Project: Sensor Proposal

Date: September 23, 2023

Group member: Elijah Weddle

Wisdom Omodiagbe

Yilin Wang

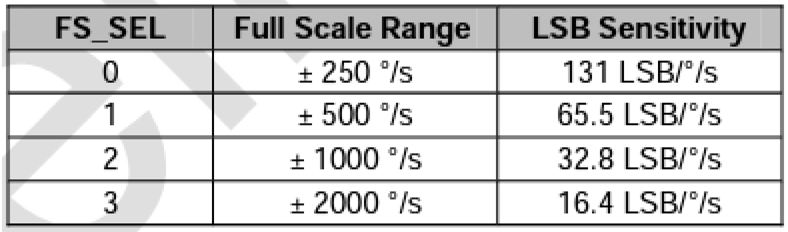
Youxin Wang

MPU6050 (Accelerometer)

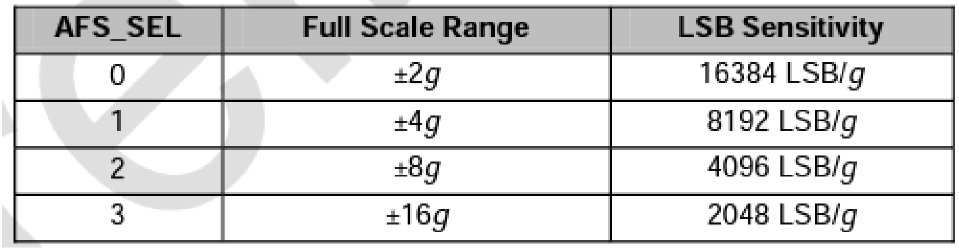
Introduction

The MPU6050 sensor uses capacitors to measure linear acceleration in 3 dimensions. and angular acceleration in 3 axes. The system contains a mass held in position by springs that contains a potential difference to a fixed reference component. When the MPU6050 sensor is accelerated, the mass will want to stay at rest which will cause a change in distance between the reference and the mass, increasing the capacitance. The angular accelerometer works similarly by taking advantage of the Coriolis effect, also known as the centripetal force which is due to the normal acceleration from rotational motion. For this project, we are only focusing on linear acceleration in one dimension.

