TEC | Tecnológico de Costa Rica

Escuela de Física

Ingeniería Física

Instructivo de Laboratorio

Instrumentación I

11 de noviembre de 2025

Versión: 0.1

Realizado por: Juan J. Rojas

Índice general

1. Sensores de estímulos eléctricos **3**
- 1.1. Objetivos . . . 3
- 1.2. Materiales y equipo . . . 3
- 1.3. Procedimiento . . . 3
- 1.4. Preguntas para el analisis de resultados . . . 6

2. Sensores de temperatura **7**
- 2.1. Objetivos . . . 7
- 2.2. Materiales y equipo . . . 7
- 2.3. Procedimiento . . . 7
- 2.4. Análisis . . . 10

3. Sensores de presión **11**
- 3.1. Objetivos . . . 11
- 3.2. Materiales y equipo . . . 11
- 3.3. Actividad 1 . . . 11
  - 3.3.1. Procedimiento . . . 11
  - 3.3.2. Análisis . . . 14
- 3.4. Actividad 2 . . . 14
  - 3.4.1. Procedimiento . . . 14
  - 3.4.2. Análisis . . . 14

4. Sensores de peso **15**
- 4.1. Objetivos . . . 15
- 4.2. Materiales y equipo . . . 15
- 4.3. Actividad 1 . . . 15
  - 4.3.1. Procedimiento . . . 15
  - 4.3.2. Análisis . . . 17

5. Mediciones de nivel **18**
- 5.1. Objetivos . . . 18
- 5.2. Materiales y equipo . . . 18
- 5.3. Actividad 1 . . . 18
  - 5.3.1. Procedimiento . . . 18
  - 5.3.2. Análisis . . . 20
- 5.4. Actividad 2 . . . 20
  - 5.4.1. Procedimiento . . . 20

5.4.2. Análisis 20

Laboratorio 1

Sensores de estímulos eléctricos

1.1. Objetivos

Al finalizar este laboratorio el estudiante estará en capacidad de:

- Configurar un sensor digital para lograr una medición eléctrica con la mayor resolución posible.
- Calcular el error asociado a la medición utilizando una multimetro digital de alta precisión.

1.2. Materiales y equipo

A suministrar por la Escuela:

- 1 ARDUINO UNO R4 MINIMA
- 1 sensor digital de voltaje, corriente y potencia (INA219)
- 1 mini breadboard
- Cables de interconexión (jumpers) macho-macho

1.3. Procedimiento

a. Realice las conexiones del circuito tal y como se indica en la Figura 1.1.

