**ĐẠI HỌC QUỐC GIA HÀ NỘI**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

**LÊ VĂN DUẨN**

**NGHIÊN CỨU HỆ THỐNG ĐIỀU KHIỂN BÁM CHO PANEL LẮP ĐẶT PIN NĂNG LƯỢNG MẶT TRỜI**

Ngành: Công nghệ Điện tử Viễn thông

Chuyên ngành: Kỹ thuật Điện tử

Mã số: 60520203

**LUẬN VĂN THẠC SĨ CÔNG NGHỆ ĐIỆN TỬ VIỄN THÔNG**

**NGƯỜI HƯỚNG DẪN KHOA HỌC: PGS.TS. TRẦN ĐỨC TÂN**

**Hà Nội - 2015**

**LỜI CẢM ƠN**

Để hoàn thành được luận văn này em đã nhận được rất nhiều sự động viên, giúp đỡ của nhiều cá nhân và tập thể.

Trước hết, em xin bày tỏ lòng biết ơn sâu sắc đến PGS. TS Trần Đức Tân đã tận tình hướng dẫn em thực hiện luận văn này.

Em xin cảm ơn sự hỗ trợ một phần của đề tài khoa học mã số QGĐA.14.03 trong quá trình em thực hiện luận văn này.

Em xin bày tỏ lòng biết ơn chân thành tới các thầy cô giáo của Trường Đại học Công nghệ - Đại học Quốc gia Hà nội, người đã đem lại cho em những kiến thức vô cùng có ích trong những năm học vừa qua.

Em cũng xin gửi lời cảm ơn chân thành tới Ban Giám hiệu, Phòng Đào tạo, Trường Đại học Công nghệ - Đại học Quốc gia Hà nội đã tạo điều kiện cho em trong quá trình học tập.

Cuối cùng em xin gửi lời cảm ơn đến gia đình, bạn bè những người đã luôn động viên và khuyến khích em trong quá trình thực hiện đề tài này.

Hà Nội, ngày 26 tháng 10 năm 2015

Tác giả

**Lê Văn Duẩn**

**LỜI CAM ĐOAN**

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của PGS.TS Trần Đức Tân. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu, công thức, hình ảnh, bảng biểu và các dữ liệu khác phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung luận văn của mình. Trường đại học Công Nghệ không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

Hà Nội, ngày 26 tháng 10 năm 2015

Tác giả

**Lê Văn Duẩn**

**MỤC LỤC**

[MỞ ĐẦU 1](#_Toc436603060)

[I. TỔNG QUAN 3](#_Toc436603061)

[I.1. Năng lượng sạch 3](#_Toc436603062)

[I.2. Năng lượng mặt trời 4](#_Toc436603063)

[I.3. Vấn đề của luận văn 6](#_Toc436603064)

[I.3.1. Giải pháp 6](#_Toc436603065)

[I.3.2. Một số hệ thống theo dõi mặt trời hiện nay 7](#_Toc436603066)

[I.3.3. Mục tiêu của luận văn. 8](#_Toc436603067)

[II. NGUYÊN TẮC HOẠT ĐỘNG 10](#_Toc436603068)

[II.1. Quỹ đạo của mặt trời 10](#_Toc436603069)

[II.2. Tọa độ di chuyển của tàu 11](#_Toc436603070)

[II.3. Mô hình toán học 12](#_Toc436603071)

[II.4. Hệ thống đề xuất 13](#_Toc436603072)

[II.4.1. Sơ lược về cảm biến la bàn số 13](#_Toc436603073)

[II.4.2. Tính toán phương hướng dựa trên la bàn số 14](#_Toc436603074)

[II.4.3. Cảm biến IMU 16](#_Toc436603075)

[II.4.4. Xác định vị trí của tàu dựa trên các cảm biến 17](#_Toc436603076)

[II.4.5. Vị trí của tàu so với mặt trời 18](#_Toc436603077)

[II.5. Bộ lọc bù 18](#_Toc436603078)

[III. MÔ HÌNH THỰC TẾ 22](#_Toc436603079)

[III.1. Phần cứng 22](#_Toc436603080)

[III.1.1. Hệ thống điều khiển 23](#_Toc436603081)

[III.1.2. Hệ thống cảm biến. 28](#_Toc436603082)

[III.1.3. Thiết bị truyền động 29](#_Toc436603083)

[III.2. Phần mềm 33](#_Toc436603084)

[III.2.1. Các thành phần của hệ thống phần mềm 33](#_Toc436603085)

[III.2.2. Lập trình cho kit ARM BeagleBone Black và các thiết bị 34](#_Toc436603086)

[III.2.3. Xử lý dữ liệu thu thập từ các cảm biến 48](#_Toc436603087)

[III.2.4. Nâng cao độ chính xác của dữ liệu đọc được từ cảm biến 49](#_Toc436603088)

[IV. KẾT QUẢ 52](#_Toc436603089)

[IV.1. Mô hình phần cứng thực tế 52](#_Toc436603090)

[IV.2. Nhận xét các kết quả thu được 53](#_Toc436603091)

[IV.3. Kết quả chạy thử hệ thống bám mặt trời 59](#_Toc436603092)

[V. KẾT LUẬN 60](#_Toc436603093)

[VI. HƯỚNG PHÁT TRIỂN 62](#_Toc436603094)

[TÀI LIỆU THAM KHẢO 63](#_Toc436603095)

[PHỤ LỤC 65](#_Toc436603096)

**DANH SÁCH CÁC TỪ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| **Tên viết tắt** | **Tiếng Anh** | **Tiếng Việt** |
| **IMU** | Inertial Measurement Unit | Khối đo quán tính |
| **GPIO** | General Purpose Input Output | Cổng vào ra đa mục đích |
| **I2C** | Inter-Integrated Circuit | Chuẩn giao tiếp nối tiếp 2 dây |
| **IDE** | Integrated Development Environment | Môi trường phát triển tích hợp |
| **LST** | Local Solar Time | Giờ mặt trời địa phương |
| **LSTM** | Local Solar Time Meridian | Giờ kinh tuyến địa phương |
| **LT** | Local Time | Giờ địa phương |
| **TC** | Time Correction Factor | Hệ số hiệu chỉnh thời gian |
| **GMT** | Greenwich Mean Time | Giờ Greenwich |
| **EoT** | Equation of Time | Phương trình thời gian |
| **Long** | Longitude | Kinh độ |
| **Lat** | Latitude | Vĩ độ |
| **MEMS** | Micro-Electro-Mechanical Systems | Hệ thống vi cơ điện tử |
| **GMR** | Giant magneto resistance | Từ trở khổng lồ |
| **AMR** | Anisotropic Magneto Resistance | Từ trở không đẳng hướng |
| **MJT** | Magneto amplification in a bipolar magnetic junction transistor | Cảm biến tiếp giáp đường hầm |
| **ARM** | Acorn RISC Machine | Kiến trúc ARM |
| **JTAG** | Joint Test Action Group | Chuẩn giao tiếp gỡ lỗi JTAG |
| **HDMI** | High-Definition Multimedia Interface | Giao diện đa phương tiện độ phân giải cao |
| **EMMC** | Embedded Multimedia Card | Bộ nhớ đa phương tiện nhúng |
| **ADC** | Analog-Digital Converter | Bộ chuyển đổi tương tự sang số |
| **SPI** | Serial Peripheral Interface | Giao diện ngoại vi nối tiếp |
| **PWM** | Pulse Width Modulation | Điều chế độ rộng xung |
| **LCD** | Liquid Crystal Display | Màn hình hiển thị tinh thể lỏng |
| **USB OTG** | USB On The Go | Bus USB hoạt động theo mô hình chủ khách |
| **USB** | Universal Serial Bus | Chuẩn kết nối tuần tự đa dụng |
| **TF** | Trans Flash | Chuẩn thẻ nhớ Trans Flash |
| **ACK** | Acknowledged | Tín hiệu ghi nhận |
| **SDA** | Serial Data Line | Đường dữ liệu nối tiếp |
| **SCL** | Serial Clock Line | Xung đồng hồ nối tiếp |

**DANH SÁCH HÌNH VẼ**

[Hình 1: Tấm năng lượng mặt trời trên các hệ thống tĩnh 6](#_Toc436603105)

[Hình 2: Sự dịch chuyển của tàu. 12](#_Toc436603106)

[Hình 3: Mô hình toán học của trục X trên mặt phẳng Y-Z 13](#_Toc436603107)

[Hình 4: Từ trường của trái đất 14](#_Toc436603108)

[Hình 5: Từ trường của trái đất trên 3 trục tọa độ 15](#_Toc436603109)

[Hình 6: La bàn số đặt nghiêng so với mặt phẳng nằm ngang của trái đất 16](#_Toc436603110)

[Hình 7: Các góc quay trên các trục tọa độ Đề Các 17](#_Toc436603111)

[Hình 8: Mô hình bộ lọc bù 20](#_Toc436603112)

[Hình 9: Hệ thống theo dõi mặt trời 2 trục 22](#_Toc436603113)

[Hình 10: Sơ đồ khối phần cứng. 23](#_Toc436603114)

