo de llegada de la señal, en cm.

La función “obtenerDatos”, lo que hace es enviar un pulso, el cual es la señal y esta es recibida por el pin anteriormente declarado como input, y el método “pulseIN”, nos sirve para detectar dicha señal

```
void loop() {
   int distanciacm = 0;
   bool activacion_motor;
  static int porcentaje = 0;
  int humedad = analogRead(pinSensor);
  int nuevoPorcentaje = map(humedad, 0, valorSensorMaximo, 0, 100);
  lcd.setCursor(9, 0);
  lcd.print(nuevoPorcentaje);
  lcd.print("% ");
  if( nuevoPorcentaje < 10 && nuevoPorcentaje > 1){
    digitalWrite(pin1,HIGH);
    digitalWrite(pin2,LOW);
    delay(2000);
    digitalWrite(pin1,LOW);
    bool controlador = true;
    while(controlador == true){
      distanciacm = convertirCM();
    lcd.setCursor(0, 1);
    lcd.print("altura= ");
    lcd.print(distanciacm);


      if(distanciacm < 60){
        controlador = false;
        activacion_motor = true;
        digitalWrite(led, HIGH);
        delay(2000);
      }

    }
    if(activacion_motor == true){
    digitalWrite(pin1,LOW);
    digitalWrite(pin2,HIGH);
    delay(2000);
    digitalWrite(pin1,LOW);
    digitalWrite(pin2,LOW);
    activacion_motor = false;
    }
    digitalWrite(pin1,LOW);
     digitalWrite(led, LOW);
       delay(2000);

    //se tiene que cambiar la humedad en el dicho sensor al igual que el snesor ultrasonico
  }
}
```

Y aquí tenemos el “loop”, en el cual ocurre toda la lógica del programa, primero declararemos las variables a utilizar, luego leeremos el valor que nos da el sensor de humedad y lo guardamos en una variable, después imprimimos en el lcd este valor, luego ingresaremos a un “IF”, SI este valor dado por el sensor humedad, se encuentra entre el intervalo de 0 a 10, si es así, gira el motor a la izquierda(simulando que abre la compuerta) y luego entra en un bucle, el cual mirará el nivel del agua con ayuda del sensor de ultrasonido, y cuando este nivel sea menor a 60 se prende el led, se sale del ciclo de mirar el nivel del tanque y pasa a activar un motor que va hacia derecha(simulando que cierra la compuerta).

Estructura



Para ver la simulación:
***Recuerda que se debe manejar manualmente los sensores**
https://www.tinkercad.com/things/3t08z5rhyss