BÀI TIỂU LUẬN

ĐỀ TÀI: ỨNG DỤNG BỘ LỌC KALMAN TRONG ĐỊNH HƯỚNG ĐỐI TƯỢNG

Môn: Xử Lý Tín Hiệu Số Và DSP

Tên sinh viên	MSSV
Hà Đức Phi	105210045
Nguyễn Thành Thư Hoàng	105210033
Hoàng Nhật Duy	105210313
Lê Hồng Phúc	105210327

Nhóm học phần : 21N32

Giáo viên hướng dẫn : Nguyễn Thị Kim Trúc.

Đà Nẵng , 11/2024

MỤC LỤC

DANH MỤC HÌNH ẢNH

CHƯƠNG I : TỔNG QUAN ĐỀ TÀI VÀ GIỚI THIỆU CẢM BIẾN GIA TỐC VÀ CON QUAY HỒI CHUYỂN MPU6050

1.1 Tổng quan đề tài:

❖ **Đặt vấn đề:**

- Trong các ứng dụng tự động hóa ngày nay như hệ thống điều khiển Robot , xe tự hành , máy bay tự hành, hệ thống ổn định Camera, … Việc đo đạc cảm biến để thu được các tín hiệu góc hay vận tốc góc với độ chính xác cao là yêu cầu bắt buộc.
- Nhưng việc các đối tượng, hệ thống trên hoạt động trong môi trường có nhiễu cao sẽ làm ảnh hưởng đến tính chính xác của các thông tin cần đo đạc. Từ đó chất lượng của hệ thống sẽ bị giảm sút một cách nghiêm trọng.
- Việc ứng dụng bộ lọc Kalman (hay còn gọi là bộ quan sát Kalman) là một công cụ mạnh mẽ để ước tính và dự đoán trạng thái hệ thống khi có sự xuất hiện của nhiễu trong mô hình, giúp cải thiện chất lượng của các tín hiệu mong muốn trong hệ thống.

❖ **Mục tiêu của đề tài:**

- Đề tài nhằm phát triển và ứng dụng bộ lọc Kalman để xử lý dữ liệu từ cảm biến MPU6050, qua đó:
  - Cải thiện độ chính xác trong việc định hướng đối tượng.
  - Giảm thiểu nhiễu và sai số trôi trong dữ liệu cảm biến.
  - Tích hợp phương pháp xử lý dữ liệu trong thời gian thực để phục vụ các ứng dụng như cân bằng robot hoặc định hướng drone.
- Song, đề tài sẽ ứng dụng các góc quay (Roll,Pitch,Yaw) đã được ước lượng để định hướng đối tượng nhờ phép quay Roll – Pitch – Yaw.

❖ Công cụ để thực hiện đề tài:

- Phần cứng:
  - Board vi điều khiển Aduino Uno R3 đóng vai trò là bộ xử lý tín hiệu số.
  - Cảm biến MPU6050 là cảm biến cho giá trị gia tốc dài (Accelometer) và cảm biến con quay hồi chuyển (Gyroscope) cho giá trị vận tốc dài.
- Phần mềm:
  - Phần mềm Arduino IDE : phục vụ cho việc lập trình chương trình xử lý số của đề tài.
  - Phần mềm Excel : Thu thập dữ liệu và chuẩn hóa trước khi gửi cho Malab vẽ đồ thị phân tích.

+ Phần mềm Matlab : Lấy dữ liệu thu thập từ Excel để vẽ các đồ thị liên quan phục vụ quá trình phân tích.
+ Phần mềm Proccesing : Mô phỏng đối tượng trong không gian 3D dựa vào các góc Roll , Pitch , Yaw đã lọc.

❖ Ứng dụng của đề tài:

- Đề tài có ứng dụng rộng rãi đến các dự án liên quan đến các tín hiệu góc quay hay vận tốc góc quay:
  + Robot tự hành: Xác định chính xác tư thế để điều chỉnh chuyển động.
  + Drone: Đảm bảo ổn định trong các điều kiện thời tiết phức tạp.
  + Thiết bị đeo thông minh: Theo dõi chuyển động và tư thế của người dùng.v.v

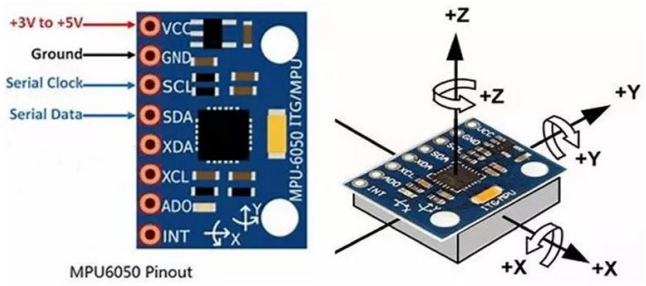
1.2 Giới thiệu chi tiết về cảm biến MPU6050:

1.2.1 Tổng quan

- MPU6050 là một cảm biến MEMS (Hệ thống vi cơ điện tử) tích hợp cả gia tốc kế và con quay hồi chuyển, cho phép đo lường đồng thời gia tốc và tốc độ quay trên cả ba trục X, Y và Z.
- Cảm biến này được sản xuất bởi **InvenSense,** hãng rất phổ biến trong các ứng dụng yêu cầu nhận diện chuyển động và định hướng. Nhờ tính nhỏ gọn, tiết kiệm năng lượng và tích hợp nhiều chức năng, MPU6050 được sử dụng rộng rãi trong nhiều lĩnh vực, từ robot tự hành, thiết bị đeo thông minh, đến các hệ thống tự động hóa và máy bay không người lái.

❖ **Sơ đồ chân :**

- **VCC: Nguồn cấp (3.3V hoặc 5V)**
- **SDA:** Chân dữ liệu (Data) trong giao tiếp I2C
- **SCL:** Chân xung Clock trong giao tiếp I2C
- **AD0:** Chân địa chỉ định danh I2C (nếu được sử dụng)
- **INT:** Chân ngắt (Interrupt) cho MPU6050
- **GND:** Chân đất (Ground)
- **XDA:** Chân dữ liệu (Data) cho bus giao tiếp I2C thứ hai
- **XCL:** Chân xung Clock cho giao tiếp I2C thứ hai



Hình 1.1 Tổng quan cảm biến MPU6050

1.2.2 Thông số kỹ thuật:

- Thông số chung:
  + Kích thước: 4×4×0.9 mm
  + Điện áp hoạt động: 2.375 ÷ 3.46 V
  + Giao thức truyền thông: I2C (Tốc độ tối đa 400 kHz)

+ Nhiệt độ hoạt động: -40°C đến +85°C

- Thông số của con quay hồi chuyển:
  + Phạm vi đo (Full Scale Range): ±250°/s, ±500°/s, ±1000°/s, ±2000°/s (có thể chọn theo cấu hình).
  + Độ nhạy (Sensitivity):

Độ nhạy °/s	LSB/(°/s)
±250	131
±500	65.5
±1000	32.8
±2000	16.4

  + Độ phân giải: 16-bit
  + Dòng hoạt động: 3.6mA.
  + Dòng ở chế độ chờ: 5µA.

  Các thanh ghi đo lường con quay hồi chuyển

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
43	67	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT[7:0]							

    - GYRO_XOUT (X-axis): Lưu giá trị tốc độ góc gần nhất đo trên trục X.
    - GYRO_YOUT (Y-axis): Lưu giá trị tốc độ góc gần nhất đo trên trục Y.
    - GYRO_ZOUT (Z-axis): Lưu giá trị tốc độ góc gần nhất đo trên trục Z.

- Thông số của cảm biến gia tốc :
  + Phạm vi đo (Full Scale Range): ±2g, ±4g, ±8g, ±16g (có thể chọn theo cấu hình).
  + Độ nhạy (Sensitivity):

Độ nhạy g	LSB/g
±2	16,384
±4	8,192
±8	4,096
±16	2,048

+ Độ phân giải: 16-bit
+ Dòng hoạt động bình thường của gia tốc kế: 500µA
+ Dòng ở chế độ công suất thấp: 10µA ở 1.25Hz, 20µA ở 5Hz, 60µA ở 20Hz, 110µA ở 40Hz

Các thanh ghi đo lường gia tốc kế

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT[7:0]							

- ACCEL_XOUT (X-axis): Lưu giá trị gia tốc gần nhất đo trên trục X.
- ACCEL_YOUT (Y-axis): Lưu giá trị gia tốc gần nhất đo trên trục Y.
- ACCEL_ZOUT (Z-axis): Lưu giá trị gia tốc gần nhất đo trên trục Z.

1.2.3 Phương thức đọc cảm biến bằng Arduino:

- Mục tiêu đọc các giá trị thô mang thông tin cảm biến gia tốc và con quay hồi chuyển từ cảm biến bằng Arduino. Sau đó chuẩn hóa.

Thiết bị sử dụng

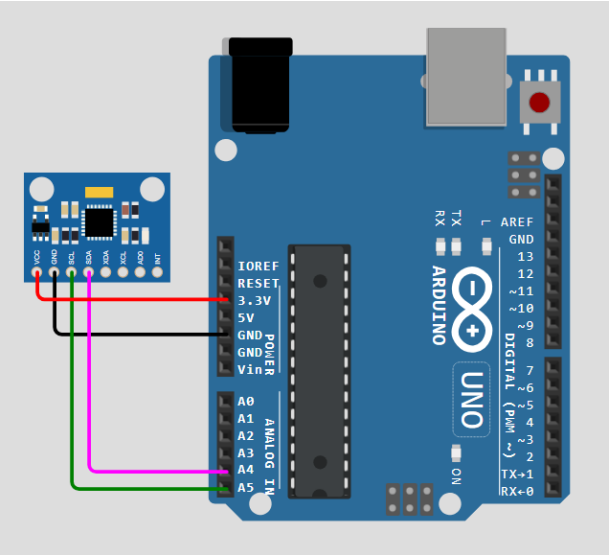
- **Arduino Uno**: Vi điều khiển làm nhiệm vụ thu thập và xử lý dữ liệu.
- **MPU6050**: Cảm biến tích hợp gia tốc kế và con quay hồi chuyển.
- **Dây kết nối**: Dùng để nối các chân giữa Arduino và MPU6050.
- **Nguồn cấp điện**: Arduino được cấp nguồn qua cổng USB.

Sơ đồ kết nối

Chân MPU6050	Chân Arduino Uno R3
VCC	3.3V
GND	GND
SCL	A5
SDA	A4

Sơ đồ kết nối các chân giữa Arduino Uno R3 và cảm biến MPU6050 được biểu diễn và mô phỏng như *Hình 2.2* trong đó:

- VCC và GND: Dùng để cấp nguồn cho MPU6050.
- SCL và SDA: Dùng để truyền và nhận dữ liệu qua giao thức I2C.



Hình 1.2 Sơ đồ đấu nối cảm biến MPU6050 với Ardunio Uno R3

2. Cấu hình phần mềm

a. Giao thức giao tiếp: I2C.

- Địa chỉ mặc định của MPU6050: 0x68.
- Tốc độ giao tiếp I2C: 400kHz.

b. Đọc dữ liệu từ cảm biến MPU6050

Để đọc cảm biến MPU6050, trước tiên cần đánh thức cảm biến bằng cách thiết lập giá trị cho bit SLEEP trong thanh ghi PWR_MGMT_1 (địa chỉ 0x6B).

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
67	103	I2C_MST_DELAY_CTRL	R/W	DELAY_ES_SHADOW	-	-	I2C_SLV4_DLY_EN	I2C_SLV3_DLY_EN	I2C_SLV2_DLY_EN	I2C_SLV1_DLY_EN	I2C_SLV0_DLY_EN
68	104	SIGNAL_PATH_RESET	R/W	-	-	-	-	-	GYRO_RESET	ACCEL_RESET	TEMP_RESET
6A	106	USER_CTRL	R/W	-	FIFO_EN	I2C_MST_EN	I2C_IF_DIS	-	FIFO_RESET	I2C_MST_RESET	SIG_COND_RESET
6B	107	PWR_MGMT_1	R/W	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT[7:0]							
41	65	TEMP_OUT_H	R	TEMP_OUT[15:8]							
42	66	TEMP_OUT_L	R	TEMP_OUT[7:0]							
43	67	GYRO_XOUT_H	R	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT_L	R	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT_H	R	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT_L	R	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT[7:0]							

Sau khi cấu hình để cảm biến thoát khỏi chế độ SLEEP, để lấy được giá trị vận tốc góc trên trục X, cần đọc dữ liệu từ hai thanh ghi: **GYRO_XOUT_H** (địa chỉ 0x43) và **GYRO_XOUT_L** (địa chỉ 0x44). Mỗi thanh ghi có độ dài 8 bit, trong đó **GYRO_XOUT_H** chứa 8 bit cao, còn **GYRO_XOUT_L** chứa 8 bit thấp. Gộp hai thanh ghi này lại sẽ tạo thành một giá trị 16 bit hoàn chỉnh, biểu diễn vận tốc góc trên trục X.

Tương tự để lấy được giá trị vận tốc góc trên trục Y, cần được dữ liệu từ hai thanh ghi: **GYRO_YOUT_H** (địa chỉ 0x45) và **GYRO_YOUT_L** (địa chỉ 0x46), còn với trục Z là 2 thanh ghi **GYRO_ZOUT_H** (địa chỉ 0x47) và **GYRO_ZOUT_L** (địa chỉ 0x48).

Phương pháp xử lý dữ liệu cảm biến để tính góc

a. Sử dụng tín hiệu từ gia tốc kế (Accelerometer)

- Gia tốc kế đo gia tốc trên các trục x,y,z. Do trọng lực luôn không đổi theo thời gian ta có thể tính các góc quay theo trục x (ϕ) và y (θ):

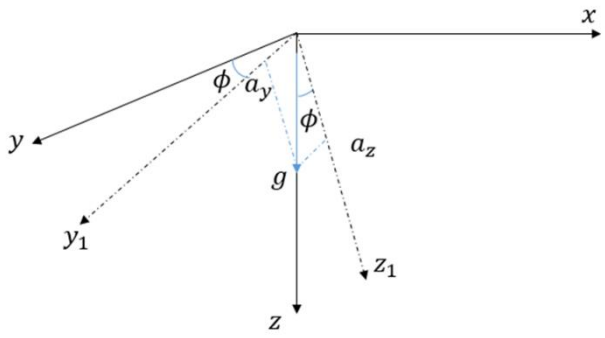
Góc quay quanh trục x (Roll)

$$\sin\phi = \frac{a_y}{g}$$

$$\cos\phi = \frac{\sqrt{a_x^2 + a_z^2}}{g}$$

$$\tan\phi = \frac{a_y}{\sqrt{a_x^2 + a_z^2}}$$

$$\Rightarrow \phi = \arctan\frac{a_y}{\sqrt{a_x^2 + a_z^2}}$$



Hình 1.3 Đồ thị biểu diễn góc quay theo trục x

Tương tự ta tính được:

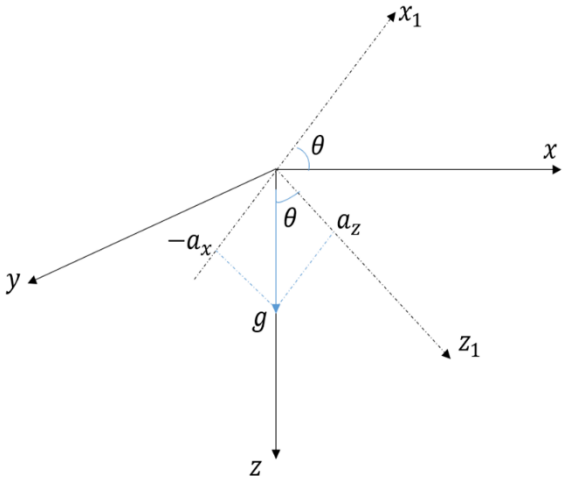
Góc quay quanh trục y (Pitch)

$$\sin\theta = \frac{-a_x}{g}$$

$$\cos\theta = \frac{\sqrt{a_y^2 + a_z^2}}{g}$$

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

$$\Rightarrow \theta = \arctan\frac{-a_x}{\sqrt{a_y^2 + a_z^2}}$$



Hình 1.4 Đồ thị biểu diễn góc quay theo trục y

Nhược điểm:

- Trục z cùng phương với trọng lực, do đó không thể tính góc quay quanh trục z ψ (Yaw) chỉ bằng gia tốc kế.
- Khi cảm biến chịu tác động gia tốc do chuyển động (ngoài trọng lực), các giá trị sẽ bị nhiễu, dẫn đến tính toán góc không chính xác, bị hạn chế góc đo

b. Sử dụng con quay hồi chuyển (Gyroscope)

Con quay đo vận tốc góc (ω) trên các trục x,y,z. Tích phân vận tốc góc theo thời gian để tính góc quay:

$$\theta(t) = \theta_0(t) + \int_{t_0}^{t} \omega(t) \cdot dt$$

Nhược điểm:

- Sai số nhỏ trong mỗi phép đo tích lũy dần theo thời gian, dẫn đến góc tính toán bị lệch (trôi) khi thời gian tăng.
- Độ chính xác phụ thuộc vào tần số lấy mẫu.
- Tồn tại giá trị sái lệch ban đầu, cần hiệu chỉnh offset để giảm thiểu các giá trị tối thiểu dự kiến trong điều kiện IMU ổn định

c. Nhận xét

Trong các ứng dụng đo lường góc quay bằng cảm biến như MPU6050, hai phương pháp cơ bản sử dụng dữ liệu từ gia tốc kế và con quay hồi chuyển đều có những ưu điểm riêng. Tuy nhiên, khi áp dụng độc lập, cả hai phương pháp này đều tồn tại các nhược điểm đáng kể:

- Gia tốc kế có khả năng đo ổn định trong điều kiện đứng yên nhưng lại bị nhiễu mạnh khi có gia tốc động hoặc rung lắc, và không thể đo được góc quay quanh trục z (Yaw).
- Con quay hồi chuyển phản ứng nhanh với các thay đổi góc nhưng lại mắc phải hiện tượng trôi góc (Drift), khiến sai số tích lũy theo thời gian, làm giảm độ chính xác trong các phép đo dài hạn.

