**Report #: (1)**

**Robt Orientation Detection Using IMU**

**Student Name:**

Hedaya Rafaat

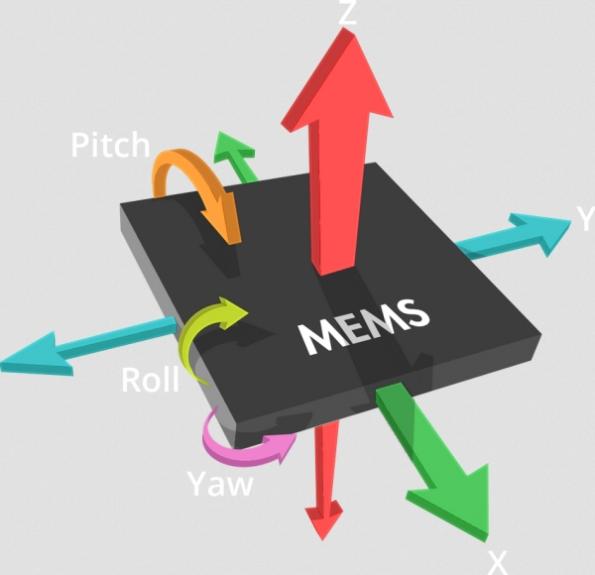
Mahmood Abdallah Saleh

Menna Allah Soliman

# Objective

* Detect robot orientation and Position using IMU (Inertia Moment Unit)
* Used IMU MPU 6050 6DOF (3DOF Gyro & 3DOF Accelerometer)

# IMU MPU 6050

* IMU Measure the Euler angles which around the rotational coordinate
* I2C Digital-output of 6-axis Motion Fusion data in rotation matrix, quaternion, Euler Angle, or raw data format
* Input Voltage: 3-5V
* Tri-Axis angular rate sensor (gyro) with a sensitivity up to 131 LSBs/dps and a full-scale range of ±250, ±500, ±1000, and ±2000dps
  + Gyro 🡪 Instance Angular Velocity
* Tri-Axis accelerometer with a programmable full-scale range of ±2g, ±4g, ±8g and ±16g
  + Accelerometer 🡪 Instance Angular Acceleration
* Digital Motion Processing™ (DMP™) engine offloads complex Motion Fusion, sensor timing synchronization and gesture detection
* Using the Complementary filter to detect exact orientation angle around exact axis assist with the data of Gyro combined with Accelerometer
  + http://www.pieter-jan.com/images/equations/CompFilter_Eq.gif

# Adruino Code

//==================================================================

//============IMU Intialization Global==============================

// MPU-6050 Accelerometer + Gyro

// I2C bus on A4, A5

#include <Wire.h>

#include <math.h> //for atan (tan^-1)

#define MPU6050\_I2C\_ADDRESS 0x68 //mp6050 address

#define FREQ 30.0 // sample freq in Hz (30 readings in one seconds)

// global angle, gyro derived

double gSensitivity = 65.5; // for 500 deg/s, check data sheet

double gx = 0, gy = 0, gz = 0;

double gyrX = 0, gyrY = 0, gyrZ = 0;

int16\_t accX = 0, accY = 0, accZ = 0;

//get offset values from calibration function

double gyrXoffs = -263, gyrYoffs = -67, gyrZoffs = 67;

//==================================================================

//===================================================================

void setup()

{

//===============================================================

//================IMU Setup====================================

int error;

uint8\_t c; //unsigned intger 8bits

uint8\_t sample\_div;

//BTSerial.begin(38400);

Serial.begin(38400);

// debug led

pinMode(13, OUTPUT);

// Initialize the 'Wire' class for the I2C-bus.

Wire.begin();

//----------------------------------------------------------

//write in PWR\_MGMT\_1 (power mangement) 0 to wake up (0x6b--reg add) (0x00--mag)

i2c\_write\_reg (MPU6050\_I2C\_ADDRESS, 0x6b, 0x00);

//----------------------------------------------------------------------

// CONFIG:

// Low pass filter samples, 1khz sample rate

i2c\_write\_reg (MPU6050\_I2C\_ADDRESS, 0x1a, 0x01);

//---------------------------------------------------------------------------------

// GYRO\_CONFIG:

// 500 deg/s, FS\_SEL=1

// This means 65.5 LSBs/deg/s

i2c\_write\_reg(MPU6050\_I2C\_ADDRESS, 0x1b, 0x08);

//--------------------------------------------------------------------------------

// CONFIG:

// set sample rate

// sample rate FREQ = Gyro sample rate / (sample\_div + 1)

// 1kHz / (div + 1) = FREQ

// reg\_value = 1khz/FREQ - 1

sample\_div = 1000 / FREQ - 1; //from data sheet, how freq of sample rate will be

i2c\_write\_reg (MPU6050\_I2C\_ADDRESS, 0x19, sample\_div);

//-------------------------------------------------------

//Serial.write("Calibrating..."); //for calibration only

digitalWrite(13, HIGH);

