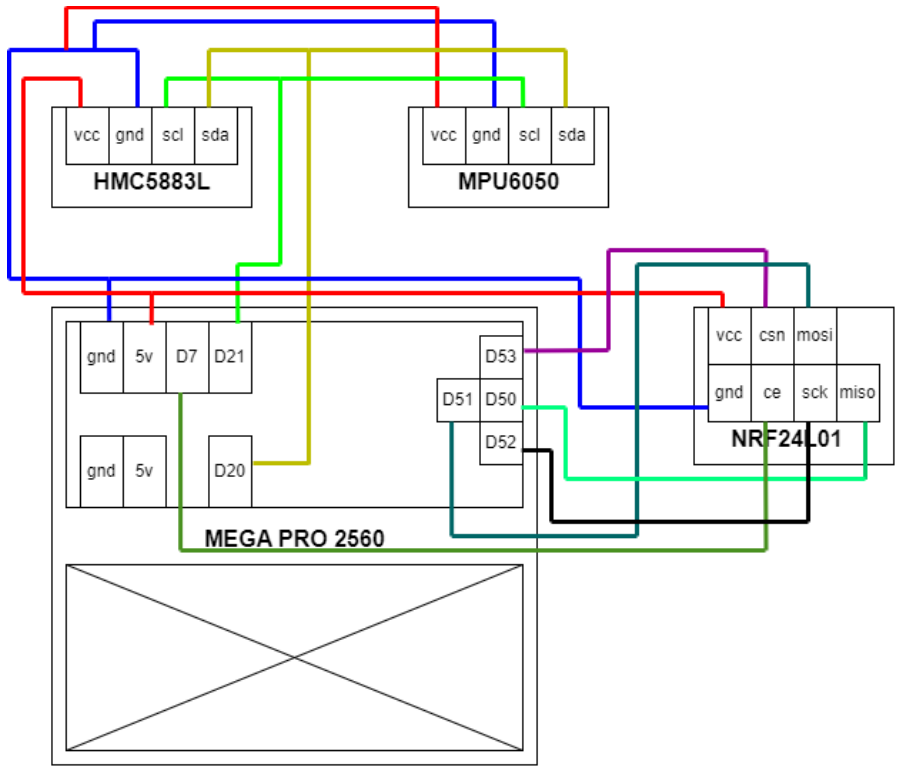
MỤC LỤC

CHƯƠNG 1: PHẦN CỨNG

1.1: SƠ ĐỒ ĐẤU DÂY



Hình 1: Sơ đồ đấu dây

CHƯƠNG 2: PHẦN MỀM

2.1: MPU6050

2.1.1: Tổng quát về MPU6050

MPU6050 là một cảm biến kết hợp bao gồm gia tốc kế 3 trục (3-axis accelerometer) và con quay hồi chuyển 3 trục (3-axis gyroscope). Cảm biến này được sử dụng để đo lường các chuyển động liên quan đến gia tốc và góc quay (roll, pitch, yaw).

Các đặc điểm chính của MPU6050:

- Đo gia tốc dọc theo ba trục (X, Y, Z). Dải đo có thể điều chỉnh từ ±2g, ±4g, ±8g đến ±16g (g là gia tốc trọng trường, khoảng 9.81 m/s^2).
- Đo tốc độ góc quanh ba trục (X, Y, Z). Dải đo có thể điều chỉnh từ ±250, ±500, ±1000 đến ±2000 độ/giây.
- Có thể giao tiếp với vi điều khiển hoặc các thiết bị khác qua giao thức I2C hoặc SPI.

2.1.2: Xử lý dữ liệu từ MPU6050

2.1.2.1: Đọc dữ liệu từ MPU6050

Sử dụng hàm Wire trong Arduino để truy cập vào địa chỉ I2C của cảm biến MPU6050 (0x68) để thực hiện giao tiếp với cảm biến.

Cấu hình thanh ghi 0x1A với 0x01 để bật bộ lọc Low Pass Filter, bộ lọc sẽ có băng thông 184Hz cho Accelerometer và 188Hz cho Gyroscope. Độ trễ của bộ lọc là 2ms.

Cấu hình thanh ghi 0x1C với 0x10 để chọn full scale ở mức 8g.

