University of Applied Sciences Hamburg	Department of Electrical Engineering	**Laboratory for Instrumentation and Measurement**
Group No**: 02**	**LAB 01** (Revision)	**Celestine Machuca 2570138**
Date: **27.04.2023**		**Soodeh Mousavi asl 2571713**
Professor: **Prof.Dr. Rasmus Retting**		

Function and Characteristics of Ultrasound-Based ToF Sensors for Distance Measurement

Contents

Objectives

- Understanding principles and characteristics of ultrasound-based sensors for measuring distance.
- Study the behaviour of the ultrasound wave.
- Measuring the time of flight (ToF) and investigating the accuracy, resolution, and quantization

Materials Used

- HC-SR04 ultrasonic sensor
- Arduino nano microcontroller
- Breadboard for prototyping
- Platform IO for code compilation and uploading.
- Python with Jupiter.

Setup

The ultrasonic sensor - HC-SR04 - was connected to the Arduino Nano to measure the distance between the sensor and an object. The hardware setup and the pin connections are as follows:

- Connect the VCC of the sensor to +5V on the Arduino board.
- Connect Trigger pin (Trig) to digital pin D3 on the Arduino board.
- Connect Echo pin to digital pin D2 on the Arduino board.
- Connect ground (GND) of the sensor to the GND pin of the Arduino board.

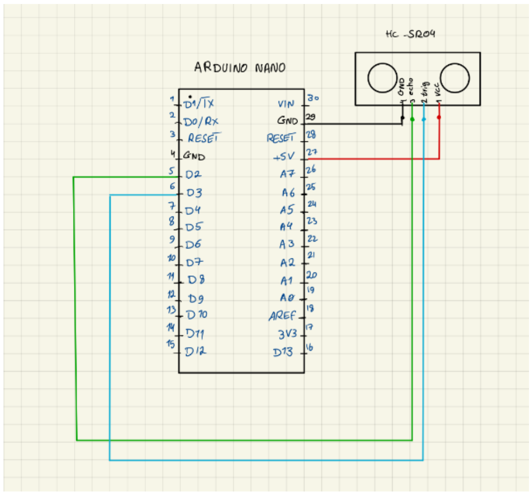


Figure 1 Connection Diagram

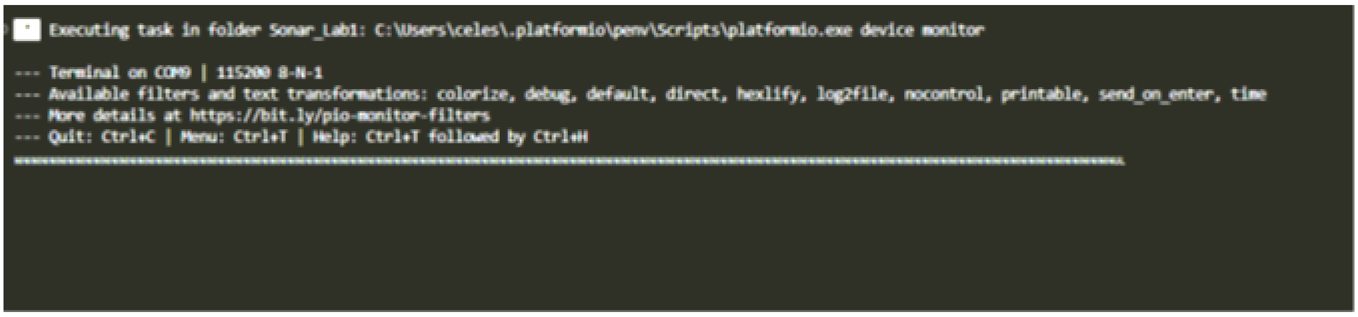
The VCC and GND pins are used to power the sensor. The Trig pin is used to send a 10us pulse to the sensor to start the measurement. The Echo pin is used to receive the echo signal from the sensor. The echo signal is a pulse that is sent back from the sensor to the microcontroller. The length of the echo signal is proportional to the distance between the sensor and the object. The sensor has a maximum range of 4m.