[Hình 11: Kit BeagleBone Black (Rev C) 25](#_Toc436603115)

[Hình 12: Sơ đồ kết nối các thành phần của hệ thống 27](#_Toc436603116)

[Hình 13: Mô đun cảm biến la bàn số GY-271 HMC5883L 28](#_Toc436603117)

[Hình 14: Mô đun cảm biến IMU GY-521 MPU6050 29](#_Toc436603118)

[Hình 15: Động cơ bước 28BYJ-48 30](#_Toc436603119)

[Hình 16: Sơ đồ cuộn dây 31](#_Toc436603120)

[Hình 17: Sơ đồ mạch điều khiển động cơ dùng ULN2803 32](#_Toc436603121)

[Hình 18: Kết nối động cơ bước với mạch điều khiển 33](#_Toc436603122)

[Hình 19: Các bước thực hiện phần mềm của hệ thống 34](#_Toc436603123)

[Hình 20: Xử lý đa luồng trong hệ thống điều khiển bám mặt trời 35](#_Toc436603124)

[Hình 21: Các chân vào ra khả dụng trên BeagleBone Black (65 chân). 37](#_Toc436603125)

[Hình 22: Các thành phần của hệ thống thực tế 52](#_Toc436603126)

[Hình 23: Mô hình hệ thống bám mặt trời thực tế 53](#_Toc436603127)

[Hình 24: Đồ thị so sánh các góc nghiêng khi IMU đứng yên 54](#_Toc436603128)

[Hình 25: Đồ thị so sánh các góc nghiêng khi IMU chuyển động 55](#_Toc436603129)

[Hình 26: Đồ thị so sánh các góc chúc đo được khi IMU đứng yên 56](#_Toc436603130)

[Hình 27: Đồ thị so sánh các góc chúc khi IMU chuyển động 56](#_Toc436603131)

[Hình 28: Đọc dữ liệu từ cảm biến la bàn số 57](#_Toc436603132)

[Hình 29: Độ lệch về góc của hệ thống so với mặt trời và góc quay của động cơ. 58](#_Toc436603133)

[Hình 30: Góc quay của động cơ (phóng to từ hình 29). 59](#_Toc436603134)

**DANH SÁCH BẢNG BIỂU**

[Bảng 1: Các thông số kỹ thuật của kit BeagleBone Black 24](#_Toc436603155)

[Bảng 2: Thứ tự cấp xung điều khiển đủ bước 31](#_Toc436603156)

[Bảng 3: Thứ tự cấp xung điều khiển nửa bước 31](#_Toc436603157)

[Bảng 4: Các tệp tin cơ bản trong thư mục SysFS của GPIO. 38](#_Toc436603158)

[Bảng 5: Thứ tự cấp xung điều khiển 46](#_Toc436603159)

**MỞ ĐẦU**

Năng lượng mặt trời ngày nay, đang trở nên rất phổ biến. Rất nhiều tấm năng lượng mặt trời đã được lắp đặt trên toàn thế giới. Hầu hết trong số chúng đều được cài đặt cố định theo hướng của bức xạ ánh sáng mặt trời tối đa. Tuy nhiên, mặt trời luôn chuyển động hằng ngày. Vì vậy, chúng ta không thể sử dụng bức xạ tối đa của mặt trời vào tất cả các thời gian trong ngày. Đối với các hệ thống năng lượng mặt trời trên các hệ thống chuyển động (ví dụ: tàu thuyền, xe quân sự, vệ tinh, cáp treo …), chúng không những bị ảnh hưởng bởi chuyển động của mặt trời mà còn bị ảnh hưởng bởi sự dịch chuyển và vị trí của chúng.

Theo dõi mặt trời là cách tốt nhất để tiếp nhận bức xạ tối đa. Bằng cách di chuyển các tấm năng lượng theo hướng dịch chuyển của mặt trời, chúng ta có thể thu được các bức xạ tối đa.

Mục tiêu của luận văn này là thiết kế hệ thống bám mặt trời 2 trục cho các tấm năng lượng được đặt trên các hệ thống chuyển động, cụ thể ở đây là trên các tàu hoạt động trên biển.

Nội dung của luận văn gồm 5 phần chính như sau:

Phần I. Tổng quan: nêu lên khái quát về năng lượng sạch nói chung và năng lượng mặt trời nói riêng. Giới thiệu chung về một số hệ thống bám mặt trời đồng thời chỉ ra mục tiêu của luận văn: *“Nghiên cứu hệ thống điều khiển bám cho panel lắp đặt pin năng lượng mặt trời*”.

Phần II. Nguyên tắc hoạt động: nghiên cứu lý thuyết về quỹ đạo chuyển động của mặt trời và sự dịch chuyển của vật thể chuyển động, giới hạn là các tàu di chuyển trên biển. Từ đó đưa ra đề xuất cho hệ thống bám cho panel lắp đặt pin mặt trời.

Phần III. Mô hình thực tế: luận văn đề xuất và phát triển một mô hình thực tế của hệ thống dựa trên mô hình đề xuất ở phần II bao gồm cả phần cứng và phần mềm.

Phần IV. Kết quả: chỉ ra kết quả thu được từ việc xây dựng mô hình thực tế.

Phần V. Kết luận: đưa ra kết luận chung về kết quả nghiên cứu và phát triển hệ thống, chỉ ra ưu và nhược điểm của hệ thống khi so sánh với các hệ thống khác.

Phần VI. Hướng phát triển: Nêu ra giải pháp đề khắc phục một số nhược điểm của hệ thống hiện tại đồng thời đưa ra hướng phát triển cho hệ thống trong tương lai.

# TỔNG QUAN

* 1. **Năng lượng sạch**

Năng lượng sạch không gây ô nhiễm không khí hay tạo ra các sản phẩm phụ gây ô nhiễm môi trường trong tiến trình phát điện. Gió, năng lượng mặt trời và địa nhiệt là những thí dụ về các tài nguyên có thể cung cấp năng lượng sạch hiện nay. Năng lượng sạch ở đây chỉ xét đến tiến trình phát điện.

Hiện nay, điện và khí đốt là những nguồn năng lượng chính và được sử dụng rộng rãi trên toàn thế giới. Tuy nhiên, chúng lại chính là những nguồn năng lượng gây ra lượng khí thải nhà kính lớn nhất. Để chống lại tác động nặng nề của biến đổi khí hậu, chúng ta cần chuyển sang sử dụng các nguồn năng lượng sạch để cung cấp cho việc sinh hoạt và sản xuất.

Hiện tại, rất nhiều các công ty, phòng nghiên cứu trên khắp thế giới cùng chính phủ các nước đang rất nỗ lực để tìm ra những biện pháp giải quyết hậu quả môi trường một cách tối ưu nhất. Tuy nhiên, một số giải pháp vẫn chưa đi vào thực tế và được kỳ vọng phát triển trong tương lai.

* **Năng lượng gió [5]**

Năng lượng gió là động năng của không khí di chuyển trong bầu khí quyển trái đất. Năng lượng gió là một hình thức gián tiếp của năng lượng mặt trời. Sử dụng năng lượng gió là một trong những cách lấy năng lượng xa xưa nhất từ môi trường tự nhiên và đã được biết đến từ thời kỳ cổ đại.

Từ hàng trăm năm nay, con người đã dùng năng lượng gió để di chuyển thuyền buồm hay khinh khí cầu, ngoài ra năng lượng gió còn được sử dụng để tạo công cơ học nhờ vào các cối xay gió. Ý tưởng dùng năng lượng gió để sản xuất điện hình thành ngay sau các phát minh ra điện và máy phát điện. Lúc đầu nguyên tắc của cối xay gió chỉ được biến đổi nhỏ và thay vì là chuyển đổi động năng của gió thành năng lượng cơ học thì dùng máy phát điện để sản xuất năng lượng điện. Khi bộ môn cơ học dòng chảy tiếp tục phát triển thì các thiết bị xây dựng và hình dáng của các cánh quạt cũng được chế tạo đặc biệt hơn. Ngày nay người ta gọi đó là tuốc bin gió, khái niệm cối xay gió không còn phù hợp nữa vì chúng không còn có thiết bị nghiền. Từ sau những cuộc khủng hoảng dầu trong thập niên 1970 việc nghiên cứu sản xuất năng lượng từ các nguồn khác được đẩy mạnh trên toàn thế giới, kể cả việc phát triển các tuốc bin gió hiện đại.

Ưu điểm nổi bật nhất của điện gió là không lo hết hay cạn kiệt nguồn nhiên liệu, không gây ô nhiễm môi trường như các nhà máy nhiệt điện, dễ chọn địa điểm và tiết kiệm đất xây dựng. Chính vì những ưu điểm này mà năng lượng gió đang được khuyến khích sử dụng ở nhiều quốc gia trên thế giới để giảm sự phụ thuộc vào nguồn năng lượng hóa thạch và giảm thiểu hiệu ứng nhà kính.

* **Địa nhiệt [2]**

Năng lượng địa nhiệt là năng lượng được tách ra từ nhiệt trong lòng trái đất. Năng lượng này có nguồn gốc từ sự hình thành ban đầu của hành tinh, từ hoạt động phân hủy phóng xạ của các khoáng vật và từ năng lượng mặt trời được hấp thụ tại bề mặt trái đất. Năng lượng địa nhiệt đã được sử dụng để nung và tắm kể từ thời La Mã cổ đại, nhưng ngày nay nó được dùng để phát điện.