Bắt đầu đọc dữ liệu của accelerometer từ thanh ghi 0x3B và đọc dữ liệu của gyroscope từ thanh ghi 0x43.

```
Wire.beginTransmission(address_MPU);
Wire.write(0x1A);
Wire.write(0x01);
Wire.endTransmission();

Wire.beginTransmission(address_MPU);
Wire.write(0x1C);
Wire.write(0x10);
Wire.endTransmission();

Wire.beginTransmission(address_MPU);
Wire.write(0x3B);
Wire.endTransmission();
```

Giá trị đọc về của mỗi trục là 16 bit. Thực hiện đọc tín hiệu từ 3 trục X, Y, Z của accelerometer và 3 trục X, Y, Z của gyroscope bằng cách cộng 8 bit MSB với LSB của mỗi trục.

```
Wire.requestFrom(0x68,6); // Accelerometer
int16_t AccXLSB = Wire.read()<<8 | Wire.read();
int16_t AccYLSB = Wire.read()<<8 | Wire.read();
int16_t AccZLSB = Wire.read()<<8 | Wire.read();

Wire.requestFrom(address_MPU,6); // Gyroscope
int16_t GyroX = Wire.read()<<8 | Wire.read();
int16_t GyroY = Wire.read()<<8 | Wire.read();
int16_t GyroZ = Wire.read()<<8 | Wire.read();
```

2.1.2.2: Calibration tín hiệu đọc từ MPU6050

Các bước hiệu chỉnh tín hiệu:

Quy đổi các dữ liệu thô của Gyroscope và Accelerometer với độ nhạy đã chọn lần lượt là 65.5 và 4096.

FS_SEL	Full Scale Range	LSB Sensitivity
0	± 250 °/s	131 LSB/°/s
1	± 500 °/s	65.5 LSB/°/s
2	± 1000 °/s	32.8 LSB/°/s
3	± 2000 °/s	16.4 LSB/°/s

Hình 2: Gyroscope sensitivity

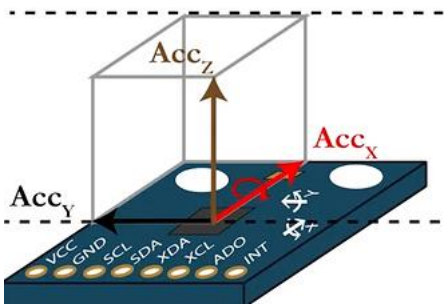
```
float RRoll = (float)GyroX/65.5;
float RPitch = (float)GyroY/65.5;
float RYaw = (float)GyroZ/65.5;
```

AFS_SEL	Full Scale Range	LSB Sensitivity
0	±2*g*	16384 LSB/*g*
1	±4*g*	8192 LSB/*g*
2	±8*g*	4096 LSB/*g*
3	±16*g*	2048 LSB/*g*

Hình 3: Accelerometer sensitivity

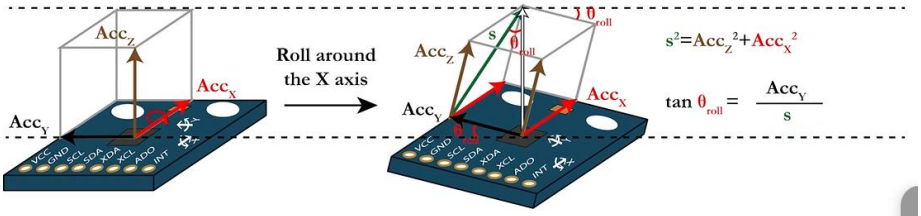
```
float AX = (float)AccXLSB/4096-0.02;
float AY = (float)AccYLSB/4096;
float AZ = (float)AccZLSB/4096-0.14;
```

Tìm phương trình chuyển đổi MPU6050 bằng cách đặt các trục X, Y, Z như sau:



Hình 4: Hình minh họa đặt trục{accelerometer, #1}

Xoay cảm biến theo trục AccX để tính góc xoay roll của cảm biến.