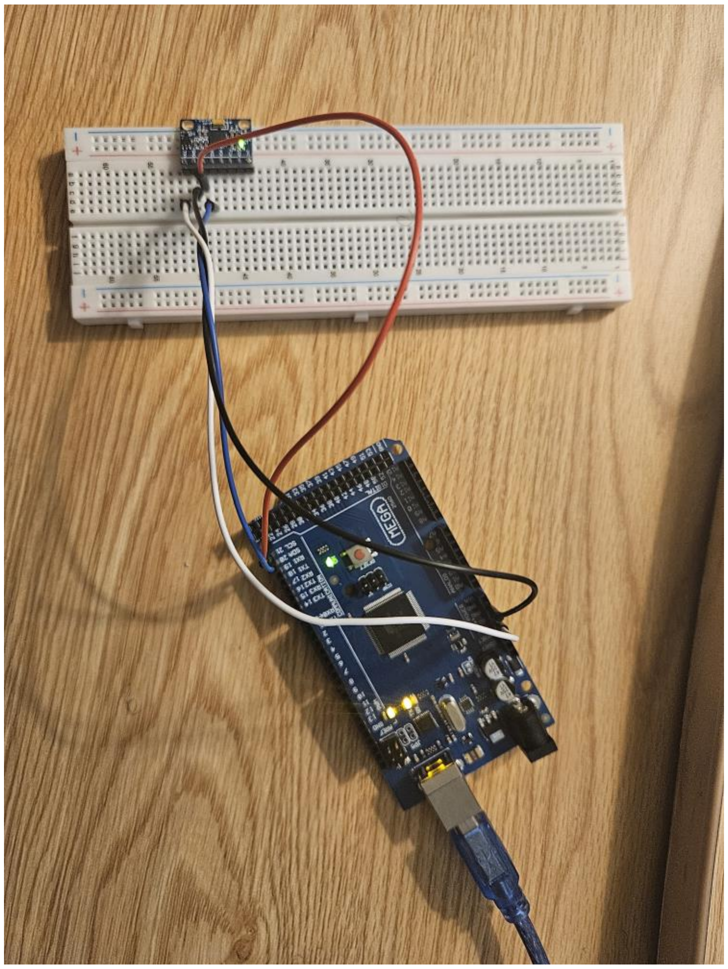
Specifications

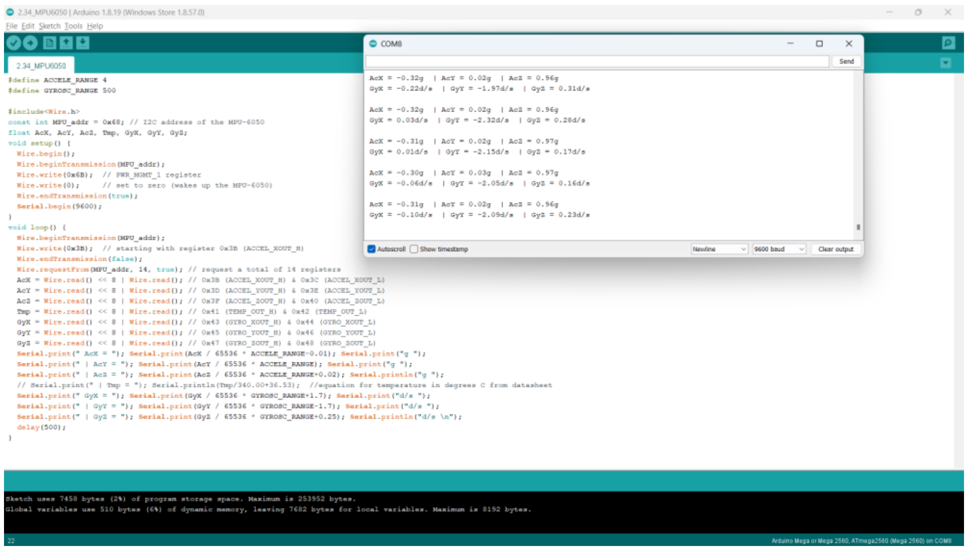


FS_SEL	Full Scale Range	LSB Sensitivity
0	± 250 °/s	131 LSB/°/s
1	± 500 °/s	65.5 LSB/°/s
2	± 1000 °/s	32.8 LSB/°/s
3	± 2000 °/s	16.4 LSB/°/s



AFS_SEL	Full Scale Range	LSB Sensitivity
0	±2*g*	16384 LSB/*g*
1	±4*g*	8192 LSB/*g*
2	±8*g*	4096 LSB/*g*
3	±16*g*	2048 LSB/*g*




2.34_MPU6050 | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help
2.34_MPU6050
COM8
Send

```
#define ACCELE_RANGE 4
#define GYROSC_RANGE 500

#include<Wire.h>
const int MPU_addr = 0x68; // I2C address of the MPU-6050
float AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;
void setup() {
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);  // PWR_MGMT_1 register
  Wire.write(0);     // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
  Serial.begin(9600);
}
void loop() {
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr, 14, true); // request a total of 14 registers
  AcX = Wire.read() << 8 | Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY = Wire.read() << 8 | Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ = Wire.read() << 8 | Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  Tmp = Wire.read() << 8 | Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  GyX = Wire.read() << 8 | Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyY = Wire.read() << 8 | Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyZ = Wire.read() << 8 | Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
  Serial.print(" AcX = "); Serial.print(AcX / 65536 * ACCELE_RANGE-0.01); Serial.print("g ");
  Serial.print(" | AcY = "); Serial.print(AcY / 65536 * ACCELE_RANGE); Serial.print("g ");
  Serial.print(" | AcZ = "); Serial.print(AcZ / 65536 * ACCELE_RANGE+0.02); Serial.println("g ");
  // Serial.print(" | Tmp = "); Serial.println(Tmp/340.00+36.53);  //equation for temperature in degrees C from datasheet
  Serial.print(" GyX = "); Serial.print(GyX / 65536 * GYROSC_RANGE+1.7); Serial.print("d/s ");
  Serial.print(" | GyY = "); Serial.print(GyY / 65536 * GYROSC_RANGE-1.7); Serial.print("d/s ");
  Serial.print(" | GyZ = "); Serial.print(GyZ / 65536 * GYROSC_RANGE+0.25); Serial.println("d/s \n");
  delay(500);
}
```

```
AcX = -0.32g  | AcY = 0.02g  | AcZ = 0.96g
GyX = -0.22d/s  | GyY = -1.97d/s  | GyZ = 0.31d/s

AcX = -0.32g  | AcY = 0.02g  | AcZ = 0.96g
GyX = 0.03d/s  | GyY = -2.32d/s  | GyZ = 0.28d/s

AcX = -0.31g  | AcY = 0.02g  | AcZ = 0.97g
GyX = 0.01d/s  | GyY = -2.15d/s  | GyZ = 0.17d/s

AcX = -0.30g  | AcY = 0.03g  | AcZ = 0.97g
GyX = -0.06d/s  | GyY = -2.05d/s  | GyZ = 0.16d/s

AcX = -0.31g  | AcY = 0.02g  | AcZ = 0.96g
GyX = -0.10d/s  | GyY = -2.09d/s  | GyZ = 0.23d/s
```