Khai thác năng lượng địa nhiệt có hiệu quả về kinh tế, có khả năng thực hiện và thân thiện với môi trường, nhưng trước đây bị giới hạn về mặt địa lý đối với các khu vực gần các ranh giới kiến tạo mảng. Các tiến bộ khoa học kỹ thuật gần đây đã từng bước mở rộng phạm vi và quy mô của các tài nguyên tiềm năng này, đặc biệt là các ứng dụng trực tiếp như dùng để sưởi trong các hộ gia đình. Các giếng địa nhiệt có khuynh hướng giải phóng khí thải nhà kính bị giữ dưới sâu trong lòng đất, nhưng sự phát thải này thấp hơn nhiều so với phát thải từ việc đốt nhiên liệu hóa thạch thông thường. Công nghệ này có khả năng giúp giảm thiểu sự nóng lên toàn cầu nếu nó được triển khai rộng rãi.

* **Năng lượng Mặt Trời [4]**

Năng lượng mặt trời thu được trên trái đất là năng lượng của dòng bức xạ điện từ xuất phát từ mặt trời đến trái đất. Chúng ta sẽ tiếp tục nhận được dòng năng lượng này cho đến khi phản ứng hạt nhân trên Mặt Trời hết nhiên liệu, vào khoảng 5 tỷ năm nữa. Có thể trực tiếp thu lấy năng lượng này thông qua hiệu ứng quang điện, chuyển năng lượng các photon của Mặt Trời thành điện năng, như trong pin Mặt Trời. Năng lượng của các photon cũng có thể được hấp thụ để làm nóng các vật thể, tức là chuyển thành nhiệt năng, sử dụng cho bình đun nước Mặt Trời, hoặc làm sôi nước trong các máy nhiệt điện của tháp Mặt Trời, hoặc vận động các hệ thống nhiệt như máy điều hòa Mặt Trời.

* 1. **Năng lượng mặt trời**

Năng lượng mặt trời là rất lớn và là nguồn năng lượng vô tận [4]. Theo nghiên cứu của các nhà khoa học, Trái đất nhận được 16 x 1018 đơn vị của năng lượng từ mặt trời hàng năm, gấp 20.000 lần so với nhu cầu sử dụng của nhân loại trên trái đất.

Năng lượng mặt trời có thể cung cấp tất cả các nhu cầu năng lượng hiện tại và tương lai. Điều này hứa hẹn làm cho nó trở thành một trong những nguồn năng lượng quan trọng trong tương lai.

Theo báo cáo dự đoán của tổ chức năng lượng quốc tế, đến năm 2050 mặt trời có thể sẽ trở trành nguồn điện năng lớn nhất, xếp trên cả nhiên liệu hóa thạch, năng lượng gió, thủy năng và năng lượng hạt nhân [3]. Hồi năm 2006 ở Úc, tháp năng lượng mặt trời khổng lồ cao 1km với 32 tuốc bin khí có tổng công suất 200MW đã đi vào sử dụng. Hệ năng lượng mặt trời này được bao quanh bởi 1 nhà kính khổng lồ có tác dụng làm nóng không khí để làm quay tuốc bin xung quanh chân tháp. Giới chuyên gia ước tính rằng các nhà máy điện sẽ có thể tạo ra 200MW điện và giảm được 700.000 tấn khí gây hiệu ứng nhà kính mỗi năm. Và gần đây nhất là nhà máy điện năng lượng mặt trời Topaz có năng suất 550MW với 9 triệu tấm pin quan điện, bao phủ hơn 20 héc ta tại California. Đây được xem là nguồn năng lượng mặt trời lớn nhất thế giới hiện nay và Topaz cung cấp điện năng cho khoảng 160.000 hộ gia đình.

Hệ thống năng lượng mặt trời không đòi hỏi thêm nhiên liệu khác để hoạt động và tác động ô nhiễm môi trường gần như là không có. Ánh sáng mặt trời có thể được lưu lại thành nhiệt để sử dụng ngay hoặc chuyển đổi thành điện năng. Ngoài ra công nghệ còn cho phép biến đổi ánh sáng thành năng lượng điện thông qua hiệu ứng quang điện…

Lợi thế của năng lượng mặt trời:

* Năng lượng mặt trời là một nguồn năng lượng sạch.
* Miễn phí và có sẵn với số lượng đầy đủ trong hầu như tất cả các phần của thế giới.

Những hạn chế của hệ thống năng lượng mặt trời bao gồm:

* Chi phí ban đầu cao.
* Đòi hỏi không gian sử dụng khá lớn
* Hiệu suất sử dụng có thể bị ảnh hưởng bởi sự ô nhiễm không khí và thời tiết làm giảm lượng ánh sáng mặt trời.
  1. **Vấn đề của luận văn**

Năng lượng mặt trời ngày nay đang trở nên rất phổ biến. Rất nhiều tấm năng lượng mặt trời đã được lắp đặt trên toàn thế giới. Hầu hết trong số chúng đều được cài đặt cố định theo hướng của bức xạ ánh sáng mặt trời tối đa. Vấn đề đặt ra là mặt trời đang chuyển động hằng ngày. Vì vậy, chúng ta không thể sử dụng bức xạ tối đa của mặt trời tất cả các thời gian. Đối với các hệ thống năng lượng mặt trời trên các hệ thống chuyển động (ví dụ: tàu thuyền, xe quân sự, vệ tinh, cáp treo …), chúng không những bị ảnh hưởng bởi chuyển động của mặt trời mà còn bị ảnh hưởng bởi sự dịch chuyển và vị trí của chúng.

Vấn đề xảy ra trên các hệ thống tĩnh: năng lượng thu được trên các tấm chênh lệch rất lớn tại các thời điểm khác nhau. Năng lượng bức xạ tối đa thu được trên tấm năng lượng mặt trời chỉ xuất hiện tại một thời điểm trong ngày.



Hình 1: Tấm năng lượng mặt trời trên các hệ thống tĩnh.

* + 1. **Giải pháp**

Theo dõi mặt trời là cách tốt nhất để tiếp nhận bức xạ tối đa. Bằng cách di chuyển các tấm năng lượng theo hướng dịch chuyển của mặt trời, chúng ta có thể thu được các bức xạ tối đa. Các hệ thống theo dõi mặt trời được chia ra làm 2 loại chính: hệ thống bám một trục, hệ thống bám 2 trục [18].

* **Hệ thống theo dõi mặt trời một trục**

Hệ thống này có thể có một trục ngang hoặc một trục dọc. Loại một trục ngang được sử dụng ở những vùng nhiệt đới, nơi nhận được rất nhiều ánh nắng mặt trời vào buổi chiều, tuy nhiên ngày ở đây ngắn. Loại một trục đứng được sử dụng ở những địa điểm có vĩ độ cao, với ánh sáng mặt trời không mạnh nhưng thời gian ban ngày vào mùa hè rất dài.

* **Hệ thống bám mặt trời 2 trục**

Hệ thống này có 2 trục: trục dọc và trục ngang. Do đó, nó có thể bám theo hướng mặt trời một cách chính xác tại bất kỳ nơi nào trên trái đất. Hệ thống này theo dõi mặt trời theo cả 2 hướng Đông-Tây, Bắc-Nam, bởi vậy cho năng suất rất cao, năng lượng điện thu được có thể tăng đến 40%.

* + 1. **Một số hệ thống theo dõi mặt trời hiện nay**

Hệ thống theo dõi mặt trời được chia làm 3 loại chính, tùy thuộc vào sự kết hợp các thành phần điều khiển, cảm biến và hệ thống định vị: hệ thống thụ động, hệ thống chủ động và hệ thống vòng lặp mở [18].

1. **Hệ thống theo dõi thụ động**

Sử dụng bức xạ của mặt trời để làm nóng không khí gây ra sự dịch chuyển của hệ thống theo dõi.

Ưu điểm: giá thành rẻ, dễ triển khai.

Nhược điểm: Đáp ứng chậm do phục thuộc vào nhiệt độ mặt trời và hệ thống khí được sử dụng, cần bảo trì thường xuyên.

1. **Hệ thống chủ động**

Sử dụng hệ thống điều khiển bằng điện hoặc thủy lực cùng với cơ cấu bánh răng, cơ cấu chấp hành để dịch chuyển hệ thống. Hệ thống này đo độ sáng của mặt trời bằng các cảm biến ánh sáng từ đó xác định được vị trí mà mô đun theo dõi cần hướng đến. Các cảm biến ánh sáng được đặt trên các vị trí của khác nhau trên hệ thống theo dõi. Nếu mặt trời không chiếu thẳng vào tấm năng lượng mặt trời. Khi đó sẽ có sự chênh lệch về cường độ sáng ở đầu ra của các cảm biến. Từ đó chúng ta có thể xác định được hướng chiếu sáng của mặt trời để dịch chuyển tấm năng lượng đến hướng chiếu sáng của mặt trời. Việc dịch chuyển các tấm năng lượng có thể được thực hiện bằng cách sử dụng động cơ bước hoặc động cơ servo.

