Sensor Proposal

BY: Aurora Clark, Malina Brown, Ethan Zumbahlen

Sensors and Actuators

MPU6050 Module:

Findings:

An MPU6050 has a 3-axis accelerometer and a 3-axis gyroscope that measures linear motion and angular motion, respectively. The accelerometer uses the Piezo electric effect to measure inclination and magnitude. The gyroscope, on the other hand, uses the Coriolis effect to measure motion around the x, y, and z axes. (Roll, Pitch, Yaw)

Piezo electric effect: Current is created when a Piezo crystal collides with a "wall." Depending on the current we can indicate orientation.

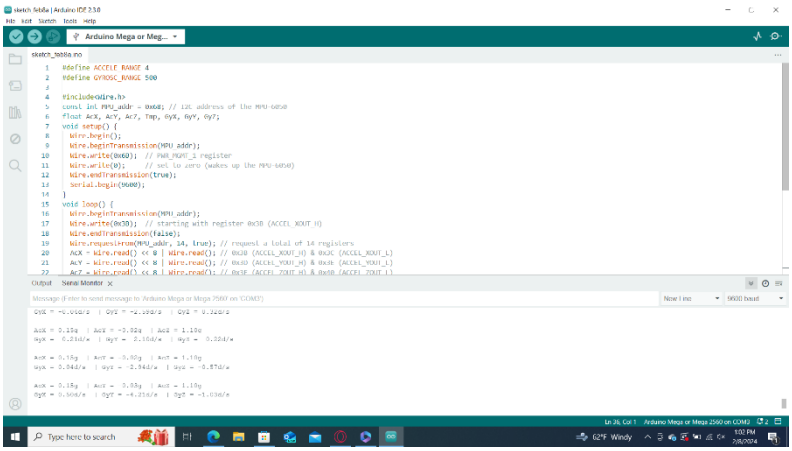
Coriolis effect: This effect works off vibration and when it is tilted/turned the piezo electric crystals experience force in said direction which in turn creates a current. When piezo electric effect current and Coriolis effect currents are in consensus the current is amplified.

Important Characteristics:

- Range
- Sensitivity
- Linearity
- Accuracy
- Drift

Diagram of connection: MPU6050

Arduino code & Sensor Reading: MPU6050



Testing plans:

We will lay the sensor on a flat surface and record values gotten from moving sensor in each direction (x, y, z) and (Roll, Pitch, Yaw). From this data we can analyze range and linearity. To test sensitivity, we can move the sensor in increments and record values to get change of output relative to our input.

The first step will be setting up the circuit on a small breadboard or something it can mount to. Lay it flat on the table with a protractor laying vertically next to it. To measure it, we will make sure the protractor's middle is close to the end of the board the MPU6050 is laying on. And starting from a stop we lift the side opposite from the protractor's center to a set number of degrees for 1 second. Using the initial velocity and the final velocity we can find the acceleration that happened when lifting it from 0 degrees to the degree we are measuring, and we repeat this process to test if it is as accurate with the first measurement. This can test the accuracy and precision of the sensor.

DHT-11:

Findings:

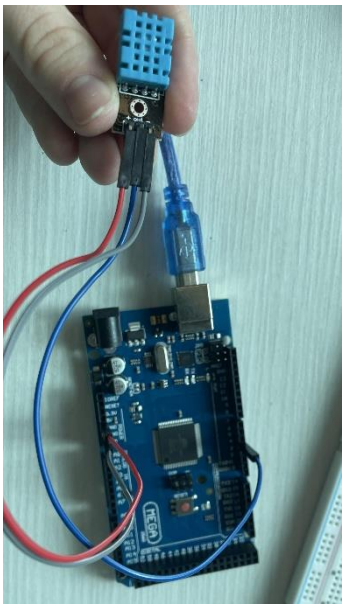
The DHT-11 is a commonly used low-cost temperature and humidity sensor, it is used to measure the air and give a reading in the output of the Arduino code every 1-2 seconds. It features a calibrated digital signal output. It has a resistive-type humidity measurement component (resistive humidity sensor), an NTC temperature measurement component (thermistor), and an 8-bit microcontroller for serial outputs for temperature and humidity. The operating voltage for the DHT-11 is 3.5V to 5.5V, it has a temperature range of 0°C to 50°C, for the humidity the range is 20% to 90%, the resolution for humidity

and temperature are both 16-bit with temperature having a resolution of 1°C, and accuracy of ±1°C and ±1% . There are 2 types of DHT-11, the sensor and the sensor module. Both the sensor and the sensor module work the same, but we have the sensor module. The sensor module comes with 3 pins, a filtering capacitor, and a pull-up resistor. Also, you must download the dht11 library to use the Arduino code to get the readings.

