TRƯỜNG ĐẠI HỌC BÁCH KHOA TP.HỒ CHÍ MINH

KHOA ĐIỆN-ĐIỆN TỬ

BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG

BÁO CÁO BÀI TẬP LỚN SỐ 2

MÔN ĐO LƯỜNG CÔNG NGHIỆP

ĐỀ TÀI: THIẾT KẾ MẠCH ĐO VÀ HIỂN THỊ GÓC CÓ KẾT HỢP BỘ LỌC NHIỄU SỬ DỤNG CẢM BIẾN MPU-9250

Giảng viên hướng dẫn: Nguyễn Đức Hoàng

Lớp: L04 – HK232

Thành viên	**MSSV**	**Công việc thực hiện**
Huỳnh Đỗ Đạt	2113126	Viết code giao diện
Hồ Nguyên Hoàng	2113395	Viết code đọc và calib MPU 9250

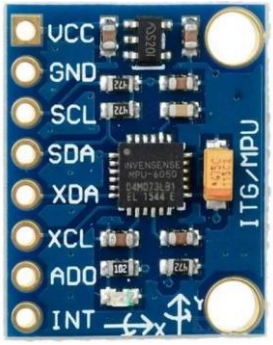
MỤC LỤC

I. Tổng quan 3
II. MPU 9250 4
III. Lọc nhiễu cho cảm biến 8
IV. Calib cảm biến MPU 9250 11
1. Calib Accelerometer 11
2. Calib Gyroscope 11
3. Calib Magnetometer 12
V. Các Frame truyền nhận dữ liệu 15
VI. GUI 19
VII. Thiết kế và thi công mạch đo 19
VIII. Kết quả 21
TÀI LIỆU THAM KHẢO 23

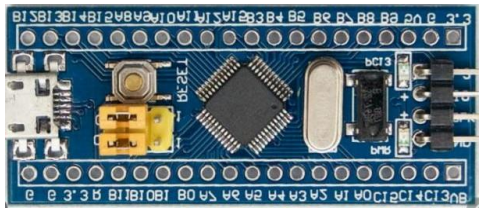
I. Tổng quan

Đề tài xây dựng thực tế một hệ thống thu thập dữ liệu cảm biến MPU 9250 (góc roll, góc pitch, góc yaw), hệ thống bao gồm:

- Cảm biến MPU 9250



- Vi điều khiển STM32F103



- Máy tính (sử dụng Visual Studio)

Mục tiêu của hệ thống là đo chính xác và đúng 3 góc roll, pitch và yaw của MPU 9250, cập nhật liên tục và nhanh, hiển thị giá trị đọc lên giao diện người dùng trên máy tính. Cấu trúc và phương án thực hiện hệ thống như hình sau:

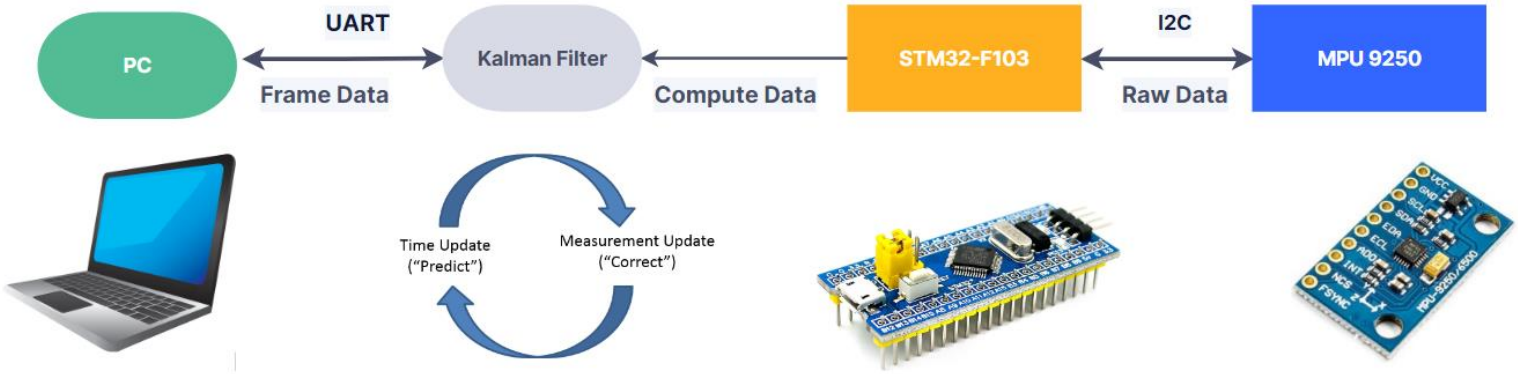


Figure 1:Cấu trúc và phương án thực hiện hệ thống

II. MPU 9250

1. Định nghĩa trục và phép quay

Trục của cảm biến và phép quay được định nghĩa dựa vào qui tắc bàn tay phải. Qui tắc như sau:

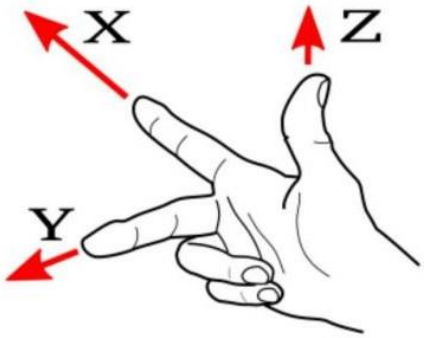


Figure 2: Qui tắc bàn tay phải

Từ đó, phép quay dương được định nghĩa là khi vật thể theo chiều kim đồng hồ quanh mỗi trục nhìn tử gốc tọa độ theo chiều dương.

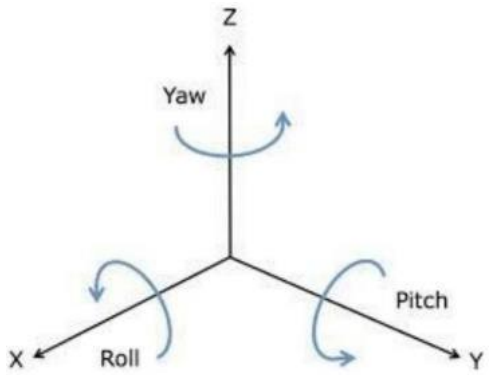


Figure 3: Định nghĩa góc dương

2. Gia tốc kế

Gia tốc kế rất chính xác trong thời gian dài, và các góc roll, pitch được tính chính xác khi vật thể đứng yên trên Trái đất. Tuy nhiên, khi cảm biến đang di chuyển, gia tốc chuyển động sẽ ảnh hưởng đến phép tính xoay. Nhưng có thể khắc phục vấn đề này bằng cách sử dụng bộ lọc thông thấp sau đầu ra của gia tốc kế, giá trị được lọc sau đó được sử dụng để tích hợp với cảm biến con quay hồi chuyển sẽ được đề cập trong phần [III Bộ lọc Kalman].

