CHƯƠNG 5: THIẾT KẾ, CHẾ TẠO VÀ LẬP TRÌNH THIẾT BỊ THEO DÕI CHUYỂN ĐỘNG CÁNH TAY

**5.1 Mục tiêu và định hướng thiết kế**

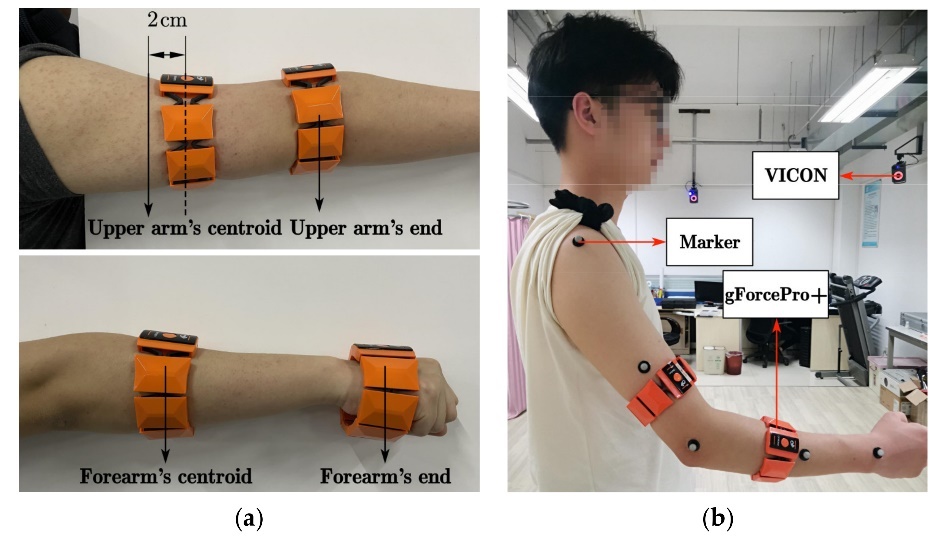
Trên thế giới hiện nay, việc điều khiển các cánh tay robot chủ yếu dựa vào việc lập trình trên các phần mềm robot hoặc điều khiển trực tiếp trên các thiết bị cầm tay như FlexPendant của hãng ABB, E-controller của Kawasaki, SmartPAD của KUKA, ... Điểm chung của các thiết bị này là đều sử dụng các nút nhấn, màn hình cảm ứng, ... để thao tác và lập trình các quỹ đạo cho robot.

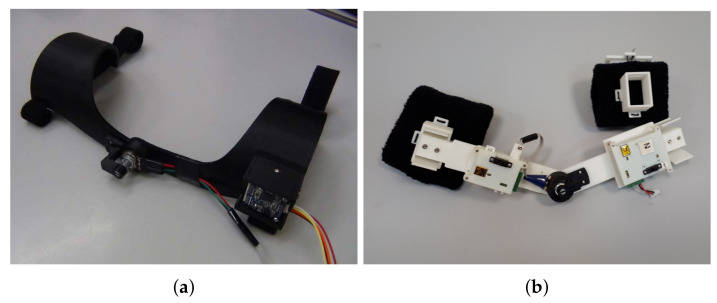


Tuy nhiên, những phương pháp này có nhược điểm là không được tự nhiên và tiện lợi khi phải thực hiện tạo từng quỹ đạo và chạy thử trong không gian thực tế.

Một trong những hướng tiếp cận mới trên thế giới trong việc điều khiển robot hiện nay chính là sử dụng các chuyển động thu được từ cơ thể con người để thực hiện điều khiển cánh tay robot:

Đề tài IMU Motion Capture Method with Adaptive Tremor Attenuation in Teleoperation Robot System năm 2022 đã sử dụng các gForcePro+ armband để thực hiện các định vị trí của cánh tay và lực tạo ra bởi các cơ để điều khiển cánh tay robot. Tuy nhiên, các thiết bị đeo có giá rất đắt và không phù hợp nếu chỉ điều khiển các cánh tay robot nhỏ.



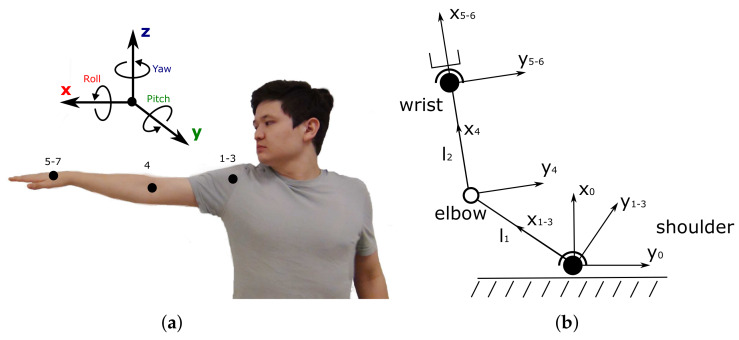
Đề tài An Open-Source 7-DOF Wireless Human Arm Motion-Tracking System for Use in Robotics Research sử dụng các cảm biến quán tính và biến trở để thực hiện tính toán. Với thiết kế và nguyên lí đơn giản, có thể giúp giảm chi phí chế tạo và điều khiển.