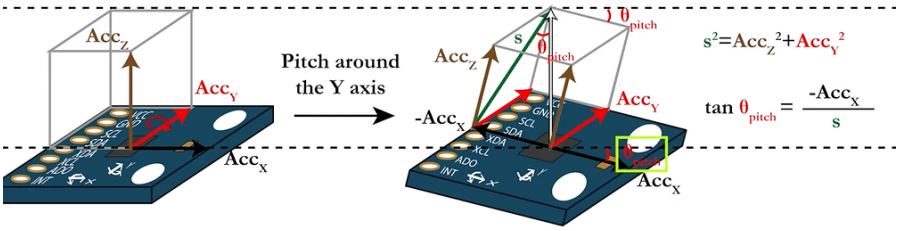


$$tan(\theta_{roll}) = \frac{Acc_Y}{s} \longrightarrow \theta_{roll} = atan\left(\frac{Acc_Y}{\sqrt{Acc_X^2 + Acc_Z^2}}\right)$$

Hình 5: Tính toán và hình minh họa

Tương tự với góc xoay pitch.

$$\theta_{pitch} = atan\left(\frac{-Acc_X}{\sqrt{Acc_Y^2 + Acc_Z^2}}\right)$$



Hình 6: Tính toán và hình minh họa

Tính góc roll và góc pitch theo phương trình tìm được như sau:

$$\theta_{pitch} = atan\left(\frac{-Acc_X}{\sqrt{Acc_Y^2 + Acc_Z^2}}\right)$$

$$\theta_{roll} = atan\left(\frac{Acc_Y}{\sqrt{Acc_X^2 + Acc_Z^2}}\right)$$

```
float Acc_angleroll  = atan(AccY/sqrt(AccX*AccX + AccZ*AccZ)) * 1/(3.142/180);
float Acc_anglepitch = atan(-AccX/sqrt(AccY*AccY + AccZ*AccZ)) * 1/(3.142/180);
```

Từ phương trình này, ta có được góc xoay theo trục X và Y của MPU6050.

Kết quả hiệu chỉnh

2.2: HMC5883L

2.2.1: Tổng quát về HMC5883L

HMC5883L là một cảm biến từ trường (la bàn số) 3 trục (three-axis magnetometer) được sử dụng để đo cường độ và hướng của từ trường xung quanh nó.

Các đặc điểm chính của HMC5883L:

- Đo từ trường theo ba trục không gian (X, Y, Z).
- Sử dụng giao thức giao tiếp I2C
- Có độ nhạy cao và khả năng đo chính xác từ trường. Từ 8 gauss đến mili-gauss.
- Có độ nhạy thấp theo trục chéo

2.2.2: Xử lý dữ liệu từ HMC5883L

2.2.2.1: Đọc dữ liệu từ HMC5883L

Sử dụng hàm Wire trong Arduino để truy cập vào địa chỉ I2C của cảm biến HMC5883L (0x1E) để thực hiện giao tiếp với cảm biến.

```
Wire.beginTransmission(address_HMC);
Wire.write(0x03);
Wire.endTransmission();
```

Giá trị đọc về của mỗi trục là 16 bit. Thực hiện đọc tín hiệu từ 3 trục X, Y, Z bằng cách cộng 8 bit MSB với LSB của mỗi trục.

03	Data Output X MSB Register	Read
04	Data Output X LSB Register	Read
05	Data Output Z MSB Register	Read
06	Data Output Z LSB Register	Read
07	Data Output Y MSB Register	Read
08	Data Output Y LSB Register	Read

```
Wire.requestFrom(address_HMC, 6);
if(6<=Wire.available())
{
  x = Wire.read()<<8;    x |= Wire.read();
  z = Wire.read()<<8;    z |= Wire.read();
  y = Wire.read()<<8;    y |= Wire.read();
}
```