Config cảm biến gia tốc với tầm đo từ -2g tới +2g (g = 9.81)

```
// Set accelerometer configuration in ACCEL_CONFIG Register
// XA_ST=0,YA_ST=0,ZA_ST=0, FS_SEL=0 -> ▯ 2g
Data = 0x00;
HAL_I2C_Mem_Write(I2Cx, MPU6050_ADDR, ACCEL_CONFIG_REG, 1, &Data, 1, i2c_timeout);
```

Chuyển giá trị thô đọc được sang gia tốc g. Do dữ liệu đọc về được chứa trong 16 bit và FS là 2g nên phải chia cho $2^{15} / 2 = 16384$

```
DataStruct->Ax = DataStruct->Accel_X_RAW / 16384.0;
DataStruct->Ay = DataStruct->Accel_Y_RAW / 16384.0;
DataStruct->Az = DataStruct->Accel_Z_RAW / 16384.0;
```

Công thức tính góc roll, pitch từ gia tốc kế như sau:

$$\boldsymbol{R}_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix}$$

$$\boldsymbol{R}_y(\theta) = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix}$$

$$\boldsymbol{R}_z(\psi) = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\tan\Phi = \frac{a_y}{\sqrt{a_x^{\,2} + a_z^{\,2}}}, \tan\theta = \frac{-a_x}{a_z}$$

$$roll = \arctan(\Phi), pitch = \arctan(\theta)$$

```
double roll_sqrt = sqrt(
    DataStruct->Accel_X_RAW * DataStruct->Accel_X_RAW + DataStruct->Accel_Z_RAW * DataStruct->Accel_Z_RAW);
if (roll_sqrt != 0.0)
{
    roll = atan(DataStruct->Accel_Y_RAW / roll_sqrt) * RAD_TO_DEG;
}
else
{
    roll = 0.0;
}
double pitch = atan2(-DataStruct->Accel_X_RAW, DataStruct->Accel_Z_RAW) * RAD_TO_DEG;
```

3. Con quay hồi chuyển

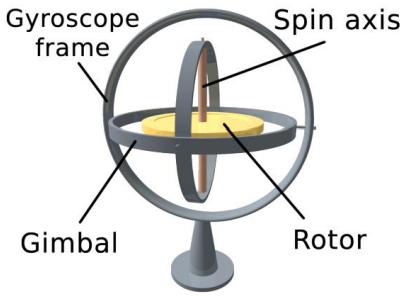


Figure 4: Cảm biến con quay hồi chuyển

Vấn đề của con quay hồi chuyển là góc tính được sẽ bị trôi theo thời gian, do công thức tính góc là một phép toán tích phân nên tại một thời điểm có sai số khi đo thì sai số đó sẽ tiếp tục được tích phân lên. Thêm vào đó, do quán tính, tốc độ của con quay hồi chuyển sẽ không trở về 0 khi đối tượng đứng yên. Tuy nhiên, con quay hồi chuyển rất chính xác khi đo trong thời gian ngắn. Vì vậy, có thể sử dụng bộ lọc thông cao cho con quay hồi chuyển.

Config cảm biến con quay hồi chuyển với tầm đo từ -250 độ/s đến +250 độ/s

```
// Set Gyroscopic configuration in GYRO_CONFIG Register
// XG_ST=0,YG_ST=0,ZG_ST=0, FS_SEL=0 -> ± 250 °/s
Data = 0x00;
HAL_I2C_Mem_Write(I2Cx, MPU6050_ADDR, GYRO_CONFIG_REG, 1, &Data, 1, i2c_timeout);
```

Chuyển giá trị thô đọc được sang dps (độ/s). Do dữ liệu đọc về được chứa trong 16 bit và FS là 250 nên phải chia cho $\frac{2^{15}}{250} = 131$

```
DataStruct->Gx = DataStruct->Gyro_X_RAW / 131.0;
DataStruct->Gy = DataStruct->Gyro_Y_RAW / 131.0;
DataStruct->Gz = DataStruct->Gyro_Z_RAW / 131.0;
```

Công thức tính góc roll, pitch từ con quay hồi chuyển:

$$roll = \omega_x \Delta t,\ pitch = \omega_y \Delta t$$

4. Từ kế

- Hướng của từ trường Trái đất đo được được sử dụng làm la bàn (3D) để xác định hướng Bắc (hướng yaw). Cảm biến này thường được ứng dụng để đo góc yaw. Trên thực tế ta có thể tính toán góc yaw bằng gyroscope, nhưng nếu ta kết hợp thêm cảm biến từ trường nó sẽ cho kết quả chính xác hơn, lấp đi nhược điểm khi dùng gyroscope.
- Khi từ kế đặt trên một bề mặt phẳng, không có trường nhiễu khác xung quanh thì góc lệch được tính như sau:

$$\psi = tan^{-1}\left(\frac{m_y}{m_x}\right)$$

Tuy nhiên công thức này không chính xác nếu cảm biến bị nghiêng, khi đó ta sẽ dựa vào góc roll, pitch đã biết theo công thức:

$$M_x = m_x \cdot \cos(\phi) + m_z \cdot \sin(\phi)$$

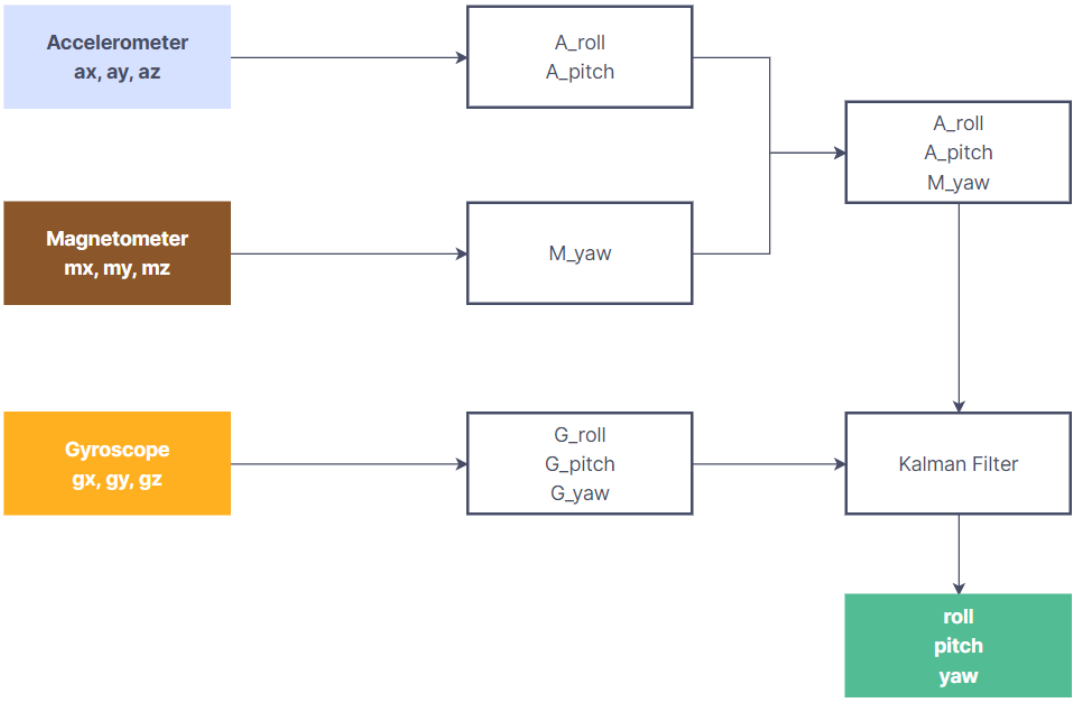
$$M_y = m_x \cdot \sin(\theta) \cdot \sin(\phi) + m_y \cdot \cos(\theta) - m_z \cdot \sin(\theta) \cdot \cos(\phi)$$

$$\psi = \tan^{-1}\frac{M_y}{M_x}$$

```
float Mx = mx * cos(pitch) + mz * sin(pitch);
float My = mx * sin(roll) * sin(pitch) + my * cos(roll) - mz * sin(roll) * cos(pitch);
float yaw = atan2(-My,Mx);
```