Ưu điểm: đáp ứng nhanh hơn so với hệ thống thụ động.

Nhược điểm: vào những ngày trời u ám hoặc trời mưa hệ thống có thể hoạt động không chính xác.

1. **Hệ thống theo dõi vòng lặp mở**

Hệ thống theo dõi vòng lặp mở tính toán vị trí của mặt trời bằng cách ứng dụng các kết quả nghiên cứu về quỹ đạo mặt trời. Hệ thống này sử dụng thuật toán trên vi điều khiển để tính toán vị trí của mặt trời dựa vào thời gian, tọa độ và góc phương vị của mặt trời.

* **Theo dõi theo thời gian**:

Một bộ định thời được dùng để dịch chuyển hệ thống theo một chu kỳ nhất định để giữ tấm năng lượng hướng về phía mặt trời. Chu kỳ dịch chuyển của hệ thống phụ thuộc vào chu kỳ dịch chuyển của mặt trời.

Ưu điểm: có thể sử dụng trong hệ thống một trục hoặc hai trục.

Nhược điểm: bị ảnh hưởng bởi sự thay đổi của các mùa trong năm.

* **Theo dõi theo tọa độ và góc phương vị của mặt trời**

Sử dụng các dữ liệu thiên văn hoặc các thuật toán để tính toán vị trí của mặt trời để xác định vị trí của mặt trời tài một thời điểm và địa điểm xác định. Hệ thống này sử dụng vi điều khiển để tính toán các vị trí đó, đồng thời điều khiển các động cơ (động cơ bước hoặc động cơ servo) để dịch chuyển tấm năng lượng đến vị trí mong muốn.

* + 1. **Mục tiêu của luận văn.**

Mục tiêu của luận văn này là thiết kế hệ thống bám mặt trời cho các tấm năng lượng được đặt trên các hệ thống chuyển động, cụ thể ở đây là trên các tàu hoạt động trên biển. Hệ thống điều khiển tấm năng lượng mặt trời trên tàu đề xuất gồm các thành phần sau.

* Xác định vị trí của mặt trời bằng cách tính góc nâng và góc phương vị của mặt trời dựa trên các nghiên cứu về quỹ đạo của mặt trời.
* Xác định vị trí của tàu dựa trên các cảm biến mà cụ thể là cảm biến la bàn số và IMU. Trong đó, cảm biến la bàn số được sử dụng để xác định hướng của tàu so với trục bắc-nam. IMU được sử dụng để điều chỉnh dữ liệu đầu ra của la bàn [10] do la bàn ở vị trí nằm nghiêng so với mặt phẳng nằm ngang của trái đất và độ nghiêng của tàu so với phương thẳng đứng. Hệ thống sử dụng bộ lọc bù để kết hợp ưu điểm của 2 thành phần của cảm biến IMU (gia tốc góc và con quay hồi chuyển) để đưa ra dữ liệu chính xác về góc.
* Xác định độ lệch của tấm năng lượng mặt trời đặt trên tàu so với mặt trời theo cả phương thẳng đứng và phương ngang.
* Điểu khiển tấm năng lượng mặt trời xoay đến vị trí nhiều bức xạ nhất. Việc điều khiển được thực hiện bởi 2 động cơ bước. Một động cơ bước được dùng đề điều khiển tấm năng lượng mặt trời quay theo phương ngang và động cơ còn lại được dùng để quay tấm năng lượng theo phương thẳng đứng.

Hệ thống này dựa trên hệ thống theo dõi vòng lặp mở ở trên. Tuy nhiên, hệ thống này được gắn trên tàu, do tàu luôn dịch chuyển nên chúng ta cần tính đến cả vị trí của tàu cùng với các tác động của các thành phần bên ngoài lên tàu (ví dụ: lực đẩy của sóng biến…).

# NGUYÊN TẮC HOẠT ĐỘNG

* 1. **Quỹ đạo của mặt trời**

Quỹ đạo mặt trời trên hệ tọa độ trái đất thay đổi theo hướng từ tây sang đông trên cùng một đường với kinh độ trái đất. Kinh độ của mặt trời thay đổi từ 00 đến 1800 so với đường xích đạo khi mặt trời chiếu sáng. Tốc độ góc nâng của mặt trời là 150/giờ. Góc phương vị của mặt trời dịch chuyển từ 23.450 hướng bắc tới -23.450 hướng nam và quay lại theo chu kỳ 1 năm. Gây ra bởi sự nghiêng của Trái đất về phía mặt trời hàng năm. Góc nghiêng và góc lệch theo giờ được tính toán theo công thức sau [6]:

 ,

 ,

trong đó:

δs : Góc lệch của mặt trời theo hướng bắc-nam.

n: Số thứ tự của ngày trong năm, tính từ ngày 01 tháng 01.

hsn: Giờ mặt trời tính theo gốc là 12h trưa.

**Tính góc lệch theo giờ của mặt trời:**

Giờ kinh tuyến địa phương (LSTM) được tính theo công thức [12]:

,

trong đó:

ΔTGMT: là múi giờ so với giờ tiêu chuẩn GMT

Phương trình của thời gian (EoT) (tính bằng phút) là một phương trình thực nghiệm dùng để sửa chữa độ lệch tâm của quỹ đạo của trái đất và độ nghiêng trục quay của trái đất [12]:

,

với B được tính theo công thức sau [12]:

,

trong đó:

n: Số thứ tự của ngày trong năm.

Độ lệch giữa giờ địa phương (LT) và giờ mặt trời địa phương (LST):

,

trong đó:

Long: Kinh độ tại vị trí xác định.

LSTM: Giờ kinh tuyến địa phương.

EoT: Phương trình của thời gian.

Giờ mặt trời địa phương (LST) được tính theo công thức sau [12]:

 .

Góc lệch theo giờ của mặt trời được tính theo công thức sau [12]:

 ,

trong đó:

LST: giờ mặt trời địa phương

* 1. **Tọa độ di chuyển của tàu**

Khi một tàu di chuyển, tọa độ của nó luôn luôn thay đổi. Vì vậy góc nâng và góc phương vị luôn thay đổi theo thời gian. Khi tàu ở một vị trí nhất định trên trái đất, với vĩ độ của nó (*Lat*) ta có thể xác định được góc nâng của nó từ công thức [6]:

,

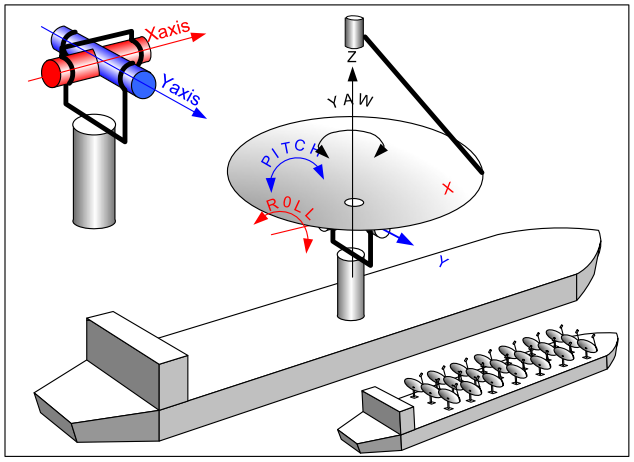
và góc phương vị A được tính theo công thức [6]:

 .

**Sự dịch chuyển của tàu:**

Một tàu lớn trên biển thực hiện các dịch chuyển với 6 bậc tự do bao gồm:

*  - Dịch chuyển theo phương dọc.
*  - Dịch chuyển lắc lư theo phương ngang.
*  - Nâng lên theo phương thẳng đứng.
*  - Quay theo trục dọc.
*  - Quay quanh trục ngang.
*  - Quay quanh trục thẳng đứng.



Hình 2: Sự dịch chuyển của tàu.

So với mặt trời, các dịch chuyển x, y, z của tàu không ảnh hưởng đến vị trí góc của mặt trời. Trong khi đó,  và ảnh hưởng trực tiếp đến vị trí của góc của mặt trời. Sự ảnh hưởng trên trục X bao gồm: góc nghiêng  (quay quanh trục X, theo phương vuông góc với trục X) và góc hướng  (quay quanh trục Z, theo phương vuông góc với trục Z) thay vì góc nâng của mặt trời trên trục X . Sự ảnh hưởng lên trục Y bao gồm: góc chúc  (quay quanh trục Y, theo phương vuông góc với trục Y) và góc hướng, thay vì góc nâng của mặt trời trên trục Y.