2.2.2.2: Hiệu chỉnh (Calibration) tín hiệu đọc từ HMC5883L

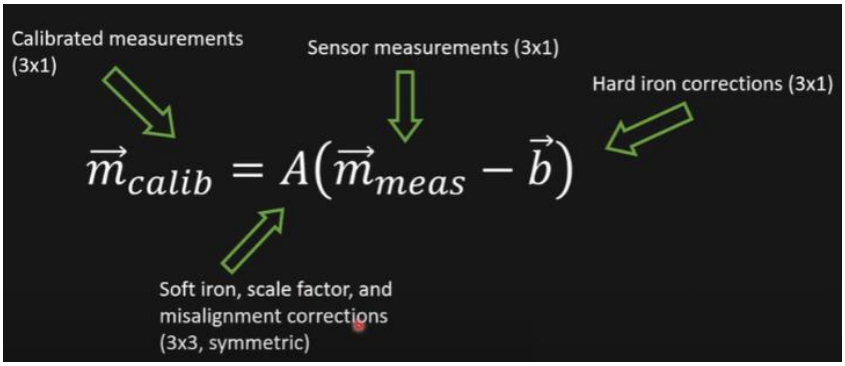
Đối với cảm biến HMC5883L, có hai loại nhiễu chính là Hard Iron Distortions (Biến dạng sắt cứng) gây ra bởi các từ trường cố định, liên tục tồn tại xung quanh cảm biến (thường có ở nam châm kim loại, các dây dẫn mang dòng điện cao,v.v), và Soft Iron Distortions (Biến dạng sắt mềm) gây ra bởi các vật liệu paramagnetic (vật liệu có tính từ kém, không giữ từ trường lâu dài) hoặc vật liệu từ tính khác làm thay đổi hình dạng từ trường địa phương (thường có ở các vật liệu như sắt, thép, v.v)

Hai loại nhiễu trên sẽ làm cho tín hiệu đọc về của hmc5883L bị méo dạng, do đó, tín hiệu đọc về cần được hiệu chỉnh. Phương trình cho việc hiểu chỉnh như sau:

$$\boldsymbol{m_c} = S_I(\tilde{\boldsymbol{m}} - \boldsymbol{b_{HI}})$$

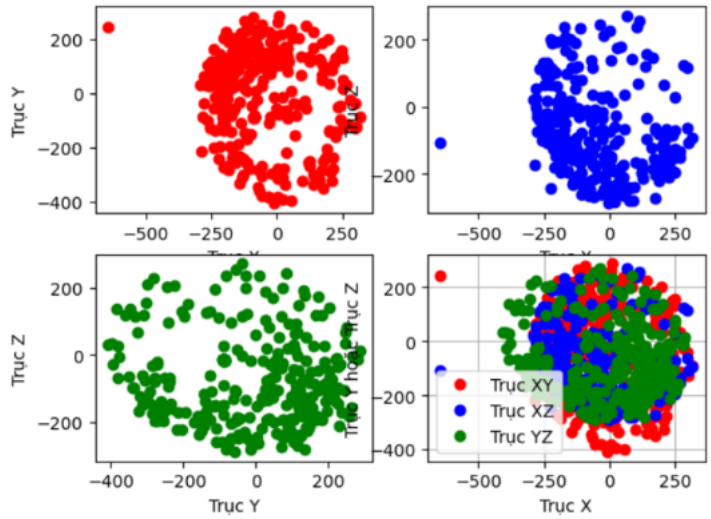
$$\begin{bmatrix} m_{c_x} \\ m_{c_y} \\ m_{c_z} \end{bmatrix} = \begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} \tilde{m}_x - b_{H_0} \\ \tilde{m}_y - b_{H_1} \\ \tilde{m}_z - b_{H_2} \end{bmatrix}$$

Chú thích:



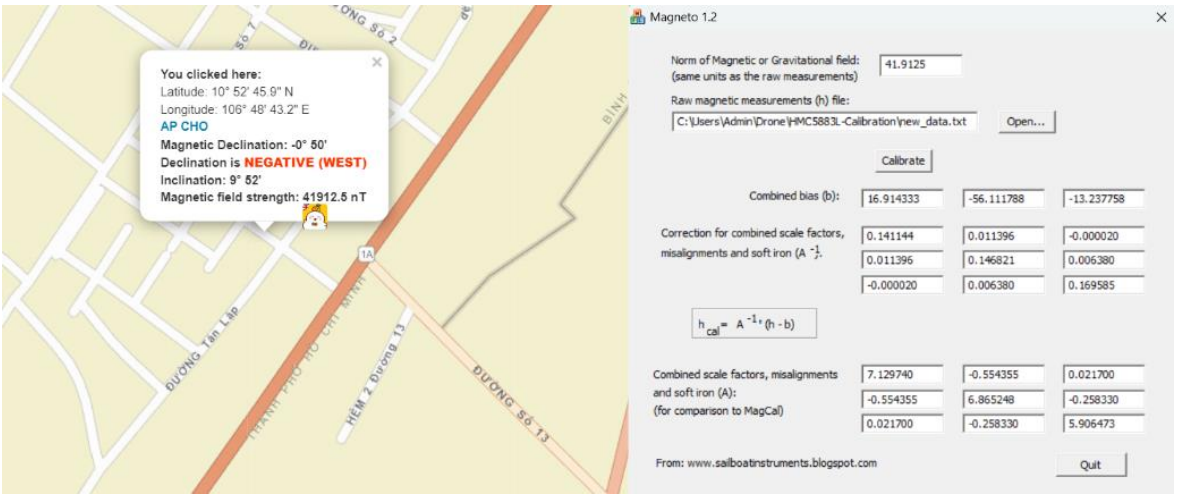
Các bước hiệu chỉnh tín hiệu:

- Lấy dữ liệu thô



Hình 7: Dữ liệu thô

- Sử dụng phần mềm để tìm ma trận A, b và góc từ thiên (góc giữa hướng Bắc từ trường và hướng Bắc thực tế): https://www.magnetic-declination.com/ và phần mềm magneto



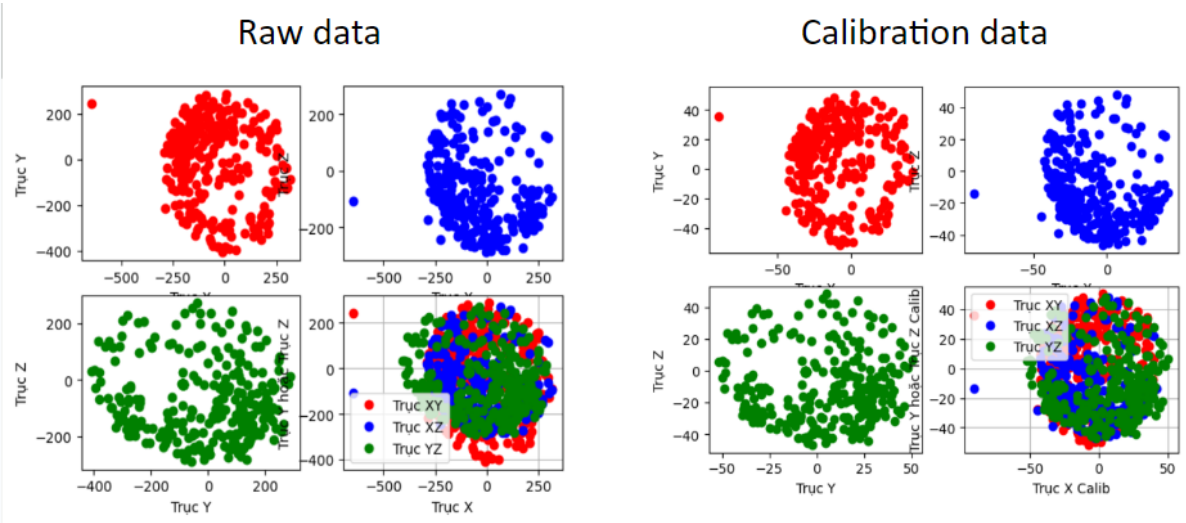
- Nhập các ma trận tìm được vào Arduino IDE và sử dụng phương trình Calib để tính toán.

```
float mag_data[]={x, y, z};
const float hard_iron[3] = {16.914333, -56.111788, -13.237758};
const float soft_iron[3][3] = {
{0.141144 , 0.011396, -0.000020},
{0.011396 , 0.146821, 0.006380},
{-0.000020, 0.006380, 0.169585},};

for (uint8_t i = 0; i < 3; i++)   hard_cal[i] = mag_data[i] - hard_iron [i];
for (uint8_t i = 0; i < 3; i++) {
mag_data[i] = (soft_iron[i][0] * hard_cal [0]) +
              (soft_iron[i][1] * hard_cal [1]) +
              (soft_iron[i][2] * hard_cal [2]);
}

float heading_true = atan2(mag_data[1], mag_data[0]);
float declinationAngle = 0.22;
float heading = heading_true + declinationAngle;
```

- Thêm các điều kiện để chỉ lấy góc từ 0 đến 360 độ.