[✓ Autoscroll] Show timestamp   Newline   9600 baud   Clear output

```
Sketch uses 7458 bytes (2%) of program storage space. Maximum is 253952 bytes.
Global variables use 510 bytes (6%) of dynamic memory, leaving 7682 bytes for local variables. Maximum is 8192 bytes.
```

22   Arduino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM8

Sensor Measurement Proposal

To measure/observe some of the characteristics, we decided to use a ramp with variable slope to obtain multiple accelerations and multiple data points.

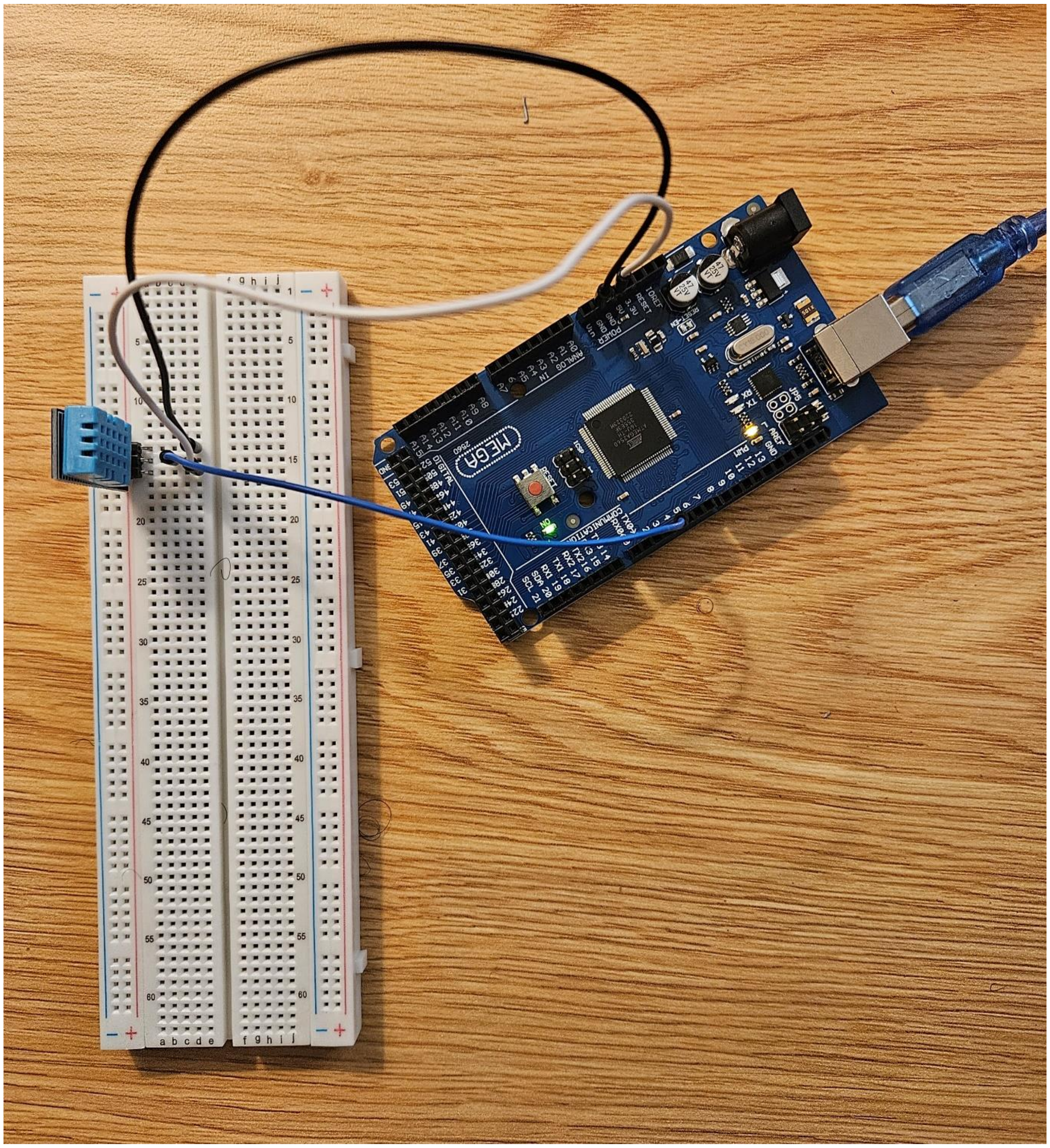
Below are some of the static characteristics that we believe we can measure from this approach.