Để giải quyết nhược điểm của hai phương pháp, các bộ lọc tín hiệu, đặc biệt là bộ lọc Kalman, đã trở thành giải pháp lý tưởng. Bộ lọc Kalman không chỉ khắc phục được các hạn chế riêng lẻ mà còn giúp ước lượng trạng thái hệ thống một cách chính xác ngay cả khi dữ liệu bị nhiễu.

CHƯƠNG II : CƠ SỞ LÝ THUYẾT BỘ LỌC KALMAN

- Bộ lọc Kalman (Kalman Filter) là một thuật toán toán học dùng để ước lượng trạng thái của một hệ thống động, ngay cả khi các quan sát đo được bị nhiễu. Bộ lọc Kalman là bộ lọc số được thiết kế trong miền không gian trạng thái.
- Được sử dụng rộng rãi như một thành phần cơ bản trong các ứng dụng như điều khiển tự động, robot, hàng không vũ trụ, tài chính, và xử lý tín hiệu.Bộ lọc được đặt theo tên của Rudolf E. Kálmán (19 tháng 5 năm 1930 – 2 tháng 7 năm 2016).

2.1 Mô hình trạng thái mô tả đối tương trong môi trường nhiễu

- Ta giả định rằng hệ thống có thể mô tả thành phương trình trạng thái rời rạc:

$$x_{k+1} = A \cdot x_k + B \cdot u_k + w_k$$

Với

- x_{k+1} : Trạng thái thời điểm k+1.
- x_k : Trạng thái thời điểm k.
- u_k : Biến đầu vào hệ thống.
- w_k : Biến đặc trưng cho nhiễu hệ thống.
- A : Ma trận hệ thống.
- B : Ma trận đầu vào.

- Nếu có thể quan sát một vài thành phần trong biến trạng thái thì ta xây dựng được phương trình đo lường tuyến tính:

$$z_k = H \cdot x_k + v_k$$

Với

- z_k : Biến đo lường
- v_k : Biến đặc trưng cho nhiễu đo lường.
- H : Ma trận quan sát.

- Trong quá trình hoạt động , nhiễu hệ thống w_k và nhiễu đo lường v_k có một số tính chất sau:
  - w_k và v_k là biến ngẫu nhiên độc lập với nhau:

$$E[w_k \cdot v_l^T] = 0 \qquad với \quad \forall l, k$$

  - Nhiễu hệ thống tại mỗi thời điểm khác nhau là độc lập và tại thời điểm k sẽ ứng với một ma trận hiệp phương sai $\mathbf{Q}_k$.

$$E[w_k \cdot w_l^T] = \begin{cases} Q_k & , k = l \\ 0 & , k \neq \end{cases}$$

  - Nhiễu đo lường tại mỗi thời điểm khác nhau là độc lập và tại thời điểm k sẽ ứng với một ma trận hiệp phương sai $\mathbf{R}_k$.

$$E[v_k \cdot v_l^T] = \begin{cases} Q_k & , k = l \\ 0 & , k \neq \end{cases}$$

- Các ma trận $\mathbf{Q}_k$ và $\mathbf{R}_k$ có tính chất:
  - Đối xứng : $\mathbf{Q}_k=\mathbf{Q}_k^T$, $\mathbf{R}_k=\mathbf{R}_k^T$
  - Bán xác định dương: $x^T\mathbf{Q}_k x\geq 0$, $x^T\mathbf{R}_k x \geq 0$,$\forall x$ *(* để mức độ không chắc chắn không âm*)*.
- Như vậy, có thể xem các biến w_k và v_k là các biến ngẫu nhiên tuân theo phân phối chuẩn $\boldsymbol{w(k)} \sim \boldsymbol{N(0}\,,\ \boldsymbol{Q_k)}$, $\boldsymbol{v(k)} \sim \boldsymbol{N(0}\,, \boldsymbol{R_k)}$**.** Với $\mathbf{Q}_k$ và $\mathbf{R}_k$ là các ma trận đường chéo.

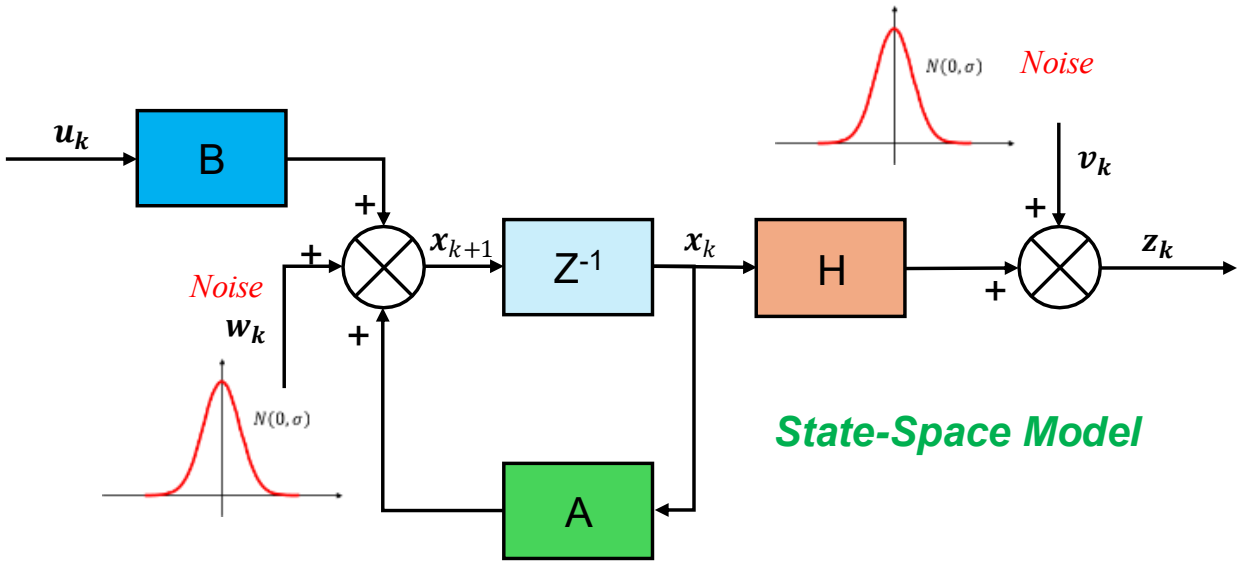
$$Q_k = \begin{bmatrix} \sigma_{x_1}^2 & \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots & \sigma_{x_n}^2 \end{bmatrix} \quad v\text{à} \quad R_k = \begin{bmatrix} \sigma_{z_1}^2 & \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots & \sigma_{z_m}^2 \end{bmatrix}$$

Với

- $\sigma_{x_1}^2 \rightarrow \sigma_{x_n}^2$: Các phương sai của thành phần biến quá trình.
- $\sigma_{z_1}^2 \rightarrow \sigma_{z_n}^2$: Các phương sai của thành phần biến đo lường.

- Để đơn giản bài toán cũng như thuận tiện cho việc tính toán, khối lượng xử lý của bộ xử lý số, ta xem các ma trận $\mathbf{Q}_k$ và $\mathbf{R}_k$ là hằng số cố định và sẽ được đo đạc thực nghiệm.

- Ta có sơ đồ mô hình không gian trạng thái của hệ thống:



Hình 2.1 Sơ đồ mô hình không gian trạng thái của hệ thống

2.2 Lý thuyết hình thành của bộ lọc Kalman:

- Mục tiêu của bộ lọc Kalman là giảm nhiễu sai số của bình phương kỳ vọng trong quá trình ước lượng trạng thái $\hat{x}_k$.Tức là đi giải quyết bài toán tối ưu hóa :

$$\hat{P}_{k|k} = agmin\ E(\|x_k - \hat{x}_k\|^2) = E((x_k - \hat{x}_k)^T(x_k - \hat{x}_k))$$

  + Với nghiệm tối ưu là biến trạng thái ước lượng $\hat{x}_k$.
- Trạng thái $\hat{x}_k$ được ước lượng dựa trên tất cả các quan sát thu thập được trước đó, tức là $\hat{x}_k|z_{k-1}$ (việc tính toán này dựa vào mô hình không gian của hệ thống và trạng thái z_k từ 0 đến k-1).
- Như vậy, để giải bài toán tối ưu trên ta cần tính toán giá trị kỳ vọng của trạng thái $\hat{x}_k$ với điều kiện là các quan sát đến thời điểm z_{k-1}, tức $E(\hat{x}_k|z_{k-1})$. Ta có:

$$\hat{x}_{k|k-1} = E(\hat{x}_k|z_{k-1}) = E(A \cdot x_{k-1} + B \cdot u_{k-1} + w_{k-1}|z_{k-1})$$

  Suy ra, ta có phương trình dự đoán trạng thái hiện tại từ trạng thái cũ:

$$\hat{x}_{k|k-1} = A.\hat{x}_{k-1} + B.u_k$$

- Khi đó ta tính được hiệp phương sai dự đoán là kỳ vọng của sai số bình phương :

$$P_{k|k-1} = E\big((x_k - \hat{x}_{k|k-1})(x_k - \hat{x}_{k|k-1})^T|z_k\big)$$

- Từ các công thức toán học ta tính được phương trình dự đoán ma trận phương sai dự đoán từ trạng thái trước:

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

- Thuật toán Kalman thực hiện tính giá trị tối thiểu hóa của phương sai sai số ước lượng sau cập nhật $\hat{P}_{k|k}$:

Khi đó: $\hat{P}_k = Cov(x_k - \hat{x}_{k|k-1})$

Hay: $K_k = argmin\ Trace\big(\hat{P}_k\big)$

Suy ra: $K_k = P_{k|k-1} \cdot H^T \cdot \big(H \cdot P_{k|k-1} \cdot H^T + R_k\big)^{-1}$ (Ma trận độ lợi).

- Khi nhận quan sát mới z_{k-1}, trạng thái được cập nhật bằng cách kết hợp trạng thái dự đoán $\hat{x}_{k|k-1}$ với thông tin từ quan sát mới z_{k-1}:
  + Với phương trình cập nhật biến trạng thái:

$$\hat{x}_k = \hat{x}_{k|k-1} + K_k \cdot \big(z_k - H \cdot \hat{x}_{k|k-1}\big)$$

  + Phương trình cập nhật ma trận phương sai của sai số ước lượng:

$$\hat{P}_k = P_{k|k-1} - K_k \cdot H \cdot P_{k|k-1}$$

- Tổng quát: Bộ lọc Kalman bao gồm hai quá trình chính: dự đoán (prediction) và cập nhật (update).
- Bước dự đoán: Bộ lọc Kalman sử dụng mô hình trạng thái của hệ thống để dự đoán **trạng thái hiện tại** và **ma trận hiệp phương sai** dựa trên trạng thái trước đó.
  + Dự đoán trạng thái mà không cần thông tin đo lường mới:

$$\hat{x}_{k|k-1} = A.\hat{x}_{k-1} + B.u_k$$

Với

- $\hat{x}_{k|k-1}$: Trạng thái dự đoán.
- $\hat{x}_{k-1}$: Trạng thái ước lượng từ thời điểm trước.
- A: Ma trận trạng thái hệ thống.
- B: Ma trận điều khiển (nếu có đầu vào u_k).

+ Dự đoán ma trận hiệp phương sai:

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

Với

- $P_{k|k-1}$: Ma trận hiệp phương sai dự đoán.
- P_{k-1}: Ma trận hiệp phương sai ước lượng từ trạng thái trước.
- Q_k: Ma trận nhiễu trạng thái (phản ánh độ bất định của mô hình).

- Bước cập nhật: Bước này sử dụng thông tin đo lường mới để cập nhật trạng thái và ma trận hiệp phương sai, cải thiện độ chính xác của dự đoán.

+ Tính ma trận độ lợi :

$$K_k = P_{k|k-1} \cdot H^T \cdot \left(H \cdot P_{k|k-1} \cdot H^T + R_k\right)^{-1}$$

Với

- K_k : Kalman gain, hệ số xác định mức độ tin cậy vào đo lường so với dự đoán.
- R_k: Ma trận nhiễu đo lường (phản ánh độ không chính xác của cảm biến).
- H : Ma trận đầu ra của hệ thống.

+ Cập nhật biến trạng thái:

$$\hat{x}_k = \hat{x}_{k|k-1} + K_k \cdot \left(z_k - H \cdot \hat{x}_{k|k-1}\right)$$

Với

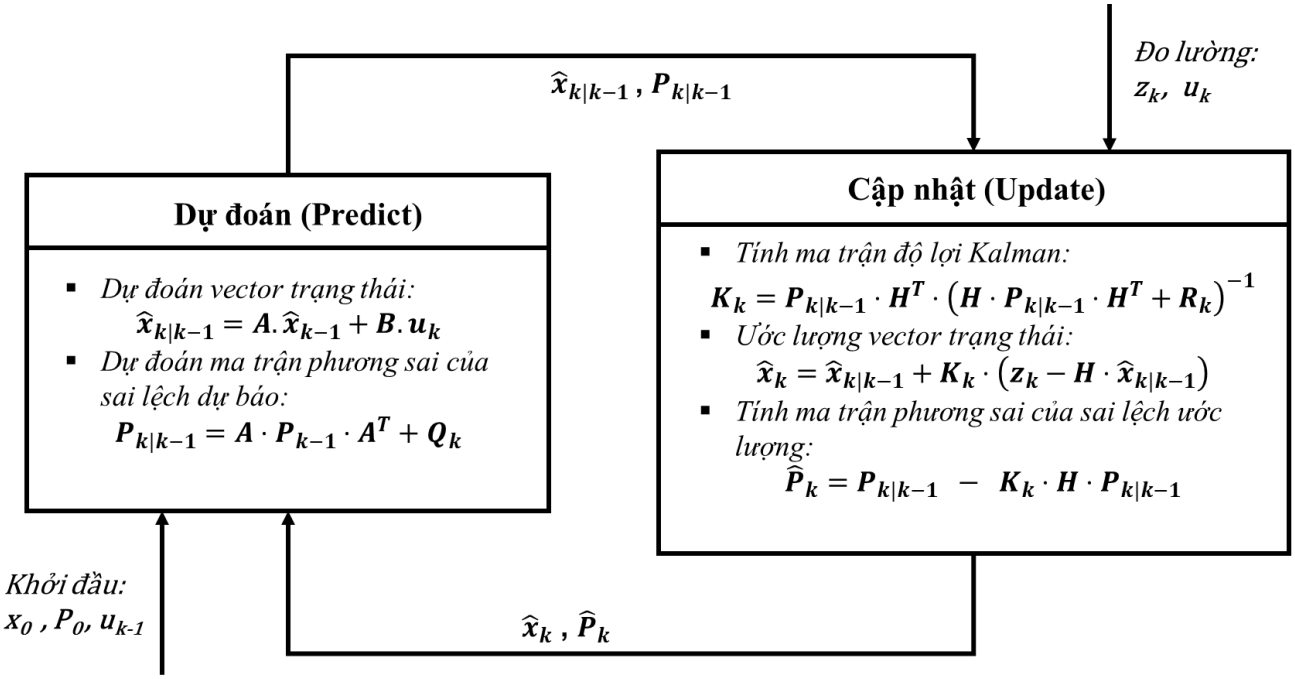
- $\hat{x}_k$: Biến trạng thái được cập nhật lại ở giá trị hiện tại.
- z_k: Tín hiệu thực đo lường từ cảm biến.
- $H \cdot \hat{x}_{k|k-1}$: Biến ước lượng từ hệ thống.

