

# **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)

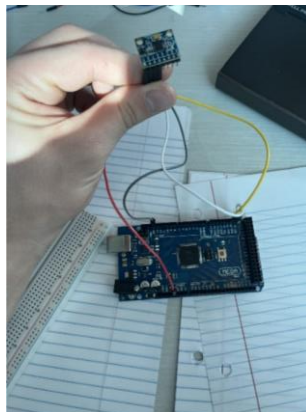
Piezo electric effect: Current is created when a Piezo crystal collides with a “wall.” Depending on the current we can indicate orientation.

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.

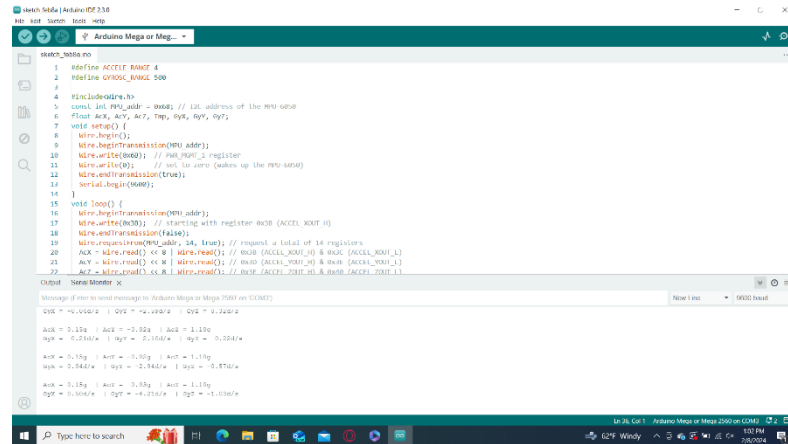
### **Important Characteristics:**

- Range
- Sensitivity
- Linearity
- Accuracy
- Drift

### **Diagram of connection: MPU6050**



## Arduino code & Sensor Reading: MPU6050



```
1 #include <Wire.h>
2 #define ACCEL_RANGE 4
3 #define GYROSC_RANGE 500
4 #include <Wire.h>
5 const int MPU_addr = 0x68; // I2C address of the MPU sensor
6 float Accx, Accy, Accz, Tmp, Gyro_x, Gyro_y, Gyro_z;
7 void setup() {
8   Wire.begin();
9   Wire.beginTransmission(MPU_addr);
10  Wire.write(0x6B); // PWR_MGMT_1 register
11  Wire.write(0); // set to zero (wakes up the MPU sensor)
12  Wire.endTransmission(true);
13  Serial.begin(9600);
14 }
15 void loop() {
16   Wire.beginTransmission(MPU_addr);
17   Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
18   Wire.endTransmission(false);
19   Wire.requestFrom(MPU_addr, 14, true); // request a total of 14 registers
20   Accx = Wire.read() << 8 | Wire.read(); // Read (ACCEL_XOUT_H) & Read (ACCEL_XOUT_L)
21   Accy = Wire.read() << 8 | Wire.read(); // Read (ACCEL_YOUT_H) & Read (ACCEL_YOUT_L)
22   Accz = Wire.read() << 8 | Wire.read(); // Read (ACCEL_ZOUT_H) & Read (ACCEL_ZOUT_L)
23   Tmp = Wire.read();
24   Gyro_x = Wire.read() << 8 | Wire.read(); // Read (GYRO_XOUT_H) & Read (GYRO_XOUT_L)
25   Gyro_y = Wire.read() << 8 | Wire.read(); // Read (GYRO_YOUT_H) & Read (GYRO_YOUT_L)
26   Gyro_z = Wire.read() << 8 | Wire.read(); // Read (GYRO_ZOUT_H) & Read (GYRO_ZOUT_L)
27 }
28
```

Serial Monitor

Message (Click to send message to Arduino Mega or Mega 2560 via COM4)

QW = 0.000000 | QV = 0.000000 | QW = 0.000000

Accx = 0.150 | Accy = -0.020 | Accz = 1.150  
Gyro\_x = 0.000000 | Gyro\_y = 0.000000 | Gyro\_z = 0.000000

Accx = 0.150 | Accy = -0.020 | Accz = 1.150  
Gyro\_x = 0.000000 | Gyro\_y = -0.000000 | Gyro\_z = -0.000000

Accx = 0.150 | Accy = -0.020 | Accz = 1.150  
Gyro\_x = 0.000000 | Gyro\_y = -0.000000 | Gyro\_z = -0.000000

### 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:**

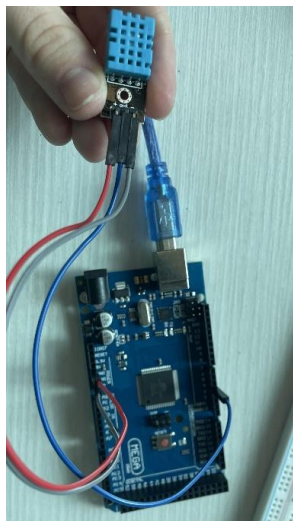
### **Findings:**

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  $\pm 1^\circ\text{C}$  and  $\pm 1\%$ . 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.