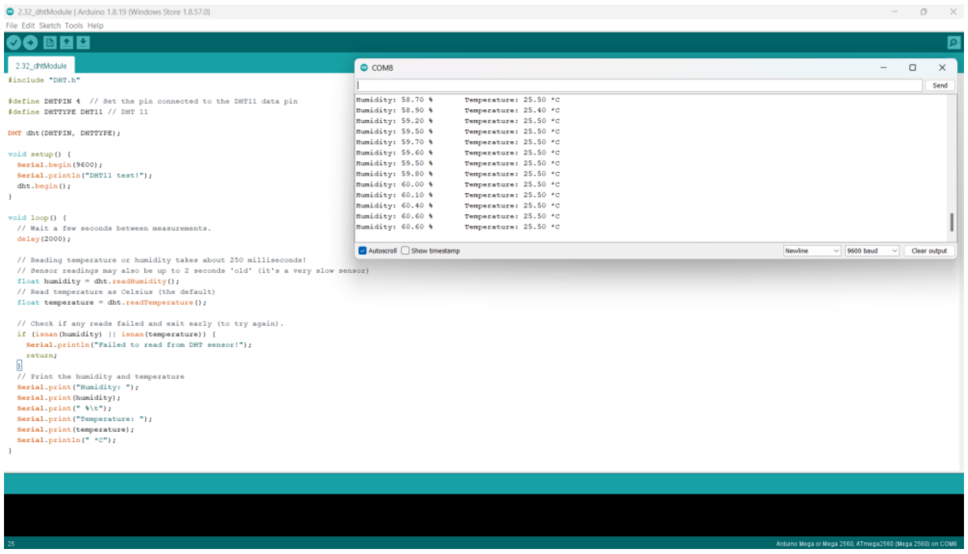
- Accuracy: We obtain multiple readings for each slope and get the average after removing the outliers
- Precision: We can obtain the standard deviation in readings for a fixed slope and use that as our precision.
- Sensitivity: We plot the average of our readings and obtain an approximate transfer function. The slope of the transfer function at any given point will be the sensitivity.
- Linearity: From the transfer function, we could check how best a linear function fits the readings.
- Repeatability: This is the difference between the highest and lowest reading we get when measuring on a fixed ramp slope which is a fixed acceleration.

DHT-11 (Temperature Sensor)

Introduction

The DHT-11 sensor is used to measure both temperature and humidity. The temperature is sensed with a thermistor while the humidity sensor uses a capacitance. The thermistor simply increases resistance as temperature increases. The resistance is measured, and a transfer function associates the temperature. The humidity sensor has a hydrogenic material that absorbs water. As the surroundings change, the hydrogenic sensor will change its dielectric constant based on how much water it is able to absorb. The dielectric constant can be measured and associated to the humidity level.