III. Lọc nhiễu cho cảm biến

- Bộ lọc Kalman là một thuật toán lọc nhiễu ra khỏi thông tin, có khả năng kết hợp nhiều tín hiệu với độ không chắc chắn khác nhau để cho ra kết quả ước lượng tốt nhất, trạng thái ước lượng hiện tại chỉ phụ thuộc vào trạng thái trước đó 1 chu kỳ tính toán nên tốc độ tính toán của bộ lọc Kalman khá nhanh nên được sử dụng rộng rãi trong nhiều lĩnh vực điều khiển.
- Bộ lọc Kalman là một thuật toán tính đến một loại các phép đo theo thời gian, trong đề tài này chúng là phép đo từ 3 cảm biến gia tốc kế (Accelerometer), từ kế (Magnetometer) và con quay hồi chuyển (Gyroscope) với sai số hệ thống (model noise) và sai số ngẫu nhiên (measurement noise).



Bộ lọc Kalman gồm 2 bước chính: Cập nhật phép đo (update) và cập nhật thời gian (predict).

Trong đề tài này 2 góc roll, pitch là độc lập, do đó mỗi góc có thể được cập nhật riêng theo công thức sau:

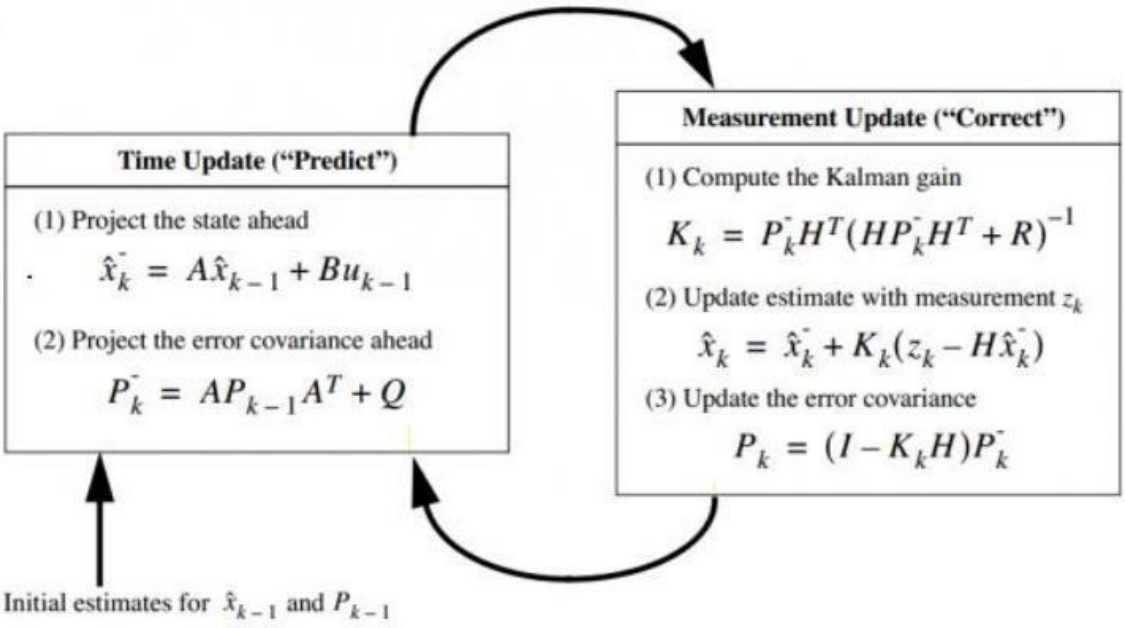


Figure 5: Qui trình tính toán của bộ lọc Kalman

Theo Kristian Sloth Lauszus [4] các ma trận tính toán là:

$$x_k = \begin{bmatrix} \theta \\ \theta_b \end{bmatrix}, \quad A = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \quad u_{k-1} = \theta', \quad I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$Q = \begin{bmatrix} Q_\theta & 0 \\ 0 & Q'_{\theta b} \end{bmatrix} \Delta t, \quad H = [1 \quad 0], \quad P = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}, \text{ and } \quad K = \begin{bmatrix} K_0 \\ K_1 \end{bmatrix}$$

Trong đó:

x_k là biến trạng thái gồm hai thành phần: góc và vận tốc góc lệch

A là ma trận chuyển đổi trạng thái được nhân với trạng thái trước đó $x_{k|k-1}$

B là ma trận điều khiển đầu vào được nhân với vector điều khiển trước u_{k-1}

H là ma trận quan sát, ánh xạ không gian trạng thái thực vào không gian quan sát

Q là hiệp phương sai của nhiễu quá trình (sai số hệ thống)

R là hiệp phương sai của nhiễu phép đo (sai số ngẫu nhiên)

P là ma trận hiệp phương sai ước lượng

K là hệ số Kalman

Bước Time Update (predict)

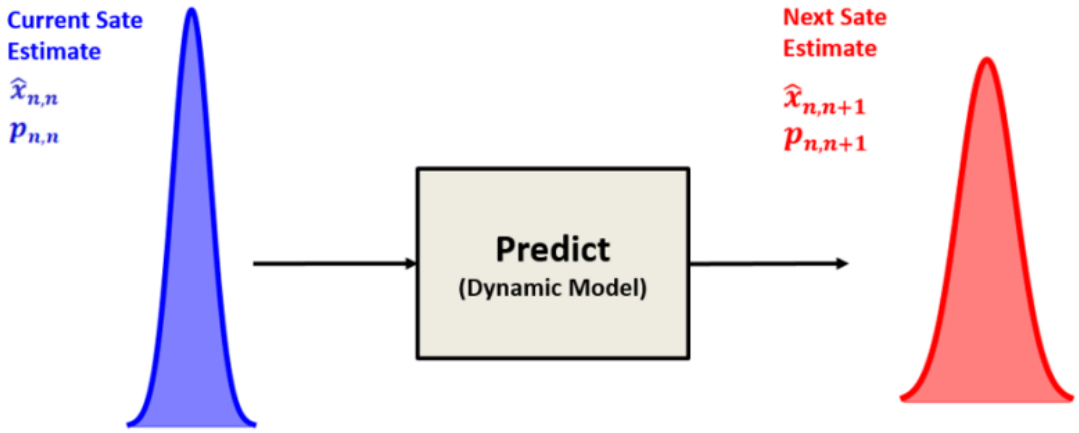


Figure 6: Time update

Bước Measurement Update (correct)