The following code is used to measure the distance between the sensor and an object. The code is written in C++ and uses the Arduino framework. The code is compiled and uploaded to the microcontroller using Platform IO.

```cpp
#include <Arduino.h>

#define ECHO_PIN 2
#define TRIGGER_PIN 3
#define LED_PIN 13

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial.println("Init");
  pinMode(ECHO_PIN, INPUT);
  pinMode(TRIGGER_PIN, OUTPUT);
  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(TRIGGER_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGGER_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER_PIN, LOW);
  long microsencondsFlightTime = pulseIn(ECHO_PIN, HIGH);
  Serial.println(microsencondsFlightTime);
  if (microsencondsFlightTime < 2900) {
    digitalWrite(LED_PIN, HIGH);
  } else {
    digitalWrite(LED_PIN, LOW);
  }
}
```

Question: *What happens if the transmission parameters of Arduino and Computer do not match?*

The baud rate of the Arduino was set to a different value than the baud rate of the computer. This results in an error due to mismatched transmission parameters. The baud rate of the Arduino can be changed in the code by changing the value of the Serial.begin() function. If the values mismatch, the computer will not be able to read the data from the serial monitor properly. The screenshot below shows the corrupted output:



```
 Executing task in folder Sonar_Lab1: C:\Users\celes\.platformio\penv\Scripts\platformio.exe device monitor

--- Terminal on COM9 | 115200 8-N-1
--- Available filters and text transformations: colorize, debug, default, direct, hexlify, log2file, nocontrol, printable, send_on_enter, time
--- More details at https://bit.ly/pio-monitor-filters
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H
```

Figure 2 Baudrate Mistmatch

Question: *What is the maximum Baud-rate?*

The maximum baud rate depends on the microcontroller. The Arduino nano can reportedly support a baud rate of higher than 115200 if not using the cap from the Arduino IDE.

Question: *How is the TOF measurement calculated?*

The TOF measurement is calculated by measuring the time it takes for the ultrasonic signal to travel from the sensor to the object and back to the sensor. The time is measured in microseconds. The distance is calculated by multiplying the time by the speed of sound (343m/s) and dividing the result by 2. The result is in meters.

The full procedure involves the following steps:

1. Send a 10us pulse to the sensor to start the measurement.

2. Wait for the echo signal to be received.

3. Measure the time it takes for the echo signal to be received.

4. Calculate the distance by multiplying the time by the speed of sound and dividing

the result by 2, in our case we reported back the raw time in microseconds for later.

processing in the notebook.

Question: *What is the dataflow from the sensor to the Arduino to the Computer?*

In general, the data flow from the sensor to the Arduino to the computer involves the sensor sending raw data to the Arduino that will be processed and transmitted to the computer via serial communications. Then the computer will process the received data to perform different tasks. The flow chart below shows the data flow between Sensor, Arduino, and the computer:

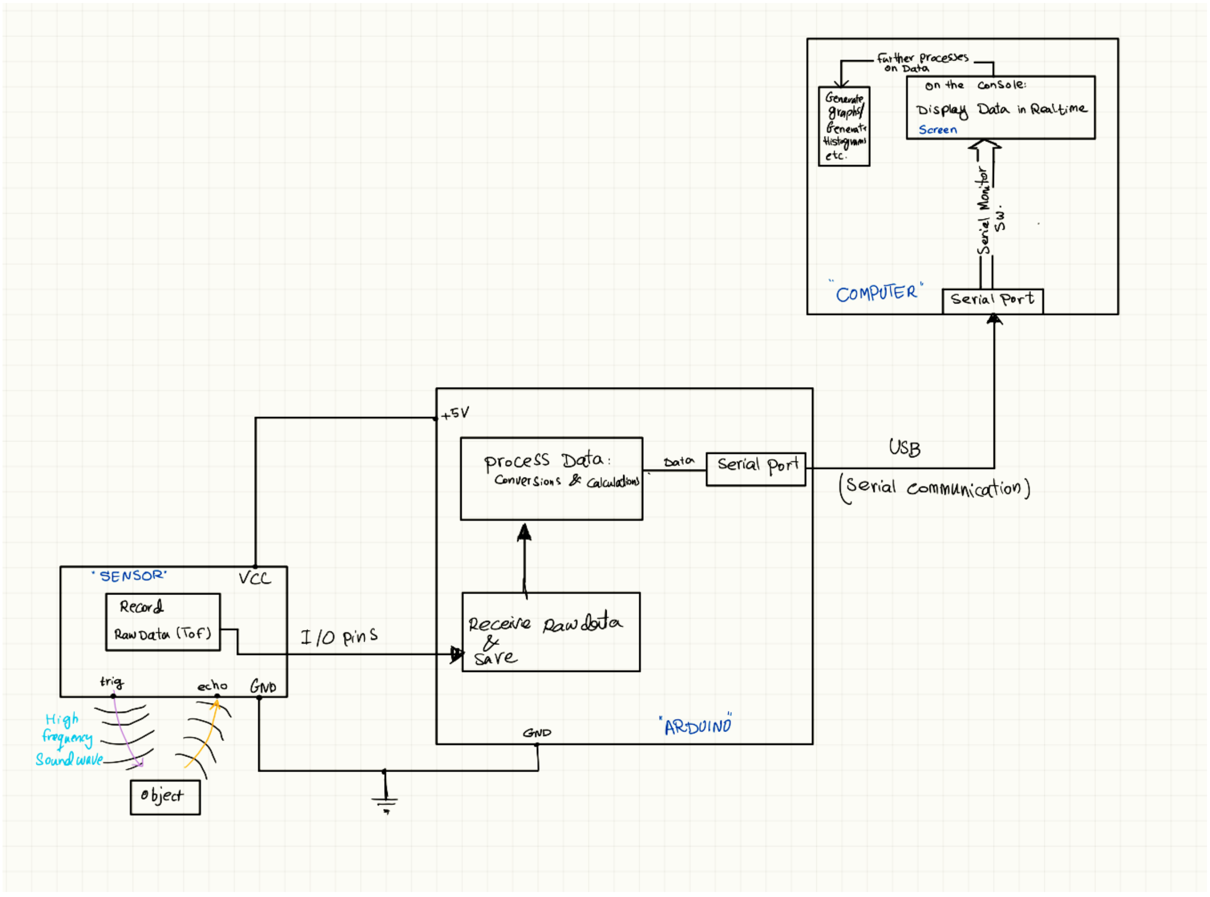


Figure 3 Data Flow

The Interface

In this part the sensor was connected to the oscilloscope with two probes while it was working. One probe to the trigger pin and the other one to the echo pin. The results were recorded and can be seen in the following screenshots.

The following lines are related to the trigger and echo signals:

```
//Trigger Signal

digitalWrite(TRIGGER_PIN,  HIGH);
delayMicroseconds(10);
digitalWrite(TRIGGER_PIN,  LOW);
```

```
//echo signal
long microsencondsFlightTime  = pulseIn(ECHO_PIN, HIGH);
```