2.32_dhtModule | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help
2.32_dhtModule
COM8
Send
Autoscroll
Show timestamp
Newline
9600 baud
Clear output
25
Arduino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM8

```
#include "DHT.h"

#define DHTPIN 4  // Set the pin connected to the DHT11 data pin
#define DHTTYPE DHT11 // DHT 11

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  Serial.println("DHT11 test!");
  dht.begin();
}

void loop() {
  // Wait a few seconds between measurements.
  delay(2000);

  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (it's a very slow sensor)
  float humidity = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float temperature = dht.readTemperature();

  // Check if any reads failed and exit early (to try again).
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  // Print the humidity and temperature
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" *C");
}
```

```
Humidity: 58.70 %       Temperature: 25.50 *C
Humidity: 58.90 %       Temperature: 25.40 *C
Humidity: 59.20 %       Temperature: 25.50 *C
Humidity: 59.50 %       Temperature: 25.50 *C
Humidity: 59.70 %       Temperature: 25.50 *C
Humidity: 59.60 %       Temperature: 25.50 *C
Humidity: 59.50 %       Temperature: 25.50 *C
Humidity: 59.80 %       Temperature: 25.50 *C
Humidity: 60.00 %       Temperature: 25.50 *C
Humidity: 60.10 %       Temperature: 25.50 *C
Humidity: 60.40 %       Temperature: 25.50 *C
Humidity: 60.60 %       Temperature: 25.50 *C
Humidity: 60.60 %       Temperature: 25.50 *C
```

Specifications

Item	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4 Pin Single Row

Parameters	**Conditions**	**Minimum**	**Typical**	**Maximum**
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 Bit	
Repeatability			±1%RH	
Accuracy	25℃		±4%RH	
	0-50℃			±5%RH
Interchangeability	Fully Interchangeable			
Measurement Range	0℃	30%RH		90%RH
	25℃	20%RH		90%RH
	50℃	20%RH		80%RH
Response Time (Seconds)	1/e(63%)25℃, 1m/s Air	6 S	10 S	15 S
Hysteresis			±1%RH	
Long-Term Stability	Typical		±1%RH/year	
Temperature				
Resolution		1℃	1℃	1℃
		8 Bit	8 Bit	8 Bit
Repeatability			±1℃	
Accuracy		±1℃		±2℃
Measurement Range		0℃		50℃
Response Time (Seconds)	1/e(63%)	6 S		30 S

Sensor Measurement Proposal

To obtain temperature readings we decided to use two different methods; an airconditioned car to obtain colder temperatures and a heat box from the school of engineering to obtain warmer temperatures. To establish a ground truth, we will use another temperature sensor with better accuracy than the DHT-11.

Here are the static characteristics we believe we can measure.

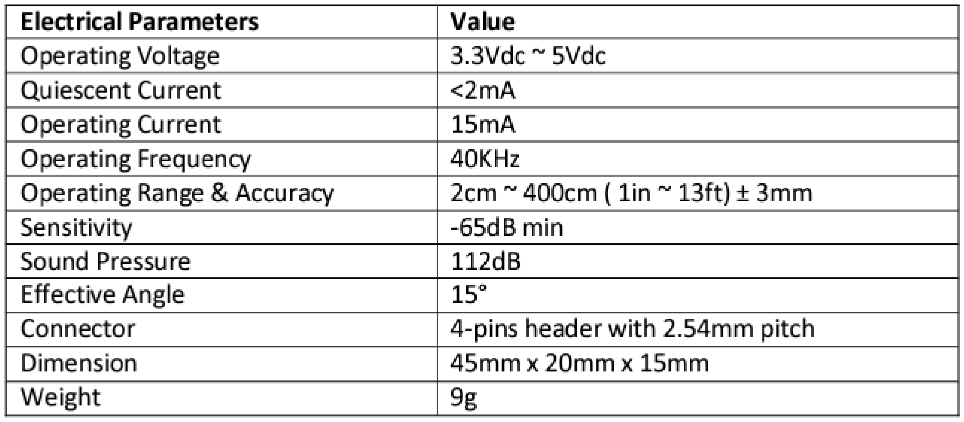
- Accuracy: We obtain multiple readings and compare it with the ground truth.
- Precision: We obtain multiple readings at a certain temperature and get the standard deviation of those readings.
- Repeatability: This will be the deviation/ difference in the readings at a fixed temperature.
- Sensitivity: We plot the average readings at multiple data points and obtain an approximate transfer function. The slope of the transfer function at any point will be the sensitivity.
- Linearity: From the transfer function, we can check how best a linear function fits the readings.
- Drift: We leave our temperature sensor at a fixed temperature for about a minute and observe for any drift in the readings.
- Hysteresis: We take a reading of room temperature after we take it out of the cooling chamber and after we take it out of the heating chamber and check for any difference in average results.