+ Cập nhật ma trận hiệp phương sai:

$$\hat{P}_k = P_{k|k-1} \; - \; K_k \cdot H \cdot P_{k|k-1}$$

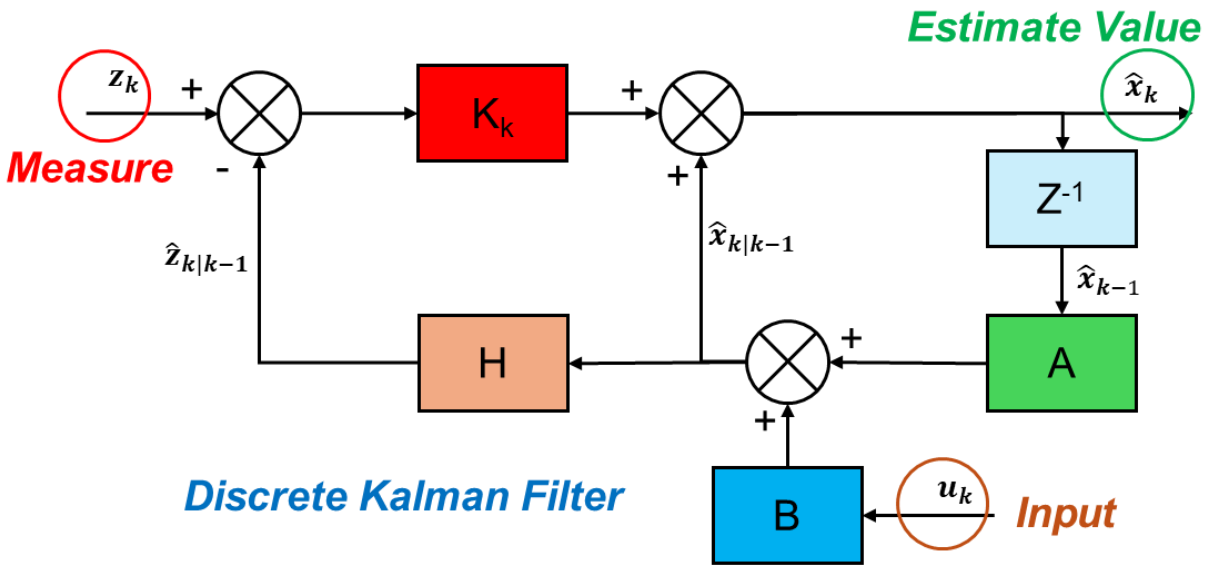
Với

- $\hat{P}_k$: Ma trận hiệp phương sai cập nhật.



Hình 2.1a Sơ đồ khối bộ lọc Kalman rời rạc

- Sơ đồ khối bộ lọc Kalman rời rạc:



Hình 2.2 Sơ đồ khối bộ lọc Kalman rời rạc

- Cần lựa chọn các giá trị đầu như $\hat{x}_0$ và P_0 .

❖ **Ưu điểm:**

- Kalman Filter là thuật toán hiệu quả và tối ưu trong việc ước lượng trạng thái của hệ thống tuyến tính với nhiễu Gaussian.
- Chỉ cần trạng thái hiện tại và quan sát mới, giúp tiết kiệm tài nguyên và phù hợp với ứng dụng thời gian thực.
- Dễ tích hợp vào các hệ thống điều khiển, có khả năng xử lý hệ thống đa biến và dự đoán trạng thái tương lai.

❖ **Nhược điểm:**

- Kalman Filter yêu cầu mô hình hệ thống chính xác và nhạy cảm với sai số trong tham số hoặc ma trận nhiễu, dẫn đến hiệu suất kém nếu mô hình không phù hợp.
- Nó không hiệu quả với hệ thống phi tuyến hoặc nhiễu phi Gaussian mà cần sử dụng các biến thể như EKF hoặc UKF.
- Hệ thống phức tạp, thuật toán đòi hỏi tài nguyên tính toán lớn và cần thời gian hội tụ lâu nếu khởi tạo không chính xác, trong khi việc điều chỉnh các tham số lại phụ thuộc vào kinh nghiệm hoặc phương pháp bổ trợ.

CHƯƠNG III : THIẾT KẾ BỘ LỌC KALMAN

3.1 Cơ sở nguyên lý :

3.1.1 Kalman cho các quay Roll và Pitch ứng với trục x và y:

- Ta có công thức tính góc quay dựa vào con quay hồi chuyển:

$$\theta(t) = \theta_0(t) + \int_{t_0}^{t} \omega(t) \cdot dt - \int_{t_0}^{t} \omega_{offset}(t) \cdot dt$$

- Trong đó:
  - θ : Góc xoay quanh 3 trục x,y,z (Roll , Pitch , Yaw).
  - θ_0 : Góc quay ban đầu hoặc ở trạng thái trước.
  - ω : Tốc độ góc từ gyroscope.
  - ω_{offset} : tốc độ góc bị trôi so với điểm zero.
  - t : thời gian.
- Thực hiện quá trình lấy mẫu :

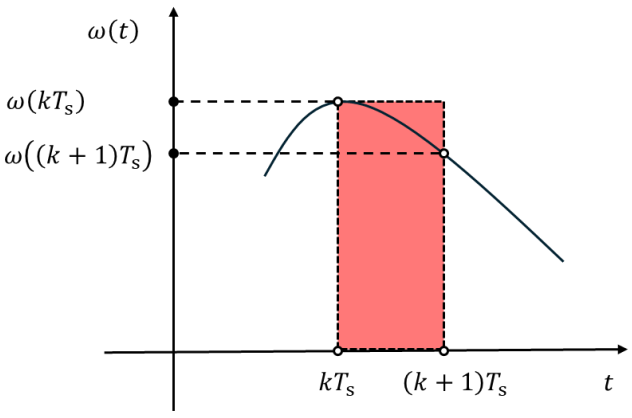
$$\theta((k+1)T_s) = \theta(kT_s) + \int_{kT_s}^{(k+1)T_s} \omega(t) \cdot dt - \int_{kT_s}^{t(k+1)T_s} \omega_{offset}(t) \cdot dt$$

- Sử dụng công thức xấp xỉ tích phân lùi cho $\int_{kT_s}^{(k+1)T_s} \omega(t) \cdot dt$ và tích phân lùi cho $\int_{kT_s}^{t(k+1)T_s} \omega_{offset}(t) \cdot dt$ ta được:

$$\theta((k+1)T_s = \theta(kT_s) + T_s \cdot \omega((k+1)T_s) - T_s \cdot \omega_{offset}((k+1)T_s)$$

- Rút gọn :

$$\theta(k+1) = \theta(k) + T_s \cdot \omega(k) - T_s \cdot \omega_{offset}(k)$$



$$\int_{kT_s}^{(k+1)T_s} \omega(t) \cdot dt \approx T_s \cdot \omega(k)$$

Hình 3.1 Tích phân lùi

- Chọn :

$$\boldsymbol{x}(k) = \begin{bmatrix} \theta(k) \\ \omega_{offset}(k) \end{bmatrix} \quad \text{và} \quad u(k) = \omega(k)$$

- Ta viết được phương trình trạng thái lý tưởng :

$$\boldsymbol{x}(k+1) = \boldsymbol{A} \cdot x(k) + \boldsymbol{B} \cdot u(k)$$

Với :

$$\boldsymbol{A} = \begin{bmatrix} 1 & -T_s \\ 0 & 1 \end{bmatrix} \quad và \quad \boldsymbol{B} = \begin{bmatrix} T_s \\ 0 \end{bmatrix}$$

- Phương trình đo lường :

$$\boldsymbol{z}(k) = \boldsymbol{H} \cdot \boldsymbol{x}(k) = \theta_{meas}(k)$$

Với :

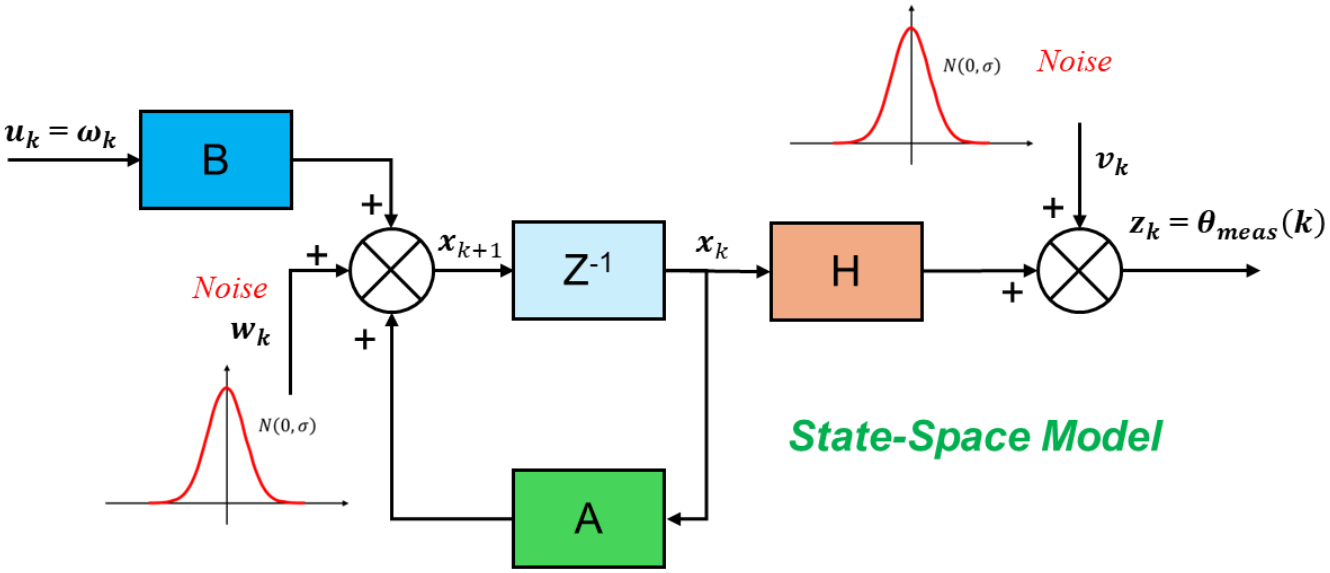
$$\boldsymbol{H} = [1 \quad 0]$$

- Hệ thống hoạt động trong môi trường nhiễu sẽ có hệ phương trình trạng thái :

$$\begin{cases} \boldsymbol{x}(k+1) = \boldsymbol{A} \cdot \boldsymbol{x}(k) + \boldsymbol{B} \cdot u(k) + \boldsymbol{w}(k) \\ \boldsymbol{z}(k) = \boldsymbol{H} \cdot \boldsymbol{x}(k) + \boldsymbol{v}(k) \end{cases}$$

Với :

$$\boldsymbol{A} = \begin{bmatrix} 1 & -T_s \\ 0 & 1 \end{bmatrix} ; \ \boldsymbol{B} = \begin{bmatrix} T_s \\ 0 \end{bmatrix} ; \ \boldsymbol{H} = [1 \quad 0] ; \ \boldsymbol{w}(k) \sim N(0, \boldsymbol{Q}_k) ; \ \boldsymbol{v}(k) \sim N(0, \boldsymbol{R}_k)$$



Hình 3.2 Sơ đồ khối của phép đo góc Roll và Pitch

❖ Thiết kế bộ lọc Kalman cho góc Roll và Pitch:

- Các bước tính toán trạng thái của bộ lọc :

❖ **Bước 1 :** Dự đoán :

+ Dự đoán trạng thái hiện tại từ trạng thái trước đó:

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{A}.\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{B}.u_{k-1}$$

$$\Rightarrow \quad \hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}} = \begin{bmatrix} \hat{\theta}_{k|k-1} \\ \widehat{\omega_o}_{k|k-1} \end{bmatrix} = \begin{bmatrix} 1 & -T_s \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \hat{\theta}_{k-1} \\ \widehat{\omega_o}_{k-1} \end{bmatrix} + \begin{bmatrix} T_s \\ 0 \end{bmatrix} \cdot \omega_k$$

$$= \begin{bmatrix} \hat{\theta}_{k-1} - T_s \cdot \widehat{\omega_o}_{k-1} + T_s \cdot \omega_k \\ \widehat{\omega_o}_{k-1} \end{bmatrix}$$

+ Dự đoán ma trận phương sai tương quan từ trạng thái trước đó:

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

- Giả sử : $\boldsymbol{P}_{k-1} = \begin{bmatrix} p_{k-1}^1 & p_{k-1}^2 \\ p_{k-1}^3 & p_{k-1}^4 \end{bmatrix}$; $\boldsymbol{Q}_k = \begin{bmatrix} q_1 & 0 \\ 0 & q_2 \end{bmatrix}$ (phương sai nhiễu mô hình với q_1 là phương sai góc quay , q_2 là phương sai tốc độ góc offset)
- Khi đó :

$$P_{k|k-1} = \begin{bmatrix} 1 & -T_s \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} p_{k-1}^1 & p_{k-1}^2 \\ p_{k-1}^3 & p_{k-1}^4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ -T_s & 1 \end{bmatrix} + \begin{bmatrix} q_1 & 0 \\ 0 & q_2 \end{bmatrix}$$

Suy ra

$$\begin{bmatrix} p_{k|k-1}^1 & p_{k|k-1}^2 \\ p_{k|k-1}^3 & p_{k|k-1}^4 \end{bmatrix} = \begin{bmatrix} \{ p_{k-1}^1 - T_s \cdot (p_{k-1}^2 + p_{k-1}^3) + T_s^2 \cdot p_{k-1}^4 + q_1 \} & \{ p_{k-1}^2 - T_s \cdot p_{k-1}^4 \} \\ \{ p_{k-1}^3 - T_s \cdot p_{k-1}^4 \} & \{ p_{k-1}^4 + q_2 \} \end{bmatrix}$$

Đặt :

$$\begin{bmatrix} p_{k|k-1}^1 & p_{k|k-1}^2 \\ p_{k|k-1}^3 & p_{k|k-1}^4 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Với :

$$\begin{cases} a = p_{k-1}^1 - T_s \cdot (p_{k-1}^2 + p_{k-1}^3) + T_s^2 \cdot p_{k-1}^4 + q_1 \\ b = p_{k-1}^2 - T_s \cdot p_{k-1}^4 \\ c = p_{k-1}^3 - T_s \cdot p_{k-1}^4 \\ d = p_{k-1}^4 + q_2 \end{cases}$$

❖ **Bước 2 :** Tính ma trận độ lợi :

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

+ Với : $R_k = r$ (phương sai của nhiễu đo lường ở đây là góc quay ước lượng từ gia tốc).

$$\boldsymbol{P}_{k|k-1} \cdot \boldsymbol{H}^T = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} a \\ c \end{bmatrix}$$

$$\Rightarrow \boldsymbol{H} \cdot \boldsymbol{P}_{\boldsymbol{k|k-1}} \cdot \boldsymbol{H}^T = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} a \\ c \end{bmatrix} = a$$

+ Vậy :

$$\boldsymbol{K}_k = \begin{bmatrix} a \\ c \end{bmatrix} \cdot \frac{1}{a+r}$$

Với :

$$\begin{cases} a = \ p_{k-1}^1 - T_s \cdot (p_{k-1}^2 + p_{k-1}^3) \ + \ T_s^2 \cdot p_{k-1}^4 \ + q_1 \\ c = \ p_{k-1}^3 - \ T_s \cdot p_{k-1}^4 \end{cases}$$

❖ **Bước 3:** Ước lượng trạng thái và ma trận phương sai:

+ Ước lượng biến trạng thái :

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \cdot \left(z(k) - \boldsymbol{H} \cdot \hat{\boldsymbol{x}}_{k|k-1}\right)$$

Suy ra :

$$\hat{\boldsymbol{x}}_k \ = \begin{bmatrix} \hat{\theta}_{k-1} \ - \ T_s \cdot \widehat{\omega_o}_{k-1} \ + \ T_s \cdot \omega_k \\ \widehat{\omega_o}_{k-1} \end{bmatrix} + \begin{bmatrix} a \\ c \end{bmatrix} \cdot \frac{(\theta_{meas}(k) - \hat{\theta}_{k|k-1})}{a+r}$$

+ Ước lượng ma trận phương sai :

$$\hat{\boldsymbol{P}}_k = \boldsymbol{P}_{k|k-1} \ - \ \boldsymbol{K}_k \cdot \boldsymbol{H} \cdot \boldsymbol{P}_{k|k-1}$$

Suy ra :

$$\hat{\boldsymbol{P}}_k \ = \begin{bmatrix} a & b \\ c & d \end{bmatrix} - \begin{bmatrix} a \\ c \end{bmatrix} \cdot \frac{1}{a+r} \cdot [1 \quad 0] \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Suy ra :

$$\hat{\boldsymbol{P}}_k \ = \begin{bmatrix} a & b \\ c & d \end{bmatrix} - \begin{bmatrix} a \\ c \end{bmatrix} \cdot [a \quad b] \cdot \frac{1}{a+r}$$