b. Abra el programa Arduino IDE

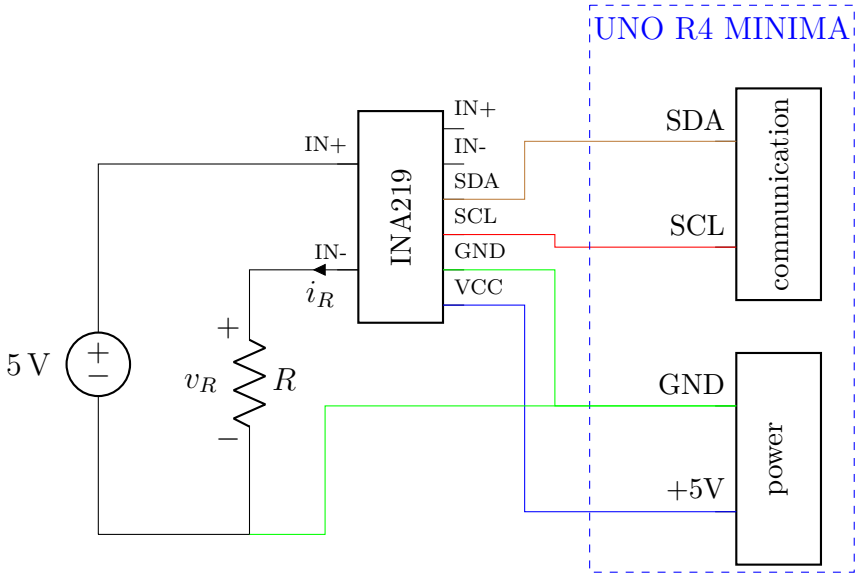

Figura 1.1: Conexión de sensor de corriente y voltaje INA219

c. Incluya el siguiente código en el Arduino IDE

```
#include <Wire.h>
#include <INA219.h>
INA219 ina219 (0x40);

//Instrucciones: Una vez subido, correr el monitor serial y presionar Enter para
    iniciar la medición, en caso de que no inicie automáticamente.
// Usar libreria INA219 (programada por Rob Tillart), de la lista de librerías.

void setup() {
  Serial.begin(115200);
  Wire.begin();
  delay(2000);
  if (!ina219.begin()) {
    Serial.println("INA219 not found");
    while (1){ delay(10); }}
  Serial.println("INA219 initialized");
  //Here we can set max current shunt value to calibrate the Calibration
   Register (8.5.1)
  ina219.setMaxCurrentShunt(1, 0.1);    //  adjust if needed
  Serial.print("Shunt value:\t");
  Serial.println(ina219.getShunt(), 4);
  Serial.print("Max current:\t");
  Serial.println(ina219.getMaxCurrent(), 4);
  //Set gain based on desired range
  ina219.setGain(1);
  Serial.print("Gain:\t");
  Serial.println(ina219.getGain());
  //Set bus voltage range
  Serial.print("Bus voltage range:\t");
  ina219.setBusVoltageRange(16);
  Serial.println(ina219.getBusVoltageRange());

  Serial.print("Calibrated?:\t");
```

```
  Serial.println(ina219.isCalibrated());
  Serial.print("CLSB:\t");
  Serial.println(ina219.getCurrentLSB());
  Serial.println("press any key");
  while (!Serial.available()) {
  }
}

float curre = 0;
float volta = 0;
float power = 0;
float shunt = 0;
int counter = 0;
// Here we set the oversampling
int samples = 100;
int delayTime = 1000/samples;

void loop() {
  counter++;
  curre += ina219.getCurrent_mA();
  volta += ina219.getBusVoltage();
  power += ina219.getPower_mW();
  shunt += ina219.getShuntVoltage_mV();
  if (counter >= samples) {
    Serial.print(curre/samples);
    Serial.print(",");
    Serial.print(volta/samples);
    Serial.print(",");
    Serial.print(power/samples);
    Serial.print(",");
    Serial.println(shunt/samples);
    counter = 0;
    curre = 0;
    volta = 0;
    power = 0;
    shunt = 0;
  }
  delay(delayTime);
}
```

d. Tomando en cuenta el valor de la resistencia que se le asignó, calcule el valor de la corriete esperada en el circuito utilizando la Ley de Ohm.

e. Investigue en el datasheet que significa el parametro PGA y como se relaciona con el rango del sensor. Ver 8.5.1

f. Configure el sensor para que el rango de medición sea el más adecuado para la resistencia que se le asignó, suba el código modificado al Arduino.

g. Realice dos mediciones de 10s de duración con una frecuencia de 1Hz al mismo tiempo que mide la corriente con un multímetro digital de alta precisión.

h. Calibre el sensor utilizando un multímetro digital de alta precisión y cambie nuevamente el código para que el valor medido por el sensor sea lo más cercano posible al valor medido por el multímetro. Ver 8.5.2.1. Suba el código modificado al Arduino.

i. Realice dos mediciones de 10s de duración con una frecuencia de 1Hz al mismo tiempo que mide la corriente con un multímetro digital de alta precisión.

j. Modifique el código para que tener oversampling de 100 muestras por segundo y suba el código modificado al Arduino.

k. Realice dos mediciones de 10s de duración con una frecuencia de 100Hz al mismo tiempo que mide la corriente con un multímetro digital de alta precisión.

l. Para cada ser de medicines calcule lo siguiente e incluya en una tabla:

- El valor promedio, $\overline{i_R}$
- La desviación estándar, σ
- El valor de la incertidumbre estándar, $\sigma_x = \sigma/\sqrt{n}$
- La exactitud
- La precisión
- La repetibilidad

m. Tome en cuenta lo siguiente:

- La exactitud se calculará como el mayor error relativo obtenido en cualquier punto de los datos en todas las corridas de medición.
- La precisión se calculará como la mayor desviación estándar entre todas las corridas de medición.
- La repetibilidad se calculará de la siguiente manera:

$$\text{repetibilidad} = \sqrt{\frac{\sum_{i=1}^{n}(\text{error absoluto})^2}{n}}$$

1.4. Preguntas para el analisis de resultados

- ¿Cómo afecta el valor de la resistencia al rango de medición del sensor?
- ¿Cómo afecta el valor de PGA al rango de medición del sensor?
- ¿En que consiste el oversampling y cómo afecta la resolución de la medición?
- ¿Que es lo que más afecta la precisión de la medición?
- ¿Qué es lo que más afecta la exactitud de la medición?

Laboratorio 2

Sensores de temperatura

2.1. Objetivos

Al finalizar este laboratorio el estudiante estará en capacidad de:

- Utilizar un microcontrolador para realizar mediciones de temperatura.
- Realizar un algoritmo básico para obtener mediciones e imprimirlas en la terminal.

2.2. Materiales y equipo

A suministrar por la Escuela:

- 1 Arduino UNO R4 MINIMA
- 1 termistor de 10 kΩ NTC marca TDK
- 1 breadboard
- Cables de conexión macho-macho

2.3. Procedimiento

a. Arme el circuito tal como se muestra en la Figura 2.1.

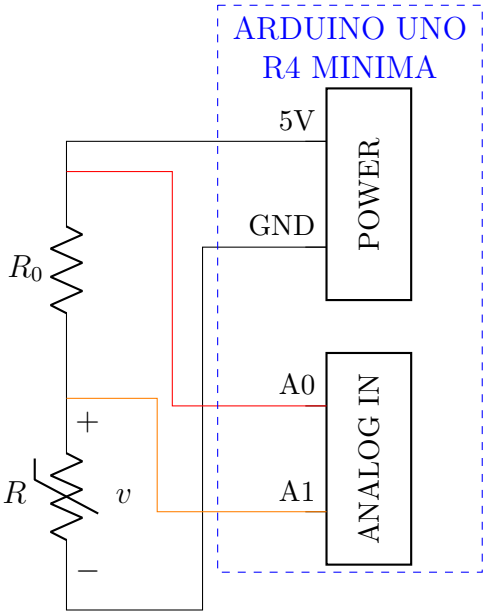

Figura 2.1: Conexión de circuito para el Laboratorio 3

b. Usando la formula del divisor de voltaje y simplificando se llega a que:

$$\frac{R_0}{R} = \frac{5}{v} - 1$$

c. El valor de la temperatura se puede aproximar usando la siguiente ecuación:

$$\frac{1}{T} = \frac{1}{T_R} + \frac{1}{B_{25/100}} \ln\left(\frac{R}{R_R}\right)$$

donde T es la temperatura del termistor en K, T_R es la temperatura de referencia en K y R_R es la resistencia de referencia en Ω

d. Según la hoja de datos del termistor el valor de $B_{25/100}$ es igual a 4300.