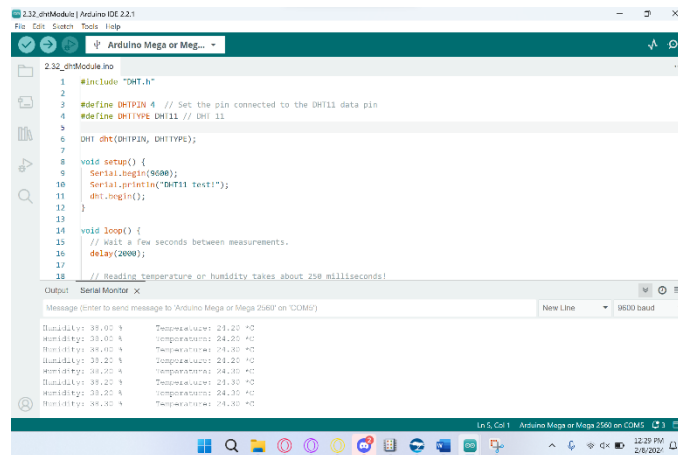
### **Important characteristics:**

- Accuracy
- Range
- Resolution

### **Diagram of connection: DHT-11**



## Arduino code & Sensor Reading: DHT-11



The screenshot shows the Arduino IDE interface with a file named '232\_dhtModule.ino'. The code defines pins for the DHT11 sensor and prints temperature and humidity readings in the serial monitor. The output shows alternating humidity and temperature values.

```
1 #include "DHT.h"
2
3 #define DHTPIN 4 // Set the pin connected to the DHT11 data pin
4 #define DHTTYPE DHT11 // DHT 11
5
6 DHT dht(DHTPIN, DHTTYPE);
7
8 void setup() {
9   Serial.begin(9600);
10  Serial.println("DHT11 test!");
11  dht.begin();
12 }
13
14 void loop() {
15   // Wait a few seconds between measurements.
16   delay(2000);
17
18   // Reading temperature or humidity takes about 250 milliseconds!
```

Serial Monitor Output:

Humidity: 35.00 %	Temperature: 23.20 °C
Humidity: 35.00 %	Temperature: 24.20 °C
Humidity: 35.00 %	Temperature: 24.20 °C
Humidity: 35.20 %	Temperature: 23.20 °C
Humidity: 35.20 %	Temperature: 24.20 °C
Humidity: 35.20 %	Temperature: 24.20 °C
Humidity: 35.20 %	Temperature: 24.20 °C
Humidity: 35.20 %	Temperature: 24.20 °C

### Testing plans:

For accuracy, we will sit outside with the sensor to measure the temperature and humidity and compare the data received to the weather app (ground truth) and an indoor/outdoor AcuRite. We will then take the sensor inside of a building to get measurements data and compare it to the thermostat (ground truth), an indoor AcuRite and indoor/outdoor AcuRite. For range, we will use a deep freezer, oven, shower, and baking soda to measure the highest and lowest values for temperature and humidity that can be accurately read by the sensor. For resolution, we 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:

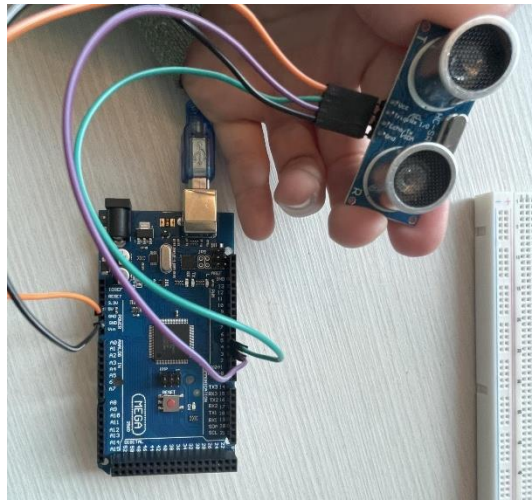
### Findings:

The ultrasonic ranging module consists of 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

The screenshot shows the Arduino IDE 2.2.1 interface. The main editor window displays a C++ program named `2.33_ultrasonicModule.ino`. The code defines two pins, `echoPin = 4` and `trigPin = 5`. In the `setup()` function, the serial port is initialized at 9600 baud, and the pins are configured as `INPUT` and `OUTPUT` respectively. A message "Ultrasonic sensor:" is printed. The `loop()` function reads the sensor data, prints the distance in centimeters, and delays for 400ms. The Serial Monitor window at the bottom shows the output of the program, displaying a series of distance measurements: 1207.72 cm, 1707.79 cm, 1207.67 cm, 1207.65 cm, 1207.50 cm, 1207.62 cm, 1207.72 cm, and 1207.64 cm. The status bar at the bottom indicates the board is set to "Arduino Mega or Mega 2560 on COM3" and the time is 12:08 PM on 2/8/2024.

```
1 const int echoPin = 4;
2 const int trigPin = 5;
3
4
5 void setup(){
6   Serial.begin(9600);
7   pinMode(echoPin, INPUT);
8   pinMode(trigPin, OUTPUT);
9   Serial.println("Ultrasonic sensor:");
10 }
11
12 void loop(){
13   float distance = readSensorData();
14   Serial.print(distance);
15   Serial.println(" cm");
16   delay(400);
17 }
18
```

