TRƯỜNG ĐẠI HỌC BÁCH KHOA TP.HỒ CHÍ MINH

KHOA ĐIỆN-ĐIỆN TỬ

BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG

BÁO CÁO BÀI TẬP LỚN MỞ RỘNG

MÔN ĐO LƯỜNG CÔNG NGHIỆP

ĐỀ TÀI: ĐỌC CẢM BIẾN IMU 6050 LỌC NHIỄU BẰNG BỘ LỌC KALMAN

Giảng viên hướng dẫn: Trần Hoàng Khôi Nguyên

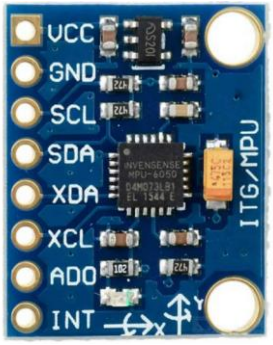
Thành viên	MSSV	Công việc thực hiện
Dương Ngọc Hoàn	2010020	Viết code giao diện
Nguyễn Võ Hồng Mỹ Hiền	2010260	Viết code đọc IMU 6050

Mục lục

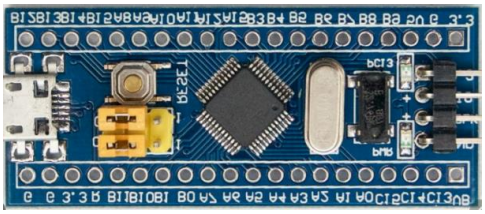
I. Tổng quan

Đề tài xây dựng thực tế một hệ thống thu thập dữ liệu cảm biến IMU 6050 (góc roll, góc pitch), hệ thống bao gồm:

- Cảm biến IMU 6050



- Vi điều khiển STM32F103



- Bàn xoay trong phòng thí nghiệm 207B3 (encoder và STM32F4)
- Máy tính (sử dụng Visual Studio)

Mục tiêu của hệ thống là đo chính xác và đúng góc roll và góc pitch của imu 6050, 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:

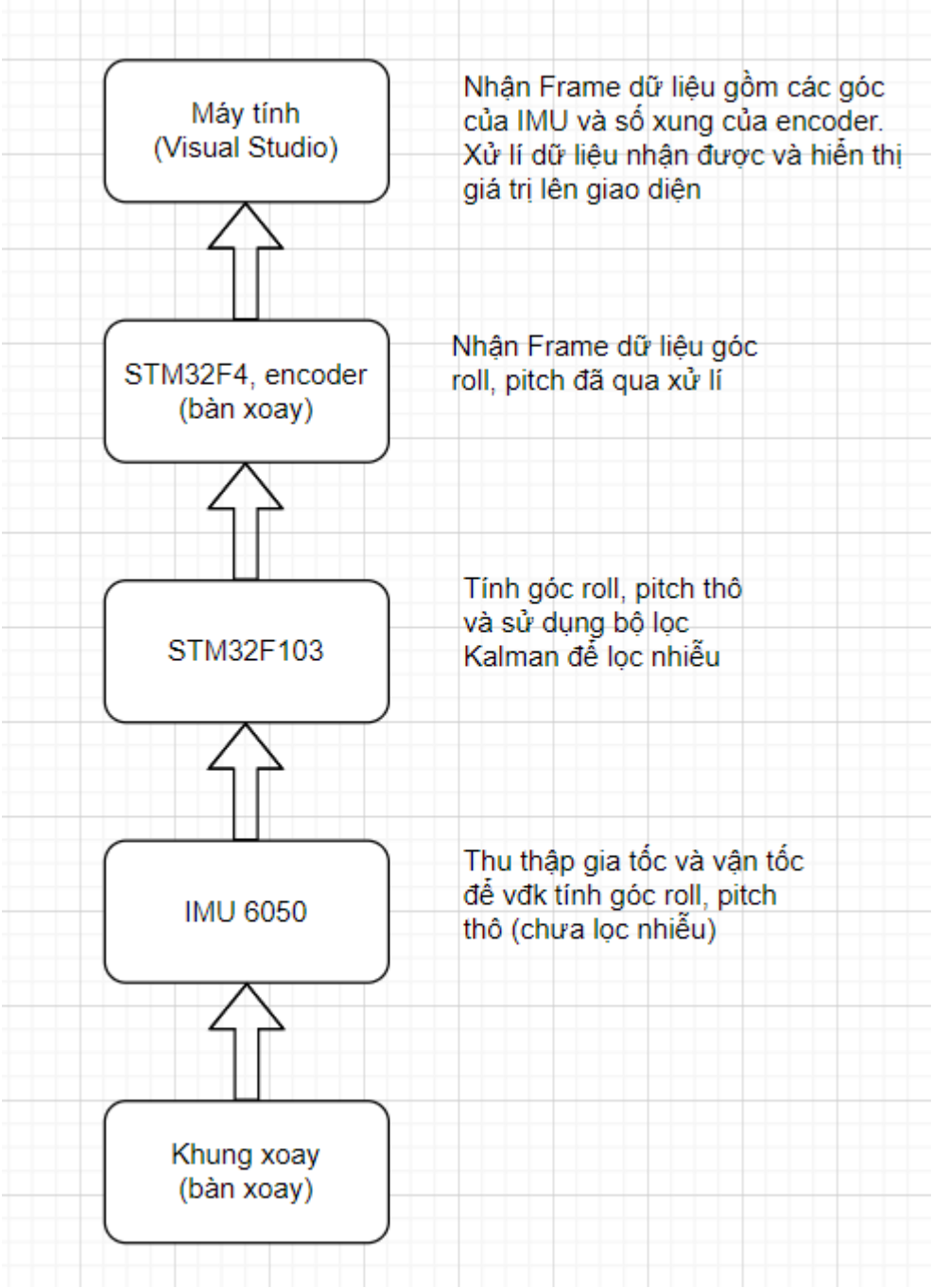


Figure 1:Cấu trúc và phương án thực hiện hệ thống

II. IMU 6050

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:

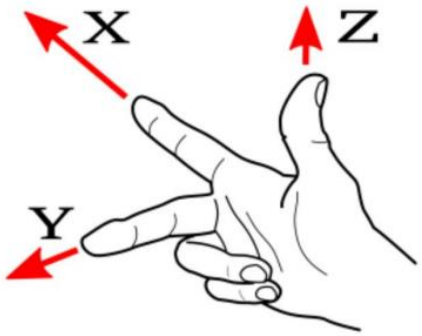


Figure 2: Qui tắc bàn tay phải

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.

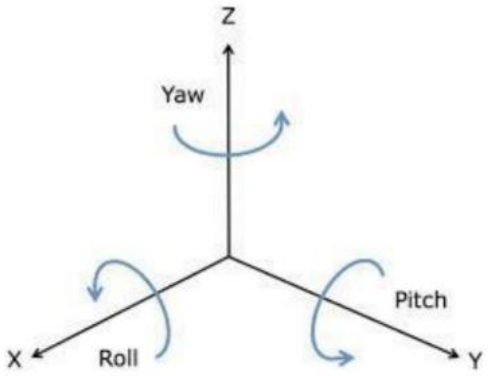


Figure 3: Định nghĩa góc dương

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 $\frac{2^{15}}{2} = 16384$

```
/*** convert the RAW values into acceleration in 'g'
     we have to divide according to the Full scale value set in FS_SEL
     I have configured FS_SEL = 0. So I am dividing by 16384.0
     for more details check ACCEL_CONFIG Register              ****/

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

Công thức tính góc roll, pitch từ gia tốc kế tham khảo bài báo số AN3461 từ Freescale Semiconductor [1] 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

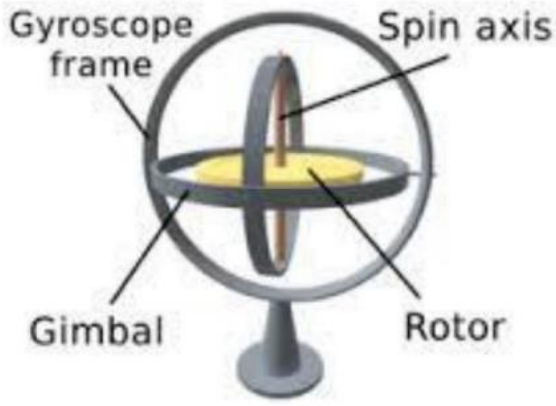


Figure 4: Cảm biến con quay hồi chuyển

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 $\frac{2^{15}}{250} = 131$