* 1. **Mô hình toán học**

Từ kết quả của phân tích toán học cho hệ thống theo dõi mặt trời ở trên ta thấy, các trục tọa độ là trung tâm của các góc đầu vào và đầu ra. Từ hình 3 ta thấy góc của trục X được tính theo công thức sau [7]:

 ,

và của trục Y được tính theo công thức sau [7]:

,

trong đó:

 - Góc lệch của mặt trời trên trục X so với góc nghiêng*.*

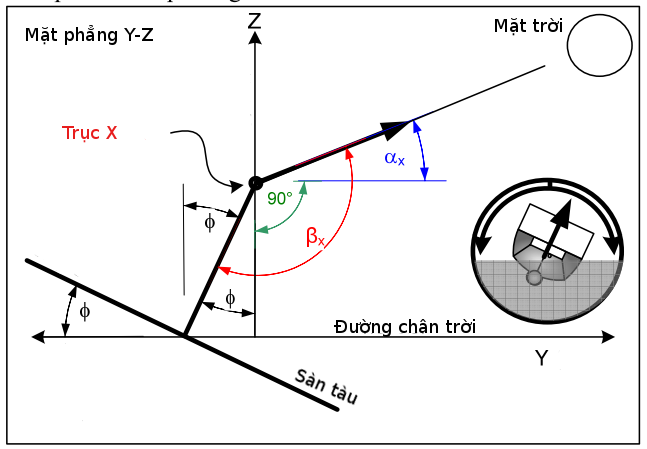
 - Góc lệch của mặt trời trên trục Y so với góc chúc hướng*.*

 - Góc lệch của mặt trời trên trục X so với phương ngang.

 - Góc lệch của mặt trời trên trục Y so với phương ngang.

 - Góc cuộn so với phương thẳng đứng.

 - Góc chúc ngóc so với phướng thẳng đứng.



Hình 3: Mô hình toán học của trục X trên mặt phẳng Y-Z.

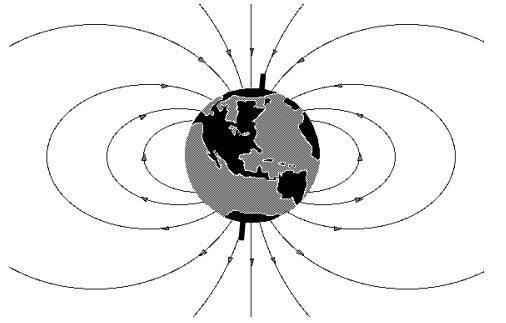
* 1. **Hệ thống đề xuất**

Vị trí cần được xác định là độ lệch của tàu so với hướng bắc-nam. Trong nội dung đề tài này, cảm biến la bàn số được sử dụng để xác định độ lệch của tàu so với hướng bắc-nam.

* + 1. **Sơ lược về cảm biến la bàn số**

Cảm biến la bàn số được dùng để đo từ trường của trái đất nhằm xác định các hướng bắc, nam, đông, tây ở trên mặt trái đất. Ban đầu nó được phát triển và sử dụng gần như dành riêng cho mục đích định vị và theo dõi, chủ yếu là trong quân đội và hàng không vũ trụ. Tuy nhiên ngày nay nó đang được sử dụng rộng rãi trong rất nhiều lĩnh vực: Ô tô, hàng hải, hàng không vũ trụ, công nghiệp, điện tử tiêu dùng…

Ngày nay, cảm biến la bàn số được chủ yếu được sản xuất dựa trên công nghệ MEMS và hoạt động dựa trên một số phương pháp cơ bản: hiệu ứng Hall, điện trở từ khổng lồ, từ trở khổng lồ (GMR), cảm biến tiếp giáp đường hầm (MTJ), từ trở không đẳng hướng (AMR) và lực Lorentz. Mỗi loại có những ưu và khuyết điểm riêng của nó.



Hình 4: Từ trường của trái đất.

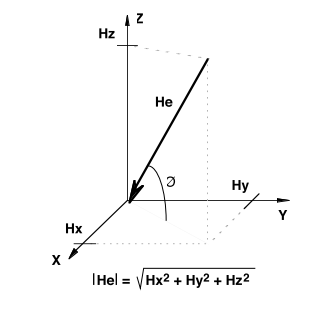
* + 1. **Tính toán phương hướng dựa trên la bàn số**

Đầu ra của la bàn số là từ trường đo được theo 3 trục X, Y, Z lần lượt là Hx, Hy, Hz như trong hình 5. Hướng của vật thể theo trục Bắc-Nam có thể được xác định chỉ bằng các giá trị từ trường Hx và Hy bằng cách đặt la bàn số trên một mặt phẳng nằm ngan song song với bề mặt Trái đất. Hướng mà la bàn số chỉ ra được tính toán dựa trên đầu ra X, Y của nó theo công thức [8]:

 .

Khi xét đến giới hạn của hàm arctan và dấu của đầu vào X, Y. Hướng của vật thể được tính theo bộ công thức sau [8]:





Hình 5: Từ trường của trái đất trên 3 trục tọa độ.

Hạn chế:

* Công thức chỉ đúng khi la bàn nằm trên mặt phẳng song song với bề mặt tiếp tuyến của trái đất.
* Khi la bàn đặt nghiêng so với bề mặt tiếp tuyến, sai số của phép đo là rất lớn.



Hình 6: La bàn số đặt nghiêng so với mặt phẳng nằm ngang của trái đất.

Để giải quyết hạn chế trên, chúng ta cần sử dụng la bàn 3 trục (X-Y-Z) và cảm biến IMU để điều chỉnh hướng của vật thể dựa trên độ nghiêng của la bàn.

* + 1. **Cảm biến IMU**

Cảm biến IMU tích hợp cảm biến gia tốc 3 trục và con quay hồi chuyển 3 trục trên cùng 1 đế silicon. Cảm biến IMU có đặc trưng là các góc nghiêng (), góc chúc () và góc hướng () theo các trục X, Y, Z tương ứng. Đầu ra của cảm biến gia tốc chính là gia tốc chính là gia tốc theo các trục X, Y, Z, đơn vị là g(với 1g = 9.8m/s2). Đầu ra của cảm biến con quay hồi chuyển 3 trục là vận tốc góc theo các trục X, Y và Z, đơn vị là 0/s.

Tính các góc quay dựa trên đầu ra của cảm biến gia tốc [9]:

* **Góc nghiêng ():**

 .

* **Góc chúc ():**

 ,

trong đó: aX, aY, aZ là đầu ra của cảm biến gia tốc theo các trục X, Y, Z.

Tính các góc quay dựa trên dữ liệu đầu ra của con quay hồi chuyển:

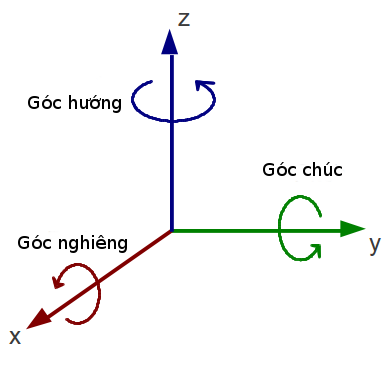
* **Góc nghiêng *():***

 .

* **Góc chúc ():**

 ,

trong đó: *gX*, *gY* là đầu ra của cảm biến con quay hồi chuyển theo các trục X, Y.



Hình 7: Các góc quay trên các trục tọa độ Đề Các.

* + 1. **Xác định vị trí của tàu dựa trên các cảm biến**

Với các góc nghiêng đo được bằng cảm biến IMU, giá trị X, Y trên mặt phẳng song song với mặt phẳng tiếp tuyến được tính như sau [10]:



Khi đó công thức được viết lại như sau:



* + 1. **Vị trí của tàu so với mặt trời**

Góc lệch của tàu so với mặt trời theo phương ngang được tính theo công thức:

,

trong đó: A và D được tính theo các công thức,.

Góc lệch của tàu so với mặt trời theo phương ngang được tính theo công thức. Trong đó giờ mặt trời hsn là xác định tại một khu vực nhất định.

* 1. **Bộ lọc bù**

Một bộ lọc bù về cơ bản là sự kết hợp của một bộ lọc thông tấp và một bộ lọc thông cao tín hiệu số để tạo ra một ước lượng chính xác về góc.

Mục đích của bộ lọc thông thấp [11] [13] là lọc ra những thay đổi ngắn hạn và loại bỏ đi những thay đổi dài hạn. Một cách để làm điều này là tính toán các thay đổi từng bước một thông qua vòng lặp chương trình. Phương trình của bộ lọc thông thấp:

,

trong đó: yn – Giá trị của đầu ra tại thời điểm n.

yn-1 – Giá trị đầu ra tại thời điểm n-1.

xn – Giá trị đầu vào tại thời điểm n.

a – Hệ số của bộ lọc.

Hệ số của bộ lọc được tính theo công thức sau [11]:

,

trong đó: τ – Hằng số thời gian.

dt – Chu kỳ lấy mẫu.

Ví dụ, một góc θ thay đổi đột ngột từ giá trị 0 đến 10°, khi đó những ước lượng về góc thay đổi từng bước thay vì cập nhật thay đổi một cách ngay lập tức:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Lặp | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| θ | 0.200 | 0.400 | 0.590 | 0.780 | 0.960 | 1.140 | 1.320 | 1.490 | 1.660 | 1.830 |

Nếu giá trị góc đọc được từ cảm biến vẫn ở mức 10°, ước lượng về góc sẽ tăng lên cho đến khi nó đạt được giá trị đó. Thời gian cần để đạt được giá trị đó phụ thuộc vào các hằng số bộ lọc (trong ví dụ này là 0.98 và 0.02) và thời gian lấy mẫu (dt).