e. El siguiente sketch de Arduino IDE implementa las ecuaciones anteriores en dos funciones, una que calcula la resistencia (R) y otra que calcula la temperatura usando el valor de $B_{25/100}$ (T_1).

```
float V = 0;
float Vin = 0;
float vR = 0;
float R = 0;
float T1 = 0;
float T2 = 0;
int counter = 0;
int samples = 100;
int delayTime = 1000/samples;

#define vR0 4587.0
#define vBETA 4300
#define vTR 298.15
#define vRR 10000
```

```
#define vZero 273.15

float calc_R (float V, float Vin, float R0) {
  float R = (Vin / V) - 1;
  R = R0 / R;
  return R;
}

float calc_T1 (float R, int RR, float TR, float Zero, int BETA) {
  float T1 = BETA + TR * log(R/RR);
  T1 = (BETA * TR) / T1;
  T1 = T1 - Zero;
  return T1;
}

void setup() {
  // initialize serial communication at 9600 bits per second
  Serial.begin(9600) ;

}

void loop() {
  counter++;
  V = (analogRead(A1) / 1023.0) * 5;
  Vin = (analogRead(A0) / 1023.0) * 5;
  vR = calc_R(V, Vin, vR0);
  R = R + vR;
  T1 += calc_T1(vR, vRR, vTR, vZero, vBETA);
  if (counter >= samples) {
    Serial.print(R/samples);
    Serial.print(",");
    Serial.print(T1/samples);
    Serial.print(",");
    Serial.println(T2/samples);
    counter = 0;
    R = 0;
    T1 = 0;
    T2 = 0;
  }
  delay(delayTime);
}
```

f. Suponga que su sensor será utilizado para medir la temperatura de la piel (20 °C a 40 °C), para reducir el error en la medición utilice lo mostrado en la sección 2.3 de este documento para determinar $C0$, $C1$ y $C3$.

g. Realice una función en el mismo sketch en Arduino IDE que permita calcular el valor de T_2 a partir de los valores de C_0, C_1, C_3 y R usando la ecuación de Steinhart-Hart

h. Imprima en la terminal los valores de R, T_1 y T_2 y léalos usando un script de Python. Realice tres mediciones de 20 segundos cada una en las que toque el sensor para provocar cambios en la temperatura, deje enfiar entre cada medición.

i. Usando el código del sensor (B57164K0103) genere una tabla de valores de resistencia usando este software, use $\Delta R/R$ de 5 % y 1 °C de resolución, exporte los datos.

j. Utilizando Python y los datos obtenidos en las tres mediciones calcule T_3 para cada valor de R usando interpolación lineal entre los puntos obtenidos por medio del software

k. Para cada una de las mediciones realice una gráfica que muestre T_1, T_2 y T_3

2.4. Análisis

- ¿Cual le parece que es el mejor método de aproximación? ¿Porqué?
- ¿Que tan importante es la precisión de la resistencia R_0?
- ¿Que tan importante es la precisión de la referencia del ADC?

Laboratorio 3

Sensores de presión

3.1. Objetivos

Al finalizar este laboratorio el estudiante estará en capacidad de:

- Utilizar un sensor para medir presión absoluta.

3.2. Materiales y equipo

- 1 Arduino UNO R4 MINIMA
- 1 sensor digital de presión BMP280
- 1 mini breadboard de 170 pines
- Jumpers macho-macho

3.3. Actividad 1

3.3.1. Procedimiento

a. Conecte el sensor BMP280 tal como se muestra en la Figura 3.1

b. Copie el siguiente código en el IDE de Arduino (el codigo se puede encontrar en: GitHub)

```
#include "Adafruit_BMP280.h"

Adafruit_BMP280 bmp;
float temperature = 0;
float pressure = 0;

void setup() {
  Serial.begin(115200);
  delay(2000);
  if (!bmp.begin(0x77)) {
    Serial.println("Sensor not found");
    while (1){}
  }
```

```
}

void loop() {
  // Read the values
  temperature = bmp.readTemperature();
  pressure = bmp.readPressure()/100; //hectopascales
  // Print to the Serial Monitor
  Serial.print(temperature);
  Serial.print(",");
  Serial.println(pressure);
  delay(1000);
}
```

c. Analice cada linea de código y cuando entienda su funcionamiento corra el *sketch* usando el botón *Upload*

d. Haga contacto con el encapsulado del sensor de forma que su dedos calienten el sensor, observe como cambia el valor en la terminal.

e. Modifique el programa de forma que lo que se imprima en la terminal sea algo como esto:

24.8,806.1
24.7,806.3

f. Instale las librerías *pyserial*, *matplotlib*, *drawnow* y *datetime* en Python para ser utilizadas luego.