```
/*** convert the RAW values into dps (°/s)
     we have to divide according to the Full scale value set in FS_SEL
     I have configured FS_SEL = 0. So I am dividing by 131.0
     for more details check GYRO_CONFIG Register                  ****/

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

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

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

III. Lọc nhiễu cho cảm biến

1. Complimentary Filter

Như tên gọi của nó, về cơ bản nó tận dụng lợi thế của từng cảm biến và bù đắp những nhược điểm mà cảm biến khác mắc phải. Trong đề tài này, như đã đề cập ở phần trên, gia tốc kế hoạt động tốt nhất ở tần số thấp trong khi con quay hồi chuyển hoạt động tốt nhất tần số cao. Bộ lọc bù rất đơn giản và dễ sử dụng, nó chứa một giá trị cố định cho trọng số của thông thấp và thông cao. Trong thời gian ngắn thì sử dụng dữ liệu từ con quay hồi chuyển, còn trong thời gian dài sử dụng dữ liệu từ gia tốc kế. Nó có công thức như sau:

$$CF\_angle = HP\_weight * (CF\_angle + gyro\_data * \Delta t) + LP\_weight * acc\_data$$

Trong đó HP_weight +LP_weight = 1

Do hai trọng số này được từ đầu nên bộ lọc bù sẽ cho kết quả bị trôi trong thời gian dài và sự thiếu chính xác trong đo đạc. Bộ lọc Kalman sẽ giải quyết vấn đề này.

2. Kalman Filter

Không giống như bộ lọc bù, nó sử dụng hệ số của thông thấp và thông cao cố định. 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 2 cảm biến gia tốc kế và con quay hồi chuyển 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:

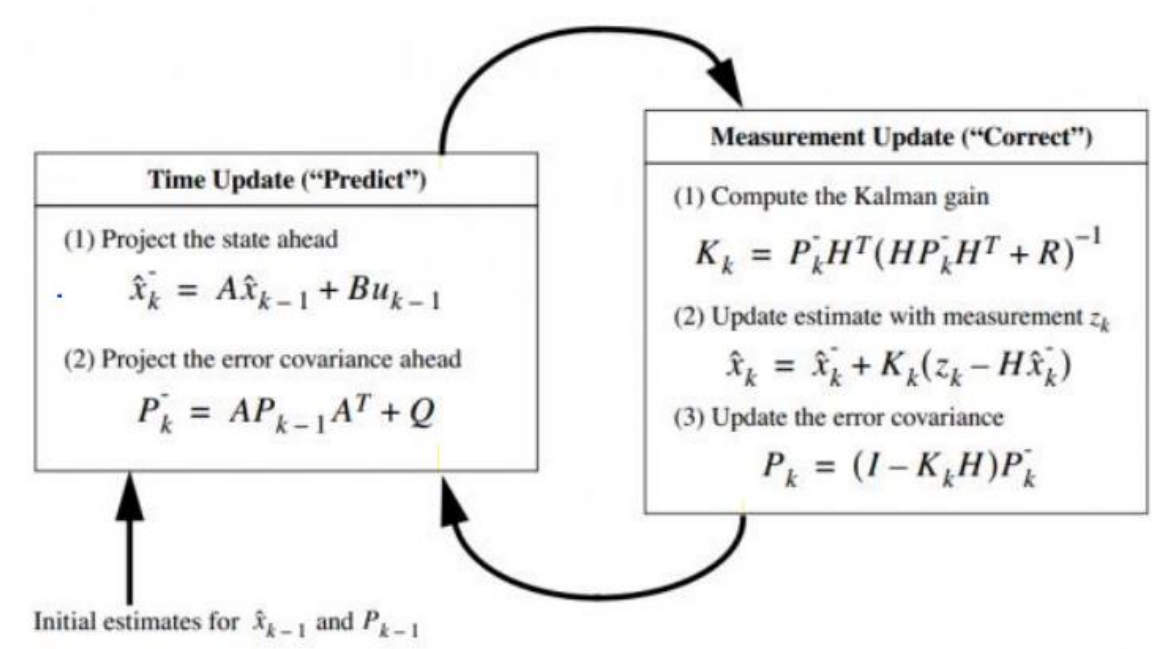


Figure 5: Qui trình tính toán của bộ lọc Kalman

Theo Kristian Sloth Lauszus [2] 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)

