Line by Line Explanation of the Code.

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

#include <Wire.h> //The wire library-which allows communication with MPU6050

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;

/\*  Configuring Sensitivities

  // 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);

\*/

//Setup is when the Aurdino is powered or reset. It initialises the baud rate to 19200 i.e. initialises serial communication, starts communication and sends commands to reset the sensor. Delay is to stabilise.

void setup() {

  Serial.begin(19200);               //19200 is the baud rate. Many factors to determine the baud rate, but 19200 is best.

  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

  delay(20);

}

void loop() {

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

//Continuously request for accelerometer data from MPU 6050//

  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. This is to convert it into gravity units.

  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) See the calculate\_IMU\_error()custom function for more details

  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

//To read gyroscope data from MPU6050 sensor. It calculates elapsed time between 2 readings, and determines time difference. Then requests, gyroscope data from the sensor.

  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

//Convert into angular velocity in degrees per second (dps). Divide by 131, as specified in the MPU6050 data sheet.

  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(Error correction.)

  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);

}