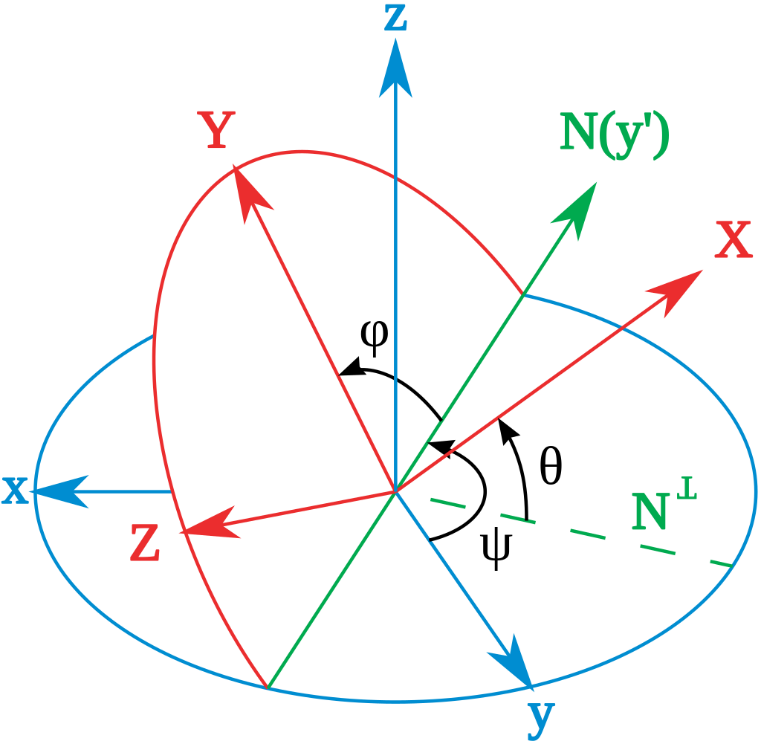
Theo đó, dựa vào nguyên lí của đề tài trên, ta sẽ thiết kế và chế tạo lại một thiết bị theo dõi chuyển động cánh tay người để thực hiện điều khiển cánh tay robot.

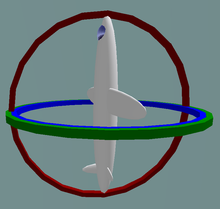
**5.2 Thiết kế nguyên lí và tính toán động học cho hệ thống**

- Kết quả cuối cùng mà hệ thống theo dõi chuyển động trả ra chính là vị trí của cánh tay trong không gian so với gốc tọa độ là vai. Ta có sơ đồ như sau:



Gỉa sử ta đã có được giá trị các góc roll, pitch, yaw của vai được đọc từ cảm biến IMU cùng với góc ở khủy tay được đọc từ biến trở,

Ta sẽ thực hiện nhân các ma trận chuyển vị để tính ra được vị trí x, y, z của đầu cánh tay.

Tuy nhiên, khi thực hiện chuyển các giá trị góc về roll, pitch, yaw, một hiện tượng gọi là Gimbal lock sẽ xảy ra làm cho các giá trị góc đọc về bị sai lệch.

Để tránh khỏi hiện tượng này, ta sẽ biểu diễn các góc xoay dưới dạng một ma trận Quaternion. Quaternion được định nghĩa như là một số phức có ba thành phần ảo:

\boldsymbol{q} = w + x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k} 

Đừng mường tượng quaternion là một cái gì đó bốn chiều, đầu bạn sẽ nổ tung. Các thành phần ảo i, j, k  có thể được coi như ba vector đơn vị của trục tọa độ x, y, z. Và một quaternion có thể được viết lại như sau:

\boldsymbol{q} = w + \boldsymbol{v} 

Với \boldsymbol{v} là một vector trong không gian 3 chiều, và w là một đại lượng vô hướng chỉ độ lớn (nói chung là một con số).

u_x^2 + u_y^2 + u_z^2 = 1 Áp dụng vào phép quay không gian của chúng ta. Giả sử bạn có một vector trục quay \boldsymbol{u}  (vector \boldsymbol{u}  phải là vector đơn vị, tức  ) và góc quay \theta (dương khi quay cùng chiều với vector, theo nguyên tắc nắm tay phải) thì chúng ta sẽ có một quaternion quay như sau:

\boldsymbol{q} = \cos \frac{1}{2} \theta + \boldsymbol{u} \sin \frac{1}{2} \theta = \cos \frac{1}{2} \theta + (u_x \boldsymbol{i} + u_y \boldsymbol{j} + u_z \boldsymbol{k}) \sin \frac{1}{2} \theta 

Trong một nghiên cứu được thực hiện vào năm 2009, Sebastian Madgwick đã phát triển một thuật toán lọc nhiễu sử dụng cho IMU và AHRS để thực hiện xử lí đọc được. Thuật toán có tên là Madgwick được open-source và cho ra kết quả là một ma trận Quaternion gồm 4 giá trị q0, q1, q2, q3, q4.

Tiếp theo, ta sẽ tiếp tục chuyển đổi ma trận Quaternion về ma trận quay để tiến hành tính toán.

1. Compute the square of the quaternion norm:

n = w^2 + x^2 + y^2 + z^2

2. If the norm is zero, the quaternion is invalid.

3. Compute the matrix elements as follows:

a11 = (w^2 + x^2 - y^2 - z^2) / n

a12 = 2 \* (x\*y - w\*z) / n

a13 = 2 \* (x\*z + w\*y) / n

a21 = 2 \* (x\*y + w\*z) / n

a22 = (w^2 - x^2 + y^2 - z^2) / n

a23 = 2 \* (y\*z - w\*x) / n

a31 = 2 \* (x\*z - w\*y) / n

a32 = 2 \* (y\*z + w\*x) / n

Sau khi ta có được ma trận xoay tại khớp vai, ta tiến hành nhân thêm các ma trận tịnh tiến và xoay để ra được vị trí cuối cùng.