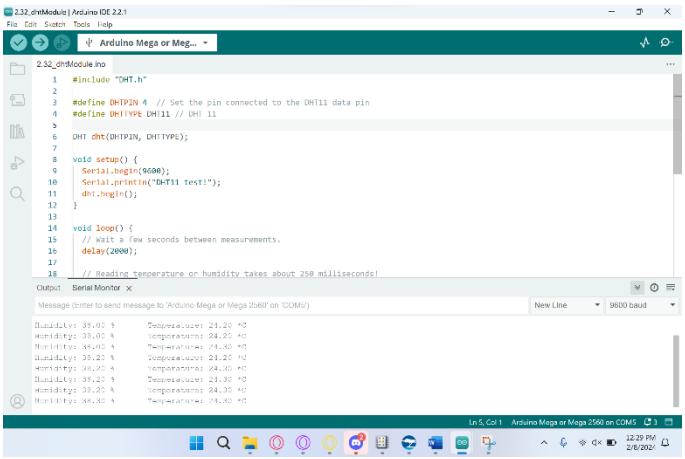
Important characteristics:

- Accuracy
- Range
- Resolution

Diagram of connection: DHT-11



Arduino code & Sensor Reading: DHT-11



Testing plans:

First, we will sit outside with the sensor to measure the temperature and humidity; we will record the values received and compare it to the weather app and an indoor/outdoor AcuRite. Next, we will take the sensor inside of a building to get measurements as well comparing it to an indoor AcuRite and indoor/outdoor AcuRite. These two tests will be done for accuracy. Then we will put it in the deep freezer to measure that as well. This will measure range. For resolution I will watch for the smallest perceived change. All these tests will give us multiple locations and different conditions to work with to test the sensors. While also getting readings from a trusted device to compare numbers.

Ultrasonic Ranging Module:

Findings:

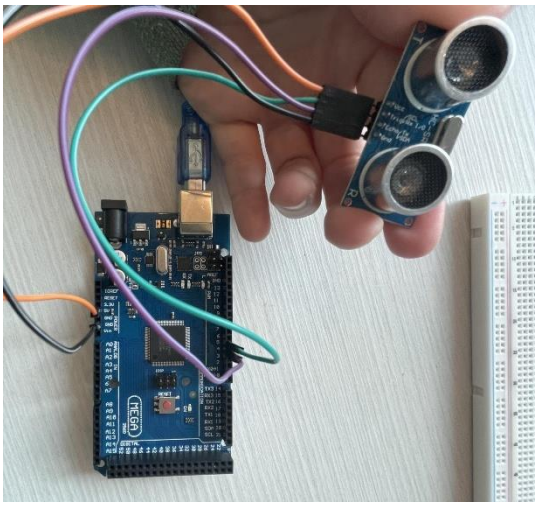
The ultrasonic ranging module consists of the ultrasonic transmitters, a receiver, and a controller unit. The module can measure between 2 cm to 400 cm with no contact using ultrasonic waves. The module uses an IO flip-flop to process a high-level signal of at least 10us. The module automatically sends out eight 40 khz and detects if there is a pulse signal return. If the signal returns, passing the high level, the high output IO duration is the time from the transmission of the ultrasonic wave to return of it.

Important characteristics:

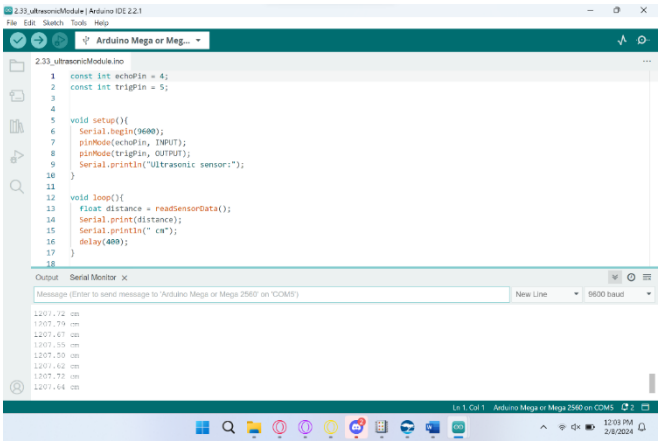
- Range

- Resolution
- Accuracy

Diagram of connection: Ultrasonic



Arduino code & Sensor Reading: Ultrasonic



Testing plans:

To test the ultrasonic sensor will be very simple. We will place an object in front of the sensor at a short distance and record the output. Then we will inch the object away from the sensor while recording the output and the actual distance of the sensor. We will continue to move the object farther away and compare the outputs with the actual distance from the sensor to see how accurate the sensor is.

Full code of each sensor:

2.34_MPU6050 | Arduino IDE 2.2.1

File Edit Sketch Tools Help

Arduino Mega or Mega 2...

2.34_MPU6050.ino

```
#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);
}
```

2.32_dhtModule | Arduino IDE 2.2.1

File Edit Sketch Tools Help

Arduino Mega or Mega 2...

2.32_dhtModule.ino

```
#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");
}
```


2.33_ultrasonicModule | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Arduino Mega or Mega 2...
2.33_ultrasonicModule.ino
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;
}