HCSR04 (Ultrasonic distance sensor)

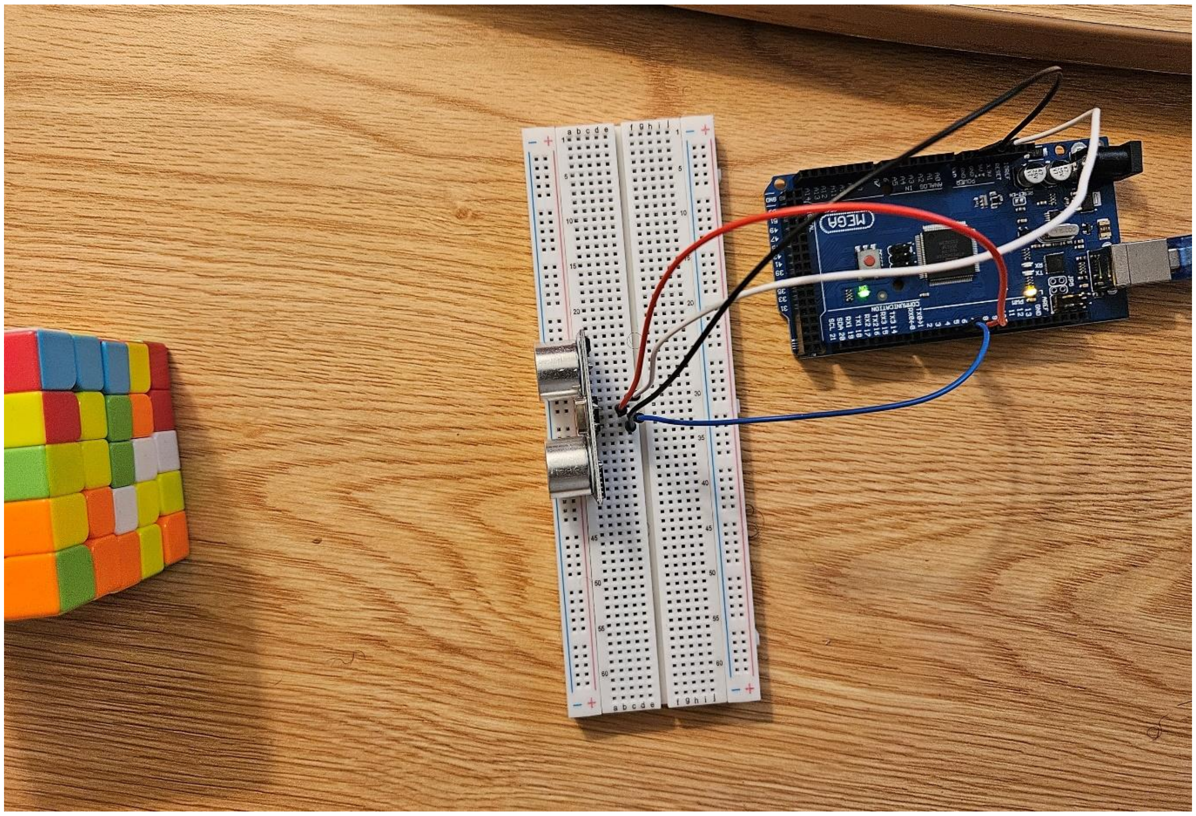
Introduction

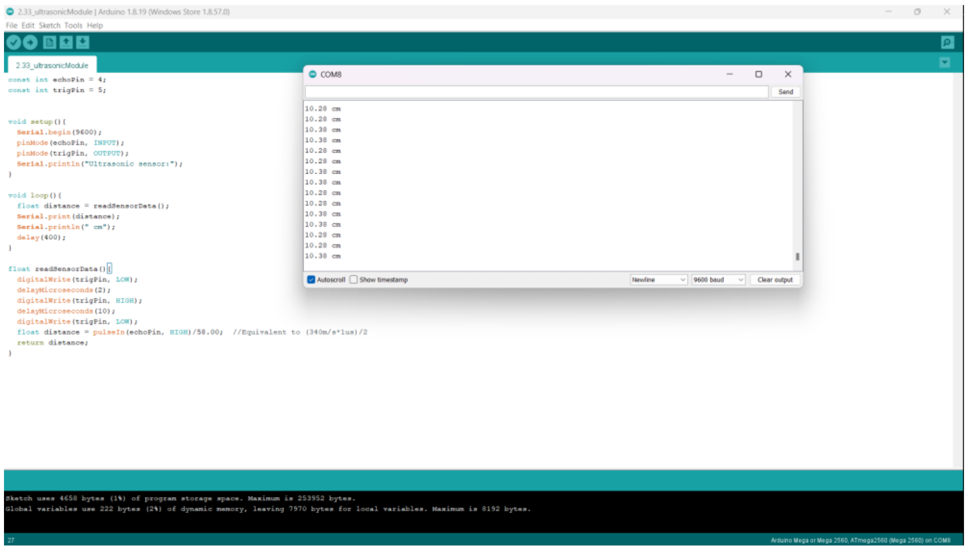
The Ultrasonic sensor is a distance measuring device. The sensor has two main components, one that sends an ultrasonic signal and one that receives it. The processor can find the time between when the signal is sent and when it is received. The time found for travel is then multiplied by the speed of the signal and divided by two. The measurement is divided by two because time taken for the signal to travel is to the point of interest and back, but the device is only interested in the distance to the device.

Specifications



Electrical Parameters	**Value**
Operating Voltage	3.3Vdc ~ 5Vdc
Quiescent Current	<2mA
Operating Current	15mA
Operating Frequency	40KHz
Operating Range & Accuracy	2cm ~ 400cm (1in ~ 13ft) ± 3mm
Sensitivity	-65dB min
Sound Pressure	112dB
Effective Angle	15°
Connector	4-pins header with 2.54mm pitch
Dimension	45mm x 20mm x 15mm
Weight	9g




2.33_ultrasonicModule | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help
2.33_ultrasonicModule
COM8
Send
Autoscroll
Show timestamp
Clear output

```
const int echoPin = 4;
const int trigPin = 5;


void setup(){
  Serial.begin(9600);
  pinMode(echoPin, INPUT);
  pinMode(trigPin, OUTPUT);
  Serial.println("Ultrasonic sensor:");
}

void loop(){
  float distance = readSensorData();
  Serial.print(distance);
  Serial.println(" cm");
  delay(400);
}

float readSensorData(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  float distance = pulseIn(echoPin, HIGH)/58.00;  //Equivalent to (340m/s*1us)/2