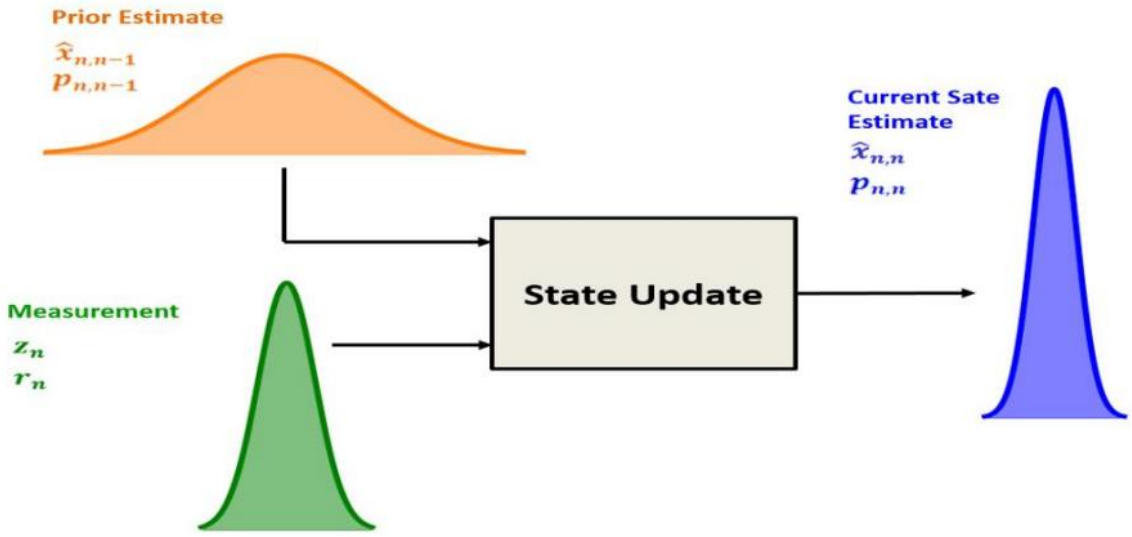


Figure 7: Measurement Update

Ma trận hiệp phương sai nhiễu quá trình Q trong trường hợp này là nhiễu gia tốc kế và con quay hồi chuyển. Ngoài ra, khi ma trận hiệp phương sai nhiễu phép đo R có giá trị cao, bộ lọc sẽ phản ứng chậm khi phép đo mới có độ không chắc chắn cao, mặc khác nếu R quá nhỏ thì kết quả đầu ra sẽ thiếu chính xác khi ta quá tinh tưởng vào phép đo mới của gia tốc kế. Dựa vào kinh nghiệm qua các lần thử nghiệm nhóm chọn các thông số như sau:

```
Kalman_t KalmanX = {
    .Q_angle = 0.001f,
    .Q_bias = 0.003f,
    .R_measure = 0.03f};

Kalman_t KalmanY = {
    .Q_angle = 0.001f,
    .Q_bias = 0.003f,
    .R_measure = 0.03f,
};
```

IV. Calib cảm biến MPU 9250

1. Calib Accelerometer

Đặt cảm biến theo hướng thẳng đứng lần lượt theo 3 trục x, y và z. Khi được đặt theo trục nào thì gia tốc của trục đó sẽ bằng gia tốc trọng trường và gia tốc của 2 trục còn lại sẽ bằng 0. Ví dụ, khi đặt cảm biến hướng thẳng đứng theo trục z thì az = g và ax = ay =0.

Sau đó, ta quan sát giá trị của gia tốc của trục đó theo 2 hướng lên và xuống, nếu 2 giá trị không bằng nhau ta thay đổi giá trị bằng cách lấy trung bình cộng 2 giá trị này.

Gía trị offset khi khảo sát:

```
#define AX_OFFSET       0.03
#define AY_OFFSET       -0.15
#define AZ_OFFSET       -1.25
```

2. Calib Gyroscope

Để calib được gyroscope trong thực tế khá khó vì phụ thuộc nhiều vào nhiệt độ (mỗi vùng làm việc sẽ có khoảng giá trị khác nhau), trong đa số trường hợp ta thực hiện calib bằng cách giữ cảm biến đứng yên trong khoảng thời gian (5s đến 10s) sau đó lấy giá trị tốc độ góc trung bình ở 3 trục, ta được giá trị offset để bù vào tương ứng.

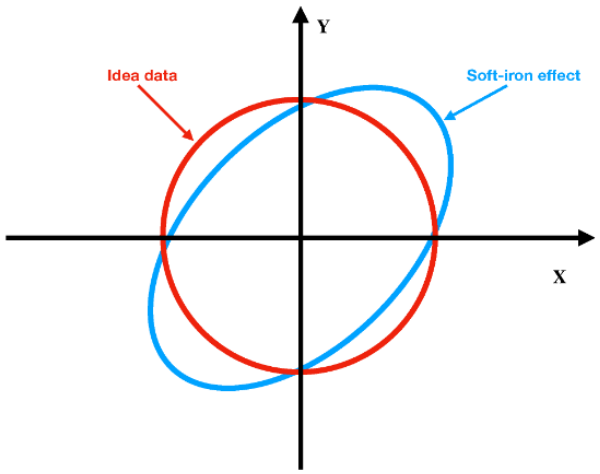
Gía trị offset khi khảo sát:

```
#define GX_OFFSET       0.0355
#define GY_OFFSET       -0.031
#define GZ_OFFSET       -0.041
```

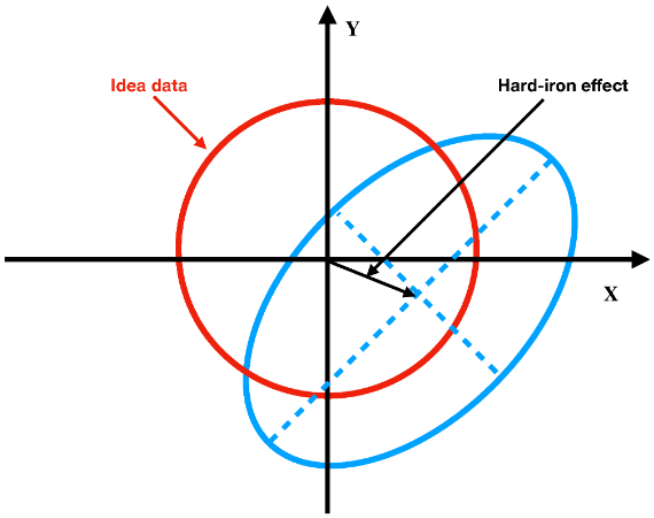
3. Calib Magnetometer

Sai số Soft Iron và Hard Iron là hai loại sai số thường gặp trong các hệ thống đo từ trường. Chúng xuất phát từ các hiện tượng từ học khác nhau trong vật liệu và có ảnh hưởng đến độ chính xác của cảm biến từ trường.

Sai số Soft Iron: Sai số này xuất phát từ sự biến dạng của từ trường do các vật liệu dễ từ (như sắt mềm). Vật liệu này dễ bị từ hoá và tạo ra các từ trường riêng, làm biến dạng từ trường tổng thể. Sai số Soft Iron thường gây ra sự biến dạng không đồng đều trên toàn bộ phạm vi đo.



Sai số Hard Iron: Sai số này xuất phát từ sự tương tác giữa từ trường của cảm biến và các vật liệu từ cứng (như nam châm vĩnh cửu). Các vật liệu này tạo ra một từ trường cố định riêng, gây ra một sai số cố định không phụ thuộc vào hướng của cảm biến. Sai số Hard Iron thường gây ra một sai số cố định.