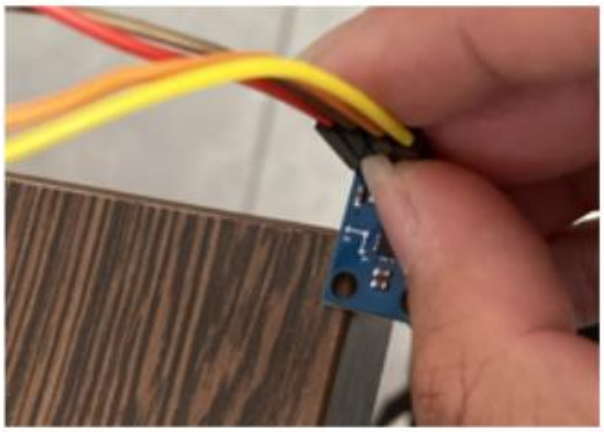
```
if(heading < 0)      heading = heading+ 2*PI;
if(heading > 2*PI)   heading =heading - 2*PI;
if(heading_true < 0)      heading_true = heading_true+ 2*PI;
if(heading_true > 2*PI)   heading_true =heading_true - 2*PI;

heading = heading * 180 / M_PI;
heading_true = heading_true * 180 / M_PI;
```

Kết quả hiệu chỉnh



Hình 8: Kết quả hiệu chỉnh

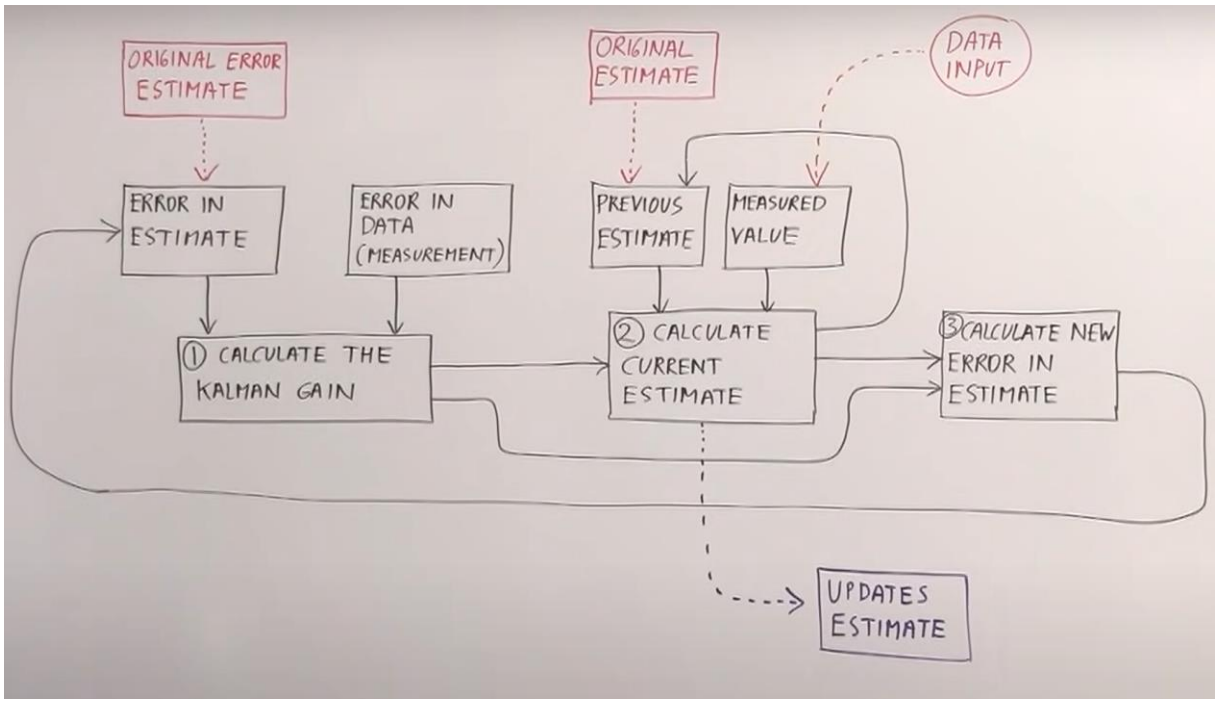
Đo góc bàn



```
x: 40.55  y: 25.96  z: -28.60
Heading (degrees): 33.44

x: 40.41  y: 25.95  z: -28.43
Heading (degrees): 33.53

x: 40.32  y: 26.67  z: -28.40
Heading (degrees): 34.30

x: 40.01  y: 26.22  z: -28.08
Heading (degrees): 34.06
```