Tổng hợp lại, tín hiệu từ cảm biến sẽ trải qua các bước sau:



Động học của hệ thống:

result =

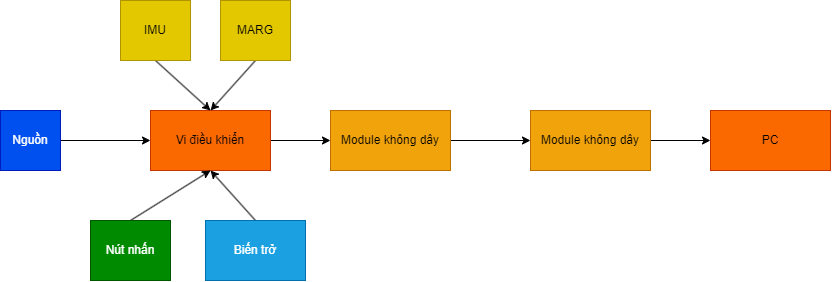
l2\*((ce\*(w^2 + x^2 - y^2 - z^2))/(w^2 + x^2 + y^2 + z^2) - (se\*(2\*w\*z - 2\*x\*y))/(w^2 + x^2 + y^2 + z^2)) + (l1\*(w^2 + x^2 - y^2 - z^2))/(w^2 + x^2 + y^2 + z^2)

l2\*((se\*(w^2 - x^2 + y^2 - z^2))/(w^2 + x^2 + y^2 + z^2) + (ce\*(2\*w\*z + 2\*x\*y))/(w^2 + x^2 + y^2 + z^2)) + (l1\*(2\*w\*z + 2\*x\*y))/(w^2 + x^2 + y^2 + z^2)

- l2\*((ce\*(2\*w\*y - 2\*x\*z))/(w^2 + x^2 + y^2 + z^2) - (se\*(2\*w\*x + 2\*y\*z))/(w^2 + x^2 + y^2 + z^2)) - (l1\*(2\*w\*y - 2\*x\*z))/(w^2 + x^2 + y^2 + z^2)

**5.3 Thiết kế hệ thống điều khiển**

Ta có tổng thể thiết kế hệ thống như sau:



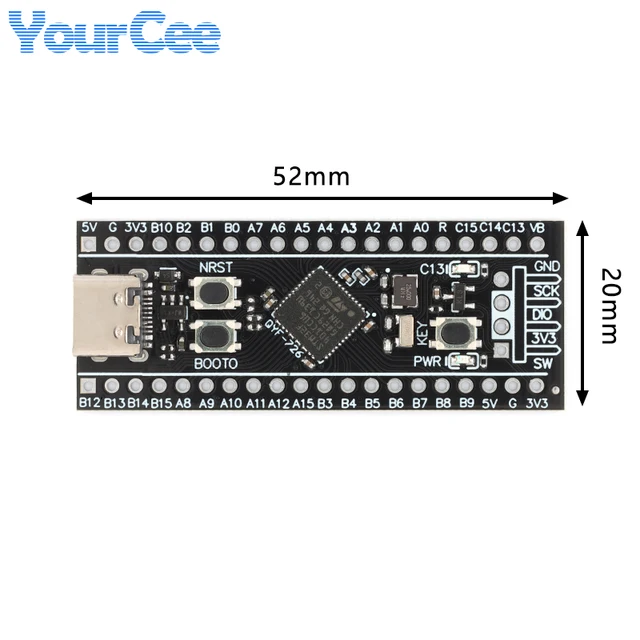
**5.4 Tính toán, chọn lựa linh kiện và thiết kế schematic**

1. Vi điều khiển

- Hiện nay, các dòng vi xử lí STM32 với kiến trúc ARM đang rất phổ biến với tốc độ tính toán nhanh và giá thành rẻ cùng nguồn hàng phổ biến.

- Với tiêu chí nhỏ gọn, tốc độ xử lí nhanh và giá thành hợp lí, ta sẽ lựa chọn các module tích hợp sẵn có trên thị trường.

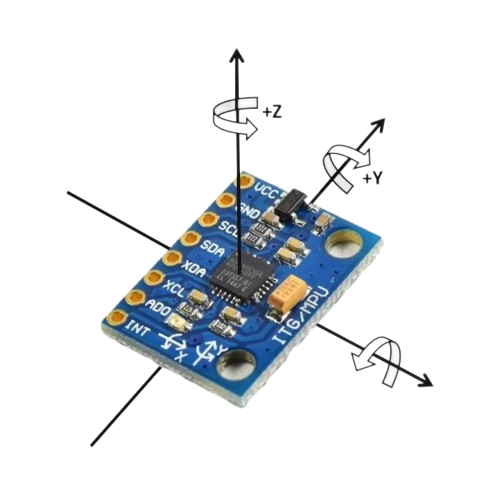
- Vi điều khiển STM32F401CCU với kiến trúc ARM Cotex-M4 với tốc độ tối đa 84 MHz và flash 256 Kb là lựa chọn phù hợp với kích thước rất nhỏ gọn cùng với tốc độ tính toán cao và giá thành rẻ.



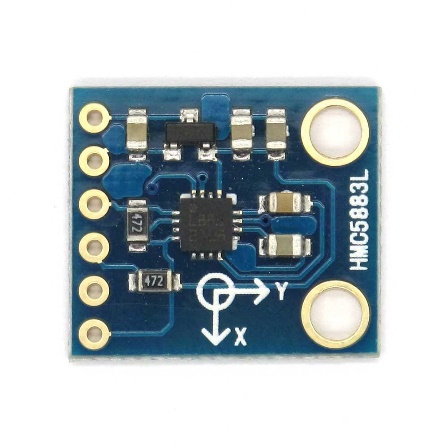
2. Cảm biến IMU và MARG

- Hiện nay có rất nhiều loại module tích hợp sẵn cảm biến IMU và MARG. Tuy nhiên, qua thực tế, các loại module này thường dễ bị làm giả làm cho tín hiệu đầu vào bị nhiễu nhiều. Một số loại module khác lại có giá thành cao và nguồn hàng hiếm. Do đó, ta sẽ lựa chọn cảm biến với tiêu chí giá thành thấp, tín hiệu ổn định và nguồn hàng phổ biến.