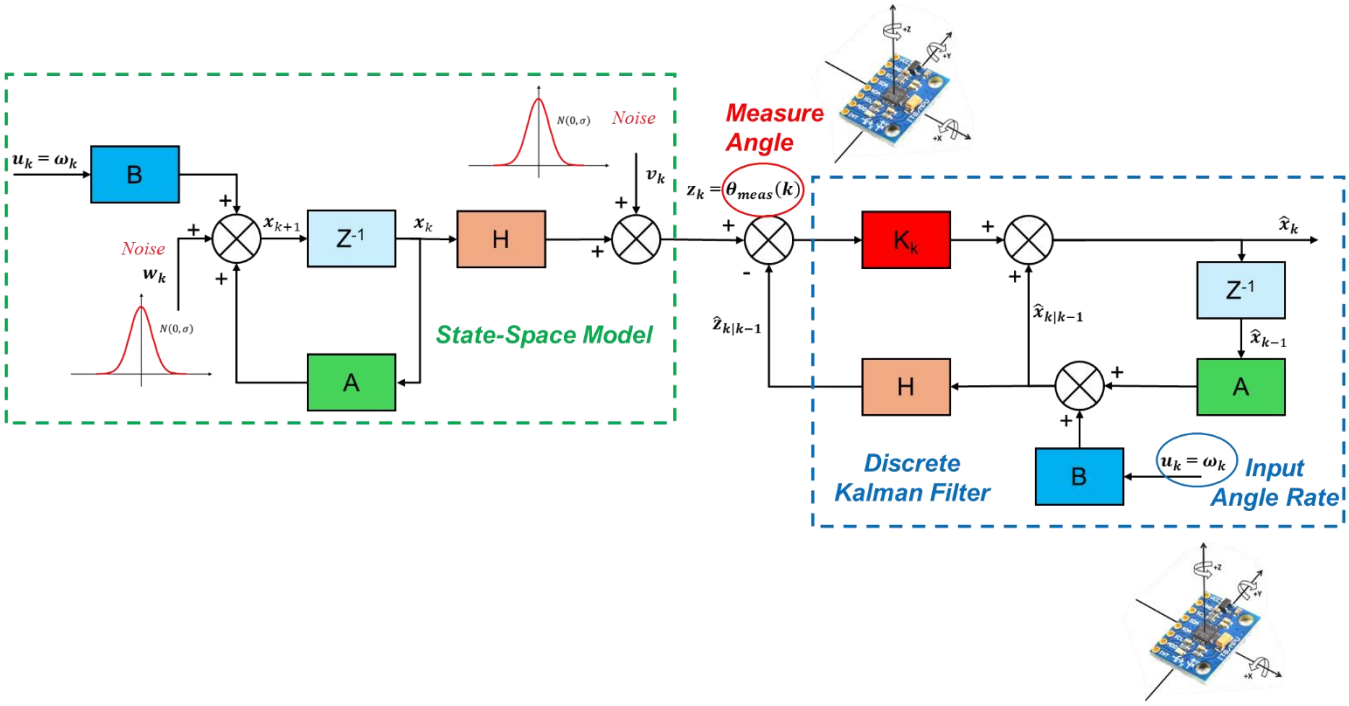
Suy ra :

$$\hat{\boldsymbol{P}}_k \ = \begin{bmatrix} a & b \\ c & d \end{bmatrix} - \begin{bmatrix} a^2 & ab \\ ac & ab \end{bmatrix} \cdot \frac{1}{a+r}$$

Quá trình sẽ được cập nhật lại ở lần thời gian lấy mẫu T_s kế tiếp . Khi đó, $\hat{\boldsymbol{P}}_k \rightarrow \boldsymbol{P}_{k-1}$ và $\hat{\boldsymbol{x}}_k \ \rightarrow \hat{\boldsymbol{x}}_{k-1}$. Biến trạng thái và ma trận phương sai ban đầu dự đoán là :

$$\hat{\boldsymbol{x}}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \text{ và } \ \boldsymbol{P}_0 \ = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- Sơ đồ khối :



Hình 3.3 Sơ đồ khối của hệ thống đối với góc Roll và Pitch

3.1.2 Kalman cho góc quay Yaw ứng với trục z:

- Hạn chế của cảm biến gia tốc không thể tính được trục Yaw , ta xây dựng mô hình trạng thái và thiết kế bộ lọc Kalman riêng cho phần ước lượng trục Yaw.
- Dựa vào kiến thức vật lý ta có quan hệ :

$$\theta(t) = \theta_0(t) + \int_{t_0}^{t} \omega(t) \cdot dt$$

- Trong đó:
  - θ : Góc xoay quanh 3 trục z (Yaw).
  - θ_0 : Góc quay ban đầu hoặc ở trạng thái trước.
  - ω : Tốc độ góc từ gyroscope.
  - t : thời gian.
- Khi thực hiện quá trình lấy mẫu với T_s ta có :

$$\theta((k+1)T_s) = \theta(kT_s) + \int_{kT_s}^{(k+1)T_s} \omega(t) \cdot dt$$

- Có thể xấp xỉ tích phân lùi thành dạng :

$$\int_{kT_S}^{(k+1)T_S} \omega(t) \cdot dt \approx T_s \cdot \omega(kT_s)$$

- Như vậy, ta có công thức sai phân tính góc xoay quanh 3 trục z (Yaw).

$$\theta((k+1)T_s) = \theta(kT_s) + T_s \cdot \omega(kT_s)$$

- Có thể rút gọn :

$$\theta(k+1) = \theta(k) + T_s \cdot \omega(k)$$

- Dựa vào quan hệ trên , ta xây dựng được phương trình trạng thái ở miền rời rạc với T_s ứng với với thời gian lấy mẫu của bộ xử lý số. Ta được phương trình trạng thái tuyến tính tuyến tính :

$$\begin{cases} \theta(k+1) = \theta(k) + T_s \cdot \omega(k) \\ \omega(k+1) = \omega(k) \end{cases}$$

- Đặt biến trạng thái :

$$\boldsymbol{x}(k) = \begin{bmatrix} \theta(k) \\ \omega(k) \end{bmatrix}$$

- Vậy ta có phương trình trạng thái không có biến điều khiển đầu vào :

$$\boldsymbol{x}(k+1) = \boldsymbol{A} \cdot \boldsymbol{x}(k) + \boldsymbol{B} \cdot u(k)$$

Với :

$$\boldsymbol{A} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \quad và \quad \boldsymbol{B} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- Phương trình đo lường, ta xem biến đo lường quan tâm đến là tốc độ góc (thành phần mang thông tin từ cảm biến) :

$$\boldsymbol{z}(k) = \boldsymbol{H} \cdot \boldsymbol{x}(k) = \omega_{meas}(k)$$

Với :

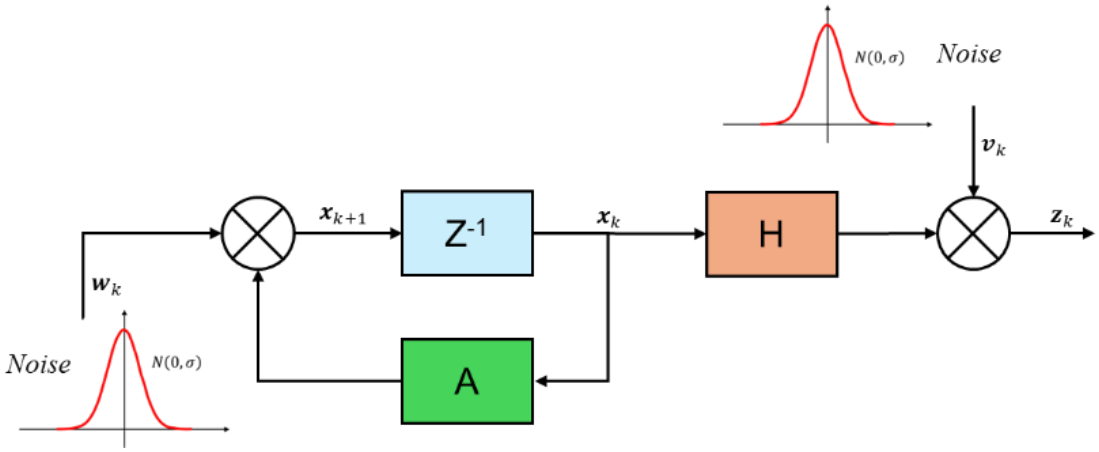
$$\boldsymbol{H} = [0 \quad 1]$$

- Hệ phương trình trạng thái của hệ thống : :

$$\begin{cases} \boldsymbol{x}(k+1) = \boldsymbol{A} \cdot \boldsymbol{x}(k) + \boldsymbol{w}(k) \\ \boldsymbol{z}(k) = \boldsymbol{H} \cdot \boldsymbol{x}(k) + v(k) \end{cases}$$

Với :

$$A = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \; ; \; H = [0 \quad 1] \; ; \; w(k) \sim N(0, Q_k) \; ; \; v(k) \sim N(0, R_k)$$



Hình 3.4 Sơ đồ khối phép đo góc Yaw

❖ Thiết kế bộ lọc Kalman cho góc Yaw:

- Các bước tính toán trạng thái của bộ lọc :

❖ **Bước 1 :** Dự đoán :

+ Dự đoán trạng thái hiện tại từ trạng thái trước đó:

$$\hat{x}_{k|k-1} = A.\hat{x}_{k-1} + B.u_{k-1}$$

$$\Rightarrow \quad \hat{x}_{k|k-1} = \begin{bmatrix} \hat{\theta}_{k|k-1} \\ \hat{\omega}_{k|k-1} \end{bmatrix} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} . \begin{bmatrix} \hat{\theta}_{k-1} \\ \hat{\omega}_{k-1} \end{bmatrix} = \begin{bmatrix} \hat{\theta}_{k-1} + T_s \cdot \hat{\omega}_{k-1} \\ \hat{\omega}_{k-1} \end{bmatrix}$$

+ Dự đoán ma trận phương sai tương quan từ trạng thái trước đó:

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

▪ Giả sử : $P_{k-1} = \begin{bmatrix} p^1_{k-1} & p^2_{k-1} \\ p^3_{k-1} & p^4_{k-1} \end{bmatrix}$; $Q_k = \begin{bmatrix} q_1 & 0 \\ 0 & q_2 \end{bmatrix}$ (phương sai nhiễu mô hình).

▪ Khi đó :

$$P_{k|k-1} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} . \begin{bmatrix} p^1_{k-1} & p^2_{k-1} \\ p^3_{k-1} & p^4_{k-1} \end{bmatrix} . \begin{bmatrix} 1 & 0 \\ T_s & 1 \end{bmatrix} + \begin{bmatrix} q_1 & 0 \\ 0 & q_2 \end{bmatrix}$$

Suy ra

$$\begin{bmatrix} p^1_{k|k-1} & p^2_{k|k-1} \\ p^3_{k|k-1} & p^4_{k|k-1} \end{bmatrix} = \begin{bmatrix} \{ p^1_{k-1} + T_s \cdot (p^2_{k-1} + p^3_{k-1}) + T_s^2 \cdot p^4_{k-1} + q_1 \} & \{ p^2_{k-1} + T_s \cdot p^4_{k-1} \} \\ \{ p^3_{k-1} + T_s \cdot p^4_{k-1} \} & \{ p^4_{k-1} + q_2 \} \end{bmatrix}$$

Đặt : $$\begin{bmatrix} p^1_{k|k-1} & p^2_{k|k-1} \\ p^3_{k|k-1} & p^4_{k|k-1} \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Với :

$$\begin{cases} a = p_{k-1}^1 + T_s \cdot (p_{k-1}^2 + p_{k-1}^3) + T_s^2 \cdot p_{k-1}^4 + q_1 \\ b = p_{k-1}^2 + T_s \cdot p_{k-1}^4 \\ c = p_{k-1}^3 + T_s \cdot p_{k-1}^4 \\ d = p_{k-1}^4 + q_2 \end{cases}$$

❖ **Bước 2 :** Tính ma trận độ lợi :

$$K_k = P_{k|k-1} \cdot H^T \cdot \left(H \cdot P_{k|k-1} \cdot H^T + R_k\right)^{-1}$$

+ Với : $R_k = r$ (phương sai của nhiễu đo lường)

$$P_{k|k-1} \cdot H^T = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}$$

$$\Rightarrow H \cdot P_{k|k-1} \cdot H^T = [0 \quad 1] \cdot \begin{bmatrix} a \\ c \end{bmatrix} = d$$

+ Vậy :

$$K_k = \begin{bmatrix} b \\ d \end{bmatrix} \cdot \frac{1}{d + r}$$

Với :

$$\begin{cases} b = p_{k-1}^2 + T_s \cdot p_{k-1}^4 \\ d = p_{k-1}^4 + q_2 \end{cases}$$

❖ **Bước 3:** Ước lượng trạng thái và ma trận phương sai:

+ Ước lượng biến trạng thái :

$$\hat{x}_k = \hat{x}_{k|k-1} + K_k \cdot \left(z(k) - H \cdot \hat{x}_{k|k-1}\right)$$

Suy ra :

$$\hat{x}_k = \begin{bmatrix} \hat{\theta}_{k-1} + T_s \cdot \hat{\omega}_{k-1} \\ \hat{\omega}_{k-1} \end{bmatrix} + \begin{bmatrix} b \\ d \end{bmatrix} \cdot \frac{(\omega_{meas}(k) - \hat{\omega}_{k-1})}{d+r}$$

+ Ước lượng ma trận phương sai :

$$\hat{P}_k = P_{k|k-1} - K_k \cdot H \cdot P_{k|k-1}$$

Suy ra :

$$\hat{P}_k = \begin{bmatrix} a & b \\ c & d \end{bmatrix} - \begin{bmatrix} b \\ d \end{bmatrix} \cdot \frac{1}{d + r} \cdot [0 \quad 1] \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Suy ra :

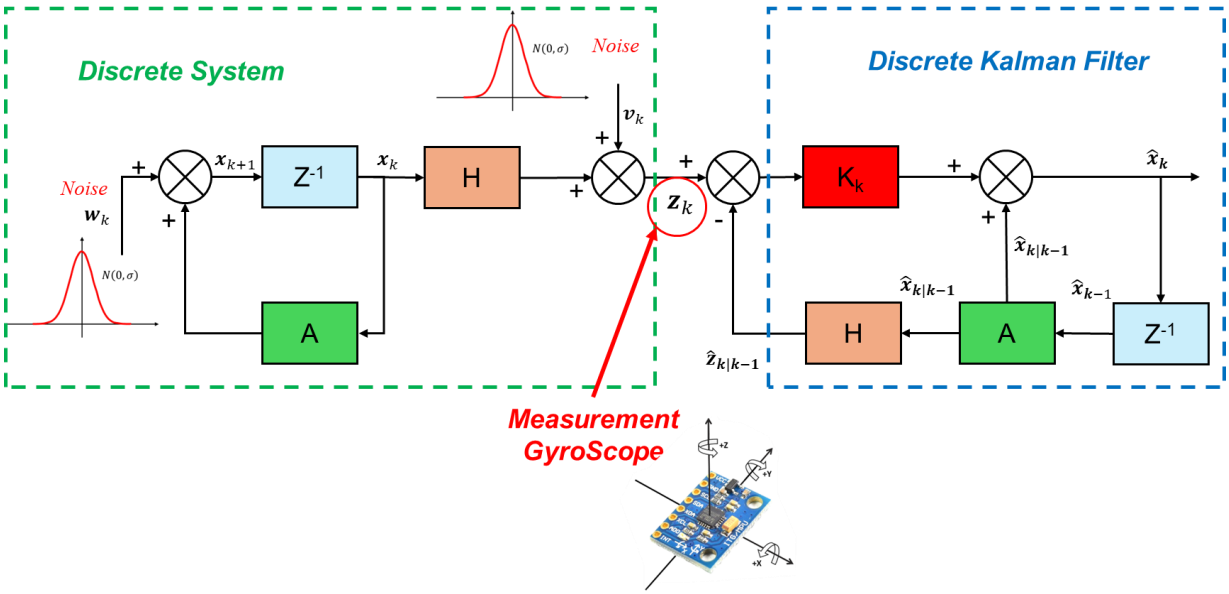
$$\hat{P}_k = \begin{bmatrix} a & b \\ c & d \end{bmatrix} - \begin{bmatrix} b \\ d \end{bmatrix} \cdot [c \quad d] \cdot \frac{1}{a+r}$$

Suy ra :

$$\hat{P}_k = \begin{bmatrix} a & b \\ c & d \end{bmatrix} - \begin{bmatrix} cb & bd \\ cd & d^2 \end{bmatrix} \cdot \frac{1}{d+r}$$

Quá trình sẽ được cập nhật lại ở lần thời gian lấy mẫu T_s kế tiếp . Khi đó, $\hat{P}_k \rightarrow P_{k-1}$ và $\hat{x}_k \rightarrow \hat{x}_{k-1}$. Biến trạng thái và ma trận phương sai ban đầu dự đoán là :

$$\hat{\boldsymbol{x}}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \text{ và } \boldsymbol{P}_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



Hình 3.5 Sơ đồ khối của hệ thống đối với góc Yaw

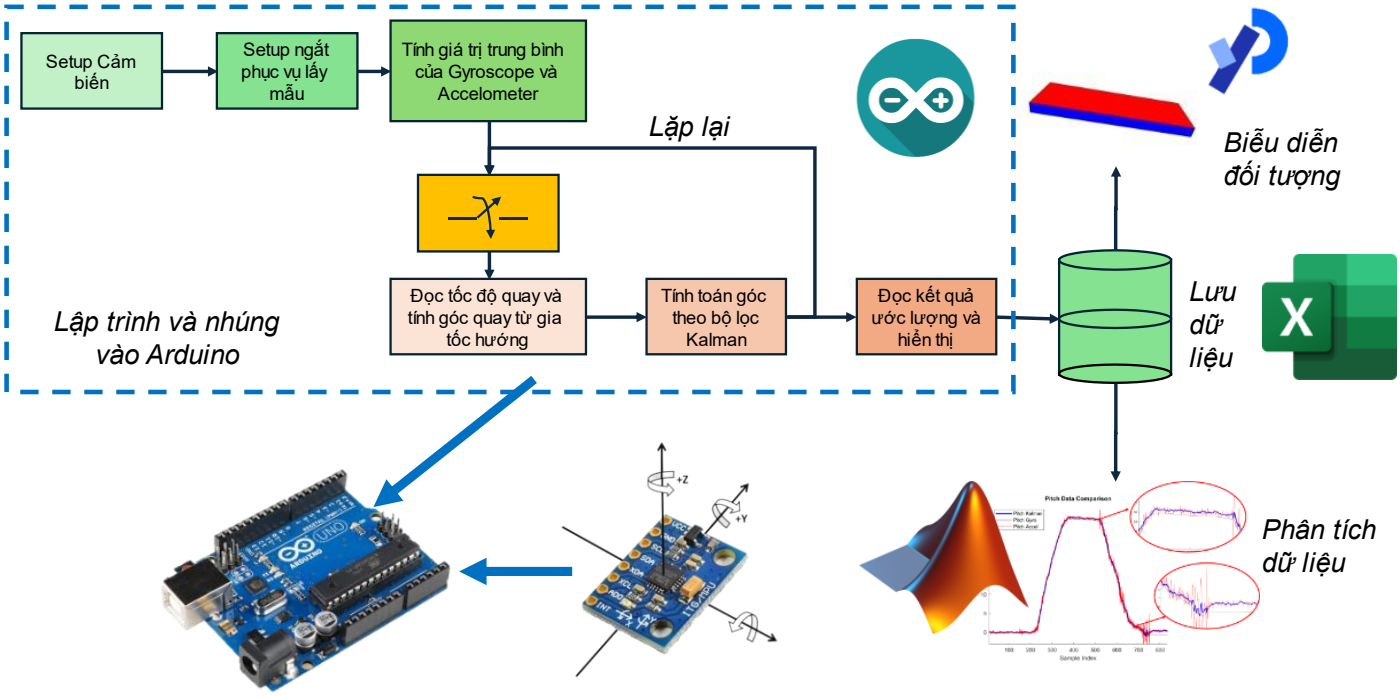
3.2 Lập trình trên bộ xử lý số:

- Ta cần khai báo một số biến cho quá trình.