//calibrate(); //for calibration only

digitalWrite(13, LOW);

//Serial.write("done."); //for calibration only

//===============================================================

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void loop()

{

//================================================================

//====================imu values generating=======================

int error;

double dT;

double ax, ay, az;

unsigned long start\_time, end\_time;

start\_time = millis(); //how many second arduino work from prog beginning

read\_sensor\_data();

// angles based on accelerometer

ay = atan2(accX, sqrt( pow(accY, 2) + pow(accZ, 2))) \* 180 / M\_PI; //pi=3.14

ax = atan2(accY, sqrt( pow(accX, 2) + pow(accZ, 2))) \* 180 / M\_PI;

// angles based on gyro (deg/s)

gx = gx + gyrX / FREQ;

gy = gy - gyrY / FREQ;

gz = gz + gyrZ / FREQ;

// complementary filter

// tau = DT\*(A)/(1-A)

// = 0.48sec

gx = gx \* 0.96 + ax \* 0.04;

gy = gy \* 0.96 + ay \* 0.04;

gy;

// check if there is some kind of request

digitalWrite(13, HIGH);

Serial.print("angle ");

Serial.print(gx, 2);

Serial.print(", ");

Serial.print(gy, 2);

Serial.print(", ");

Serial.println(gz, 2);

Serial.print("acc ");

Serial.print(gyrX, 2);

Serial.print(", ");

Serial.print(gyrY, 2);

Serial.print(", ");

Serial.println(gyrZ, 2);

end\_time = millis();

// remaining time to complete sample time

delay(((1/FREQ) \* 1000) - (end\_time - start\_time));

//Serial.println(end\_time - start\_time);

//================================================================

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//IMU Calibration and Intialization Code

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void calibrate()

{

int x;

long xSum = 0, ySum = 0, zSum = 0;

uint8\_t i2cData[6];

int num = 500;

uint8\_t error;

for (x = 0; x < num; x++)

{

error = i2c\_read(MPU6050\_I2C\_ADDRESS, 0x43, i2cData, 6);

if(error!=0)

return;

xSum += ((i2cData[0] << 8) | i2cData[1]);

ySum += ((i2cData[2] << 8) | i2cData[3]);

zSum += ((i2cData[4] << 8) | i2cData[5]);

}

gyrXoffs = xSum / num;

gyrYoffs = ySum / num;

gyrZoffs = zSum / num;

Serial.println("Calibration result:");

Serial.print(gyrXoffs);

Serial.print(", ");

Serial.print(gyrYoffs);

Serial.print(", ");

Serial.println(gyrZoffs);

while(1);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void read\_sensor\_data()

{

uint8\_t i2cData[14];

uint8\_t error;

// read imu data

error = i2c\_read(MPU6050\_I2C\_ADDRESS, 0x3b, i2cData, 14);

if(error!=0)

return;

// assemble 16 bit sensor data

accX = ((i2cData[0] << 8) | i2cData[1]);

accY = ((i2cData[2] << 8) | i2cData[3]);

accZ = ((i2cData[4] << 8) | i2cData[5]);

gyrX = (((i2cData[8] << 8) | i2cData[9]) - gyrXoffs) / gSensitivity;

gyrY = (((i2cData[10] << 8) | i2cData[11]) - gyrYoffs) / gSensitivity;

gyrZ = (((i2cData[12] << 8) | i2cData[13]) - gyrZoffs) / gSensitivity;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// ================== I2C routines===================================

int i2c\_read(int addr, int start, uint8\_t \*buffer, int size)

{

int i, n, error;

Wire.beginTransmission(addr);

n = Wire.write(start);

if (n != 1)

return (-10);

n = Wire.endTransmission(false); // hold the I2C-bus

if (n != 0)

return (n);

// Third parameter is true: relase I2C-bus after data is read.

Wire.requestFrom(addr, size, true);

i = 0;

while(Wire.available() && i<size)

{

buffer[i++]=Wire.read();

}

if ( i != size)

return (-11);

return (0); // return : no error

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int i2c\_write(int addr, int start, const uint8\_t \*pData, int size)

{

int n, error;

Wire.beginTransmission(addr);

n = Wire.write(start); // write the start address

if (n != 1)

return (-20);

n = Wire.write(pData, size); // write data bytes

if (n != size)

return (-21);

error = Wire.endTransmission(true); // release the I2C-bus

if (error != 0)

return (error);

return (0); // return : no error

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int i2c\_write\_reg(int addr, int reg, uint8\_t data)

{

int error;

error = i2c\_write(addr, reg, &data, 1);

return (error);

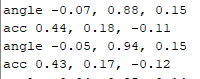
}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Results

Using IMU we could detect the orientation / inclination angle of the robot around its axis (which is Z Axis) and angular acceleration of the robot that will rotate around its self as shown from the results of IMU



To detect linear velocity or acclerarion we need a feedback from the actuator (DC Motor) to calculate exact robot linear position, linear velocity and linear acceleration. Since the IMU is caring about rotational coordinate and out robot is going on a plane not 3d Area