Bộ lọc thông cao [14] lọc ra những dữ liệu ổn định theo thời gian và loại bỏ đi những thay đổi ngắn hạn. Phương trình của bộ lọc thông cao:

,

trong đó: yn – Giá trị của đầu ra tại thời điểm n.

yn-1 – Giá trị đầu ra tại thời điểm n-1.

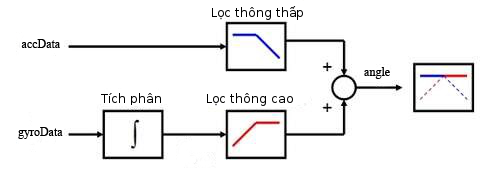
xn, xn-1 – Giá trị đầu vào tại thời điểm n và n-1.

a – Hệ số của bộ lọc.

Hình 7 là mô hình của bộ lọc bù được sử dụng để ước lượng góc quay dựa trên dữ liệu đọc được từ cảm biến của gia tốc và cảm biến con quay hồi chuyển. Bộ tích phân được dùng để tính góc quay của IMU tại một thời điểm dựa trên sự thay đổi của vận tốc góc *gyroData (0/s)* trong một khoảng thời gian *dt* (s) nhất định.



Các thành phần dữ liệu đo được từ cảm biến gia tốc ổn định trong một khoảng thời gian dài. Do đó chúng ta có thể xử dụng bộ lọc thông thấp cho các thành phần dữ liệu này. Dữ liệu từ cảm biến con quay hồi chuyển chỉ đáng tin cậy ở ngắn hạn, nó sẽ bị trôi sau một khoảng thời gian đủ dài. Vì vậy bộ lọc thông cao được sử dụng để xử lý dữ liệu từ cảm biến con quay hồi chuyển. Vì các cảm biến gia tốc và con quay hồi chuyển được đặt trên cùng một chip được gọi là IMU. Do đó chu kỳ lấy mẫu của IMU là cố định do đó hệ số của bộ lọc thông thấp và bộ lọc thông cao là như nhau trong trường hợp này.



Hình 8: Mô hình bộ lọc bù.

Khi kết hợp các thành phần của hệ thống trên ta có phương trình của bộ lọc bù như sau [11]:

 ,

trong đó: angle - Góc cần xác định.

gyroData – Vận tốc góc đo được từ cảm biến con quay hồi chuyển (0/s).

accData – Góc Dữ liệu đọc được từ cảm biến gia tốc (0).

dt - Chu kỳ lấy mẫu.

a – Hệ số của bộ lọc.

Hệ số của bộ lọc được tính theo công thức, trong đó τ là hằng số thời gian. Hằng số thời gian (τ) được chọn sao cho phải lớn hơn khoảng thời gian của nhiễu điển hình trên cảm biến gia tốc, đồng thời giá trị này phải đảm bảo sao cho điểm gốc của cảm biến của cảm biến không bị lệch trong một khoảng chấp nhận được.

Ví dụ: đối với cảm biến IMU, thành phần đầu ra từ con quay hồi chuyển xảy ra hiện tượng trôi điểm không với vận tốc ±200/s [15] trên toàn dải ±20000/s [16], như vậy hiện tượng trôi điểm không sẽ chiếm khoảng 1% toàn dải trong 1 giây. Để giảm hiện tượng trôi điểm không, chúng ta có thể chọn hằng số thời gian nhỏ hơn 1 giây. Nếu chúng ta chọn hằng số thời gian τ = 0.75s < 1s. Với tần số lấy mẫu là 500Hz, tương ứng với khoảng thời gian giữa 2 lần lấy mẫu là 0.02s. Hệ số của bộ lọc sẽ có giá trị như sau:

.

Khi đó công thức được viết lại như sau:

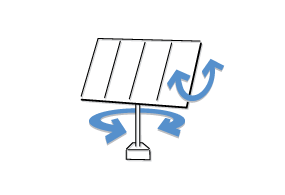
.

# MÔ HÌNH THỰC TẾ

* 1. **Phần cứng**

Hệ thống đề xuất trong nội dung luận văn này là hệ thống theo dõi mặt trời 2 trục như trong Hình 8. Tấm năng lượng mặt trời được điều khiển theo 2 hướng:

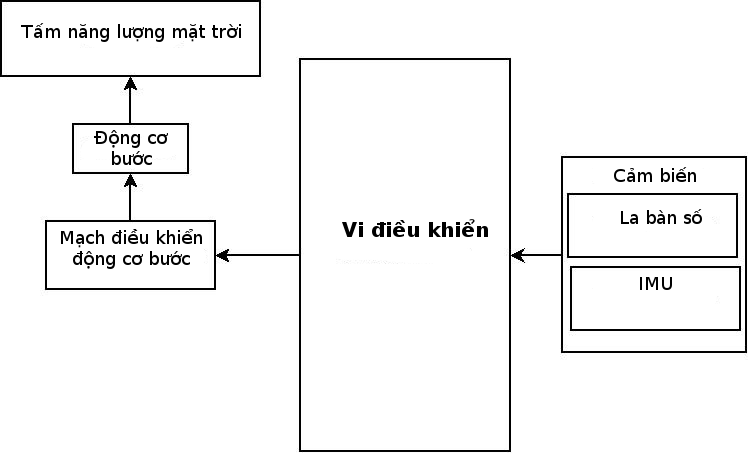
* Thay đổi góc nâng theo hướng từ đông sang tây với vận tốc 150/h. Thay đổi này được tính theo công thức.
* Thay đổi theo hướng Bắc-Nam tùy thuộc vào độ lệch của mặt trời (tính theo ngày) và độ lệch của tàu so với trục Bắc-Nam. Thay đổi này được tính theo công thức.



Hình 9: Hệ thống theo dõi mặt trời 2 trục.

Sơ đồ khối của hệ thống được mô tả trong Hình 9. Các thành phần chính của hệ thống bao gồm:

* Hệ thống điều khiển.
* Hệ thống cảm biến: cảm biến la bàn số, cảm biến IMU.
* Thiết bị truyền động: động cơ bước, mạch điều khiển động cơ bước.

****

Hình 10: Sơ đồ khối phần cứng.

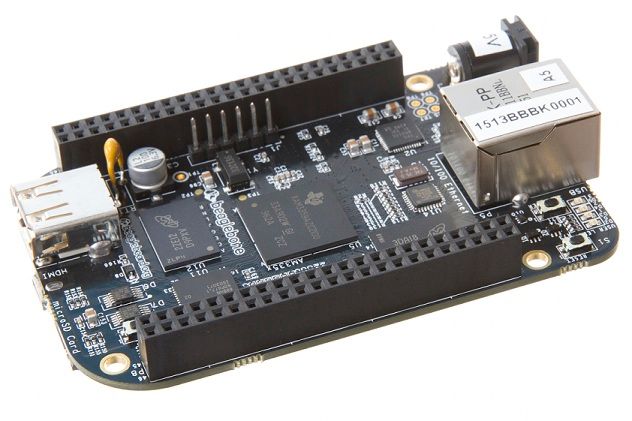
* + 1. **Hệ thống điều khiển**

1. **Giới thiệu kit Beaglebone Black (Rev C)**

Kit BeagleBone Black (Rev C) là một bộ kit phát triển dựa trên bộ xử lý AM3358 được tích hợp một lõi ARM Cortex A8 với xung nhịp hoạt động lên tới 1GHz và có các thiết bị ngoại vi phong phú. BeagleBone Black (Rev C) cung cấp rất nhiều các giao diện mở rộng như Ethernet, USB host và OTG, thẻ TF, serial, JTAG, micro HDMI, EMMC, ADC, I2C, SPI, PWM và LCD. BeagleBone Black (Rev C) được thiết kế để đáp ứng các yêu cầu khác nhau của các lĩnh vực khác nhau bao gồm: các thiết bị trò chơi, gia đình và tự động hóa công nghiệp, thiết bị y tế tiêu dùng, máy in, hệ thống thu phí thông minh, hệ thống trọng lượng của máy bán hàng tự động thông minh, thiết bị đầu cuối giáo dục và đồ chơi cao cấp.

Bảng 1: Các thông số kỹ thuật của kit BeagleBone Black

|  |  |  |
| --- | --- | --- |
|  | Tính năng | |
| Bộ vi xử lý | Sitara AM3358BZCZ100 – 1GHz, 2000 MIPS | |
| Bộ xử lý đồ họa | SGX530 3D | |
| Bộ nhớ SDRAM | 512 DDR3L 800MHz | |
| Bộ nhớ Flash | 4GB | |
| Cổng nạp và gỡ lỗi | JTAG 20 chân | |
| Nguồn | USB hoặc đầu cắm DC 5V | Nguồn DC 5V thông qua cổng mở rộng |
| Bộ chỉ thị | 1 đèn LED nguồn, 2 LED Ethernet, 4 LED có thể điều khiển được | |
| USB 2.0 client | USB0 | |
| USB 2.0 host | USB1 | |
| Cổng giao tiếp nối tiếp | UART0 thông qua cổng 6 chân 3.3V TTL | |
| Cổng mạng | 10/100, RJ45 | |
| Khe cắm thẻ nhớ SD/MMC | micro SD 3.3V | |
| Nút nhấn | Nút khởi động, nút nguồn, nút nhấn đầu ra | |
| Cổng xuất hình ảnh | 16bit HDMI, độ phân giải tối đa 1280x1024 | |
| Cổng xuất âm thanh | Thông qua giao diện HDMI | |
| Các cổng mở rộng | Nguồn 5V, 3.3V, VDD\_ADC, McASP0, SPI, I2C, GPIO (tối đa 59 chân), LCD, GPMC, MMC1, MMC2, 7 EHRPWM, ngắt XDMA, .. | |
| Trọng lương | 39.68g | |
| Nguồn | 5V, 3.3V | |



Hình 11: Kit BeagleBone Black (Rev C).