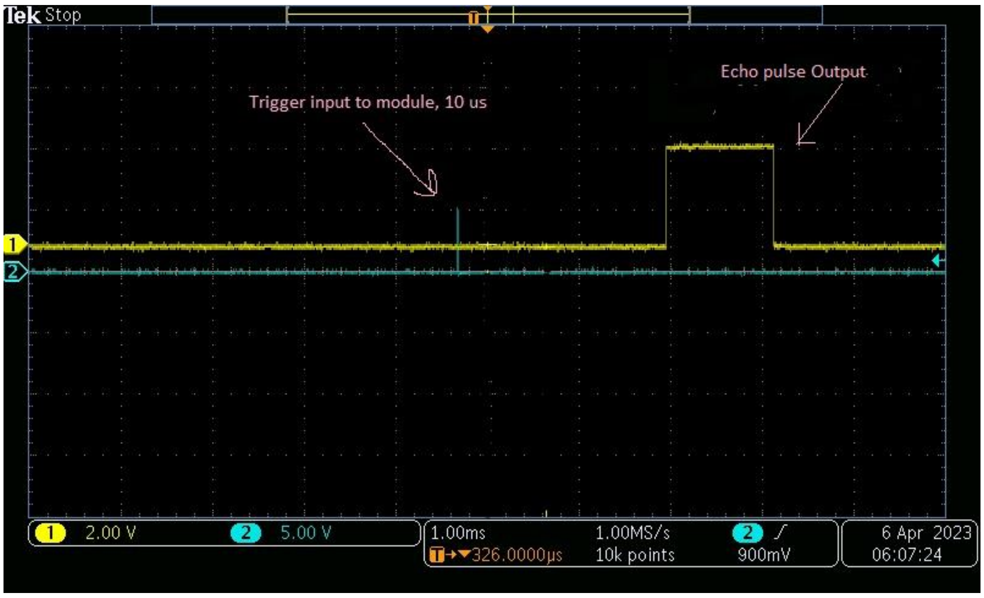


Figure 4 Voltages of echo (yellow) and trigger (blue) over time connected to the channel 1 and 2 of the oscilloscope respectively.

In this experiment the TRIG signal is a 10us pulse that is sent to the sensor to start the measurement. The ECHO signal is the pulse that is sent back from the sensor to the microcontroller. The length of the ECHO signal is proportional to the distance between the sensor and the object.

The trigger signal as seen in blue on the fig x is the programed 10us pulse as seen in the snippet below.

```
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
```

The echo signal as seen in yellow on the fig x is the pulse that is sent back from the sensor to the microcontroller. The snippet below shows how the time it takes for the echo signal to be received is measured.

```
long duration = pulseIn(echoPin, HIGH);
```

The length of the ECHO signal is proportional to the distance between the sensor and the object. The detection of that signal is done using the `pulseIn()` function in the Arduino framework.

Question: *What is the measurement frequency?*

The program has a delay of nearly 1 second after calling the function for generating the trigger pulse. Hence, the generated pulse will be 1sec plus the ToF each time. The maximum range of ToF is 24ms based on the datasheet for an object located at 4m from the sensor, so it is negligible. That means the measurement frequency is ~1Hz because every new measurement will be generated every almost 1 second.

Sensor characteristics

For this part of the experiment, an object (a laptop) was placed at the distance of 100cm initially from the sensor. 100 measurements were taken, and a histogram was generated to visualize the data distribution.

Afterwards, the measurement was repeated for 5 additional distances (10cm, 20cm, 30cm, 40cm, and 60cm).

The results can be seen in the following histograms:

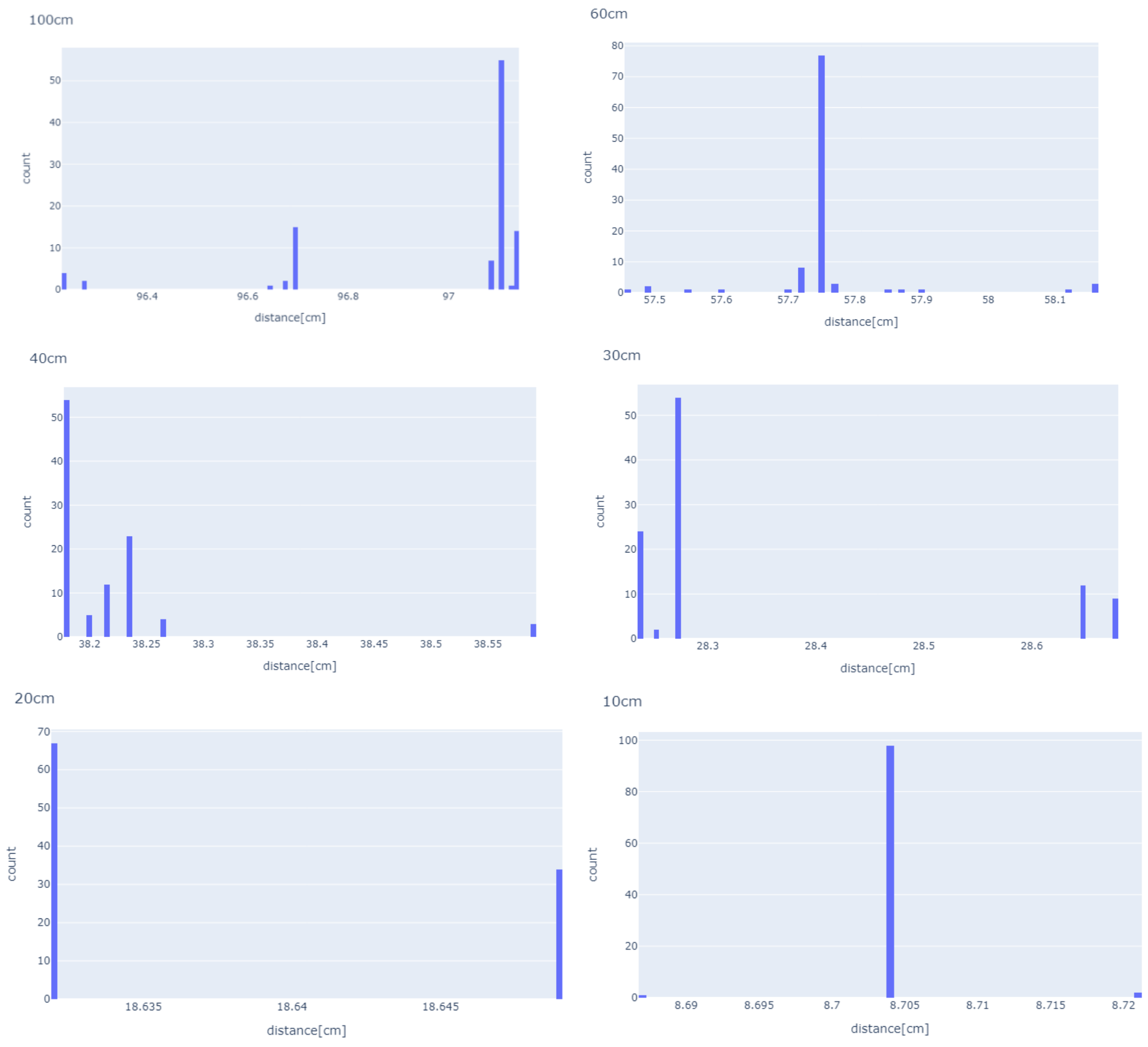


Figure 5 Histograms generated for different distances in cm from the sensor, 100 measurements for each

As can be seen all the histograms have one strong peak indicating that the sensor can measure the distance quite uniformly. However, there are some scattered data which are more at greater distances. These can be due to several potential reasons such as, measurements error caused by noise, interference from other sensors or environmental factors such as temperature or humidity. Reflection and echoes can be another reason for the error, as the ultrasonic sensors emit sound waves that bounce the object and return to the sensor, which these waves may reflect off multiple objects prior to returning to the sensor. Sensor limitations or object properties can be other possible reasons. To determine the exact reason, it is necessary to obtain more measurements and test and analyse every possible factor.

