Aim:

To use Arduino IDE and MPU6050 IMU and code a program in Arduino IDE to observe and record the pitch, yaw and roll.

Introduction:

There are six degrees of freedom for a rigid body moving in three-dimensional space. As the movement along each of the three axes is independent of each other and independent of the rotation of any of these axes, the motion has six degrees of freedom. Pitch, yaw and roll are the three dimensions of movement when an object moves through a medium. The terms may describe an aeroplane's movements through the air. They are also applied to watercraft moving through water and spacecraft moving through space. The surfaces of a plane and the fins of a fish have a similar function. They adjust the object's attitude as it moves through the fluid. Submarines face the same dynamic control problems as fish do.

Theory:

The MPU6050 IMU has a 3-Axis accelerometer and 3-Axis gyroscope integrated on a single chip.

The gyroscope measures the angular velocity along the three axes. So it is not directly able to predict roll, pitch or yaw. But, integrating angular velocity over time gives us the angle, which can be used to measure the change in roll, pitch and yaw. The gyroscope outputs are in degrees per second, so to get the angular position, we just need to integrate the angular velocity.

On the other hand, the MPU6050 Accelerometer measures acceleration.

Briefly, it can measure gravitational acceleration along the three axes and using some trigonometry math, we can calculate the angle at which the sensor is positioned. So, if we fuse or combine the accelerometer and gyroscope data, we can get very accurate information about the sensor orientation.

The Accelerometer measures and tells you the amount of force (acceleration) it is experiencing in X, Y and Z directions. Now, this data makes sense in orientation because of gravity. We know that if an object is not moving, it will

experience acceleration only due to gravity (neglecting the other minimal forces). The direction of gravitational force is always the same concerning the earth's frame. Still, based on the orientation of IMU, it will experience different amounts of acceleration along the three axes. These acceleration values can give us roll and pitch values.

```
pitch = 180 * atan2(accelX, sqrt(accelY*accelY + accelZ*accelZ))/PI;
roll = 180 * atan2(accelY, sqrt(accelX*accelX + accelZ*accelZ))/PI;
```

Roll, pitch, and yaw are rotational forces, or moments, about the X, Y, and Z axes. Just like pure linear forces, these moment forces need to be considered when calculating bearing life or determining the suitability of a linear system to withstand static loads.

Roll is the rotation about the x-axis (between -180 and 180 degree).
Pitch is the rotations about the y-axis (between -90 and 90 degree).
Yaw is the rotation about the z-axis (between -180 and 180 degree).

Hardware Components Required:

1. Arduino UNO
2. Jumper cables
3. MPU6050 IMU

Software Required:

Arduino IDE

PROCEDURE:

We can connect and read the data from the MPU6050 sensor using Arduino. We are using the I2C protocol for communication with the Arduino, so we need only two wires for connecting it, plus two wires for powering it.

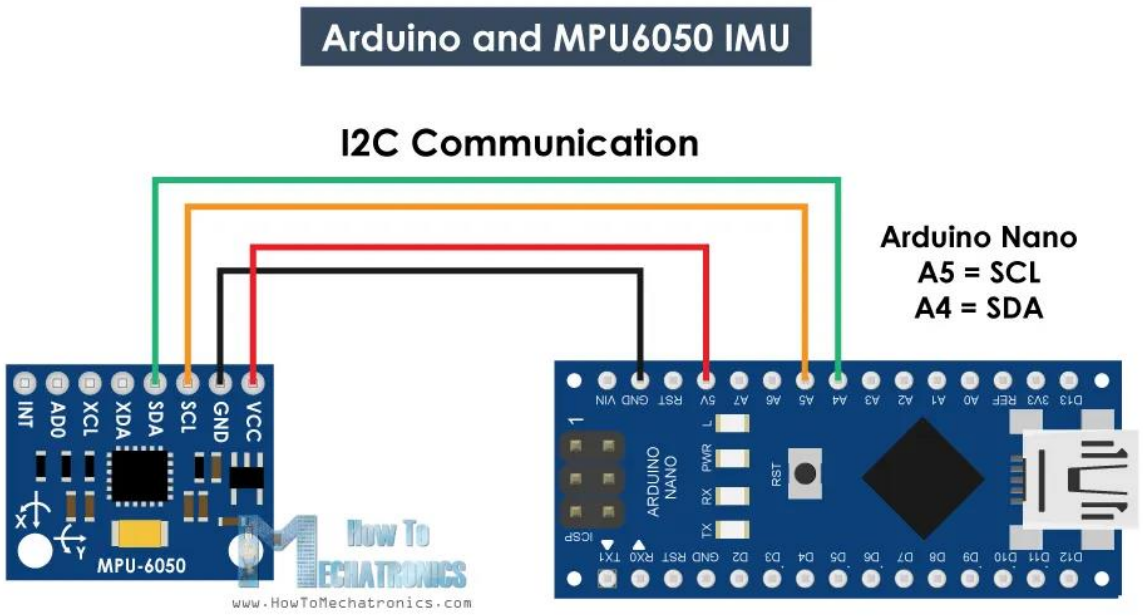
Make the following connections:

1. Connect the VCC pin of the MPU6050 with the 5 volts pin of the Arduino.

2. Connect the GND pin of the MPU6050 with the GND pin of the Arduino.

3. Join the SCL pin of the MPU6050 with the A5 pin of the Arduino.

4. Join the SDA pin of the MPU6050 with the A4 pin of the Arduino.



Upload the MPU6050 Arduino Code.

CODE:

```
/*
  Arduino and MPU6050 Accelerometer and Gyroscope Sensor
*/

#include <Wire.h>

const int MPU = 0x68;                    // MPU6050 I2C address
float AccX, AccY, AccZ;
float GyroX, GyroY, GyroZ;
float accAngleX, accAngleY, gyroAngleX, gyroAngleY, gyroAngleZ;
float roll, pitch, yaw;
float AccErrorX, AccErrorY, GyroErrorX, GyroErrorY, GyroErrorZ;
float elapsedTime, currentTime, previousTime;
```

```
int c = 0;
void setup()
{
  Serial.begin(19200);
  Wire.begin();                           // Initialize comunication
  Wire.beginTransmission(MPU);        // Start communication with MPU6050 // MPU=0x68
  Wire.write(0x6B);                        // Talk to the register 6B
  Wire.write(0x00);                       // Make reset - place a 0 into the 6B register
  Wire.endTransmission(true);          //end the transmission


  /*
  // Configure Accelerometer Sensitivity - Full Scale Range (default +/- 2g)
  Wire.beginTransmission(MPU);
  Wire.write(0x1C);                  //Talk to the ACCEL_CONFIG register (1C hex)
  Wire.write(0x10);                  //Set the register bits as 00010000 (+/- 8g full scale range)
  Wire.endTransmission(true);
  // Configure Gyro Sensitivity - Full Scale Range (default +/- 250deg/s)
  Wire.beginTransmission(MPU);
  Wire.write(0x1B);                   // Talk to the GYRO_CONFIG register (1B hex)
  Wire.write(0x10);                   // Set the register bits as 00010000 (1000deg/s full scale)
  Wire.endTransmission(true);
  delay(20);
  */

  delay(20);
}


void loop()

{

          // === Read acceleromter data === //

  Wire.beginTransmission(MPU);
```

```
  Wire.write(0x3B);
                                        // Start with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);

  Wire.requestFrom(MPU, 6, true);   // Read 6 registers total, each axis value is
stored in 2 registers

  //For a range of +-2g, we need to divide the raw values by 16384, according to
the datasheet

  AccX = (Wire.read() << 8 | Wire.read()) / 16384.0; // X-axis value
  AccY = (Wire.read() << 8 | Wire.read()) / 16384.0; // Y-axis value
  AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0; // Z-axis value


  // Calculating Roll and Pitch from the accelerometer data

  accAngleX = (atan(AccY / sqrt(pow(AccX, 2) + pow(AccZ, 2))) * 180 / PI) - 0.58;
// AccErrorX ~(0.58)
  accAngleY = (atan(-1 * AccX / sqrt(pow(AccY, 2) + pow(AccZ, 2))) * 180 / PI) +
1.58; // AccErrorY ~(-1.58)

  // === Read gyroscope data === //

  previousTime = currentTime;      // Previous time is stored before the actual
time read
  currentTime = millis();             // Current time actual time read

  elapsedTime = (currentTime - previousTime) / 1000;

                                          // Divide by 1000 to get seconds
  Wire.beginTransmission(MPU);

  Wire.write(0x43);                       // Gyro data first register address 0x43

  Wire.endTransmission(false);

  Wire.requestFrom(MPU, 6, true);      // Read 4 registers total, each axis value is
stored in 2 registers
```

```
  GyroX = (Wire.read() << 8 | Wire.read()) / 131.0; // For a 250deg/s range we have to divide first the raw value by 131.0, according to the datasheet

  GyroY = (Wire.read() << 8 | Wire.read()) / 131.0;
  GyroZ = (Wire.read() << 8 | Wire.read()) / 131.0;

  // Correct the outputs with the calculated error values

  GyroX = GyroX + 0.56; // GyroErrorX ~(-0.56)
  GyroY = GyroY - 2; // GyroErrorY ~(2)
  GyroZ = GyroZ + 0.79; // GyroErrorZ ~ (-0.8)

  // Currently the raw values are in degrees per seconds, deg/s, so we need to multiply by sendonds (s) to get the angle in degrees

  gyroAngleX = gyroAngleX + GyroX * elapsedTime;          // deg/s * s = deg
  gyroAngleY = gyroAngleY + GyroY * elapsedTime;
  yaw = yaw + GyroZ * elapsedTime;

  // Complementary filter - combine acceleromter and gyro angle values
  roll = 0.96 * gyroAngleX + 0.04 * accAngleX;
  pitch = 0.96 * gyroAngleY + 0.04 * accAngleY;

  // Print the values on the serial monitor
  Serial.print("Roll :");
  Serial.print(roll);
  Serial.print(" Pitch: ");
  Serial.print(pitch);
  Serial.print(" Yaw: ");
  Serial.println(yaw);

}
```