Beaglebone Black có thể được kết hợp với các mạch mở rộng "Capes" và các thư viện mã nguồn mở. Những tính năng này cho phép nâng cấp và mở rộng hệ thống một cách nhanh chóng. Sự đa dạng của Capes để mở rộng các tính năng xử lý có sẵn và hiện tại đang có trên 35 Capes khác nhau cho BeagleBone Black.

Về phần mềm Beaglebone Black được cài đặt sẵn hệ điều hành Angstrom Linux, môi trường phát triển tích hợp Cloud9 (IDE), thư viện và ngôn ngữ Bonescript giúp cho việc phát triển ứng dụng nhanh chóng. Ngoài ra, Beaglebone Black cũng được bổ sung các chức năng dựa trên ngôn ngữ C để tận dụng sức mạnh của bộ vi xử lý ARM Cortex A8 cũng như sử dụng các ngoại vi một cách hiệu quả.

1. **Sử dụng kit BeagleBone Black để điều khiển hệ thống.**

Trong nội dung luận văn này kit Beaglebone Black được sử dụng làm bộ phận xử lý chính của hệ thống. Các chức năng chính của kit bao gồm:

* Thu thập dữ liệu từ các cảm biến.
* Xử lý dữ liệu từ các cảm biến để xác định độ lệch về vị trí.
* Điều khiển động cơ bước dựa trên các giá trị độ lệch về vị trí.

Tài nguyên sử dụng bao gồm:

* I2C2: dùng để giao tiếp với cảm biến la bàn số.
* I2C1: dùng để giao tiếp với cảm biến IMU.
* 8 chân GPIO: điều khiển 2 động cơ bước.
* Nguồn VCC: cấp nguồn cho các cảm biến và mạch điều khiển động cơ.



Hình 12: Sơ đồ kết nối các thành phần của hệ thống.

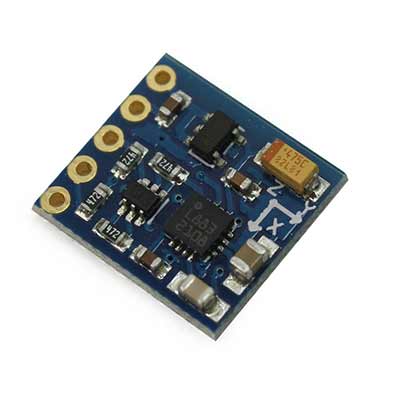
* + 1. **Hệ thống cảm biến.**

Các cảm biến được sử dụng trong hệ thống này bao gồm: cảm biến la bàn số GY-271 HMC5883L và cảm biến IMU GY-521 MPU6050.

1. **Mô đun cảm biến la bàn số GY271 HMC5883L**

GY-271 HMC5883L là mô đun cảm biến la bàn số dùng để đo từ trường theo 3 trục X, Y, Z, xác định độ lệch của vật thể so với cực Bắc, Nam. Thông số kỹ thuật của mô đun GY271 như sau:

* Chip xử lý: HMC5883L
* Nguồn cung cấp :3-5V
* Cổng giao tiếp: I2C
* Dải đo ± 1.3-8 Gaussian.



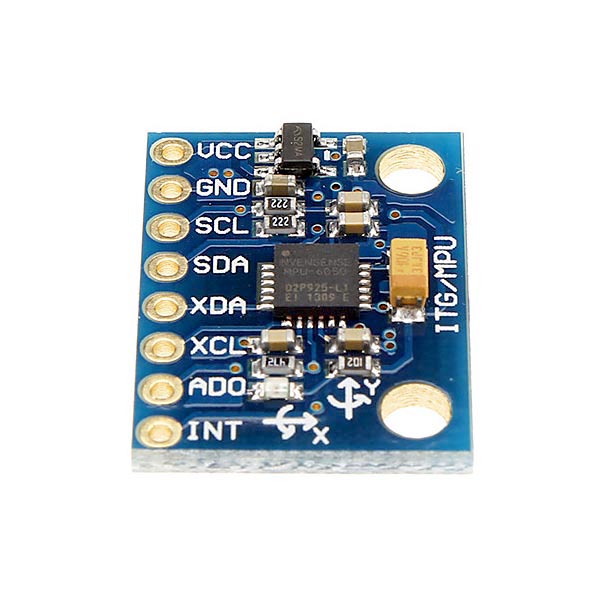
Hình 13: Mô đun cảm biến la bàn số GY-271 HMC5883L

1. **Mô đun cảm biến IMU GY-521.**

Mô đun GY-521 MPU6050 là mô đun cảm biến gia tốc và con quay hồi chuyển 3 trục.

Thông số kỹ thuật:

* Chip : MPU-6050 ( Đo được 3 trục góc + 3 trục gia tốc)
* Điện áp sử dụng: 3-5V
* Giao tiếp: I2C
* Hỗ trợ AD 16 Bit
* Độ phân giải góc : ± 250 500 1000 2000 °/s
* Độ phân giải gia tốc : ± 2 ± 4 ± 8 ± 16g



Hình 14: Mô đun cảm biến IMU GY-521 MPU6050

Cảm biến MPU6050 là một thiết bị theo dõi chuyển động từ công ty Invensense. Cảm biến này tích hợp một cảm biến gia tốc 3 trục và một con quay hồi chuyển 3 trục trên cùng một đế silicon cùng với bộ xử lý tín hiệu số chuyển động với khả năng xử lý thuật toán phức tạp.

* + 1. **Thiết bị truyền động**

1. **Động cơ bước**

Động cơ bước là một loại động cơ điện có nguyên lý và ứng dụng khác biệt với đa số các loại động cơ điện thông thường. Chúng thực chất là một động cơ đồng bộ dùng để biến đổi các tín hiệu điều khiển dưới dạng các xung điện rời rạc kế tiếp nhau thành các chuyển động góc quay hoặc các chuyển động của rô to và có khả năng cố định rô to vào những vị trí cần thiết.

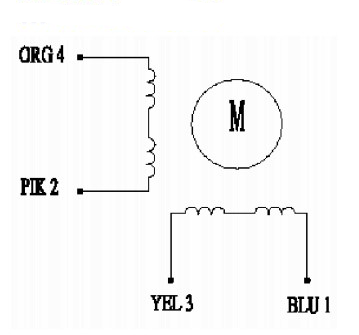
Động cơ bước phong phú về góc quay. Các động cơ kém nhất quay 900 mỗi bước, trong khi đó các động cơ nam châm vĩnh cửu xử lý cao thường quay 1.80 đến 0.720 mỗi bước. Với một bộ điều khiển, hầu hết các loại động cơ nam châm vĩnh cửu và hỗn hợp đều có thể chạy ở chế độ nửa bước, một vài bộ điều khiển có thể điều khiển các phân bước nhỏ hơn hay còn gọi là vi bước. Đối với cả động cơ nam châm vĩnh cửu hoặc động cơ biến từ trở, nếu chỉ một mấu của động cơ được kích, rô to (ở chế độ không tải) sẽ nhảy đến một góc cố định và sau đó giữ nguyên ở góc đó cho đến khi mô men xoắn vượt qua giá trị mô men xoắn giữ của động cơ. Do những ưu điểm trên mà động cơ bước thường được sử dụng trong các ứng dụng điều khiển chính xác. Trong nội dung luận văn này, động cơ bước được sử dụng để điều chỉnh góc quay một các chính xác.

Hệ thống theo dõi mặt trời trong nội dung luận văn này sử dụng 2 động cơ bước 28BYJ-48 để điều khiển tấm năng lượng xoay theo 2 trục như trong Hình 8. Các thông số kỹ thuật của động cơ 28BYJ-48:

* Điện áp hoạt động: 5-12VDC
* Số đầu vào điều khiển: 4
* Số bước: 64
* Trở kháng: 200Ω ± 7% (25 ℃)
* Điện trở cách điện: > 10MΩ (500V)
* Mô-men xoắn: > 34.3mN.m (120Hz).



Hình 15: Động cơ bước 28BYJ-48



Hình 16: Sơ đồ cuộn dây

Như trong Hình 14 và Hình 15 ta thấy động cơ bước 28BYJ-48 gồm có 5 dây kết nối gồm: dây cấp nguồn VCC màu đỏ và 4 đầu vào điều khiển (xanh, vàng, hồng và da cam). Động cơ này có thể điều khiển được ở 2 chế độ: đủ bước và nửa bước.