Kiểu dữ liệu	Tên biến	Ý nghĩa
struct Gyro_type{ float GX; float GY; float GZ; } *// Kiểu dữ liệu tự tạo cho các biến có 3 thành phần GX,GY,GZ.*	**Gyro**	*Gyro là biến lưu giá trị tốc độ quay.*
struct Accel_type { float AX; float AY; float AZ; };*// Kiểu dữ liệu tự tạo cho các biến có 3 thành phần AX,AY,AZ.*	**Accel**	*Accel là biến lưu giá trị gia tốc hướng*
struct Angles_type { float roll; float pitch; };*// Kiểu dữ liệu tự tạo cho các biến có 2 thành phần roll, pitch.*	**Angles**	*Angles là biến lưu góc quay được tính toán từ thông tin gia tốc.*
struct Matx21{ float h11; float h21; }; *// Kiểu dữ liệu tự tạo cho ma trận 2x1*	**Xk_k, Yk_k, Zk_k**	*Biến lưu giá trị biến trạng thái trục X, Y, Z*
struct Matx22{ float h11; float h12; float h21; float h22; };*// Kiểu dữ liệu tự tạo cho ma trận 2x2*	**PXk_k, PYk_k, PZk_k**	*Biến lưu giá trị ma trận phương sai trục X, Y, Z*
float	**Offset_gx, Offset_gy, Offset_gz**	*Biến lưu giá trị bù Offset tốc độ góc trục X,Y,Z*
	Ogx, Ogy, Ogz	*Biến phục vụ quá trình tính Offset tốc độ góc trục X, Y, Z.*
	Offset_ax, Offset_ay, Offset_az	*Biến lưu giá trị bù Offset gia tốc hướng trục X,Y,Z*
	Ogx, Ogy, Ogz	*Biến phục vụ quá trình tính Offset gia tốc hướng trục X, Y, Z.*

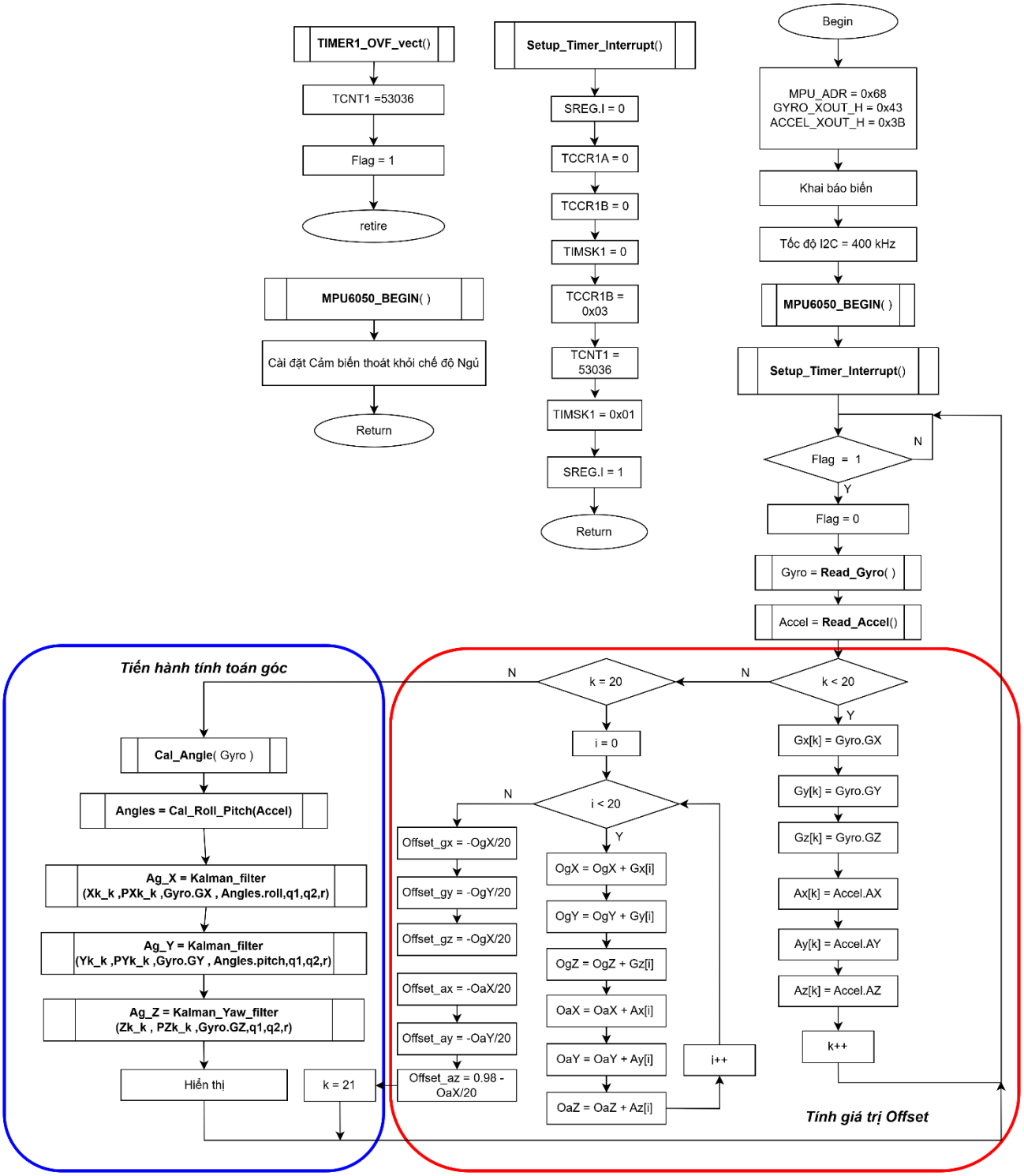
	AgX,AgX_1	*Biến tính toán góc trục X hiện tại và trạng thái trước chưa qua Kalman*
	AgY,AgY_1	*Biến tính toán góc trục Y hiện tại và trạng thái trước chưa qua Kalman*
	AgY,AgY_1	*Biến tính toán góc trục Z hiện tại và trạng thái trước chưa qua Kalman*
	Ag_Y,Ag_X,Ag_Z	*Biến góc ước lượng qua Kalman.*
	Gx[20], Gy[20], Gz[20]	*Mảng chứa giá trị tốc độ phục vụ việc tính trung bình.*
	Ax[20], Ay[20], Az[20]	*Mảng chứa giá trị gia tốc phục vụ việc tính trung bình.*
	r , q_1,q_2	*Gía trị mang thông tin phương sai đo lường và phương sai hệ thống.*
	Ts	*Thời gian lấy mẫu*
int	**k**	*Biến lặp phục vụ việc tính giá trị Offset.*
bool	**Flag**	*Cờ cho phép lấy mẫu cảm biến.*

- Quy trình thực hiện việc ước lượng góc quay theo bộ lọc Kalman:



Hình 3.6 Quy trình thực hiện việc ước lượng góc quay theo bộ lọc Kalman

❖ Lưu đồ thuật toán chương trình chính :



❖ Chương trình :

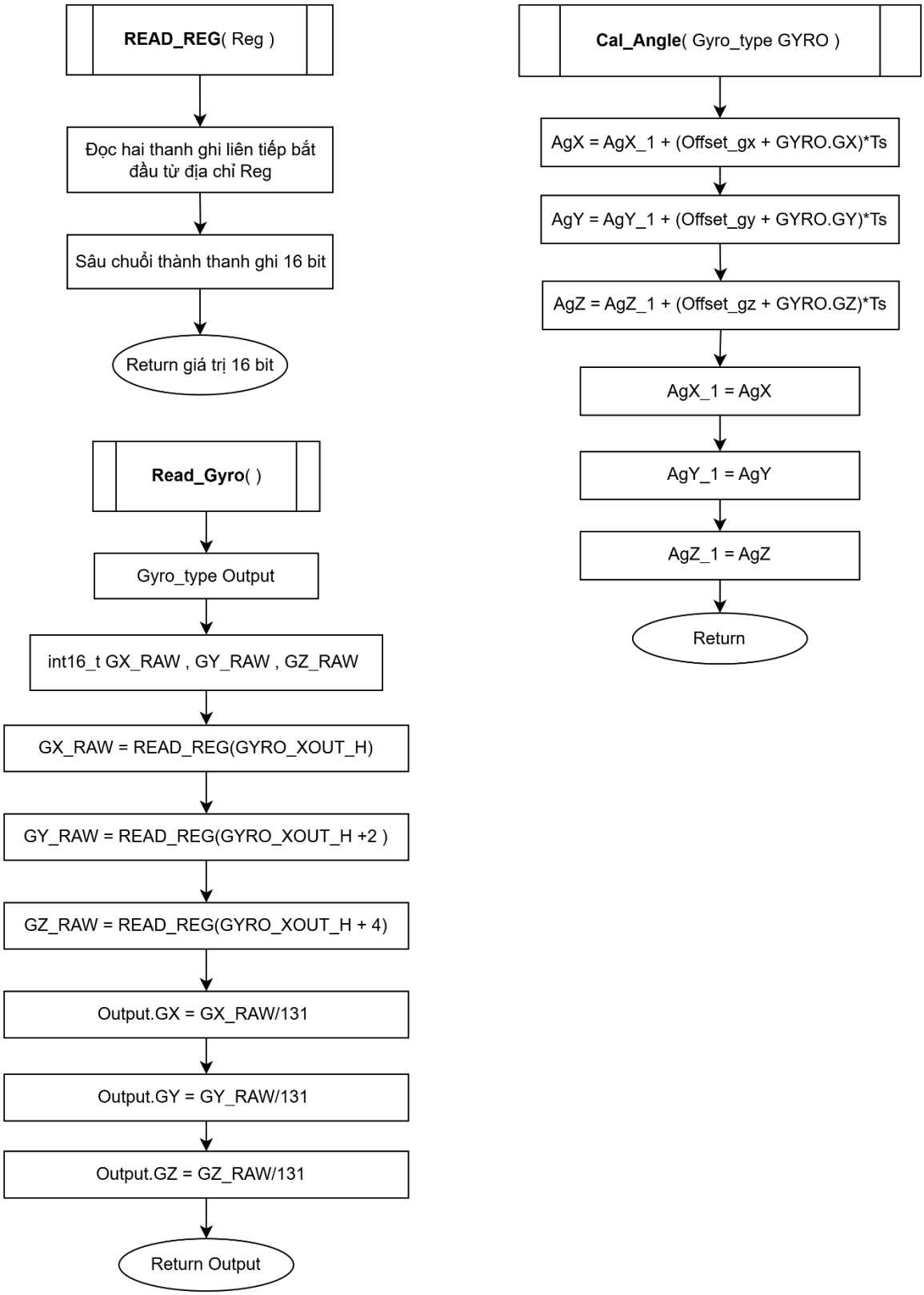
Code	Comment
void setup() { *// Cau hinh giao tiep.* *Serial.begin(9600); // Khoi tao Serial de xuat du lieu* *Wire.begin(); // Khoi tao giao tiep I2C* *Wire.setClock(400000); // Dat toc do I2C len 400kHz* *MPU6050_BEGIN(); // Khoi tao MPU6050*	*Cấu hình giao thức cũng như cấu hình ban đầu cho cảm biến .*
// Cau hinh ngat. *cli(); // Tat ngat toan cuc* *TCCR1A = 0;* *TCCR1B = 0;* *TCCR1B |= (1 << CS11) | (1 << CS10); // Prescaler = 64* *TCNT1 = 53036; // Gia tri khoi tao cho ngat 1ms* *TIMSK1 |= (1 << TOIE1); // Kich hoat ngat tran Timer1* *sei(); // Bat ngat toan cuc* *}*	*Cấu hình ngắt Timer phục vụ cho việc lấy mẫu.*
void loop() { *if(Flag == 1){* *Flag = 0;* *Gyro = Read_Gyro();* *Accel = Read_Accel();* *if(k < 20){* *Gx[k] = Gyro.GX;* *Gy[k] = Gyro.GY;* *Gz[k] = Gyro.GZ;* *//* *Ax[k] = Accel.AX;* *Ay[k] = Accel.AY;* *Az[k] = Accel.AZ;* *k++;* *}*	*Nếu k < 20 thì lấy mẫu liên tục tốc độ góc và gia tốc hướng lưu vào mảng*
else if(k == 20){ *for(int i = 0; i<20;i++){* *Ogx = Ogx + Gx[i];* *Ogy = Ogy + Gy[i];* *Ogz = Ogz + Gz[i];* *Oax = Oax + Ax[i];* *Oay = Oay + Ay[i];* *Oaz = Oaz + Az[i];* *}* *Offset_gx = -Ogx/20.0;* *Offset_gy = -Ogy/20.0;* *Offset_gz = -Ogz/20.0;* *Offset_ax = -Oax/20.0;* *Offset_ay = -Oay/20.0;* *Offset_az = 0.981 - Oaz/20;* *k = 21;* *}*	*Nếu k = 20 thì tiến hành tính giá trị trung bình và gán cho offset của mỗi trục ứng với giá trị tốc độ góc và gia tôc hướng*

```
            else{
            Cal_Angle(Gyro);
            Angles = Cal_Roll_Pitch(Accel);
            Ag_X = Kalman_filter(Xk_k ,PXk_k ,Gyro.GX ,
        Angles.roll,0.003,0.5,1.2);
            Ag_Y = Kalman_filter(Yk_k ,PYk_k ,Gyro.GY ,
        Angles.pitch,0.002,0.6,0.9);
            Ag_Z = Kalman_Yaw_filter(Zk_k , PZk_k
        ,Gyro.GZ,0.001,0.033,0.003);

            Serial.print(Ag_Y);
            Serial.print(","); // Dấu phân cách giữa các giá
        trị
            Serial.print(Ag_X);
            Serial.print(","); // Dấu phân cách giữa các giá
        trị
            Serial.println(Ag_Z); // Dấu newline để kết thúc
        một chuỗi
          }
          }
}
```

Nếu k > 21 thì bắt đầu tính toán tốc độ góc, góc quay chưa qua Kalman và đã qua Kalman