- Cảm biến 6 trục MPU6050 cho phép đo được 3 trục gia tốc và 3 trục gygroscope là 1 loại cảm biến rất phổ biến trên thị trường hiện nay.



- Cảm biến từ trường HMC5883L là một loại cảm biến có thể đo được giá trị từ trường Trái Đất theo 3 trục X, Y, Z



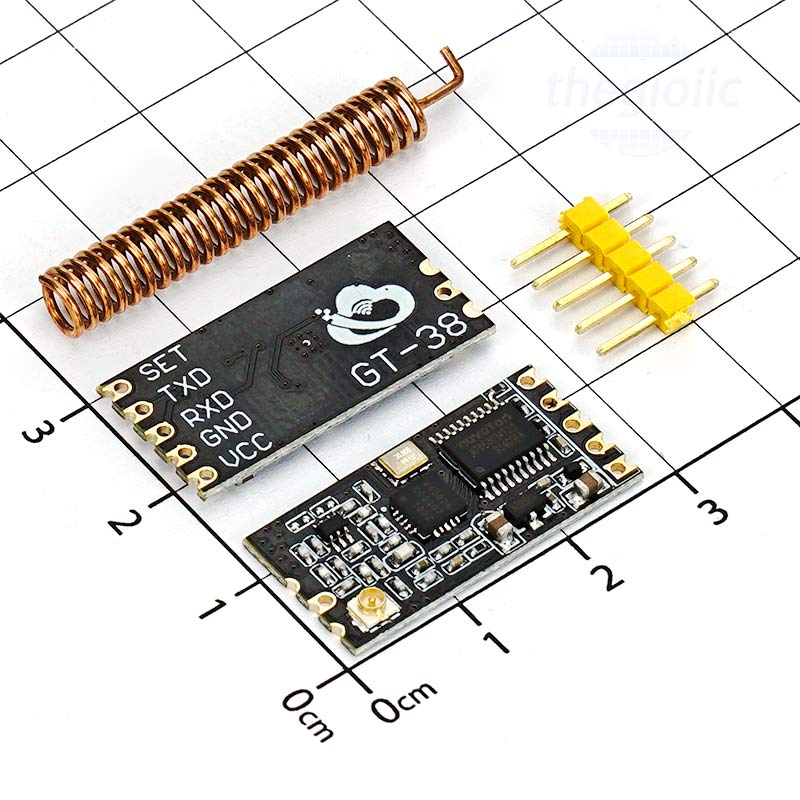
3. Biến trở

- Biến trở RV24YN 20S 10K là một loại biến trở được sản xuất tại Nhật Bản đáp ứng được tiêu chuẩn JIS. Với đường kính trục lớn có thể thuận tiện hơn cho việc thiết kế khớp nối tại khủy tay.



4. Module giao tiếp không dây

- Module UART không dây GT-38 là loại module cho phép truyền dữ liệu không dây qua giao thức UART. Với tốc độ theo datasheet có thể lên đến 100000 bps khi baudrate đạt 115200. Khoảng cách truyền có thể đạt tới 1.2 Km nếu dùng mode Ultra low distance.



5. Nguồn

Với tiêu chí nhỏ gọn và thời gian sử dụng lâu, ta sẽ sử dụng Pin Lipo 3.7V là loại pin Lithium cho khả năng sạc nhiều lần với kích thước rất nhỏ gọn.



Với dòng tiêu thụ ước tính của thiết bị là ~ 100 mA. Thiết bị có thể sử dụng được trong 20 giờ liên tục.

Để thuận tiện cho việc sạc và xả pin, ta sử dụng module mạch sạc tích hợp mạch nguồn 5V cho hệ thống



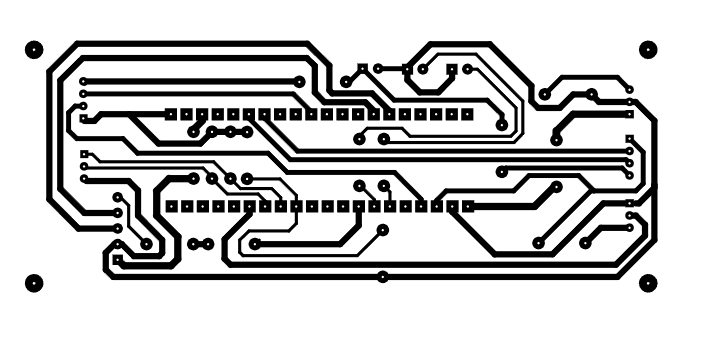
**5.5 Thiết kế và gia công mạch in**

1. Thiết kế schematic

2. Chọn lựa các linh kiện

3. Vẽ mạch

4. Gia công mạch in thủ công



**5.6 Kết nối và xử lí dữ liệu thu được từ các cảm biến**

**5.6.1 Cấu hình cho MCU**

**5.6.2. Lấy giá trị từ IMU**

1.1 Giao tiếp I2C

- Trên STM32 ta thực hiện cấu hình để giao tiếp I2C, tốc độ 1000 bps.

- Địa chỉ I2C của MPU6050 mặc định là 0x68 << 1.