The mean values of the measured data by the sensor for an object at different distances as well as calculated speed of ultrasound can be seen in table below:

Distance[cm]	Mean value of ToF[us]	Speed of ultrasound[m/s] (calculated)	Relative-Error
100	5704.86	350.578	0.0220933 (2.2%)
60	3397.31	353.221	0.0297987 (2.9%)
40	2264.87	355.892	0.0375857 (3.7%)
30	1698.65	359.884	0.0492251 (4.9%)
20	1132.43	364.852	0.063707 (6.3%)
10	566.22	390.617	0.138827 (13.8%)

From the following formula the speed of ultrasound was calculated which can be seen in the table above.

Speed = (2 x distance)/ToF

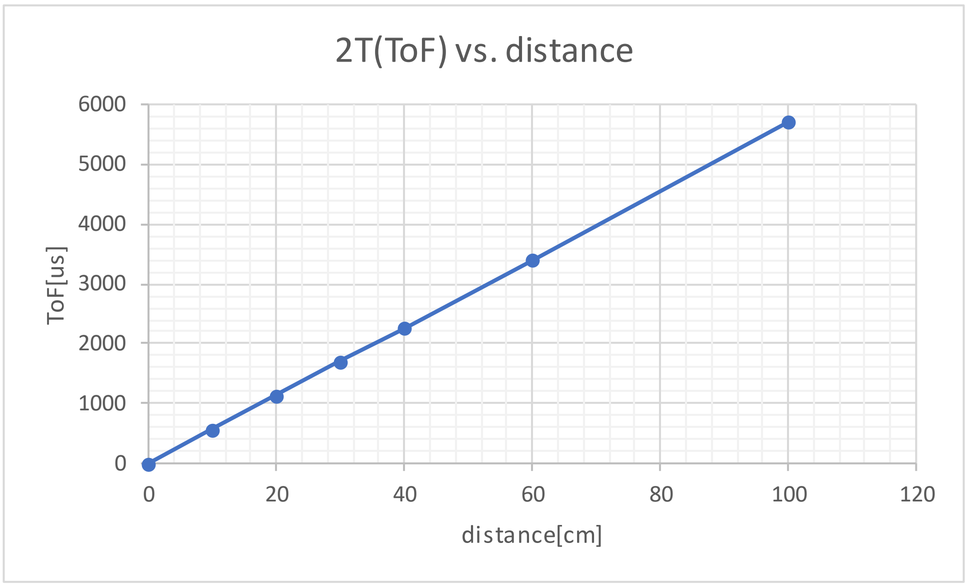
The speed of sound $\sim 343\ m/s$ was used as a reference to calculate the relative error.

The calculated speeds are higher than the standard speed of sound. This means that the medium that the sound wave were propagating was not at the standard temperature and pressure and density. As can be seen this calculated value increased at shorter distances. This means the ambience that measurement was performed in had higher density than the standard air density that the speed of sound was obtained. Other factors such as pressure, temperature, or impurities can affect the speed of sound measurements.

Further investigations and tests are needed to figure out the real reason for these differences.

Next the ToF as a function of distance was plotted indicating a trendline. The inverse of the slope of the resulting trendline was used to calculate the speed of sound by times two because the ultrasound waves travel twice the distance from an object.

The calculated speed of sound was 350.828 m/s with 2.2% approximated error as compared to the standard speed of sound 343.15 m/s at 20°C.



Accuracy of the measurements can be obtained by comparing the measured distances against the actual distance to generate the relative error. Then the accuracy of each distance measurement can be calculated by following equation:

Accuracy = (1-Relative error) x True value

Real distance[cm]	Relative error [cm]	Accuracy [cm]
100	0.03	97
60	0.04	57.6
40	0.05	38
30	0.06	28.2
20	0.07	18.6
10	0.13	8.7

The HC-SR04 ultrasonic sensor has a **resolution** of 1.0 as because it measures the Tof as an integer in microseconds. The 1us would be the **quantization** value since the ToF for the minimum distance is 1us.