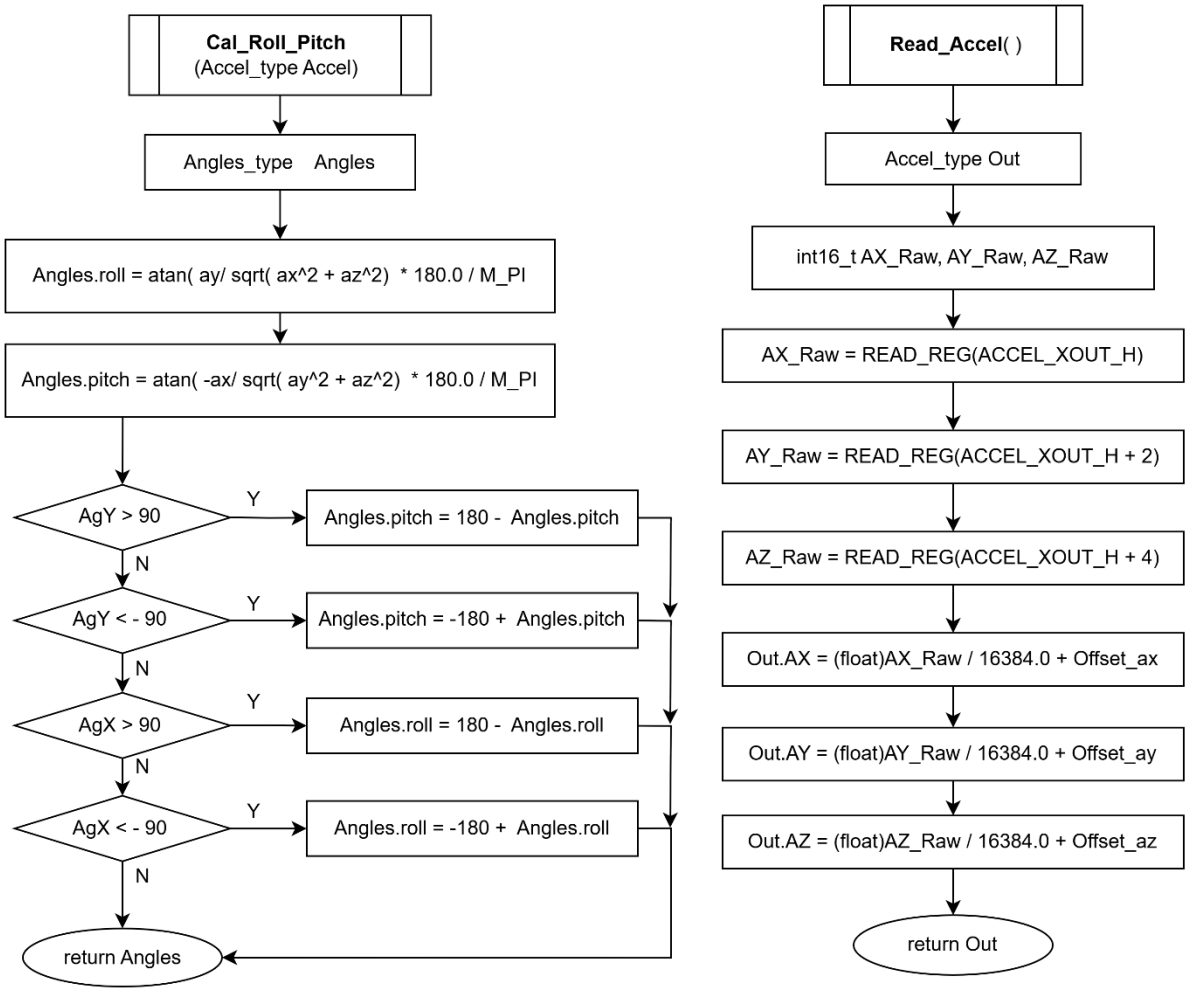
❖ Lưu đồ thuật toán chương trình phục vụ các hàm đọc tốc độ góc và tính góc quay dựa vào gyroscope:



❖ Chương trình :

Code	Comment
void MPU6050_BEGIN() { *Wire.beginTransmission(MPU6050_ADDR);* *Wire.write(0x6B); // Thanh ghi PWR_MGMT_1* *Wire.write(0); // Dat che do ngu ve 0 (danh thuc cam bien)* *Wire.endTransmission(true);* *}*	*Cấu hình cảm biến*
// Ham doc 16-bit tu mot thanh ghi bat dau *int16_t READ_REG(uint8_t reg) {* *Wire.beginTransmission(MPU6050_ADDR);* *Wire.write(reg);* *Wire.endTransmission(false);* *Wire.requestFrom(MPU6050_ADDR, 2, true); // Doc 2 byte du lieu* *int16_t value = (Wire.read() << 8) | Wire.read(); // Gop 2 byte lai thanh gia tri 16-bit* *return value;* *}*	*Hàm đọc giá trị thô cảm biến.*
Gyro_type Read_Gyro(){ *Gyro_type Out;* *int16_t GX_Raw,GY_Raw,GZ_Raw;* *GX_Raw = READ_REG(GYRO_XOUT_H);* *GY_Raw = READ_REG(GYRO_XOUT_H +2);* *GZ_Raw = READ_REG(GYRO_XOUT_H +4);* *Out.GX = GX_Raw/131.0 ;* *Out.GY = GY_Raw/131.0 ;* *Out.GZ = GZ_Raw/131.0 ;* *return Out;* *}*	*Hàm đọc tốc độ góc từ cảm biến*
void Cal_Angle(Gyro_type GYRO){ // Cong thuc so khai. *AgX = AgX_1 + (Offset_gx + GYRO.GX)*Ts;* *AgY = AgY_1 + (Offset_gy + GYRO.GY)*Ts;* *AgZ = AgZ_1 + (Offset_gz + GYRO.GZ)*Ts;* *AgX_1 = AgX;* *AgY_1 = AgY;* *AgZ_1 = AgZ;* *}*	*Hàm tính toán giá trị góc quay không qua Kalman*
ISR (TIMER1_OVF_vect) { *TCNT1 = 53036; // Tai lai gia tri cho 1ms* *if (Flag == 0) {* *Flag = 1;* *}* *}*	*Hàm Ngắt phục vụ lấy mẫu*

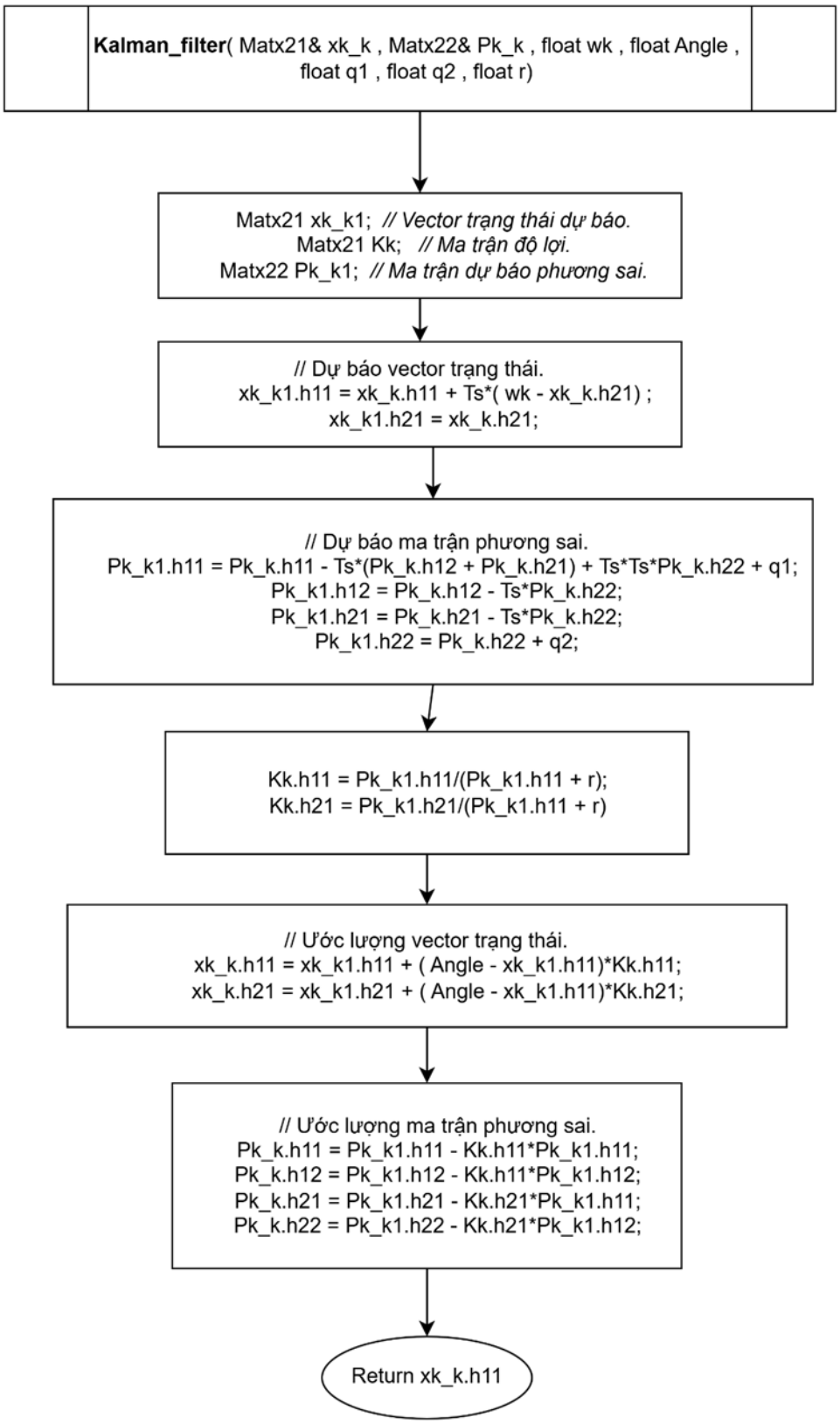
❖ Lưu đồ thuật toán chương trình phục vụ các hàm đọc gia tốc và tính góc quay dựa vào accelometer:



❖ Chương trình :

Code	Comment
Accel_type Read_Accel(){ *Accel_type Out;* *int16_t AX_Raw, AY_Raw, AZ_Raw;* *// Đọc giá trị thô từ các thanh ghi* *AX_Raw = READ_REG(ACCEL_XOUT_H); //* *Thanh ghi trục X* *AY_Raw = READ_REG(ACCEL_XOUT_H + 2); //* *Thanh ghi trục Y* *AZ_Raw = READ_REG(ACCEL_XOUT_H + 4); //* *Thanh ghi trục Z* *Out.AX = (float)AX_Raw / 16384.0 + Offset_ax ; //* *Tính gia tốc trục X* *Out.AY = (float)AY_Raw / 16384.0 + Offset_ay; //* *Tính gia tốc trục Y* *Out.AZ = (float)AZ_Raw / 16384.0 + Offset_az; //* *Tính gia tốc trục Z* *return Out;* *}*	*Hàm tính gia tốc hướng .*
Angles_type Cal_Roll_Pitch (Accel_type Accel) { *Angles_type Angles;* *// Tính Roll và Pitch* *// Tính Roll (xung quanh trục X)* *Angles.roll = atan2(Accel.AY, sqrt(Accel.AX * Accel.AX + Accel.AZ * Accel.AZ)) * 180.0 / M_PI;* *// Tính Pitch (xung quanh trục Y)* *Angles.pitch = atan2(-Accel.AX, sqrt(Accel.AY * Accel.AY + Accel.AZ * Accel.AZ)) * 180.0 / M_PI;* *if (AgY > 90) {* *Angles.pitch = 180 - Angles.pitch;* *}* *else if (AgY < -90) {* *Angles.pitch = -180 - Angles.pitch;* *}* *if (AgX > 90) {* *Angles.roll = 180 - Angles.roll;* *}* *else if (AgX < -90) {* *Angles.roll = -180 - Angles.roll;* *}* *return Angles;* *}*	*Hàm tính góc Pitch , Roll từ gia tốc hướng*

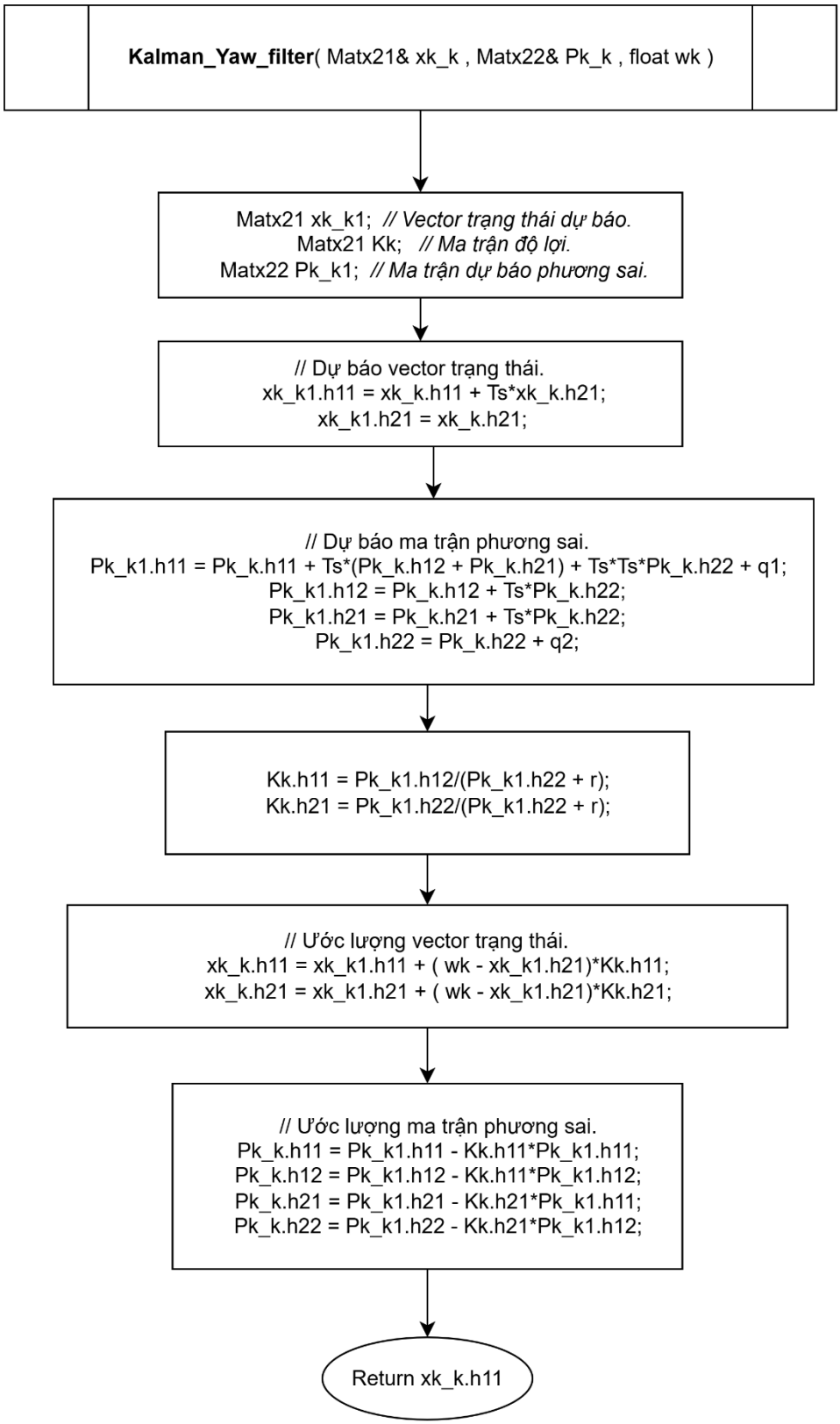
❖ Lưu đồ thuật toán Kalman cho góc Roll và Pitch :



❖ Chương trình :

```
float Kalman_filter( Matx21& xk_k , Matx22& Pk_k , float wk , float Angle , float q1 , float q2
, float r) {
    Matx21 xk_k1;  // Vector trạng thái dự báo.
    Matx21 Kk;   // Ma trận độ lợi.
    Matx22 Pk_k1;  // Ma trận dự báo phương sai.
    // Dự báo vector trạng thái.
    xk_k1.h11 = xk_k.h11 + Ts*( wk - xk_k.h21) ;
    xk_k1.h21 = xk_k.h21;
    // Dự báo ma trận phương sai.
    Pk_k1.h11 = Pk_k.h11 - Ts*(Pk_k.h12 + Pk_k.h21) + Ts*Ts*Pk_k.h22 + q1;
    Pk_k1.h12 = Pk_k.h12 - Ts*Pk_k.h22;
    Pk_k1.h21 = Pk_k.h21 - Ts*Pk_k.h22;
    Pk_k1.h22 = Pk_k.h22 + q2;
   // Tính ma trận độ lợi.
   Kk.h11 = Pk_k1.h11/(Pk_k1.h11 + r);
   Kk.h21 = Pk_k1.h21/(Pk_k1.h11 + r);
   // Ước lượng vector trạng thái.
   xk_k.h11 = xk_k1.h11 + ( Angle - xk_k1.h11)*Kk.h11;
   xk_k.h21 = xk_k1.h21 + ( Angle - xk_k1.h11)*Kk.h21;
   // Ước lượng ma trận phương sai.
   Pk_k.h11 = Pk_k1.h11 - Kk.h11*Pk_k1.h11;
   Pk_k.h12 = Pk_k1.h12 - Kk.h11*Pk_k1.h12;
   Pk_k.h21 = Pk_k1.h21 - Kk.h21*Pk_k1.h11;
   Pk_k.h22 = Pk_k1.h22 - Kk.h21*Pk_k1.h12;
   return xk_k.h11;
}
```