1.2 Init và cấu hình ban đầu

- Theo datasheet, ta có flow cấu hình cho IMU như sau:

+ Wake-up MP6050 bằng cách write 0x00 vào thanh ghi RA\_PWR\_MGMT\_1

+ Chọn Clock cho bộ PLL bằng cách tham chiếu với giá trị gygroscope của trục X

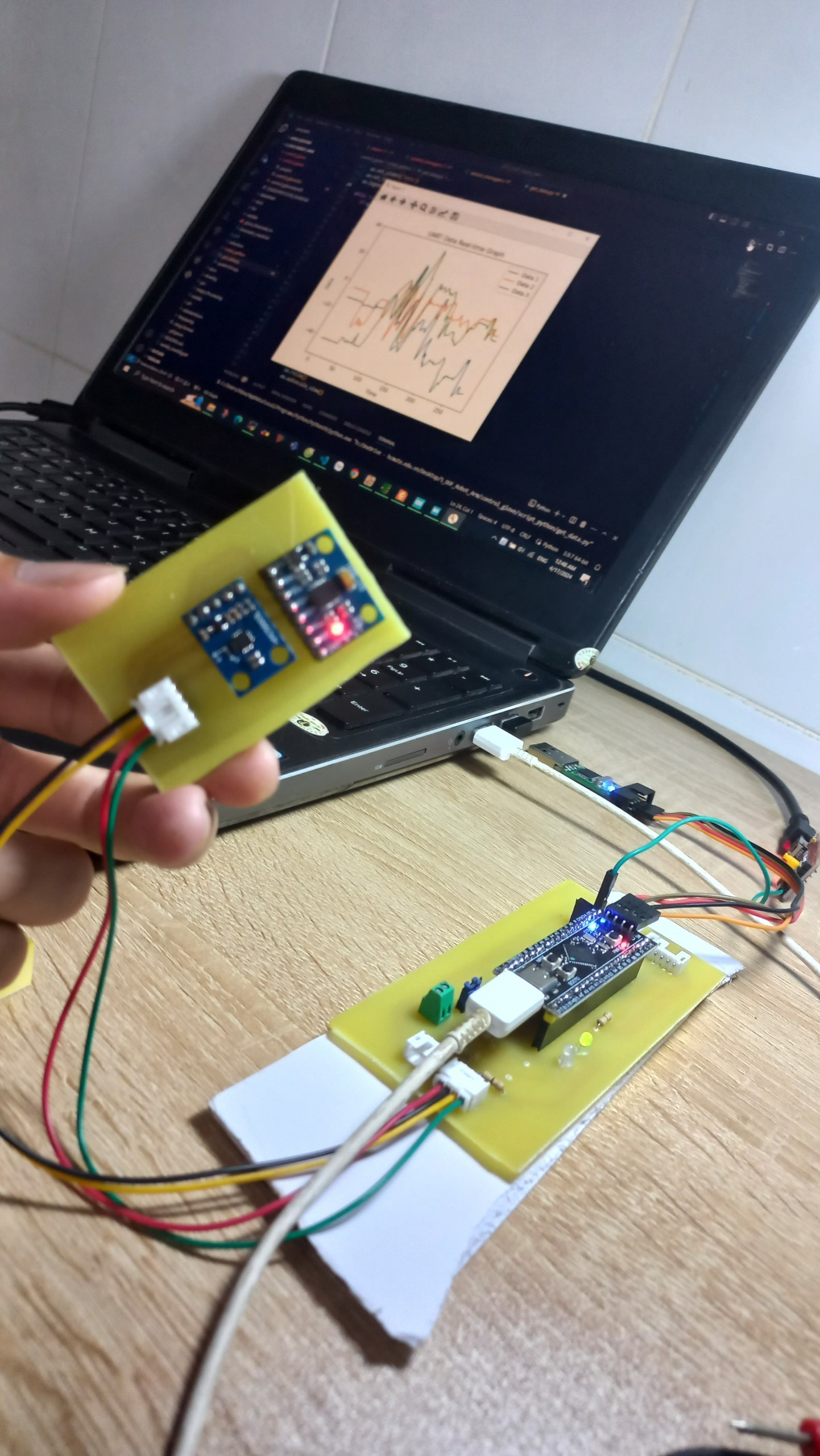
+ Set sample rate cho MPU6050

+ Cài đặt full scale range cho giá trị gygroscope là maximum 2000 độ/giây

+ Cài đặt full scale range cho giá trị gia tốc là 8g.

+ Cài đặt giá trị lọc thông thấp số Digital Low Pass Filter với acc là 260 Hz, delay 0 ms, gygro là 256 Hz, delay 0,98 ms.

1.3 Đọc giá trị gia tốc và gygro



1.4 Calib các giá trị ban đầu

- Giải thuật calib:

- So sánh giá trị trước và sau khi calib

**2. Lấy giá trị từ cảm biến từ trường**

1.1 Giao tiếp I2C

- Trên STM32 ta thực hiện cấu hình để giao tiếp I2C, tốc độ 1000 bps.

- Địa chỉ I2C của MPU6050 mặc định là 0x1E << 1.

1.2 Init cấu hình ban đầu

- Theo datasheet, ta có flow cấu hình như sau:

+ Chọn range cho cảm biến là RANGE\_1\_3GA

+ Chon mode đo là Continuos

+ Chọn tần số lấy mẫu là 75 Hz (maximum)

+ Chọn sample lấy là 1

1.3 Đọc giá trị từ trường

- Các giá trị từ trường của cảm biến có thể đọc được từ các thanh ghi:

REG\_OUT\_X\_M, REG\_OUT\_Y\_M, REG\_OUT\_Z\_M

1.4 Calib giá trị cảm biến từ trường

- Thuật toán calib:

**3. Lấy giá trị từ biến trở**

1. Đọc giá trị ADC

- Ta thực hiện cấu hình bộ ADC, chế độ 12-bits

- Ta có điện áp đầu vào có range từ 0V 🡪 3.3V, giá trị ADC từ 0 🡪 4095.