Để calib được từ trường ta sử dụng biểu thức sau:

$$\boldsymbol{M_{calib} = A(M - b)}$$

Với:

M là giá trị từ trường đo được từ cảm biến;

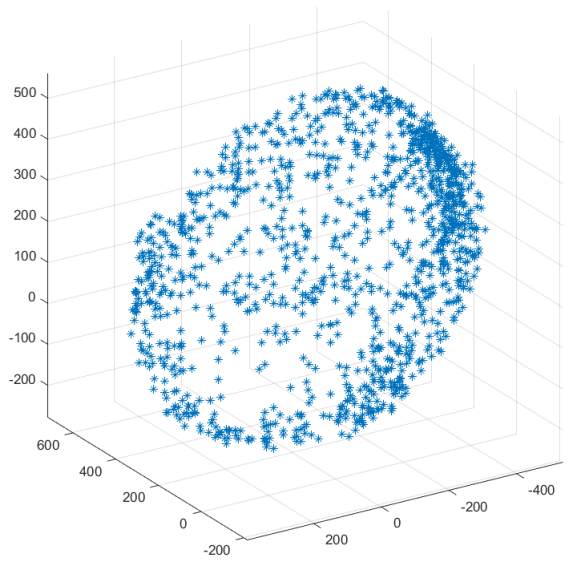
A được mô tả bằng một ma trận 3x3, là ma trận kết hợp các scale factor, sự lệch trục và hiệu ứng soft iron;

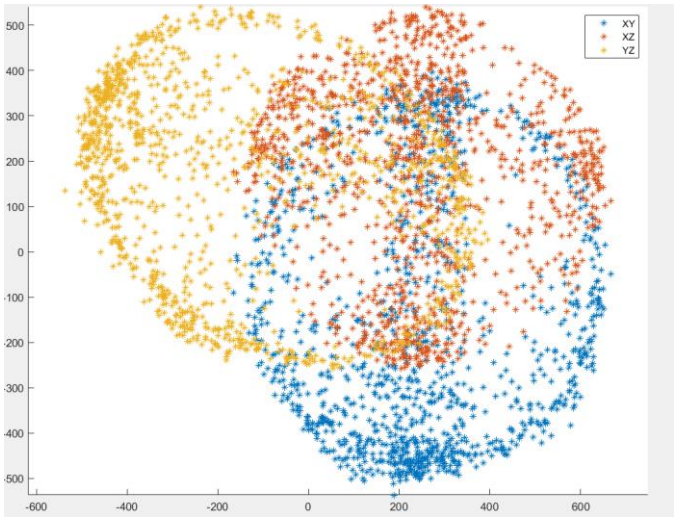
b mô tả bằng một vector 3 chiều là sự kết hợp của các vector bias bao gồm hard iron.

Để lấy được các mẫu giá trị đo cho việc calib, xoay IMU 360 độ quanh tất cả trục là đường thẳng đi qua tâm để có thể mô phỏng được nhiễu từ trường. Các mẫu giá trị từ trường trên ba trục được đưa vào phần mềm Matlab để thực hiện calib. Khi dùng Matlab vẽ các điểm (mx, my, mz) này trong không gian 3D, ta thu được mặt ellipsoid.

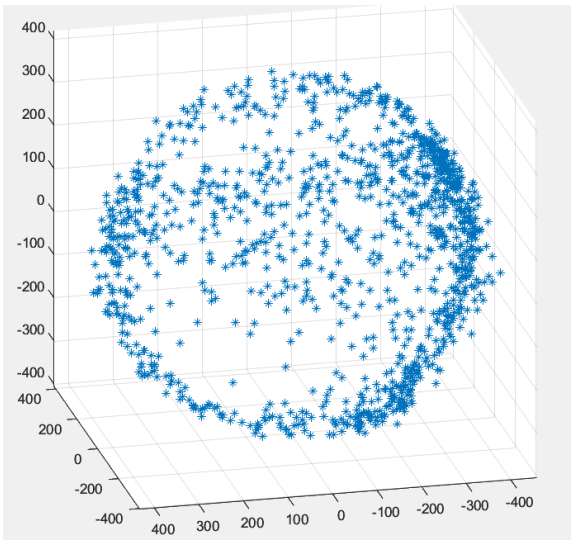
Sau khi tìm được thông số mặt ellipsoid bằng phương pháp ellipsoid fit với code Matlab, ta có thể thu được ma trận ***A***, ***b*** và dùng nó cho việc hiệu chỉnh magnetometer trong chương trình firmware đọc giá trị cảm biến

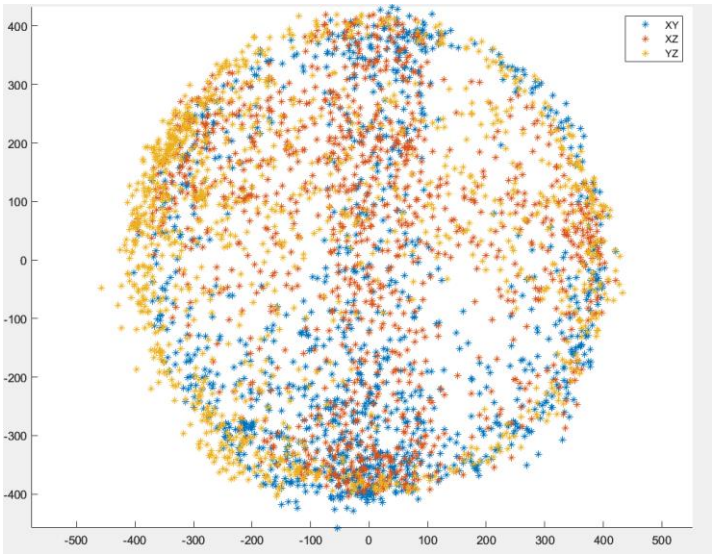
Từ trường trước khi calib (3d và 2d trên 3 mặt phẳng)





Từ trường sau khi calib bằng matlab (3d và 2d trên 3 mặt phẳng)





V. Các Frame truyền nhận dữ liệu

Frame dữ liệu STM32F1 nhận được: 26 byte gồm dữ liệu của ba góc Roll, Pitch, Yaw

Số byte	1	1	6	1	1	6	1	1	6	1	1
	\n	±	roll		±	pitch		±	yaw		\r

Code xử lí bằng C#:

- Các lựa chọn tốc độ Baudrate truyền:

```
public Form1()
{
    InitializeComponent();
    string[] Baudrate = { "1200", "2400", "4800", "9600", "115200" };
    cboBaudRate.Items.AddRange(Baudrate);
    Control.CheckForIllegalCrossThreadCalls = false;
}
```

- Khởi tạo và thiết lập vẽ biểu đồ 1 (tương ứng với góc Roll), ở các biểu đồ tương ứng với góc Pitch, Yaw được thực hiện tương tự:

```
private void Form1_Load(object sender, EventArgs e)
{
    cboComPort.DataSource = SerialPort.GetPortNames(); //detect automatically
    cboBaudRate.Text = "9600"; //default
    GraphPane myPanne = zedGraphControl1.GraphPane; //khoi tao bieu do 1
```

```
myPanne.Title.Text = "Plotted graph of roll";
myPanne.YAxis.Title.Text = "Roll angle (degree)";
myPanne.XAxis.Title.Text = "Time (s)";
```

```
RollingPointPairList list = new RollingPointPairList(500000); // 500000 points
```

```
LineItem line = myPanne.AddCurve("angle", list, Color.Red, SymbolType.Diamond); //chu thich
```

```
myPanne.XAxis.Scale.Min = 0;
myPanne.XAxis.Scale.Max = 40;
myPanne.XAxis.Scale.MinorStep = 0.05; // vach chia nho
myPanne.XAxis.Scale.MajorStep = 0.1; // vach chia to x
```

```
zedGraphControl1.AxisChange();
```