g. Copie el siguiente código en su IDE de Python. (el codigo se puede encontrar en: GitHub)

```
import serial
import csv
from datetime import datetime
import matplotlib.pyplot as plt
from drawnow import drawnow

arduino = serial.Serial('COM14')
nombre = "bmp280"
narchivo = 1
mediciones = 10

def data_fig():
    plt.plot(sec,pres)


datos = []
sec = []
pres = []

arduino.write(b'0')

for i in range(mediciones):
    dato = arduino.readline()[:-2].decode('utf-8').split(',')
    print(dato)
    datos.append([datetime.now().strftime(format="%Y-%m-%d %H:%M:%S"),
```

```
                    float(dato[0]),
                    float(dato[1])])
    sec.append(i)
    pres.append(float(dato[1]))
    drawnow(data_fig)

seguir = input("Cualquier tecla")

# Guardar en CSV
with open(f"{nombre}_{narchivo}.csv", "w", newline="") as f:
    writer = csv.writer(f)
    writer.writerow(["timestamp", "Temperatura (°C)", "Presión (hPa)"])  #
    encabezados
    writer.writerows(datos)

print("Datos guardados en mediciones.csv")
```

h. Modifique el puerto COM para que coincida con el puerto en el que esta conectado el Arduino UNO R4 MINIMA

i. Modifique el código para que cada toma de datos se almacene en un archivo *.csv* y para que se genere una gráfica en tiempo real.

j. Realice tres tomas de datos de 60 segundos cada una

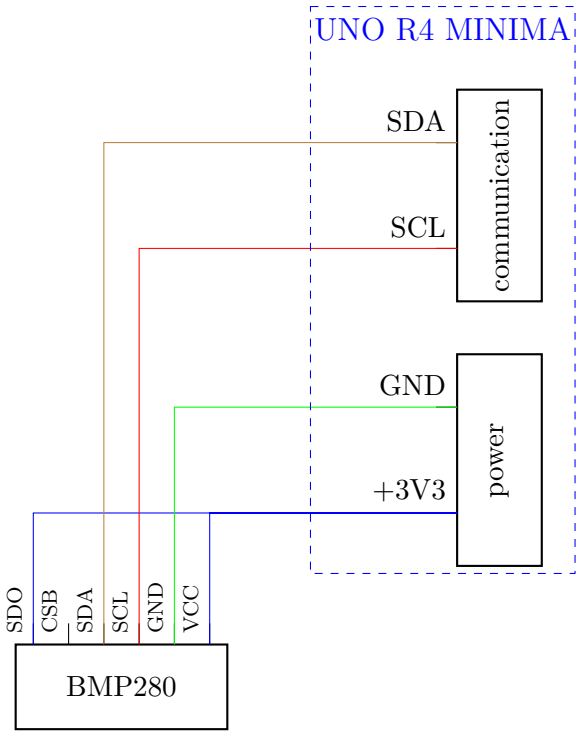

Figura 3.1: Conexión de sensor de presión absoluta BMP280

3.3.2. Análisis

a. Determine la presión atmosférica local usando el promedio de las mediciones que las tres tomas de datos, incluya la precisión de la medición

b. Determine la altitud local usando el valor de presión calculado ¿Es este valor cercano al esperado para la altitud del campus (1407 msnm)?

3.4. Actividad 2

3.4.1. Procedimiento

a. Usando la misma configuración anterior, realice una nueva toma de datos de 60s en la que lleve el sensor lo mas cerca del piso posible y luego lo mas alto posible lentamente y repita durante los 60 segundos de la toma

3.4.2. Análisis

a. ¿Se puede observar el cambio de altura en las mediciones?

b. Según la hoja de datos del sensor ¿Debería detectar estos cambios?

c. ¿El tiempo entre mediciones es realmente un segundo?