2. Lọc nhiễu sử dụng Low pass filter

- Ta sử dụng thuật toán low pass filter với hệ số Alpha là 0.03. Ta có kết quả:

**6. Tính toán giá trị vị trí cuối**

1. Áp dụng thuật toán lọc nhiễu Magdwick

- Các giá trị đầu vào: ax, ay, az, gx, gy, gz, mx, my, mz và tần số lấy mẫu

- Lưu đồ lấy data và xử lí bằng thuật toán Magdwick

- Chọn các hệ số Kp, Ki

- Kết quả khi convert về 3 góc Euler:

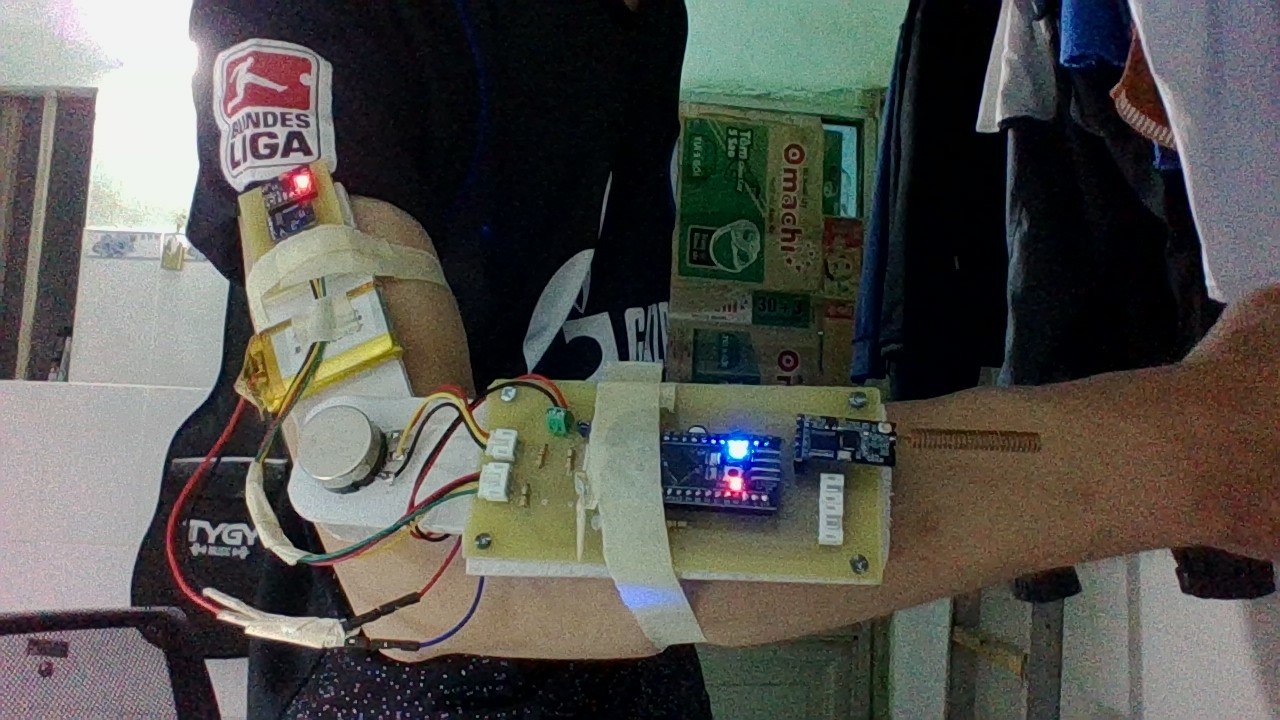
2. Chuyển đổi về giá trị vị trí cuối

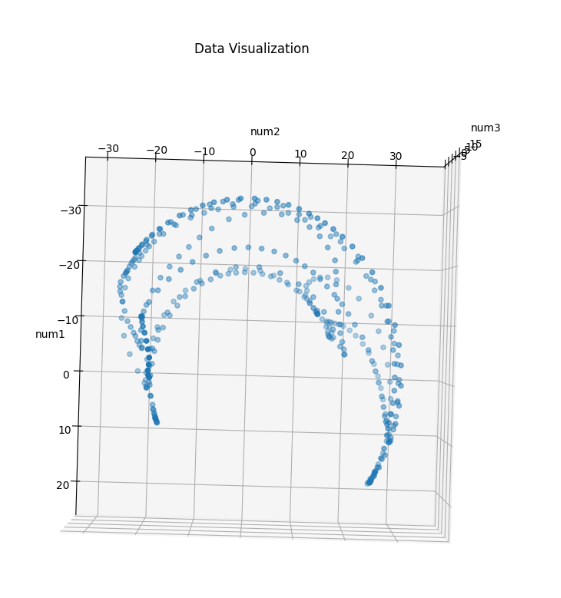
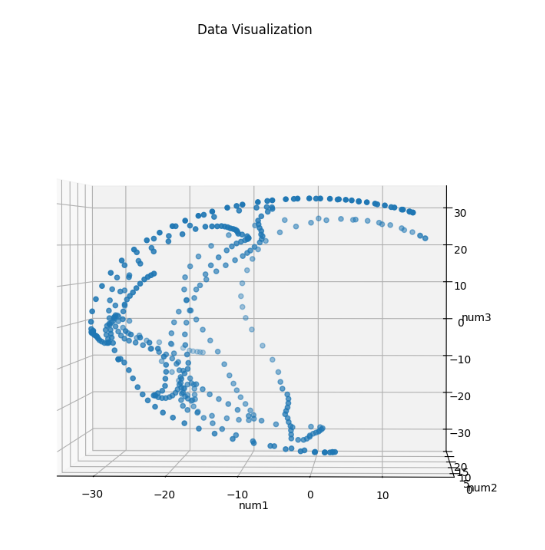
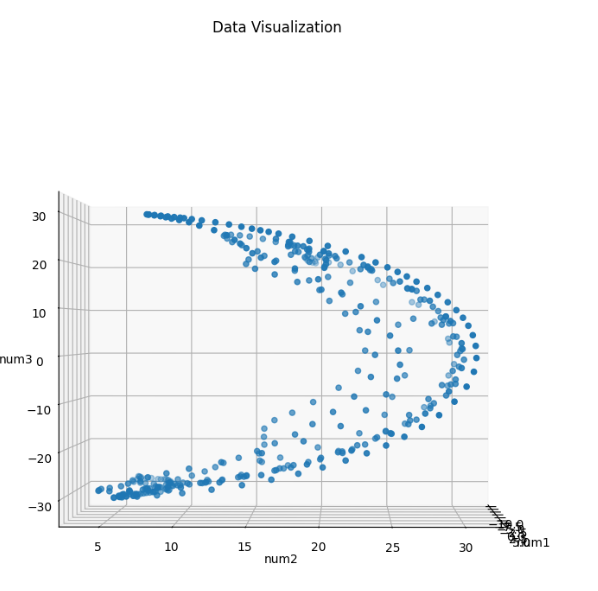
- Giá tri đầu vào: q0, q1, q2, q3 tính từ Magdwick và giá trị góc từ biến trở