Hình 9: So sánh kết quả đo của cảm biến với điện thoại

2.3: BỘ LỌC KALMAN



2.3.1: Prediction:

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix} = \begin{bmatrix} \text{Position at time } k \\ \text{Velocity at time } k \end{bmatrix} \qquad \mathbf{A} \cdot \mathbf{x} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ v \end{bmatrix} \qquad \mathbf{B} \cdot \mathbf{u} = \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} \cdot \begin{bmatrix} a \end{bmatrix}$$

$$\mathbf{x}_{k+1} = \mathbf{A} \cdot \mathbf{x}_k + \mathbf{B} \cdot \mathbf{u}$$

$$\boxed{\mathbf{x}_{k+1} = \begin{bmatrix} x_k + v_k \cdot \Delta t + \frac{\Delta t^2}{2} \cdot a_k \\ v_k + \Delta t \cdot a_k \end{bmatrix}}$$

$$\mathbf{P}_{k+1} = \mathbf{A} \cdot \mathbf{P}_k \cdot \mathbf{A}^T + \mathbf{Q}$$

Where:

- $\mathbf{P}_{k+1}$ is the updated state covariance matrix.
- $\mathbf{A}$ is the state transition matrix.
- $\mathbf{A}^T$ is the transpose of $\mathbf{A}$.
- $\mathbf{P}_k$ is the current state covariance matrix at time k.
- $\mathbf{Q}$ is the process noise covariance matrix, which accounts for the uncertainty in the model.

2. Multiply by $\mathbf{A}^T$ (the transpose of $\mathbf{A}$):

$$\mathbf{A}^T = \begin{bmatrix} 1 & 0 \\ \Delta t & 1 \end{bmatrix}$$

Now multiply:

$$\mathbf{P}' = \mathbf{A} \cdot \mathbf{P}_k \cdot \mathbf{A}^T = \begin{bmatrix} P_{xx} + \Delta t P_{vx} & P_{xv} + \Delta t P_{vv} \\ P_{vx} & P_{vv} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ \Delta t & 1 \end{bmatrix}$$

Performing this multiplication, we get:

$$\mathbf{P}' = \begin{bmatrix} P_{xx} + 2\Delta t P_{vx} + \Delta t^2 P_{vv} & P_{xv} + \Delta t P_{vv} \\ P_{vx} + \Delta t P_{vv} & P_{vv} \end{bmatrix}$$

3. Add the Process Noise Covariance $\mathbf{Q}$:

Finally, add the process noise covariance matrix $\mathbf{Q}$:

$$\mathbf{P}_{k+1} = \mathbf{P}' + \mathbf{Q} = \begin{bmatrix} P_{xx} + 2\Delta t P_{vx} + \Delta t^2 P_{vv} & P_{xv} + \Delta t P_{vv} \\ P_{vx} + \Delta t P_{vv} & P_{vv} \end{bmatrix} + \begin{bmatrix} q_1 & 0 \\ 0 & q_2 \end{bmatrix}$$

This results in the updated covariance matrix:

$$\mathbf{P}_{k+1} = \begin{bmatrix} P_{xx} + 2\Delta t P_{vx} + \Delta t^2 P_{vv} + q_1 & P_{xv} + \Delta t P_{vv} \\ P_{vx} + \Delta t P_{vv} & P_{vv} + q_2 \end{bmatrix}$$

2.3.2: Kalman Gain:

2. **Covariance Matrix**: The predicted covariance matrix $\mathbf{P}_{k+1}^{-}$ is:

$$\mathbf{P}_{k+1}^{-} = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}$$

3. **Measurement Matrix**: Since the measurement is only for position, the measurement matrix $\mathbf{H}$ is:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Nếu ma trận H là [1 0], thì phép đo **zk** chủ yếu dự vào dữ liệu vị trí (position) đo được và bỏ qua dữ liệu vận tốc đo được.