Laboratorio 4

Sensores de peso

4.1. Objetivos

Al finalizar este laboratorio el estudiante estará en capacidad de:

- Implementar un sistema de medición de peso utilizando el módulo HX711 con calibración multipunto

4.2. Materiales y equipo

- 1 Arduino UNO R4 MINIMA
- Celda de carga de 1 kg
- 1 amplicador HX711
- Pesas de referencia (500g, 1000g, 1500g)
- Cable USB y computadora con Arduino IDE

4.3. Actividad 1

4.3.1. Procedimiento

a. Conecte el amplificador HX711 tal como se muestra en la Figura

b. Copie el siguiente código en el IDE de Arduino (el codigo se puede encontrar en: GitHub)

```
#include "HX711.h"

HX711 scale;

#define DATAPIN 6
#define CLOCKPIN 7

void setup()
{
  Serial.begin(115200);
```

```
  delay(1000);

  scale.begin(DATAPIN, CLOCKPIN);
}


String input = "a";
String maxt = "500";
int maxn = 500;
float value = 0.0;
float units = 0.0;


void loop()
{
  value = scale.get_value(2);
  units = scale.get_units(10);
  Serial.print(value);
  Serial.print(",");
  Serial.println(units);
  delay(250);
  while(Serial.available()) input = Serial.readStringUntil('\n');
  if (input == "c") {
    Serial.println("***Calibration***");
    Serial.println("\nEmpty the scale, press a key to continue");
    while(!Serial.available());
    while(Serial.available()) Serial.read();
    scale.tare();
    Serial.print("Weight: ");
    Serial.println(scale.get_units(10));
    Serial.println("\nSet max weight");
    while(!Serial.available());
    while(Serial.available()) maxt = Serial.readStringUntil('\n');
    maxn = maxt.toInt();
    Serial.print("\nPut ");
    Serial.print(maxn);
    Serial.println(" gram in the scale, press a key to continue");
    while(!Serial.available());
    while(Serial.available()) Serial.read();
    scale.calibrate_scale(maxn, 10);
    Serial.print("Weight: ");
    Serial.println(scale.get_units(10));
    input = "n";
  }
}
```

c. Analice cada linea de código y cuando entienda su funcionamiento corra el *sketch* usando el botón *Upload*

d. En la terminal serial ingrese la letra ç", esto iniciará un proceso de calibración, siga las instrucciones del menú, calibre con peso máximo de 200 gramos.

e. Copie el siguiente código en su IDE de Python. (el codigo se puede encontrar en: GitHub)

```
import serial
import csv
```

```
from datetime import datetime

arduino = serial.Serial('COM9')
nombre = "scale900_800"
narchivo = 3
mediciones = 20

datos = []

arduino.write(b'0')

for i in range(mediciones):
    dato = arduino.readline()[:-2].decode('utf-8').split(',')
    print(dato)
    datos.append([datetime.now().strftime(format="%Y-%m-%d %H:%M:%S"),
                  float(dato[0]),
                  float(dato[1])
                  ])


# Guardar en CSV
with open(f"{nombre}_{narchivo}.csv", "w", newline="") as f:
    writer = csv.writer(f)
    writer.writerow(["timestamp", "raw", "peso"])  # encabezados
    writer.writerows(datos)

print("Datos guardados en mediciones.csv")
```

f. Modifique el puerto COM para que coincida con el puerto en el que esta conectado el Arduino UNO R4 MINIMA.

g. Coloque un peso de 100 gramos y realice una toma de datos de 20 muestras.

h. Coloque un peso de 300 gramos y realice una toma de datos de 20 muestras.

i. Repita este procedimiento para calibraciones con peso máximo de 500 gramos y 900 gramos.

4.3.2. Análisis

a. ¿Cuál es la resolución teórica del módulo HX711 si trabaja con un convertidor de 24 bits y una celda de carga de 1 kg? ¿Qué implicaciones tiene esto en la medición de masas pequeñas? Como recomendación, utilizar la siguiente ecuación:

$$\text{Resolución} = \frac{\text{Capacidad de la celda}}{2^{\text{Número de bits}}}$$

b. ¿Observó alguna diferencia entre las mediciones realizadas con diferentes puntos de calibración?¿Cuál fue mejor?