2. Chế tạo mẫu thử và kiểm tra kết quả

- Nguyên lí: Đặt các cảm biến IMU, MARG ở vai và biến trở ở khủy tay

- Khoảng cách từ vai đến khủy tay chọn là 12 cm, từ khủy tay đến cuối là 20 cm



**5.7 Thiết kế và gia công hệ thống cơ khí**

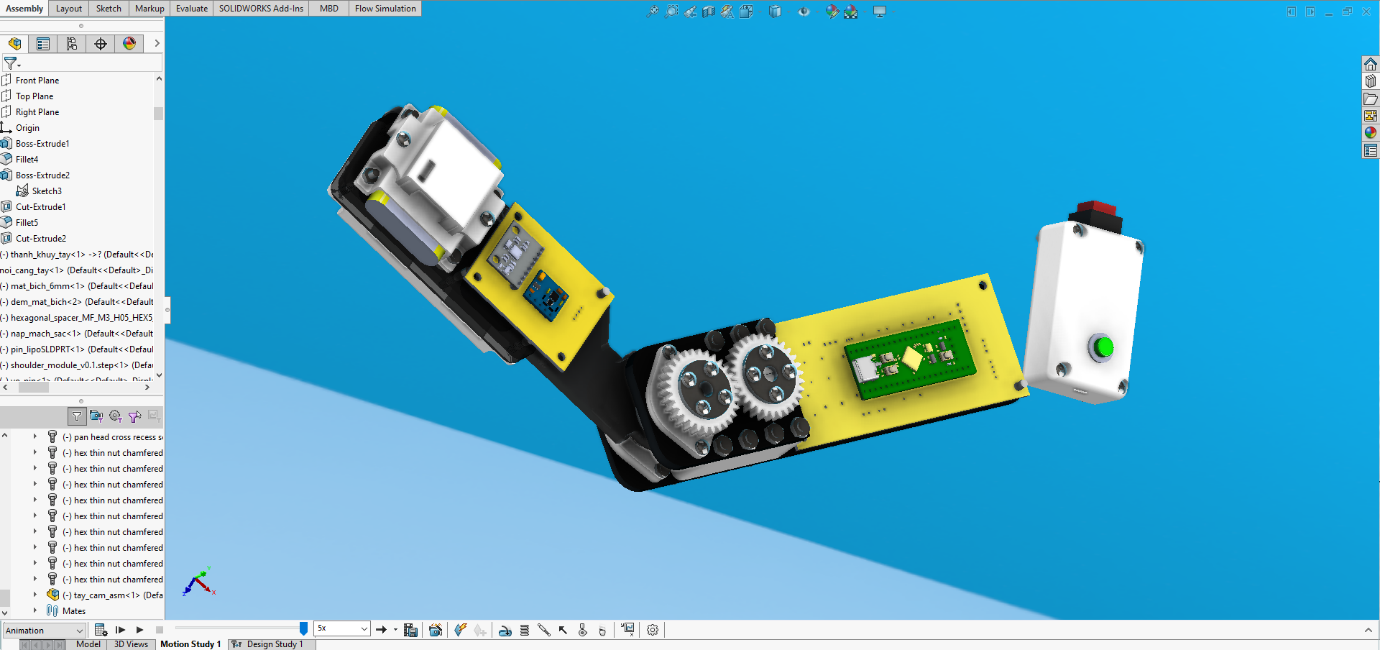
**5.7.2 Tổng quan về cấu trúc cơ khí của thiết bị**