Modifying the program

The program was updated to output the distance in cm. The following function was added to the code:

```
void distanceInCentimeters(long microseconds)
{
  double speedOfSound = 343.0;
  double distance = (microseconds * speedOfSound) / 2.0;
  return distance / 100.0;
}
```

Ultrasound sensing of different objects.

For this part of the experiment a square object was placed in front of the sensor at different positions and angles to investigate the performance of the sensor. The object was positioned in a horizontal and diagonal position for this purpose. The results can be seen in the generated relative error heatmap. The image of the setup to determine the detection maps and the error heatmaps can be seen below:

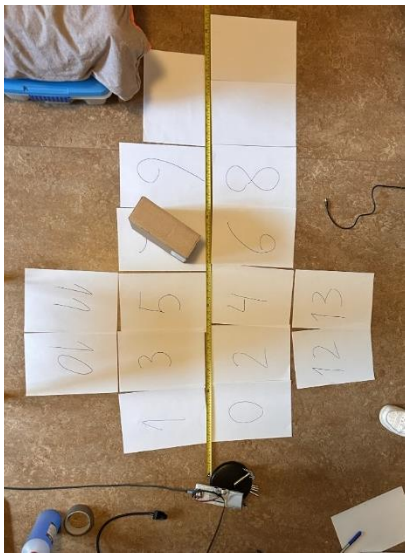


Figure 6 Grid Measurement with horizontal and diagonal.