- Thực hiện kết nối với cổng COM để nhận dữ liệu từ cảm biến:

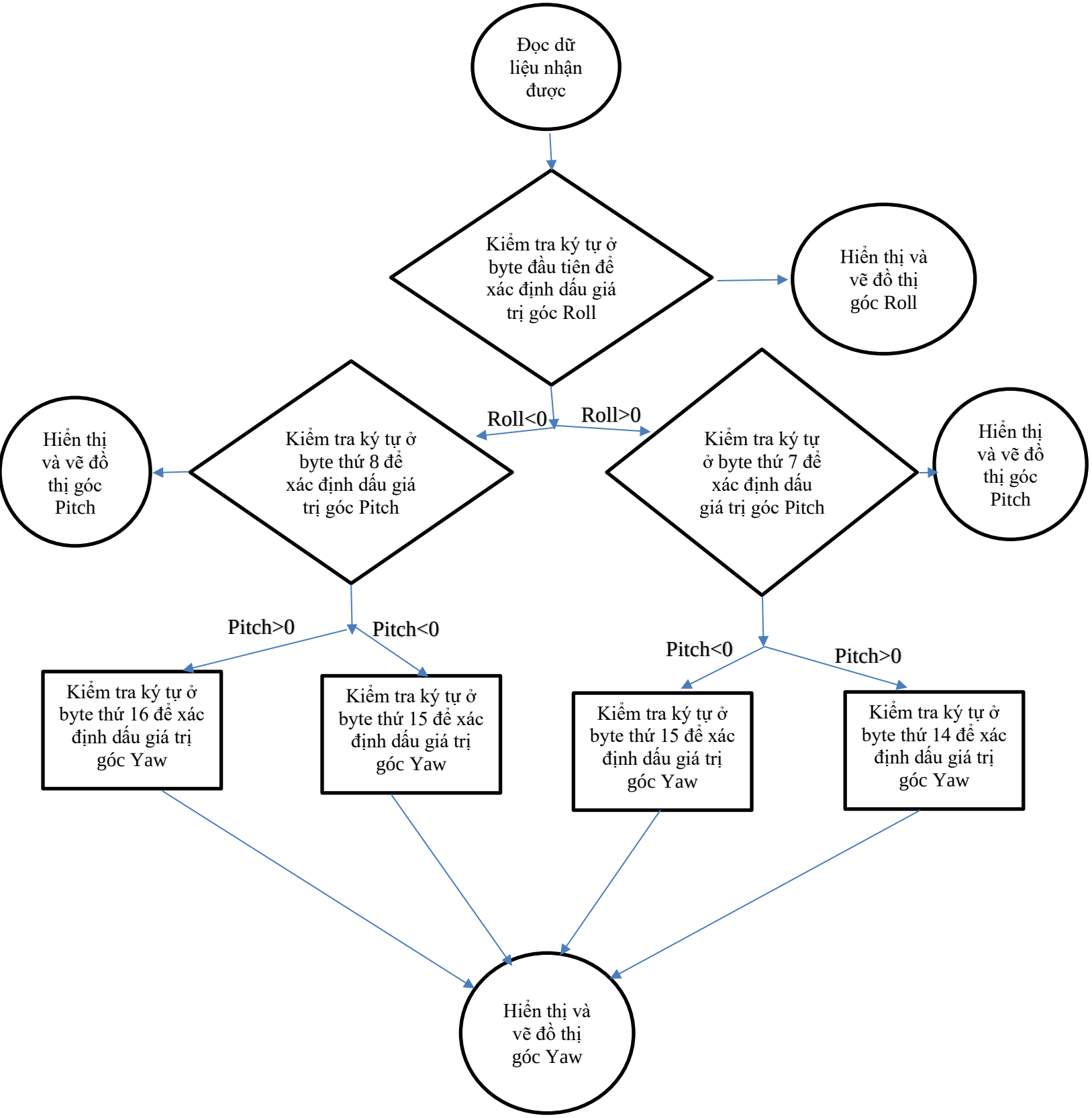
```
private void butConnect_Click(object sender, EventArgs e)
{
    if (!serCOM.IsOpen)
    {
        butConnect.Text = "Disconnected";
        serCOM.PortName = cboComPort.Text;
        serCOM.BaudRate = Convert.ToInt32(cboBaudRate.Text);
        serCOM.Open();
    }
    else
    {
        butConnect.Text = "Connected";
        serCOM.Close();
    }
}
```

- Thực hiện đọc và hiển thị dữ liệu :

```
private void serCOM_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    string revdata = serCOM.ReadExisting();
    txtAllData.Text = revdata;

    revdata = revdata.Trim();
    int len = revdata.Length;
```

- Dựa vào frame truyền dữ liệu nhận được, thực hiện kiểm tra dữ liệu, bóc tách và hiển thị giá trị 3 góc Roll, Pitch, Yaw theo giải thuật sau:


Đọc dữ liệu nhận được
Kiểm tra ký tự ở byte đầu tiên để xác định dấu giá trị góc Roll
Hiển thị và vẽ đồ thị góc Roll
Roll<0
Roll>0
Kiểm tra ký tự ở byte thứ 8 để xác định dấu giá trị góc Pitch
Hiển thị và vẽ đồ thị góc Pitch
Kiểm tra ký tự ở byte thứ 7 để xác định dấu giá trị góc Pitch
Hiển thị và vẽ đồ thị góc Pitch
Pitch>0
Pitch<0
Pitch<0
Pitch>0
Kiểm tra ký tự ở byte thứ 16 để xác định dấu giá trị góc Yaw
Kiểm tra ký tự ở byte thứ 15 để xác định dấu giá trị góc Yaw
Kiểm tra ký tự ở byte thứ 15 để xác định dấu giá trị góc Yaw
Kiểm tra ký tự ở byte thứ 14 để xác định dấu giá trị góc Yaw
Hiển thị và vẽ đồ thị góc Yaw

- Hàm vẽ biểu đồ 1 (tương ứng với góc Roll), ở các biểu đồ tương ứng với góc Pitch, Yaw được thực hiện tương tự:

```
double tong1 = 0;
```

```
public void draw(double line)
{
    LineItem duongline = zedGraphControl1.GraphPane.CurveList[0] as LineItem;
    if (duongline == null)
    {
        return;
    }
    IPointListEdit list = duongline.Points as IPointListEdit;
    if (list == null)
    {
        return;
    }

    list.Add(tong1, line);
    zedGraphControl1.AxisChange();
    zedGraphControl1.Invalidate();
    tong1 += 0.1; //1 step x = vach chia
}
```

- Khi đang quan sát các biểu đồ, ta có thể nhấn nút “Save Data” để lưu lại dữ liệu tức thời tại thời điểm đó dưới dạng text:

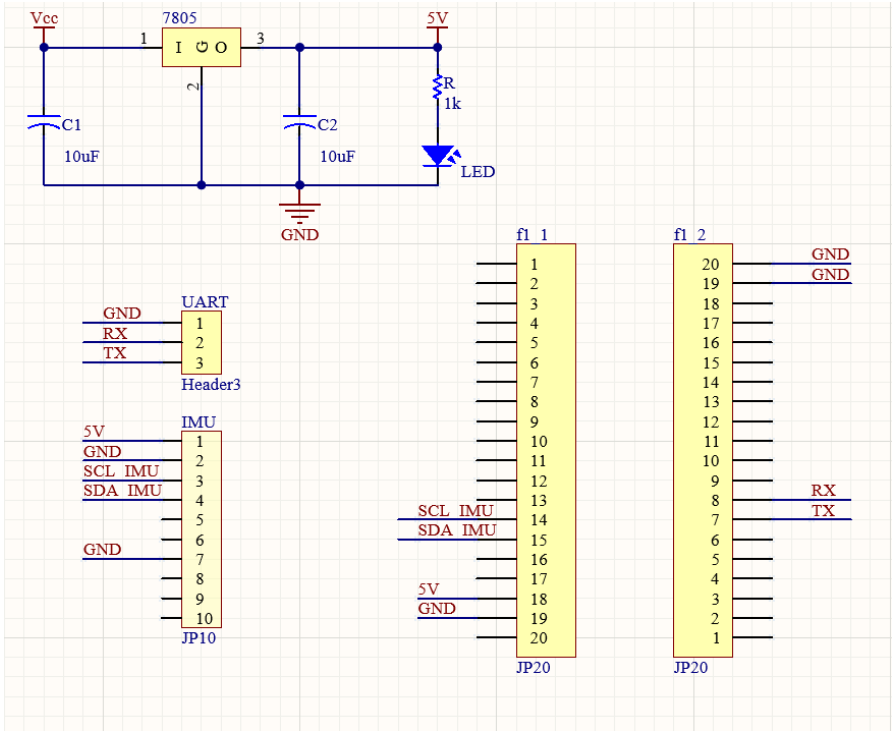
```
private void butSaveData_Click(object sender, EventArgs e)
{
    TextWriter text1 = new StreamWriter("G:\\Data.txt");
    text1.WriteLine(txtAllData);
    text1.Close();
}
```