Cấu trúc cánh tay người: Chiều dài, rộng, độ mở tại khớp khủy tay, …

**5.7.1 Thiết kế, tính toán tại khớp nối**

**5.7.1.1 Xác định cơ cấu và các lực tác động**

**5.7.1.2 Tính toán bền cho trục**

**5.7.1.3 Tính toán lựa chọn ổ lăn**

**5.7.2 Thiết kế phần dây đeo**

**5.7.3 Thiết kế các vị trí đặt các linh kiện**

**5.7.4 Thiết kế tay cầm**

**5.7.5 Gia công và lắp ráp**

**5.7.5.1 Cắt mica**

**5.7.5.2 In 3D**

**5.7.5.3 Lắp ráp và hoàn thiện**

**5.8 Thử nghiệm và đánh giá độ chính xác dữ liệu tính được**

**5.8.1 Thử nghiệm và đánh giá độ ổn định của tín hiệu ở trạng thái nghỉ**

**5.8.2 Thử nghiệm và đánh giá độ chính xác của các giá trị vị trí khi quay trong không gian**

**5.8.3 Thử nghiệm và đánh giá về độ trễ của tín hiệu khi truyền không dây**

**5.9 Thiết kế và lập trình các tính năng cho thiết bị**

**5.9.1 Thiết kế kiến trúc cho chương trình nhúng**

Hiện nay, cấu trúc của một chương trình nhúng thường được chia thành nhiều lớp khác nhau để có thể dễ dàng tương tác và quản lí. Theo đó, ta sẽ chia chương trình thành các lớp: System Manager, Sensor Manager, Driver, Middleware và BSP (Board Support Package). Cấu trúc chi tiết như sau:

**5.9.2 Thiết kế và lập trình các mode hoạt động**

**5.9.2.2 Thiết kế giao thức truyền nhận dữ liệu tới GUI tool**

1. Xác định frame truyền

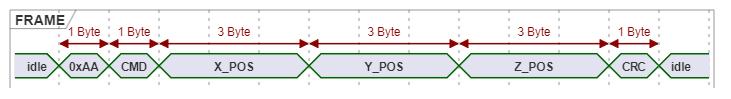
- Khi thực hiện truyền dữ liệu từ găng tay qua C#, ta cần truyền 3 thông số vị trí x, y, z là tọa độ của đầu găng tay. Các thông số vị trí có giá trị là số thực có giá trị từ -99,9999 đến 100,0000 mm (lấy 4 chữ số sau số thập phân). Vậy tổng số giá trị cần truyền cho mỗi giá trị vị trí là 2.000.000 giá trị. Suy ra cần ít nhất 3 bytes hay 24 bits để mã hóa. Ta có: 224 = 16.777.216 giá trị, lớn hơn số lượng giá trị cần thiết.

- Để thực thiện các câu lệnh từ găng tay lên C#, ta cần thêm 1 byte để mã hóa. Với 1 byte ta có thể mã hóa được 256 loại câu lệnh khác nhau.

- Trong quá trình tuyền nhận dữ liệu, có thể có xảy ra hiện tượng data bị hư hại do tác động của môi trường truyền dẫn, vì vậy cần thêm 1 byte CRC để đảm bảo tính an toàn của dữ liệu truyền đi.

- Để dễ dàng cho việc xử lí, ta sử dụng thêm 1 byte đầu tiên để báo hiệu 1 frame bắt đầu.

- Ta có kết quả sau khi cân nhắc các yếu tố:



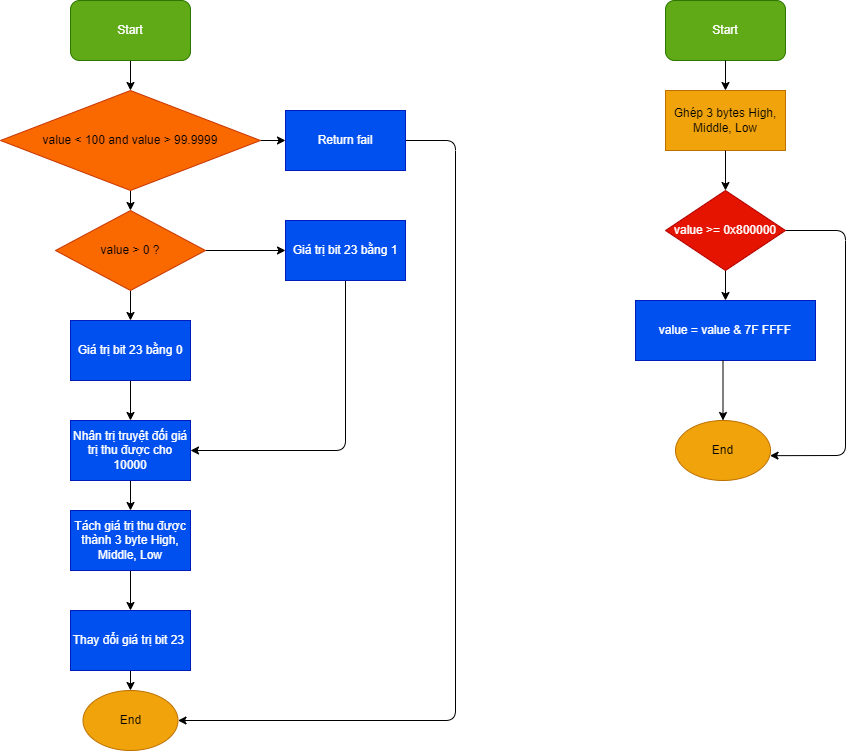
2. Quy ước câu lệnh

|  |  |
| --- | --- |
| **Command ID** | **Ý nghĩa** |
| 0x00 | Chỉ truyền giá trị vị trí |
| 0x01 | Truyền giá trị vị trí và start record |
| 0x02 | Truyền giá trị vị trí và stop record |
| 0x03 | Xóa record vừa tạo ra |
| 0x04 | Truyền giá trị vị trí và quay khớp 5 theo chiều dương |
| 0x05 | Truyền giá trị vị trí và quay khớp 5 theo chiều âm |

**3. Xử lí các giá trị vị trí**

- Các giá trị vị trí là các số thực (float) được mã hóa bởi 32 bits dữ liệu (ở dòng vi điều khiển 32 bits). Ta cần thêm 1 bit để mã hóa dấu.

Ta có: [1 bit dấu][23 bit giá trị]



Hình : Thuật toán mã hóa và giải mã giá trị vị trí