❖ Lưu đồ thuật toán Kalman cho góc Yaw:



❖ Chương trình :

```
float Kalman_Yaw_filter( Matx21& xk_k , Matx22& Pk_k , float wk , float q1 , float q2 , float r) {
    Matx21 xk_k1;  // Vector trạng thái dự báo.
    Matx21 Kk;   // Ma trận độ lợi.
    Matx22 Pk_k1;  // Ma trận dự báo phương sai.
    // Dự báo vector trạng thái.
    xk_k1.h11 = xk_k.h11 + Ts*xk_k.h21;
    xk_k1.h21 = xk_k.h21;
    // Dự báo ma trận phương sai.
    Pk_k1.h11 = Pk_k.h11 + Ts*(Pk_k.h12 + Pk_k.h21) + Ts*Ts*Pk_k.h22 + q1;
    Pk_k1.h12 = Pk_k.h12 + Ts*Pk_k.h22;
    Pk_k1.h21 = Pk_k.h21 + Ts*Pk_k.h22;
    Pk_k1.h22 = Pk_k.h22 + q2;
   // Tính ma trận độ lợi.
   Kk.h11 = Pk_k1.h12/(Pk_k1.h22 + r);
   Kk.h21 = Pk_k1.h22/(Pk_k1.h22 + r);
   // Ước lượng vector trạng thái.
   xk_k.h11 = xk_k1.h11 + ( wk - xk_k1.h21)*Kk.h11;
   xk_k.h21 = xk_k1.h21 + ( wk - xk_k1.h21)*Kk.h21;
   // Ước lượng ma trận phương sai.
   Pk_k.h11 = Pk_k1.h11 - Kk.h11*Pk_k1.h11;
   Pk_k.h12 = Pk_k1.h12 - Kk.h11*Pk_k1.h12;
   Pk_k.h21 = Pk_k1.h21 - Kk.h21*Pk_k1.h11;
   Pk_k.h22 = Pk_k1.h22 - Kk.h21*Pk_k1.h12;
   return xk_k.h11;
}
```

CHƯƠNG IV : KẾT QUẢ THỰC NGHIỆM

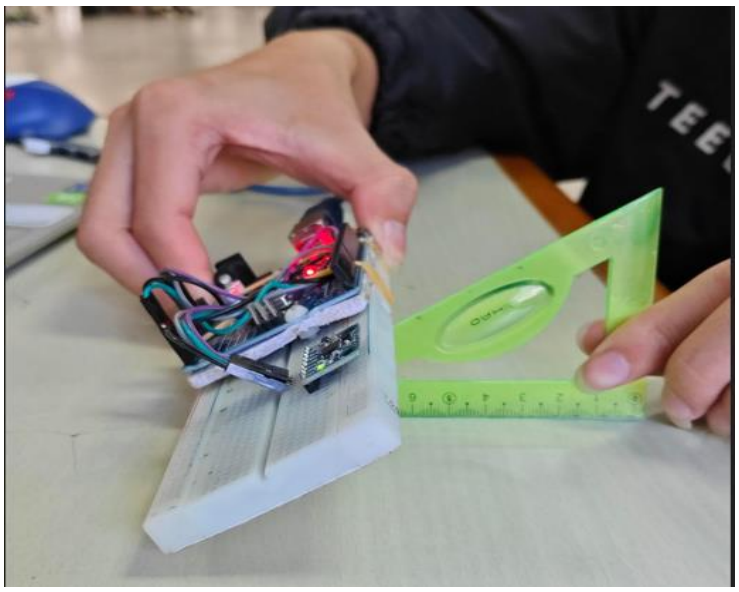
- Chọn thời gian lấy mẫu Ts = 0.02 s . Bằng thực nghiệm ta đo 200 mẫu ta tìm được phương sai q1 = , q2 = , r = ứng với 3 góc quay. Tiến hành thực nghiệm.

4.1 Góc quay xoay quanh trục y (Pitch):

- Đo đạc 200 mẫu các biến quá trình như góc quay ước lượng , tốc độ quay offset và góc quay thực từ cảm biến gia tốc. Ta tính được phương sai các biến trên :

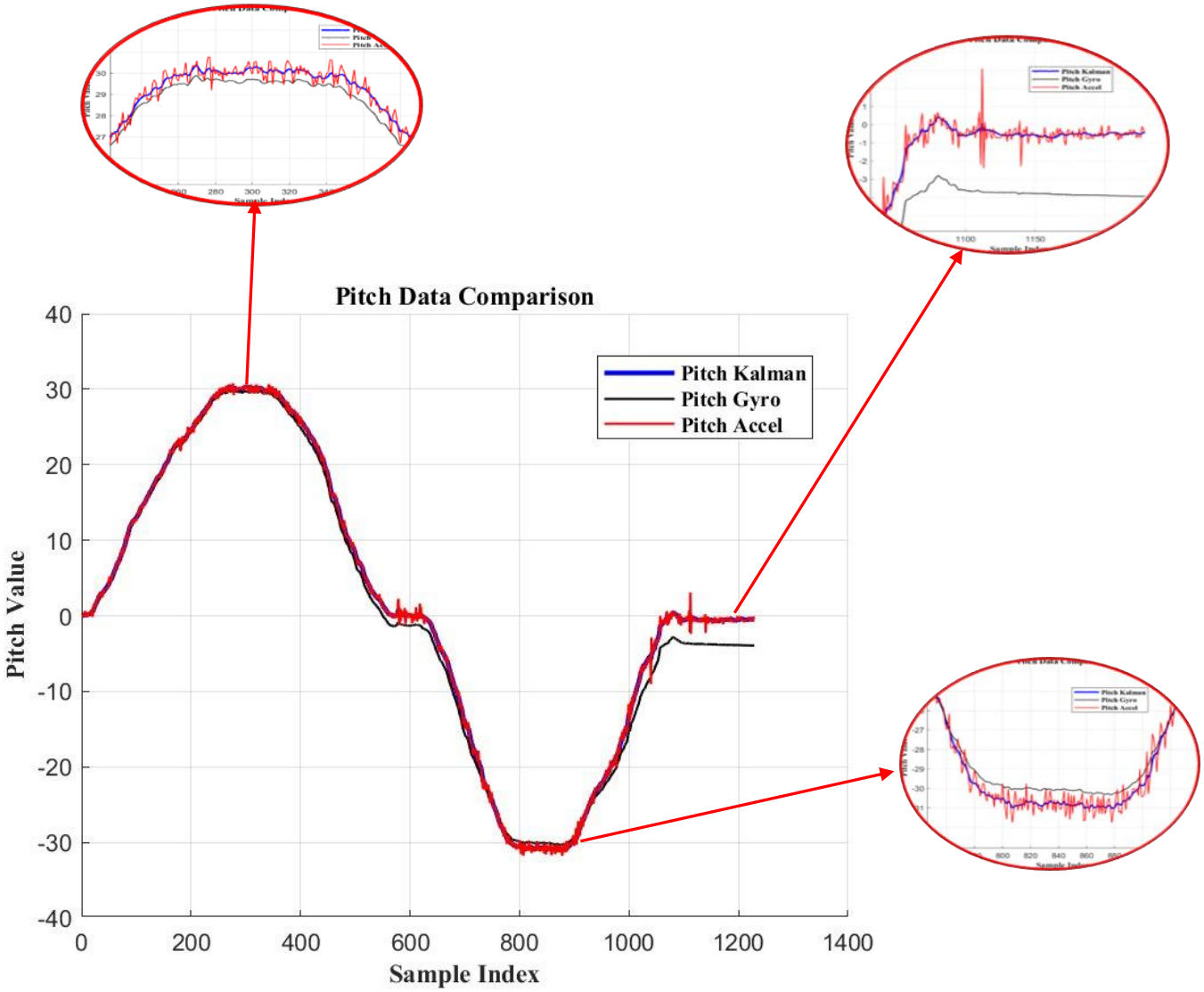
$$\begin{cases} \hat{\theta}_{k|k-1} = 0.003 \\ \widehat{\omega_o}_{k|k-1} = q_2 = 0.5 \\ \theta_{meas}(k) = r = 1.2 \end{cases}$$

- Thực nghiệm với các góc quay Pitch = $\pm 30^o$.



Hình 4.1 Thực nghiệm với các góc quay Pitch = $\pm 30^o$

- Kết quả:



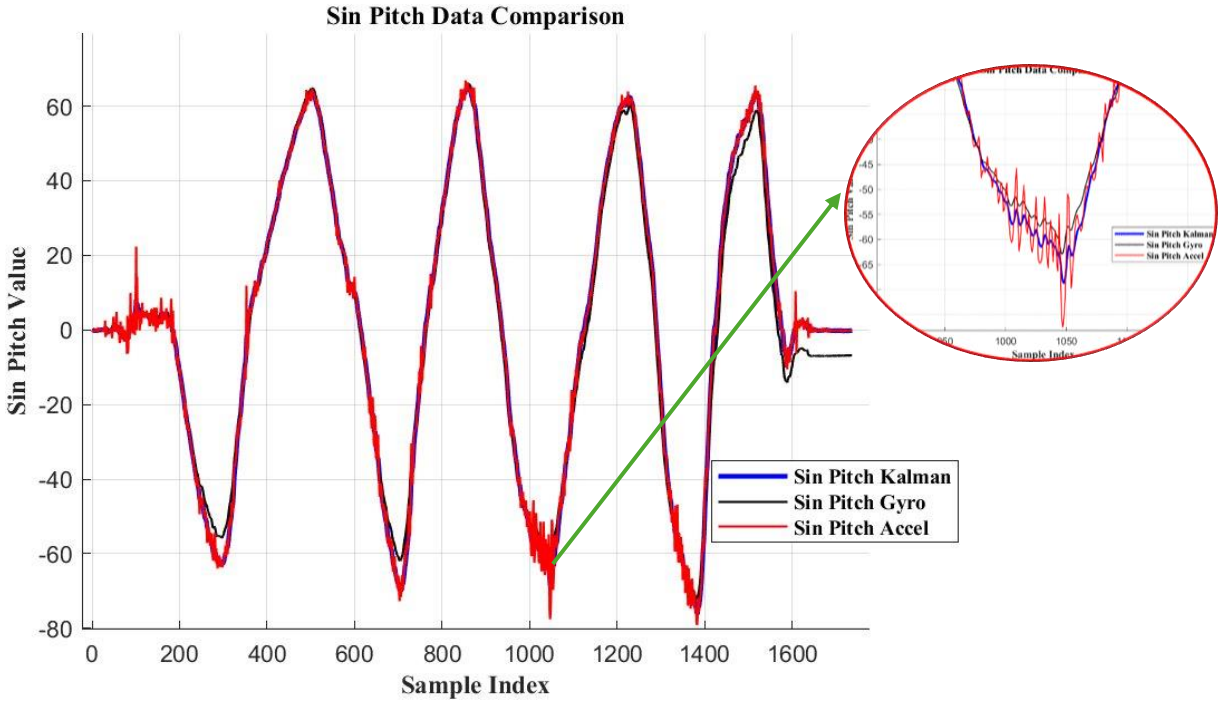
Hình 4.2 Kết quả thực nghiệm với góc quay Pitch = ± 30^o
(Đường màu xanh là góc đo bởi bộ lọc Kalman, đường màu đỏ góc đo bởi con quay hồi chuyển, màu đen đo bởi gia tốc kế)

- Nhận xét:
  - Tín hiệu góc Pitch được tính toán từ Gyroscope (đường màu đen): Dữ liệu từ con quay hồi chuyển tương đối ổn định tuy nhiên có xu hướng bị trôi (drift) theo thời gian. Ở những vùng giá trị ổn định dữ liệu bị lệch khá rõ so với thực tế. Số lần lấy mẫu càng tăng độ drift càng lớn, khi góc Pitch bằng 0 ở cuối quá trình, tín hiệu từ cách đo này không còn có thể tiến tới giá trị mong muốn.
  - Tín hiệu góc Pitch được tính toán từ Accelerometer (đường màu đỏ): Dữ liệu từ gia tốc kế bị nhiễu nhiều hơn các phương pháp khác do sự dịch chuyển của cảm biến

dẫn đến sinh ra gia tốc hướng, gây sai lệch gia tốc trọng trường. Tuy nhiên, không bị trôi như dữ liệu Gyroscope, mà thường ổn định hơn trong thời gian dài.

- Tín hiệu góc Pitch được tính toán từ Kalman filter (đường màu xanh): Đường biểu diễn từ bộ lọc Kalman mượt mà và chính xác hơn so với các phương pháp khác. Có khả năng loại bỏ nhiễu tốt, đặc biệt ở những vùng dữ liệu dao động mạnh (khoảng lấy mẫu 550 ÷ 600, 800 ÷ 900, …).

- Thực nghiệm cho góc Pitch dao động:



Hình 4.3 Kết quả thực nghiệm với góc quay Pitch dao động (Đường màu xanh là góc đo bởi bộ lọc Kalman, đường màu đỏ góc đo bởi con quay hồi chuyển, màu đen đo bởi gia tốc kế)

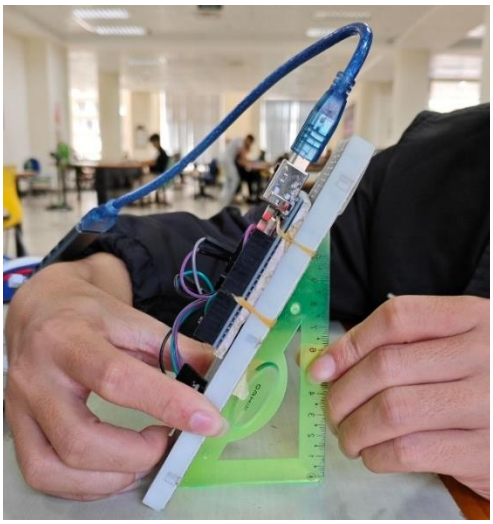
- Nhận xét: Qua các lần dao động mạnh, độ trôi của góc đo bằng con quay hồi chuyển ngày càng tăng dẫn đến sai số ngày càng lớn so với 2 phương pháp tính toán bằng bộ lọc Kalman và gia tốc kế. Ở cuối quá trình lấy mẫu độ sai lệch trở nên rõ rệt khi giá trị dữ liệu góc Pitch đo bằng con quay hồi chuyển không tiến tới 0 như trong thực tế.

4.2 Góc quay xoay quanh trục x (Roll):

- Đo đạc 200 mẫu các biến quá trình như góc quay ước lượng , tốc độ quay offset và góc quay thực từ cảm biến gia tốc. Ta tính được phương sai các biến trên :

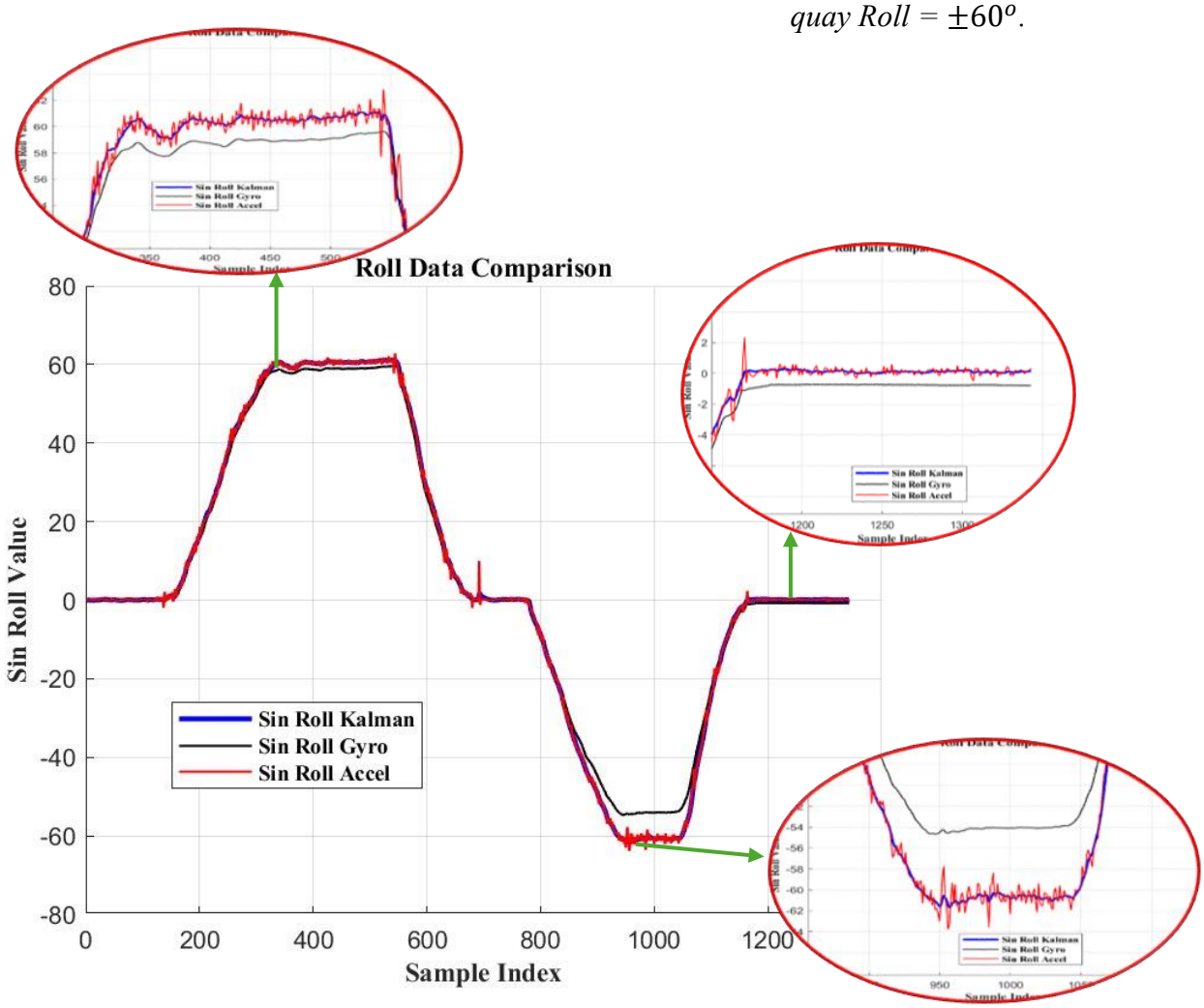
$$\begin{cases} \hat{\theta}_{k|k-1} = 0.002 \\ \widehat{\omega_o}_{k|k-1} = q_2 = 0.6 \\ \theta_{meas}(k) = r = 0.9 \end{cases}$$

- Thực nghiệm với các góc quay Roll = $\pm 60^o$.