NOTE:

- Nếu ma trận R $\rightarrow$ 0 thì K $\rightarrow$ 1, từ đó khâu cập nhật của bộ lọc sẽ chủ yếu dựa vào dữ liệu thực đo được.
- Nếu ma trận R càng lớn thì K $\rightarrow$ 0, từ đó khâu cập nhật của bộ lọc sẽ chủ yếu dựa vào giá trị dự đoán của bộ lọc.
- Ma trận Q được dùng để thêm vào giá trị sai số (nhiễu của quá trình dự đoán) để tránh ma trận phương sai của giá trị dự đoán bằng 0 (nếu P = 0, dữ liệu từ dữ liệu thực hầu như bị bỏ qua, dẫn đến bộ lọc không còn hiệu quả).

4. **Kalman Gain Calculation**: The Kalman gain $\mathbf{K}$ is calculated as:

$$\mathbf{K}_k = \mathbf{P}_{k+1}^{-}\mathbf{H}^T \left(\mathbf{H}\mathbf{P}_{k+1}^{-}\mathbf{H}^T + R\right)^{-1}$$

Breaking this down:

- $\mathbf{H}\mathbf{P}_{k+1}^{-}\mathbf{H}^T = P_{11}$.
- Therefore, the scalar denominator is $P_{11} + R$.

Thus, the Kalman gain matrix $\mathbf{K}$ becomes:

$$\mathbf{K}_k = \begin{bmatrix} P_{11} \\ P_{21} \end{bmatrix} \cdot \frac{1}{P_{11} + R}$$

So, the individual Kalman gains are:

$$K_1 = \frac{P_{11}}{P_{11} + R}, \quad K_2 = \frac{P_{21}}{P_{11} + R}$$

2.3.3: Updation:

Final Update Equation:

Substitute the Kalman gains back into the update equation:

$$\begin{bmatrix}\hat{x}_{k+1}\\ \hat{v}_{k+1}\end{bmatrix} = \begin{bmatrix}\hat{x}^-_{k+1}\\ \hat{v}^-_{k+1}\end{bmatrix} + \begin{bmatrix}\frac{P_{11}}{P_{11}+R}\\ \frac{P_{21}}{P_{11}+R}\end{bmatrix} \cdot \left(z_k - \hat{x}^-_{k+1}\right)$$

4. **Compute $\mathbf{K}_k \cdot \mathbf{H}$**: The product of the Kalman gain matrix and the measurement matrix is:

$$\mathbf{K}_k \cdot \mathbf{H} = \begin{bmatrix}\frac{P_{11}}{P_{11}+R}\\ \frac{P_{21}}{P_{11}+R}\end{bmatrix} \cdot [1\,0] = \begin{bmatrix}\frac{P_{11}}{P_{11}+R} & 0\\ \frac{P_{21}}{P_{11}+R} & 0\end{bmatrix}$$

5. **Compute $\mathbf{I} - \mathbf{K}_k \cdot \mathbf{H}$**: Subtract this result from the identity matrix:

$$\mathbf{I} - \mathbf{K}_k \cdot \mathbf{H} = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} - \begin{bmatrix}\frac{P_{11}}{P_{11}+R} & 0\\ \frac{P_{21}}{P_{11}+R} & 0\end{bmatrix} = \begin{bmatrix}1-\frac{P_{11}}{P_{11}+R} & 0\\ -\frac{P_{21}}{P_{11}+R} & 1\end{bmatrix}$$

6. **Update $\mathbf{P}_{k+1}$**: Finally, multiply this result by the predicted covariance matrix $\mathbf{P}^-_{k+1}$:

$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}_k \cdot \mathbf{H}) \cdot \mathbf{P}^-_{k+1}$$

Substituting the matrices:

$$\mathbf{P}_{k+1} = \begin{bmatrix}1-\frac{P_{11}}{P_{11}+R} & 0\\ -\frac{P_{21}}{P_{11}+R} & 1\end{bmatrix} \cdot \begin{bmatrix}P_{11} & P_{12}\\ P_{21} & P_{22}\end{bmatrix}$$

Performing the matrix multiplication:

$$\mathbf{P}_{k+1} = \begin{bmatrix}\left(1-\frac{P_{11}}{P_{11}+R}\right)P_{11} & \left(1-\frac{P_{11}}{P_{11}+R}\right)P_{12}\\ \left(-\frac{P_{21}}{P_{11}+R}\right)P_{11}+P_{21} & \left(-\frac{P_{21}}{P_{11}+R}\right)P_{12}+P_{22}\end{bmatrix}$$