Serial Monitor X  
Message (Error to send message to 'Arduino Mega or Mega 2560' on 'COM3') New Line 9600 baud

1207.72 cm  
1707.79 cm  
1207.67 cm  
1207.65 cm  
1207.50 cm  
1207.62 cm  
1207.72 cm  
1207.64 cm

Ln 1, Col 1 Arduino Mega or Mega 2560 on COM3 12:08 PM 2/8/2024

## 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
1  #define ACCELE_RANGE 4
2  #define GYROSC_RANGE 500
3
4  #include<Wire.h>
5  const int MPU_addr = 0x68; // I2C address of the MPU-6050
6  float AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;
7  void setup() {
8      Wire.begin();
9      Wire.beginTransmission(MPU_addr);
10     Wire.write(0x6B); // PWR_MGMT_1 register
11     Wire.write(0); // set to zero (wakes up the MPU-6050)
12     Wire.endTransmission(true);
13     Serial.begin(9600);
14 }
15 void loop() {
16     Wire.beginTransmission(MPU_addr);
17     Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
18     Wire.endTransmission(false);
19     Wire.requestFrom(MPU_addr, 14, true); // request a total of 14 registers
20     AcX = Wire.read() << 8 | Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
21     AcY = Wire.read() << 8 | Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
22     AcZ = Wire.read() << 8 | Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
23     Tmp = Wire.read() << 8 | Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
24     GyX = Wire.read() << 8 | Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
25     GyY = Wire.read() << 8 | Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
26     GyZ = Wire.read() << 8 | Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
27     Serial.print(" AcX = "); Serial.print(AcX / 65536 * ACCELE_RANGE+0.01); Serial.print("g ");
28     Serial.print(" AcY = "); Serial.print(AcY / 65536 * ACCELE_RANGE); Serial.print("g ");
29     Serial.print(" AcZ = "); Serial.print(AcZ / 65536 * ACCELE_RANGE+0.02); Serial.print("g ");
30     // Serial.print(" Tmp = "); Serial.println(Tmp/340.00+36.53); //equation for temperature in degrees C from datasheet
31     Serial.print(" GyX = "); Serial.print(GyX / 65536 * GYROSC_RANGE+1.7); Serial.print("d/s ");
32     Serial.print(" GyY = "); Serial.print(GyY / 65536 * GYROSC_RANGE-1.7); Serial.print("d/s ");
33     Serial.print(" GyZ = "); Serial.print(GyZ / 65536 * GYROSC_RANGE+0.25); Serial.println("d/s \n");
34     delay(500);
35 }
```

### 2.32\_dhtModule | Arduino IDE 2.2.1

File Edit Sketch Tools Help

Arduino Mega or Mega 2...

```
2.32_dhtModule.ino
1  #include "DHT.h"
2
3  #define DHTPIN 4 // Set the pin connected to the DHT11 data pin
4  #define DHTTYPE DHT11 // DHT 11
5
6  DHT dht(DHTPIN, DHTTYPE);
7
8  void setup() {
9      Serial.begin(9600);
10     Serial.println("DHT11 test!");
11     dht.begin();
12 }
13
14 void loop() {
15     // Wait a few seconds between measurements.
16     delay(2000);
17
18     // Reading temperature or humidity takes about 250 milliseconds!
19     // Sensor readings may also be up to 2 seconds 'old' (it's a very slow sensor)
20     float humidity = dht.readHumidity();
21     // Read temperature as Celsius (the default)
22     float temperature = dht.readTemperature();
23
24     // Check if any reads failed and exit early (to try again).
25     if (isnan(humidity) || isnan(temperature)) {
26         Serial.println("Failed to read from DHT sensor!");
27         return;
28     }
29     // Print the humidity and temperature
30     Serial.print("Humidity: ");
31     Serial.print(humidity);
32     Serial.print(" %\t");
33     Serial.print("Temperature: ");
34     Serial.print(temperature);
35     Serial.println(" °C");
36 }
```



2.33\_ultrasonicModule.ino

```
1  const int echoPin = 4;
2  const int trigPin = 5;
3
4
5  void setup(){
6      Serial.begin(9600);
7      pinMode(echoPin, INPUT);
8      pinMode(trigPin, OUTPUT);
9      Serial.println("Ultrasonic sensor:");
10 }
11
12 void loop(){
13     float distance = readSensorData();
14     Serial.print(distance);
15     Serial.println(" cm");
16     delay(400);
17 }
18
19 float readSensorData(){
20     digitalWrite(trigPin, LOW);
21     delayMicroseconds(2);
22     digitalWrite(trigPin, HIGH);
23     delayMicroseconds(10);
24     digitalWrite(trigPin, LOW);
25     float distance = pulseIn(echoPin, HIGH)/58.00; //Equivalent to (340m/s*1us)/2
26     return distance;
27 }
28
```