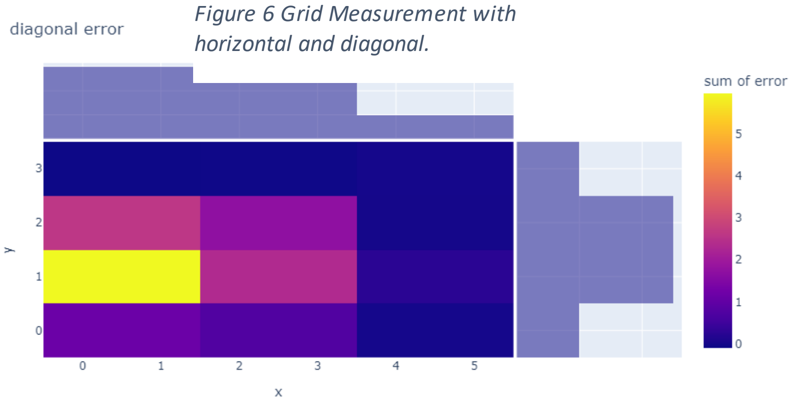


Figure 7 Diagonal 45% Relastive Error to Grid Position

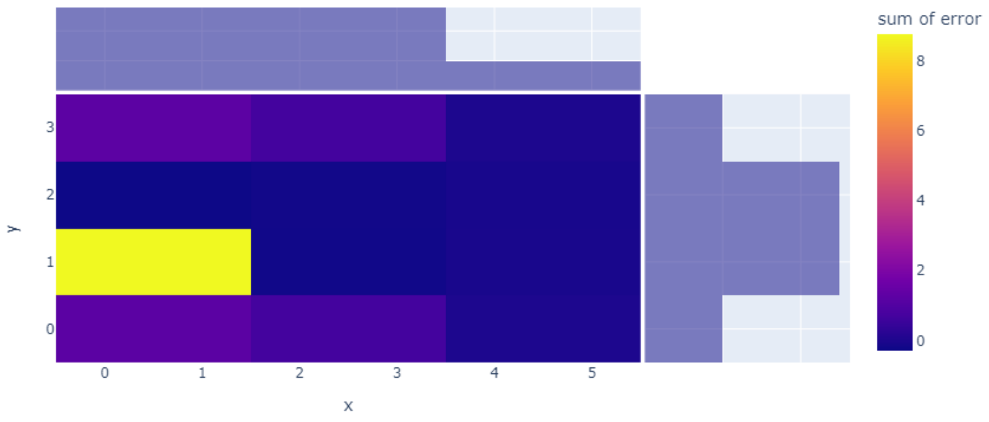


Figure 8 Horizontal Position Relative Error to Grid Position

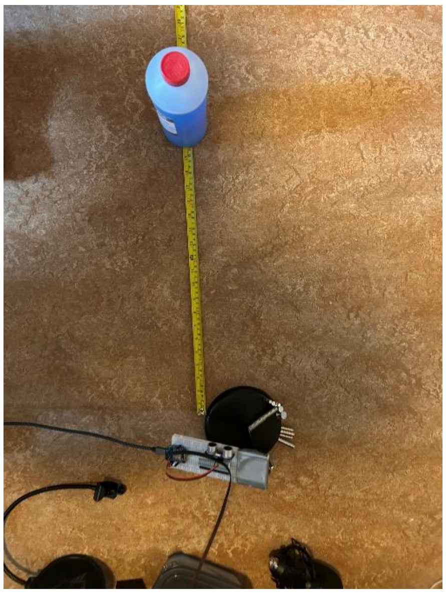


Figure 9 Round Object

From the measurement done on the grid it can be seen an increase on the relative error depending on angle of the flat surface finding that a diagonal position result on the worse results.

Programming a threshold

In this part, the program was modified for the Arduino to switch On LED (13) while an object comes closer than 2m and turn Off when the distance is greater than that.

```
#include <Arduino.h>
#define ECHO_PIN 2
#define TRIGGER_PIN 3
#define LED_PIN 13

void setup()
{

  Serial.begin(115200);
  Serial.println("Init");
  pinMode(ECHO_PIN, INPUT);
  pinMode(TRIGGER_PIN, OUTPUT);
  pinMode(LED_PIN, OUTPUT);
}
void loop()
{

  digitalWrite(TRIGGER_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGGER_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER_PIN, LOW);
  long microsencondsFlightTime = pulseIn(ECHO_PIN, HIGH);

  // programming threshold for distances lower and higher than 2m=> 5800us=Δt
  Serial.println(microsencondsFlightTime);
  if (microsencondsFlightTime < 5800)
  {
    digitalWrite(LED_PIN, HIGH);
  }
  else
  {
    digitalWrite(LED_PIN, LOW);
  }
}
```

From the measurement done on the grid it can be seen an increase on the relative error depending on the angle of the flat surface. This indicates that at the diagonal position measured a distance which immensely inaccurate especially at the angle ~45°. This says that the ultrasound waves are scattered after bouncing the object when it was at diagonal positions. During the lab a rounded object (a bottle of water) was placed in front of the sensor primarily at the distance of ~40cm. To test the performance of the sensor the object was placed in a wider range area. The results were accurate at the first position as was expected. Getting farther from the side (changing the position of the object horizontally) affected the accuracy of the results indicating that the sensor was not able to detect the object properly.

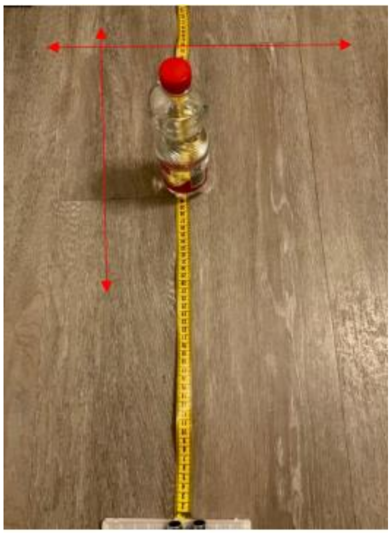


Figure 10 Water bottle used for round object measurements.

Note: this photo was taken after the lab just representing the process of the measurements

BUL2_LAB_1