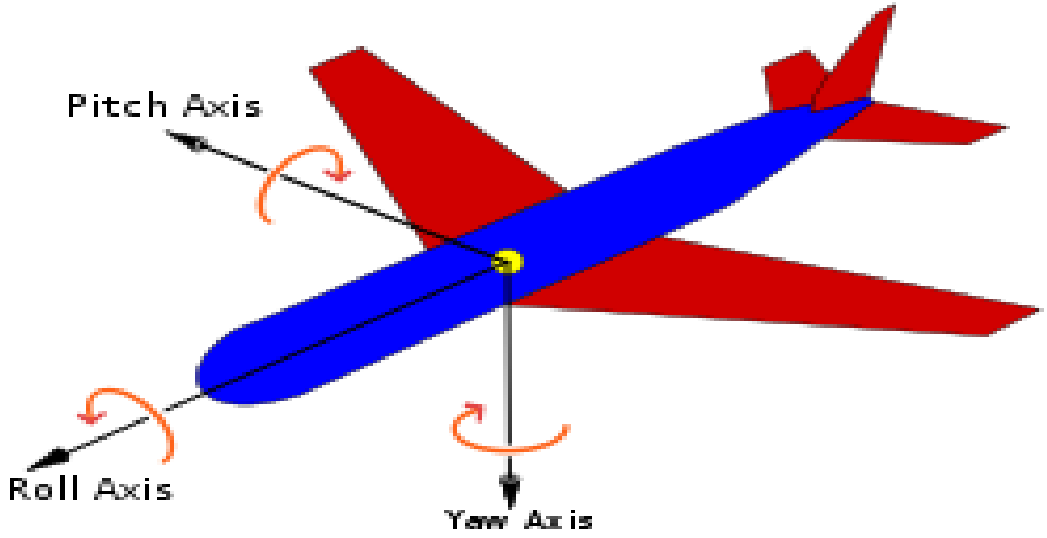
Application:

Aircraft and Aeroplanes:

A gyroscope is a vitally needed instrument or device on an aircraft, whether flown by a human pilot, an autopilot under human supervision, or autonomously.

Imagine three lines running through an aeroplane and intersecting at right angles at the aeroplane's centre of gravity.