Hình 4.4 Thực nghiệm với các góc quay Roll = $\pm 60^o$.

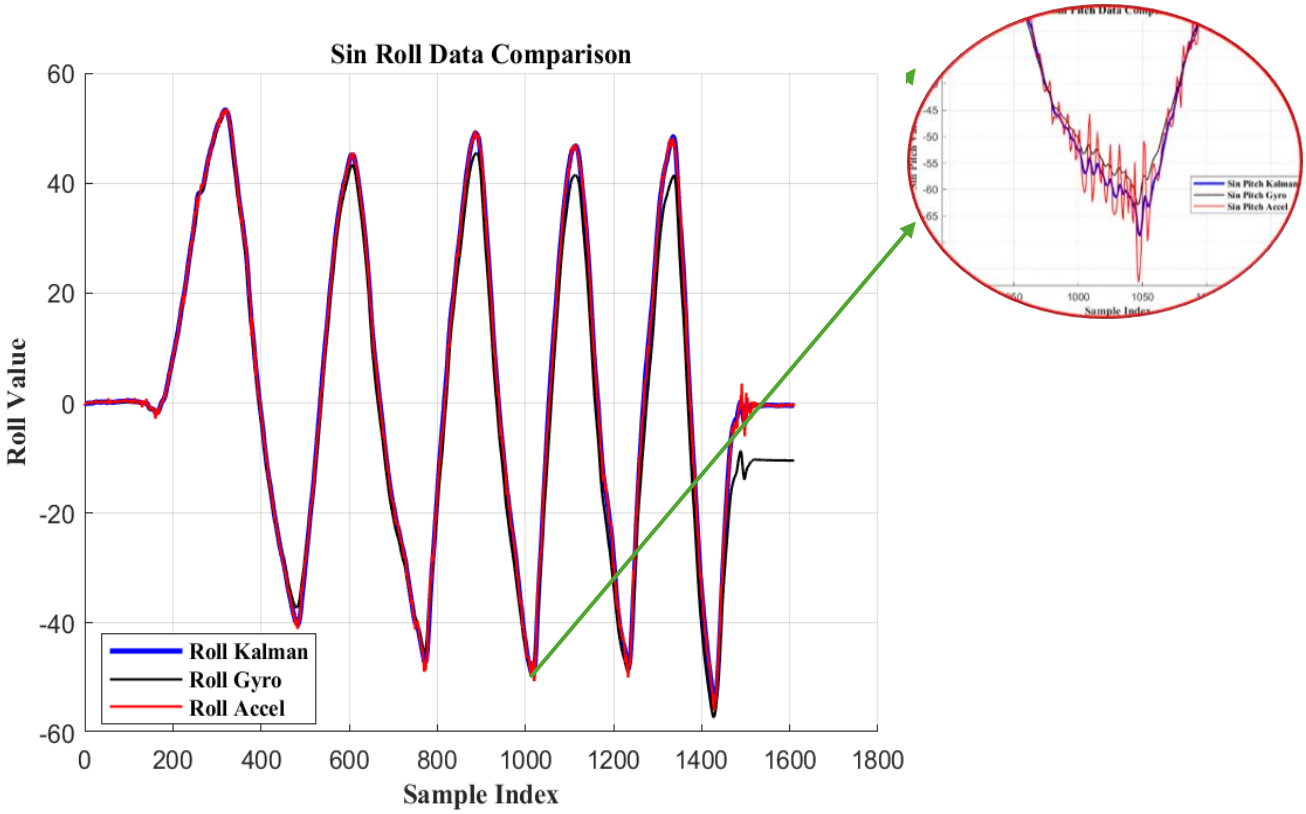
- Kết quả



Hình 4.5 Kết quả thực nghiệm với góc quay quay Roll = ± 30^o (Đường màu xanh là góc đo bởi bộ lọc Kalman, đường màu đỏ góc đo bởi con quay hồi chuyển, màu đen đo bởi gia tốc kế)

- Nhận xét:
  - Tín hiệu góc Pitch được tính toán từ Gyroscope (đường màu đen): Dữ liệu từ con quay hồi chuyển tương đối ổn định tuy nhiên có xu hướng bị trôi (drift) theo thời gian. Ở những vùng giá trị ổn định dữ liệu bị lệch khá rõ so với thực tế. Số lần lấy mẫu càng tăng độ drift càng lớn, khi góc Pitch bằng 0 ở cuối quá trình, tín hiệu từ cách đo này không còn có thể tiến tới giá trị mong muốn.
  - Tín hiệu góc Pitch được tính toán từ Accelerometer (đường màu đỏ): Dữ liệu từ gia tốc kế bị nhiễu nhiều hơn các phương pháp khác do sự dịch chuyển của cảm biến dẫn đến sinh ra gia tốc hướng, gây sai lệch gia tốc trọng trường. Tuy nhiên, không bị trôi như dữ liệu Gyroscope, mà thường ổn định hơn trong thời gian dài.
  - Tín hiệu góc Pitch được tính toán từ Kalman filter (đường màu xanh): Đường biểu diễn từ bộ lọc Kalman mượt mà và chính xác hơn so với các phương pháp khác. Có khả năng loại bỏ nhiễu tốt, đặc biệt ở những vùng dữ liệu dao động mạnh (khoảng lấy mẫu 350 ÷ 550, 950 ÷ 1050, …).

- Thực nghiệm cho góc Roll giao động:



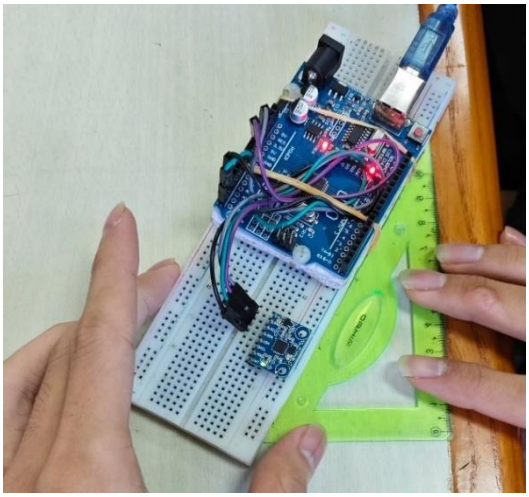
Hình 4.6 Kết quả thực nghiệm với góc quay quay Pitch Dao động (Đường màu xanh là góc đo bởi bộ lọc Kalman, đường màu đỏ góc đo bởi con quay hồi chuyển, màu đen đo bởi gia tốc kế)

4.3 Góc quay xoay quanh trục y (Yaw):

- Đo đạc 200 mẫu các biến quá trình như góc quay ước lượng , tốc độ quay offset và góc quay thực từ cảm biến gia tốc. Ta tính được phương sai các biến trên :

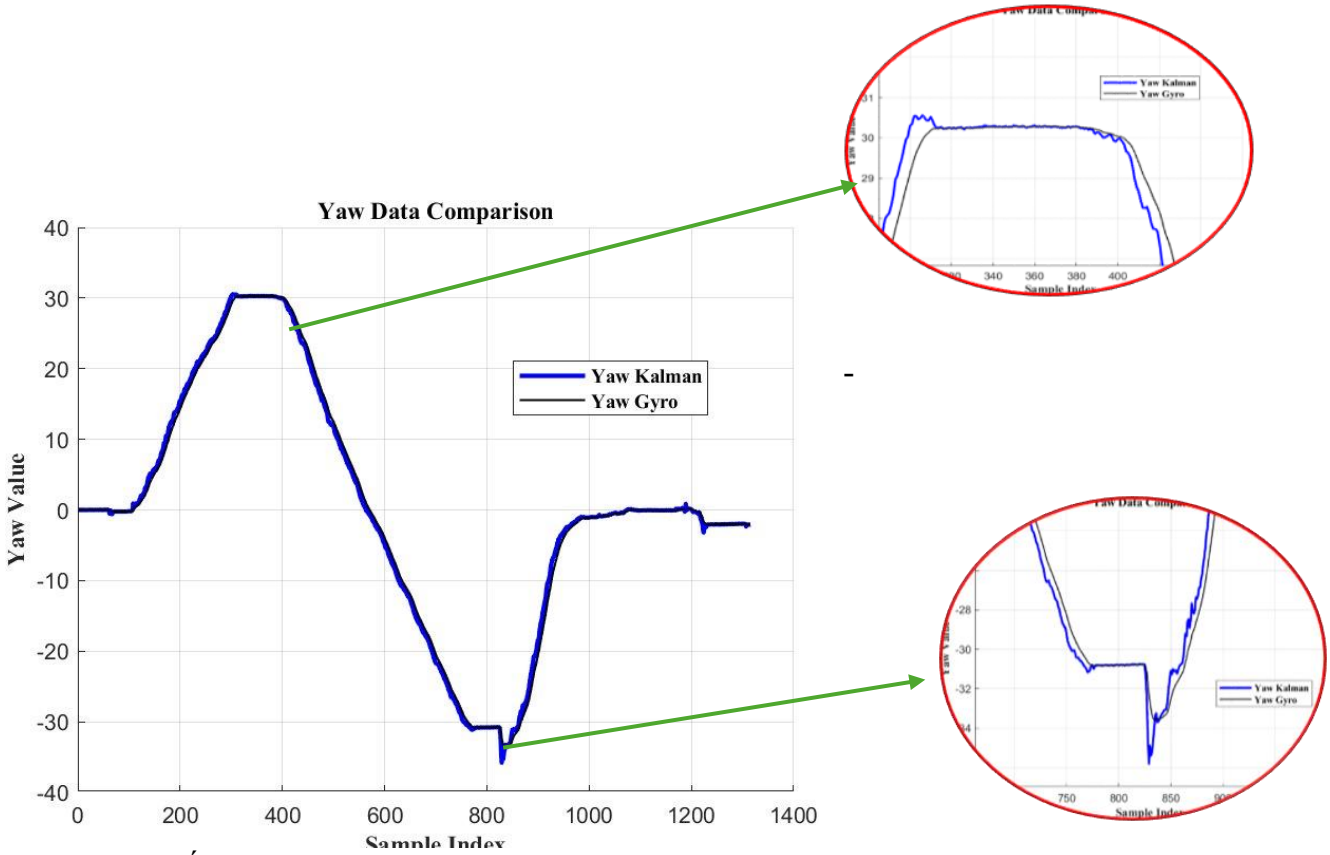
$$\begin{cases} \hat{\theta}_{k|k-1} = 0.001 \\ \hat{\omega}_{k|k-1} = \omega_{meas}(k) = q_2 = r = 0.003 \end{cases}$$

- Thực nghiệm với các góc quay Yaw = $\pm 30^o$.



Hình 3.7 Thực nghiệm với góc quay xoanh trục Yaw

- Kết quả:



Hình 4.8 Kết quả thực nghiệm với góc quay quay Yaw = ± 30^o (Đường màu xanh là góc đo bởi bộ lọc Kalman, đường màu đỏ góc đo bởi con quay hồi chuyển, màu đen đo bởi gia tốc kế)

Nhận xét: Bộ lọc Kalman chỉ sử dụng dữ liệu từ vận tốc góc (Gyroscope) để tính toán góc Yaw, vì cảm biến gia tốc (Accelerometer) không hiệu quả trên trục Z (trục Z song song với gia tốc trọng trường). Do đó kết quả tín toán góc Yaw từ bộ lọc Kalman và Gyroscope nhìn chung bám sát và tương đồng nhau. Cả hai đều có xu hướng bị trôi (drift) theo thời gian. Càng về sau độ trôi càng lớn do đó ở cuối quá trình lấy mẫu dữ liệu góc Yaw không thể tiến tới 0 như trong thực tế.

- Hướng phát triển: Sử dụng cảm biến MPU9250 có thêm thông tin về từ trường ngang ứng dụng được giá trị giá tốc kế vào bộ lọc Kalman.

4.4 Thực nghiệm định hướng đối tượng với phép quay Roll – Pitch – Yaw

- Khi biết được góc quay Roll – Pitch – Yaw của một đối tượng xoay tương ứng với 3 trục x – y – z . Ta xây dựng được phép quay Roll – Pitch – Yaw:
  + Phép quay quanh trục x (Roll ϕ):

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

  + Phép quay quanh trục y (Pitch θ):

$$R_y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

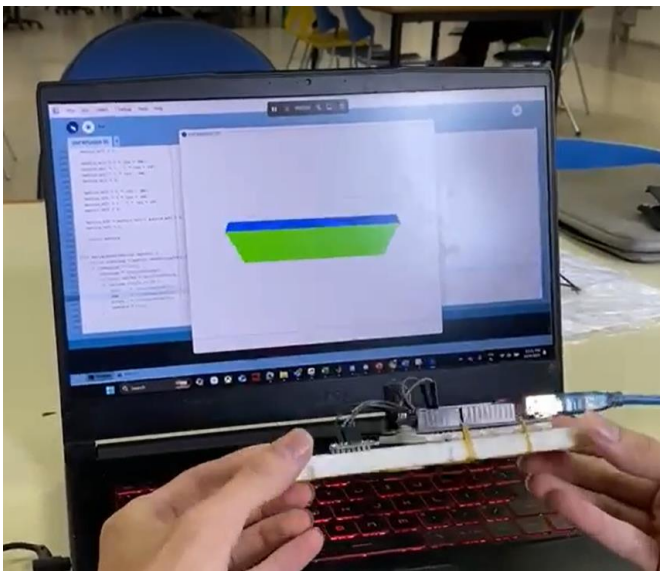
  + Phép quay quanh trục z (Yaw ψ):

$$R_z = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

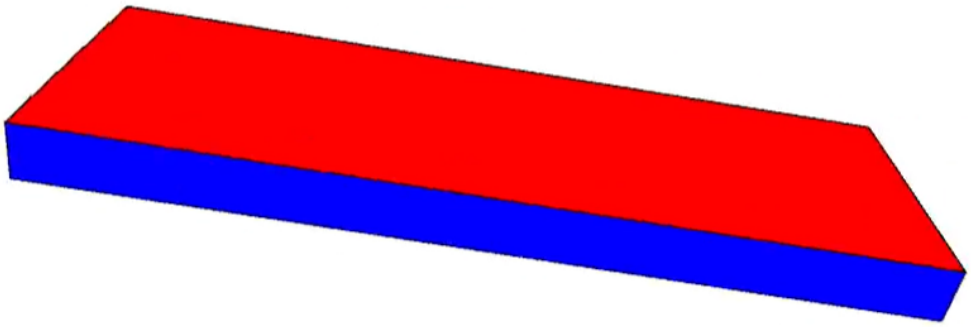
- Khi đó tổng hợp ma trận định hướng ta được:

$$R = R_x \cdot R_y \cdot R_z$$

- Dựa vào phép quay này, ta xây dựng được chương trình định hướng cho đối tượng trong không gian 3 chiều. Nhược điểm của phương pháp này là phép quay không phân biệt được sự đổi trục trong chuyển động của đối tượng (do tính chất không giao hoán của ma trận).
- Nhưng khi xét trong trường hợp không đổi trục thì việc định hướng vẫn cho kết quả tốt.



Hình 4.9 Thực nghiệm định hướng đối tượng



Hình 4.10 Mô phỏng đối tượng

- Nhận xét : Đánh giá các chuyển động quanh ở trục x (góc Roll) và y (góc Yaw) cho kết quả khá tốt, trục z (phép quay góc Yaw) do mô hình định hướng Kalman ít tín hiệu đầu vào hơn nên cho kết quả chưa tốt (vẫn xuất hiện tình trạng driff nếu để trong thời gian dài).

HẾT