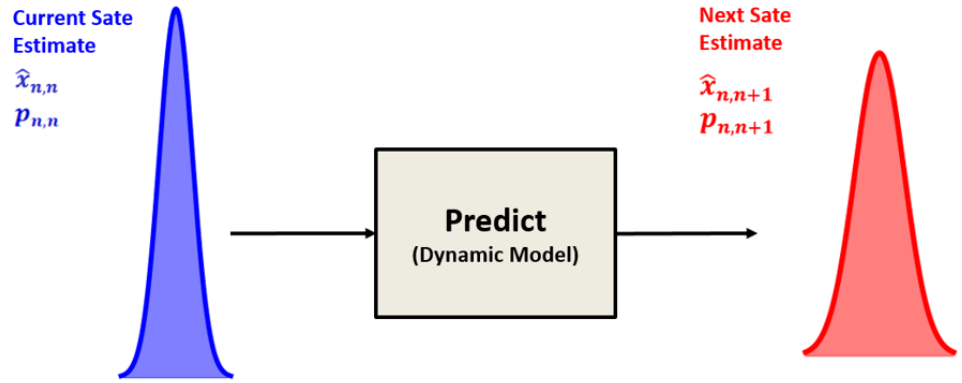


Figure 6: Time update

- Bước 1:

$$X_{k|k-1} = A\, x_{k-1|k-1} + B\theta'$$

$$\begin{bmatrix} \theta \\ \theta_b \end{bmatrix}_{k\,|\,k-1} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \theta_b \end{bmatrix}_{k-1|k-1} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \theta'$$

$$\begin{bmatrix} \theta \\ \theta_b \end{bmatrix}_{k|k-1} = \begin{bmatrix} \theta + \Delta t\,(\theta' - \theta_b) \\ \theta_b \end{bmatrix}_{k-1|k-1}$$

Có thể thấy, góc mới sẽ được cập nhật bằng góc cũ cộng với tốc độ góc nhân với thời gian

```
double rate = newRate - Kalman->bias;
Kalman->angle += dt * rate;
```

- Bước 2:

$$P_{k|k-1} = A\, P_{k-1|k-1}\, A^T + Q_k$$

$$\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k-1|k-1} \begin{bmatrix} 1 & 0 \\ -\Delta t & 1 \end{bmatrix} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q'_{\theta b} \end{bmatrix} \Delta t$$

$$\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} = \begin{bmatrix} P_{00} + \Delta t(\Delta t P_{11} - P_{01} - P_{10} + Q_\theta) & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} + Q'_{\theta b}\Delta t \end{bmatrix}$$

Từ các phương trình toán trên chuyển sang code C như sau:

```
Kalman->P[0][0] += dt * (dt * Kalman->P[1][1] - Kalman->P[0][1]
                                - Kalman->P[1][0] + Kalman->Q_angle);
Kalman->P[0][1] -= dt * Kalman->P[1][1];
Kalman->P[1][0] -= dt * Kalman->P[1][1];
Kalman->P[1][1] += Kalman->Q_bias * dt;
```

Bước Measurement Update (correct)