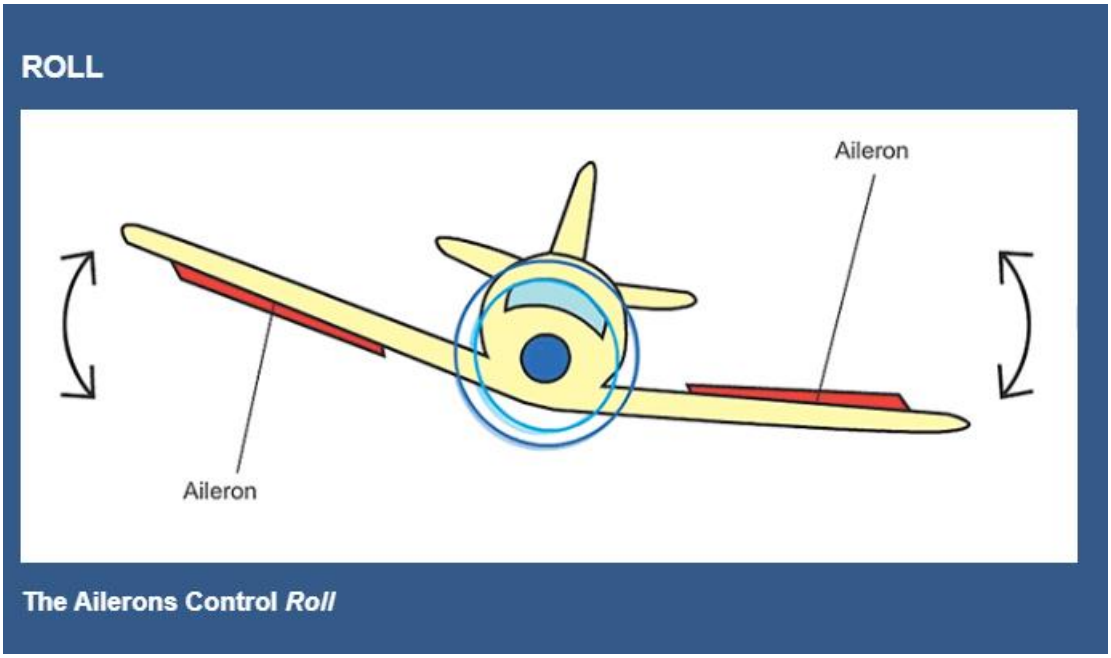
- Rotation around the front-to-back axis is called **roll**.
- Rotation around the side-to-side axis is called **pitch**.
- Rotation around the vertical axis is called **yaw**.