  return distance;
}
```

```
10.28 cm
10.28 cm
10.38 cm
10.38 cm
10.28 cm
10.28 cm
10.38 cm
10.38 cm
10.28 cm
10.28 cm
10.38 cm
10.38 cm
10.28 cm
10.28 cm
10.38 cm
```

Newline | 9600 baud

```
Sketch uses 4658 bytes (1%) of program storage space. Maximum is 253952 bytes.
Global variables use 222 bytes (2%) of dynamic memory, leaving 7970 bytes for local variables. Maximum is 8192 bytes.
```

27 Arduino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM8

Sensor Measurement Proposal

To obtain our readings, we will fix our sensor and use it to measure the distance from it to a moveable board. The ground truth for the distance will be obtained using a measuring tape.

Below is the list of properties we can measure.

- Accuracy: We obtain multiple readings and compare it with the ground truth.
- Precision: We obtain multiple readings at a certain distance and get the standard deviation of those readings.

- Repeatability: This will be the deviation/ difference in the readings at a fixed temperature.
- Range: The range from the specs is between 2cm to 4m and that is within our control so we can measure the range.
- Sensitivity: We plot the average readings at multiple data points and obtain an approximate transfer function. The slope of the transfer function at any point will be the sensitivity.
- Linearity: From the transfer function, we can check how best a linear function fits the readings.
- Drift: We fix the distance for a prolonged time and check for any drift in the measurement.
- Resolution: We can make changes to the distance up to 1mm. We can use these small changes to measure the resolution.