Nile University

OPT-08: Robots and Vision

Report #: (1)

Robt Orientation Detection Using IMU

Student Name:

Hedaya Rafaat

Mahmood Abdallah Saleh

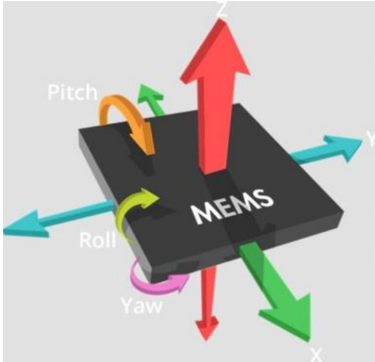
Menna Allah Soliman

Spring 2018

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
    - $angle = 0.98 * (angle + gyrData * dt) + 0.02 * (accData)$

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

```
angle -0.07, 0.88, 0.15
acc 0.44, 0.18, -0.11
angle -0.05, 0.94, 0.15
acc 0.43, 0.17, -0.12
```

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