MAINTAINING CONTROL

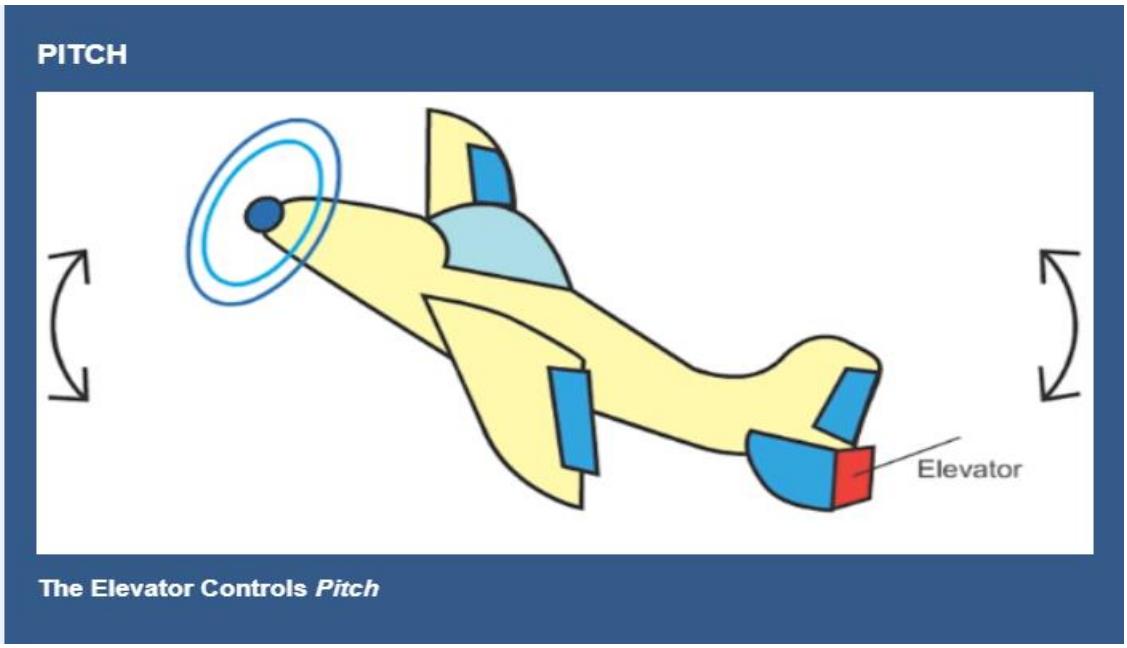
The Ailerons Control Roll

On the outer rear edge of each wing, the two ailerons move in opposite directions, up and down, decreasing lift on one wing while increasing it on the other. This causes the airplane to roll to the left or right. To turn the airplane, the pilot uses the ailerons to tilt the wings in the desired direction.



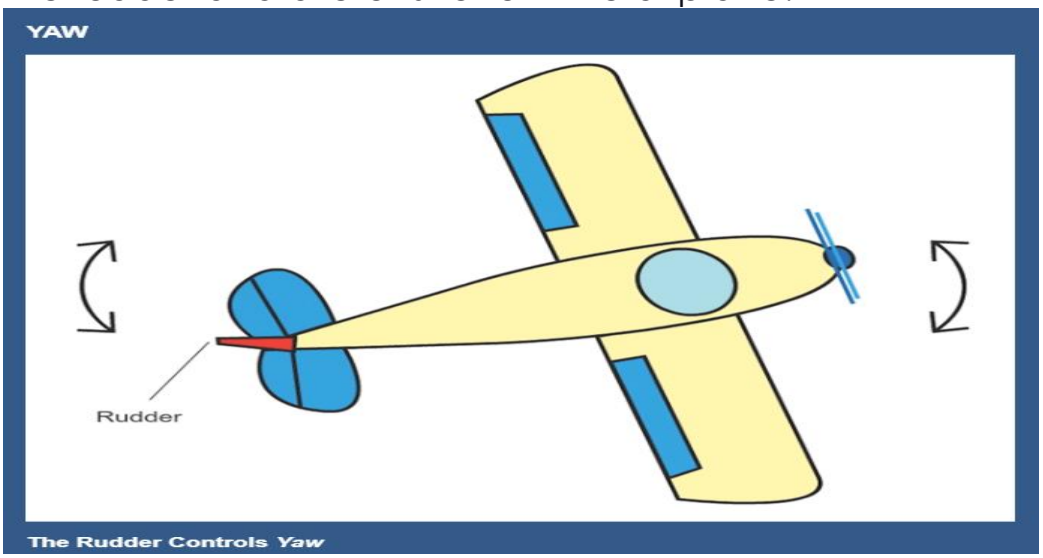
The Elevator Controls Pitch

On the horizontal tail surface, the elevator tilts up or down, decreasing or increasing lift on the tail. This tilts the nose of the airplane up and down.



The Rudder Controls Yaw

On the vertical tail fin, the rudder swivels from side to side, pushing the tail in a left or right direction. A pilot usually uses the rudder and ailerons to turn the airplane.



The two axes of the horizontal plane are typically defined as X and Y, with the X axis being in the direction of motion. The Y axis is orthogonal (perpendicular) to the direction of motion and is also in the horizontal plane. The Z axis is orthogonal to both the X and Y axes but is located in the vertical plane. To find the positive direction of the Z axis, we use the right-hand rule: point the index finger in the direction of positive X, then curl it in the direction of positive Y, and the thumb will indicate positive Z.

References:

https://howthingsfly.si.edu/flight-dynamics/roll-pitch-and-yaw#:~:text=Rotation%20around%20the%20front%2Dto,vertical%20axis%20is%20called%20yaw.

https://howtomechatronics.com/tutorials/arduino/arduino-and-mpu6050-accelerometer-and-gyroscope-tutorial/