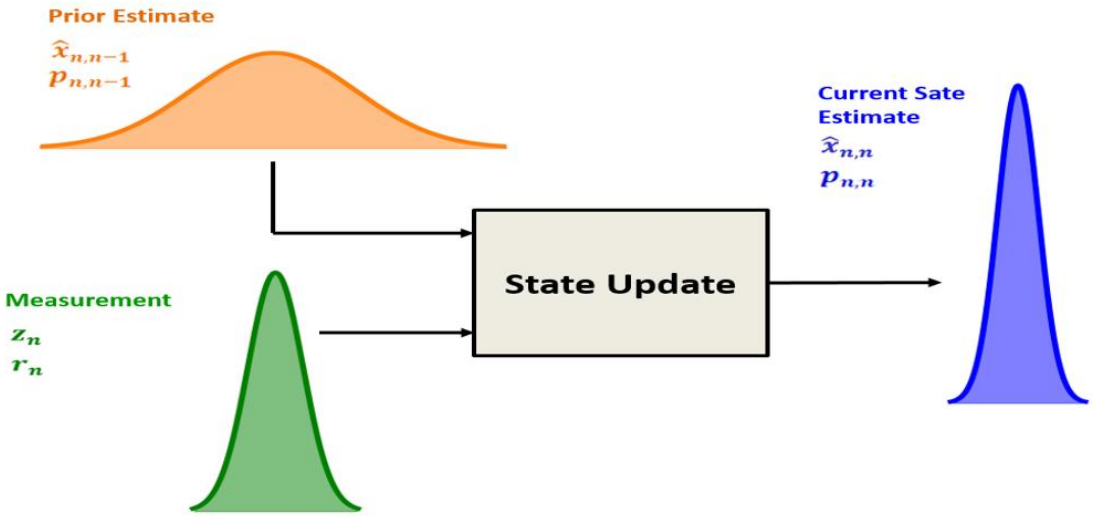


Figure 7: Measurement Update

- Bước 1:

Define $S = H\, P_{k|k-1}\, H^T + R$

$$S = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + R = P_{00\,k|k-1} + R$$

```
double S = Kalman->P[0][0] + Kalman->R_measure;
```

$K_k = P_{k|k-1} H^T S^{-1}$

$$\begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} * \frac{1}{S} = \frac{\begin{bmatrix} P_{00} \\ P_{10} \end{bmatrix}_{k|k-1}}{S}$$

```
double K[2];
K[0] = Kalman->P[0][0] / S;
K[1] = Kalman->P[1][0] / S;
```

- Bước 2

Define $y_k = z_k - H\, x_{k|k-1}$

$$y_k = z_k - \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \theta_b \end{bmatrix}_{k|k-1} = z_k - \theta_{k|k-1}$$

$$x_{k|k} = x_{k|k-1} + K_k\, y_k$$

$$\begin{bmatrix} \theta \\ \theta_b \end{bmatrix}_{k|k} = \begin{bmatrix} \theta \\ \theta_b \end{bmatrix}_{k|k-1} + \begin{bmatrix} K_0\, y \\ K_1\, y \end{bmatrix}_k$$

```
double y = newAngle - Kalman->angle;
Kalman->angle += K[0] * y;
Kalman->bias += K[1] * y;
```

- Bước 3:

$P_{k|k-1} = (I - K_k H)\, P_{k|k-1}$

$$\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k} = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} K_0 \\ K_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \right) \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1}$$

$$\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k} = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} - \begin{bmatrix} K_0 P_{00} & K_0 P_{01} \\ K_1 P_{00} & K_1 P_{01} \end{bmatrix}$$

```
double P00_temp = Kalman->P[0][0];
double P01_temp = Kalman->P[0][1];

Kalman->P[0][0] -= K[0] * P00_temp;
Kalman->P[0][1] -= K[0] * P01_temp;
Kalman->P[1][0] -= K[1] * P00_temp;
Kalman->P[1][1] -= K[1] * P01_temp;
```

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. Các Frame truyền nhận dữ liệu

Frame dữ liệu STM32F1 truyền cho STM32F4 của bàn xoay: Frame có 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

Figure 8: Frame dữ liệu truyền đi

```
#define FRAME  "\n%s%06d %s%06d %s%06d \r"
#define SIGN(X) ((X)>=0)?" ":"-"
#define ABS(X) ((X)>=0)?((X)):(-(X))
```

```
sprintf(data,FRAME,SIGN(datax),ABS(datax),SIGN(datay),ABS(datay),SIGN(dataz),ABS(dataz));

HAL_UART_Transmit(&huart1,(uint8_t*)data, strlen(data), 5);
```

Frame dữ liệu máy tính nhận được từ STM32F4: Có 43 byte gồm các góc roll, pitch, yaw và số xung của 3 encoder

Số byte	1	1	6	1	1	6	1	1	6	1	5	1	5	1	5	1
	\n	±	roll		±	pitch		±	yaw		số xung trục x		số xung trục y		số xung trục z	\r