VI. GUI

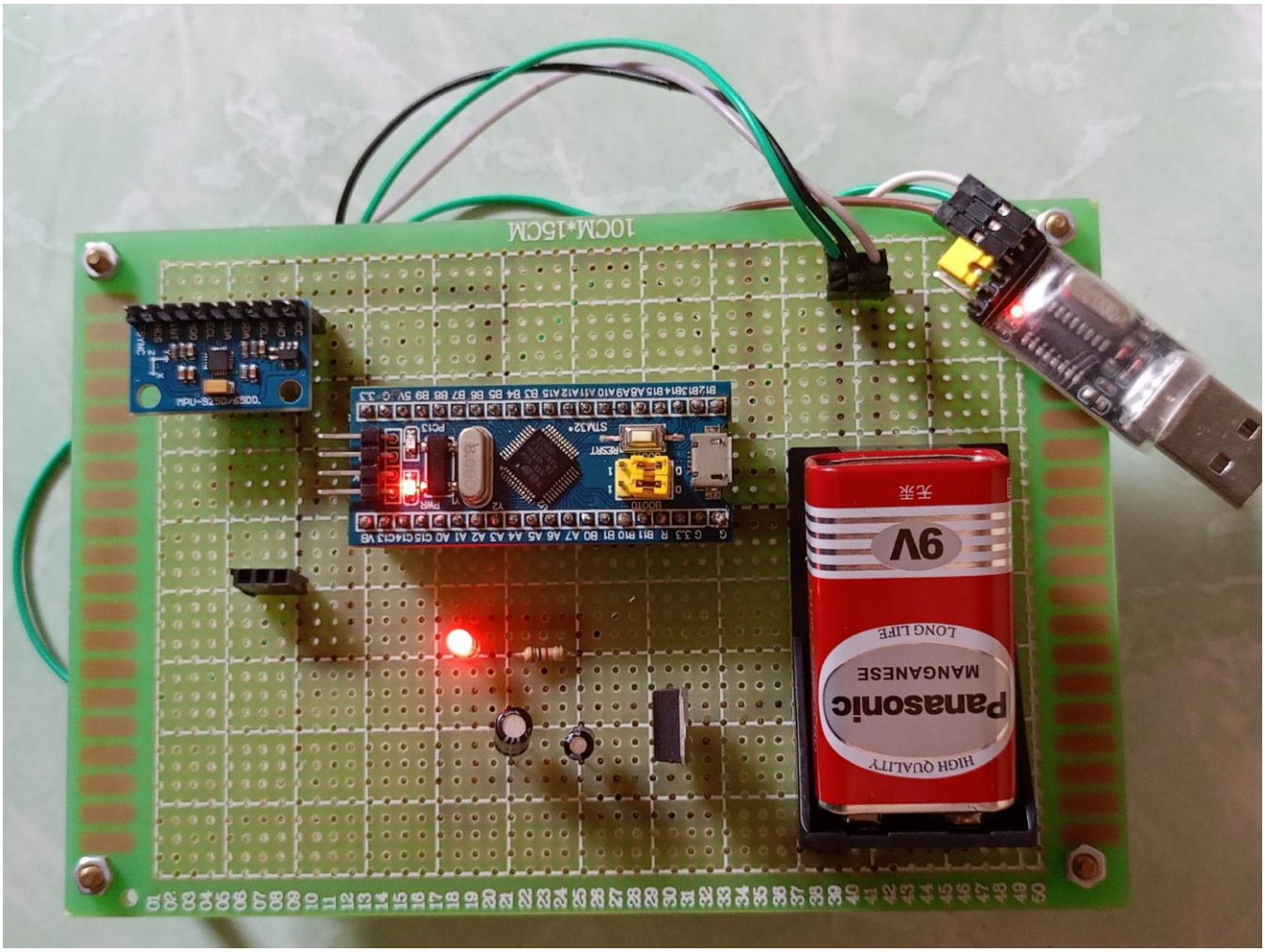
Figure 10: Giao diện hiển thị

VII. Thiết kế và thi công mạch đo

- **Sơ đồ nguyên lý mạch đo**

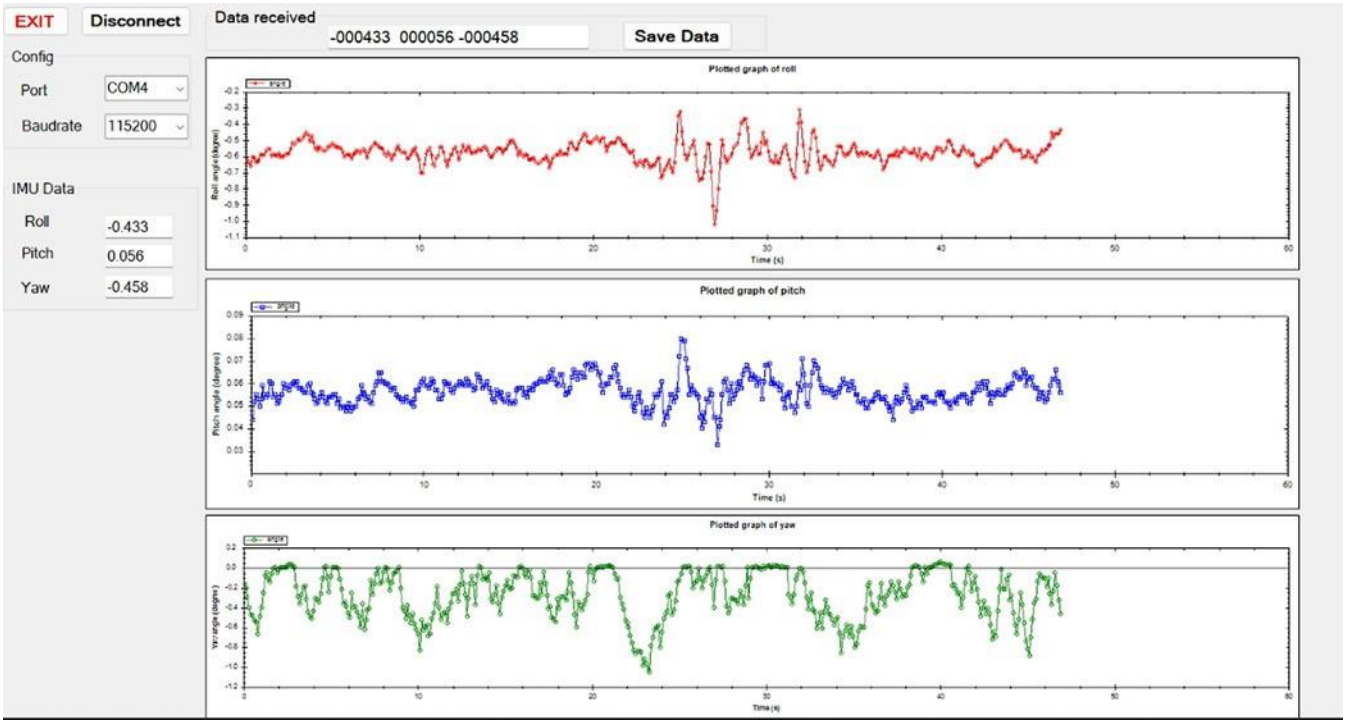


- **Mạch thực tế**



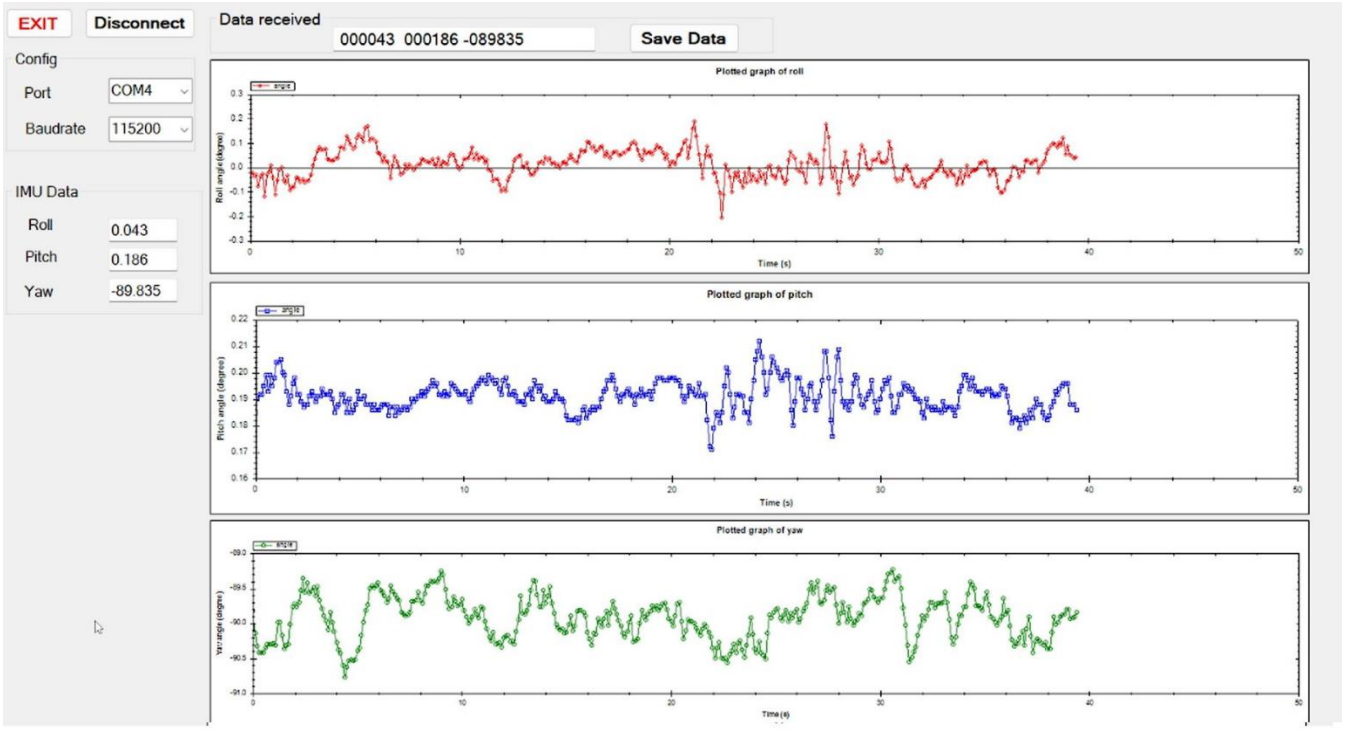
VIII. Kết quả

- **Đặt cảm biến ở gốc (ban đầu)**



Nhận xét: Số liệu thu được khá chính xác. Sai số góc roll, pitch, yaw lần lượt nằm trong khoảng $\pm\ 0.6^0$, $\pm\ 0.06^0$, $-\ 0.8^0$. Nguyên nhân sai số chủ yếu là do ảnh hưởng của quá trình lắp vị trí cảm biến đo, nhiễu từ trường, nhiệt độ thay đổi…. ở mỗi lần đo nên không thể tránh khỏi sai số.

- **Xoay cảm biến 90^0 so với góc ban đầu**



Nhận xét: Số liệu thu được khá chính xác, sai số góc roll, pitch, yaw lần lượt nằm trong khoảng $\pm 0.1^0$, $\pm 0.2^0$ và $\pm 0.5^0$. Nguyên nhân sai số chủ yếu là do ảnh hưởng của quá trình lắp vị trí cảm biến đo, nhiễu từ trường, nhiệt độ thay đổi…. ở mỗi lần đo nên không thể tránh khỏi sai số.

TÀI LIỆU THAM KHẢO

[1] Nguyễn Đức Hoàng. *Bài giảng Đo lường Công nghiệp.* Đại học Bách Khoa TP.HCM. 2016

[2] Nguyễn Đức Hoàng. *Tài liệu hướng dẫn thí nghiệm Đo lường Công nghiệp.* Đại học Bách Khoa TP.HCM. 2016.

[3] MarkPedley, "Tilt Sensing Using a Three-Axis Accelerometer," vol. AN3461, pp. 5-10, 03 2013.

[4] K. S. Lauszus, "A practical approach to Kalman filter and how to implement it," 10 9 2012. [Online]. Available: http://blog.tkjelectronics.dk/2012/09/a-practical approach-to-kalmanfilter-and-how-to-implement-it/. [Accessed 2017].