Laboratorio 5

Mediciones de nivel

5.1. Objetivos

Al finalizar este laboratorio el estudiante estará en capacidad de:

- Utilizar un sensor para medir nivel.
- Determinar la curva de calibración para medición de volumen.

5.2. Materiales y equipo

- 1 Arduino UNO R4 MINIMA
- 1 sensor ultrasónico HC-SR04
- 1 mini breadboard de 55 pines
- 1 beaker de 250 ml
- Jumpers macho-macho y hembra-hembra

5.3. Actividad 1

5.3.1. Procedimiento

a. Conecte el sensor HC-SR04 tal como se muestra en la Figura 5.1

b. Copie el siguiente código en el IDE de Arduino (el codigo se puede encontrar en: GitHub)

```
/*
 * Nivel
*/

const int trigPin = 9;
const int echoPin = 10;

float duration, distance, volume;
```

```
void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = (duration*.0343)/2;
  // distance = 7.96 - distance;
  // volume = 32.05 * distance + 29.492;
  Serial.print("H: ");
  Serial.println(distance);
  // Serial.print(" V:");
  // Serial.println(volume);
  delay(100);
}
```

c. Analice cada linea de código y cuando entienda su funcionamiento corra el *sketch* usando el botón *Upload*

d. Fije el sensor en el aditamento plástico y coloquelo en la parte superior del beaker

e. Realice una medición de distancia para poder calibrar el valor cero.

f. Calibre el sensor en cero de forma que pueda medir la altura desde la base el beaker

g. Llene el beaker hasta 50ml, tome 10 datos y utilice el promedio para relacionar altura y volumen en ese punto

h. Llene el beaker hasta 150ml, tome 10 datos y utilice el promedio para relacionar altura y volumen en ese punto

i. Calcule la curva de calibración de volumen usando esos dos puntos

j. Vacie el beaker

k. Llene el beaker a 50, 100, 150 y 200ml y en cada uno de esos puntos tome el promedio de 10 datos de volumen como valor medido

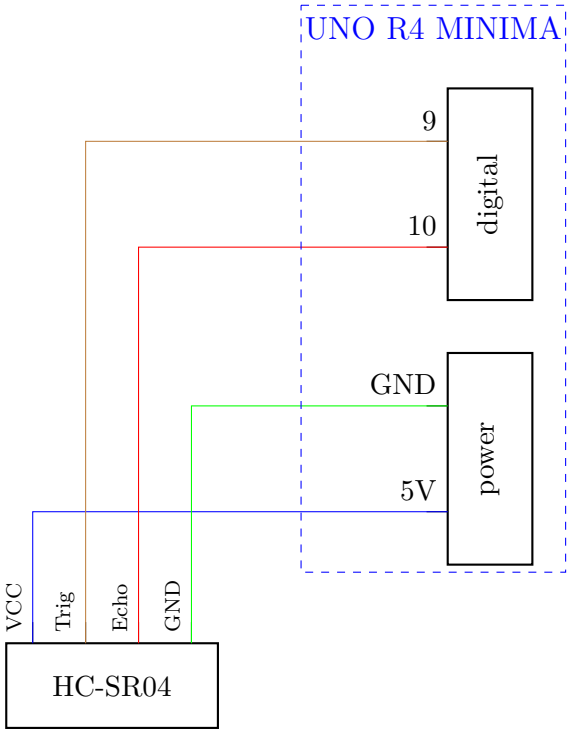

Figura 5.1: Conexión de sensor ultrasónico de distancia

5.3.2. Análisis

a. Compare los valores medidos con los valores teóricos para 50, 100, 150 y 200ml

b. Determine la incertidumbre por repetibilidad de la medición de volumen

5.4. Actividad 2

5.4.1. Procedimiento

a. Usando la misma configuración anterior, realice una nueva toma de datos de 15 segundos en la que inicie con el beaker vacío y lo llene hasta aproximadamente 200ml

5.4.2. Análisis

a. Incluya una gráfica del proceso de llenado

b. ¿Se observar algún tipo de distorsión en la medición? Si es así, ¿a que se debe?