Figure 9:Frame dữ liệu nhận được

Trong đó số xung encoder trả về khi khởi động là 32000. Măc khác encoder có độ phân giải là 2500 xung/ vòng và mode đọc encoder là mode x4 nên khi đổi từ đơn vị xung sang độ ta làm như sau:

$$angle = \frac{32000 - pulse\_current}{4*2500} 360$$

Code xử lí bằng C#:

```
Ain[3] = double.Parse(sub_data_v1[3]);
double tmp3 = (Ain[3] - 32000) /( 10000);
while (Math.Abs(tmp3) > 1)
{
    tmp3 = Math.Abs(tmp3) - 1;
}
Ain[3] = tmp3 * 360;
if (Ain[3] > 90) Ain[3] = 180 - Ain[3] ;
if (Ain[3] < -90) Ain[3] = Math.Abs(Ain[3]) - 180;
rollEnc_txt.Text = Ain[3].ToString();

Ain[4] = double.Parse(sub_data_v1[4]);
double tmp4 = (Ain[4] - 32000) / 10000;
while (Math.Abs(tmp4) > 1)
{
    tmp4 = Math.Abs(tmp4) - 1;
}
Ain[4] = (tmp4 * 360);
if (Ain[4] > 180) Ain[4] = Ain[4] - 180;
if (Ain[4] < -180) Ain[4] = Ain[4] + 180;
pitchEnc_txt.Text = Ain[4].ToString();
```

V. GUI

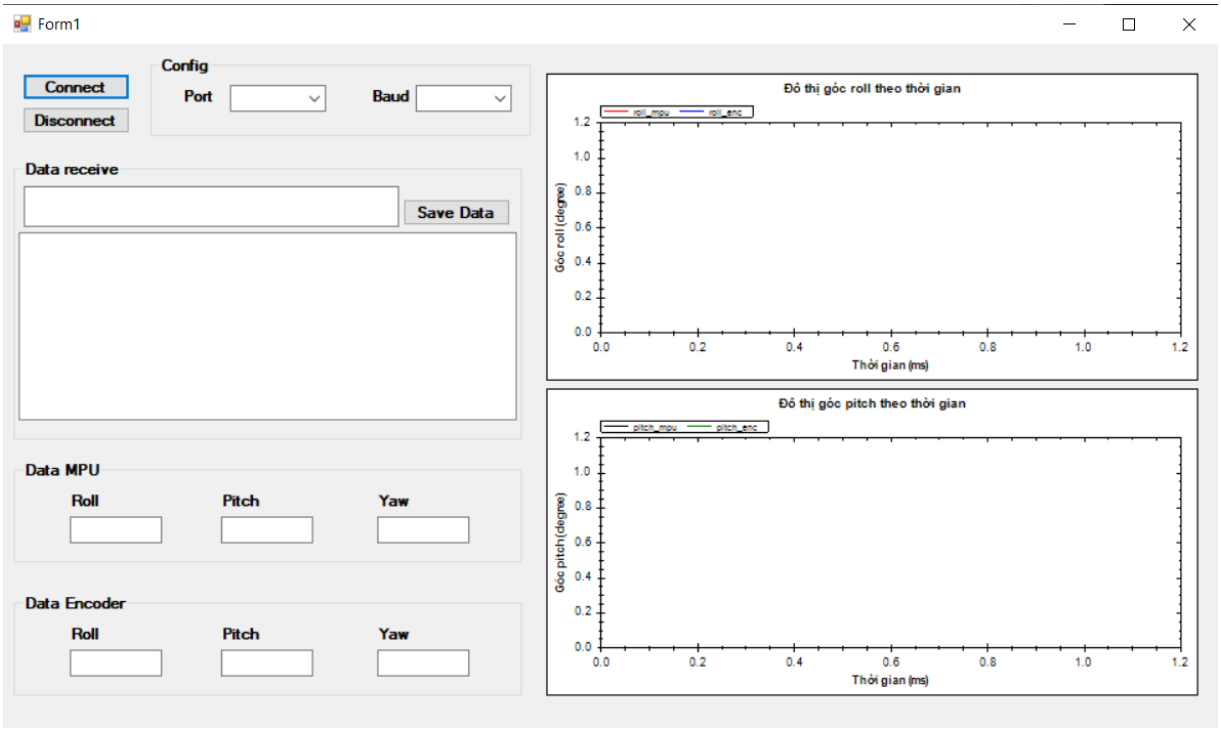


Figure 10: Giao diện hiển thị

Tài liệu tham khảo

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

[2] 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]

Danh mục hình ảnh