* Điều khiển đủ bước: mỗi lần cấp điện áp cho các cuộn dây, động cơ sẽ quay đủ 1 bước (5.6250), thứ tự cấp xung điều khiển như sau:

Bảng 2: Thứ tự cấp xung điều khiển đủ bước

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cuộn dây | Các bước cấp nguồn | | | |
| Bước 1 | Bước 2 | Bước 3 | Bước 4 |
| 4 ORG | 1 | 1 | 0 | 0 |
| 3 YEL | 0 | 1 | 1 | 0 |
| 2 PIK | 0 | 0 | 1 | 1 |
| 1 BLU | 1 | 0 | 0 | 1 |

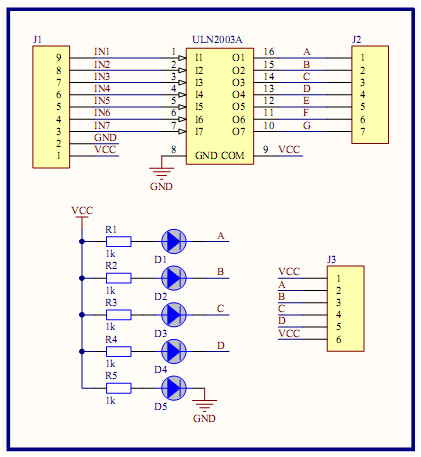
* Điều khiển nửa bước: để điều khiển động cơ quay nửa bước, thứ tự các xung cấp vào các cuộn dây như ở bảng dưới. Khi đó mỗi bước động cơ quay được 2.81250.

Bảng 3: Thứ tự cấp xung điều khiển nửa bước

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cuộn dây | Các bước cấp nguồn | | | | | | | |
| B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 |
| 4 ORG | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 3 YEL | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 PIK | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 BLU | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

1. **Mạch điều khiển động cơ bước:**

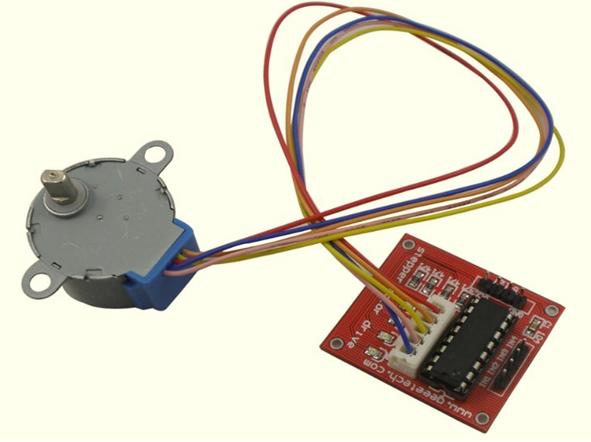
Mô đun điều khiển động cơ sử dụng mạch đệm ULN2803 với ưu điểm nhỏ gọn, có sẵn mạch đệm cho động cơ. ULN2003 là tổ hợp của 7 mạch Darlington được tích hợp lại. Nó được dùng rộng dãi trong các ứng dụng điều khiển động cơ, đèn LED…Mỗi kênh của UNL2803 có thể cho dòng đi qua tối đa là 0.5A.



Hình 17: Sơ đồ mạch điều khiển động cơ dùng ULN2803

Thông số kỹ thuật của mạch điều khiển động cơ ULN2803:

* Mạch đệm ULN2003.
* Điện áp cung cấp: 5 ~ 12VDC.
* Tín hiệu ngõ vào: 4 chân IN1, IN2, IN3, IN4.
* Tín hiệu ngõ ra: giắc cắm động cơ bước 28BYJ-48.
* Đèn led hiển thị trạng thái hoạt động của động cơ.



Hình 18: Kết nối động cơ bước với mạch điều khiển

* 1. **Phần mềm**
     1. **Các thành phần của hệ thống phần mềm**

Các thành phần chính của hệ thống phần mềm bao gồm:

* Hệ điều hành Angstrom Linux.
* Trình điều khiển thiết bị (I2C, GPIO, Ethernet…).
* Thư viện điều khiển phần cứng.
* Thư viện xử lý số.

Các bước thực hiện phần mềm của hệ thống bám mặt trời về cơ bản được mô tả như trong hình sau:



Hình 19: Các bước thực hiện phần mềm của hệ thống

* + 1. **Lập trình cho kit ARM BeagleBone Black và các thiết bị** 
       1. **Lập trình cho KIT BeagleBone**

Trong nội dung của luận văn này, phần mềm của hệ thống chạy trên hệ điều hành Linux và sử dụng cơ chế lập trình đa luồng giúp cho các thành phần của hệ thống (cảm biến, điều khiển, xử lý dữ liệu…) có thể được thực thi đồng thời. Các thành phần ngoại vi trên BeagleBone black được sử dụng bao gồm: GPIO, I2C, UART, Ethernet. Các ngoại vi này được sử dụng để điều khiển động cơ và giao tiếp với các cảm biến và gỡ lỗi chương trình.

1. **Lập trình giao tiếp đa luồng**

Phần mềm của hệ thống được phân thành nhiều luồng bằng cách sử dụng thư viện lập trình đa luồng “*pthread*” trên hệ điều hành Linux. Việc tạo ra nhiều luồng giúp cho nhiều luồng xử lý được thực hiện đồng thời. Cụ thể, trong hệ thống này, phần mềm xử lý được chia làm 3 luồng chính:

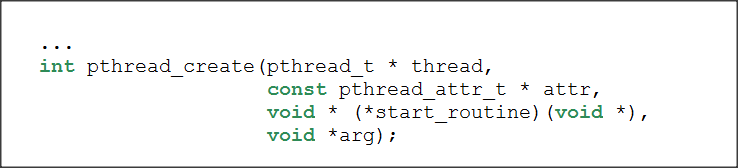
* Giao tiếp với cảm biến la bàn số và xử lý dữ liệu đọc được (compass\_thread).
* Giao tiếp với cảm biến IMU và xử lý dữ liệu đọc được (ins\_thread).
* Tổng hợp và xử lý các dữ lý đọc được từ cảm biến và điều chỉnh vị trí của tấm năng lượng thông qua điều khiển động cơ (data\_process\_thread).



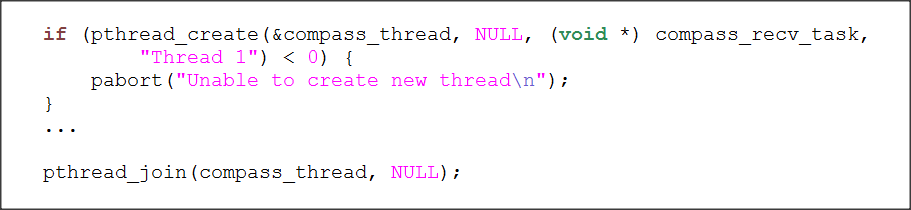
Hình 20: Xử lý đa luồng trong hệ thống điều khiển bám mặt trời

Mỗi luồng được tạo mới bằng cách sử dụng hàm pthread\_create với tham số đầu tiên là handle của luồng được tạo, nó được sử dụng để luồng chính và các luồng khác có thể tương tác với luồng đó, tham số thứ 3 là con trỏ một hàm. Hàm đó sẽ được thực thi khi luồng được tạo. Ở đây các hàm xử lý đó bao gồm các đoạn code xử lý dữ liệu được đặt trong vòng lặp vô tận, vì quá trình xử lý của các luồng này là liên tục và chỉ dừng lại khi hệ thống bị dừng.

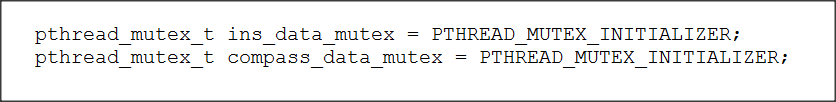
Khai báo của hàm pthread:



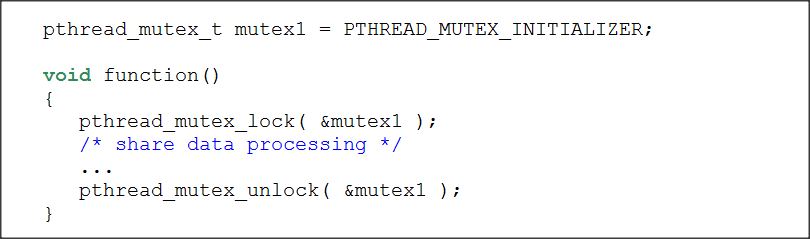
Ví dụ: tạo compass thread



Các luồng của hệ thống chia sẽ các dữ liệu với nhau, đó chính là dữ liệu đọc được từ các cảm biến. Do đó, trong quá trình thao tác với dữ liệu dùng chung có thể gây ra sự không nhất quán về dữ liệu cũng như xung đột về bộ nhớ dẫn đến làm mất dữ liệu. Để tránh tình trạng này, các mutex (được định nghĩa trong thư viện pthread) được sử dụng để tránh tình trạng trên. Ở đây, 2 mutex được khai báo đề bảo vệ các quá trình xử lý dữ liệu từ 2 cảm biến la bàn số và IMU.

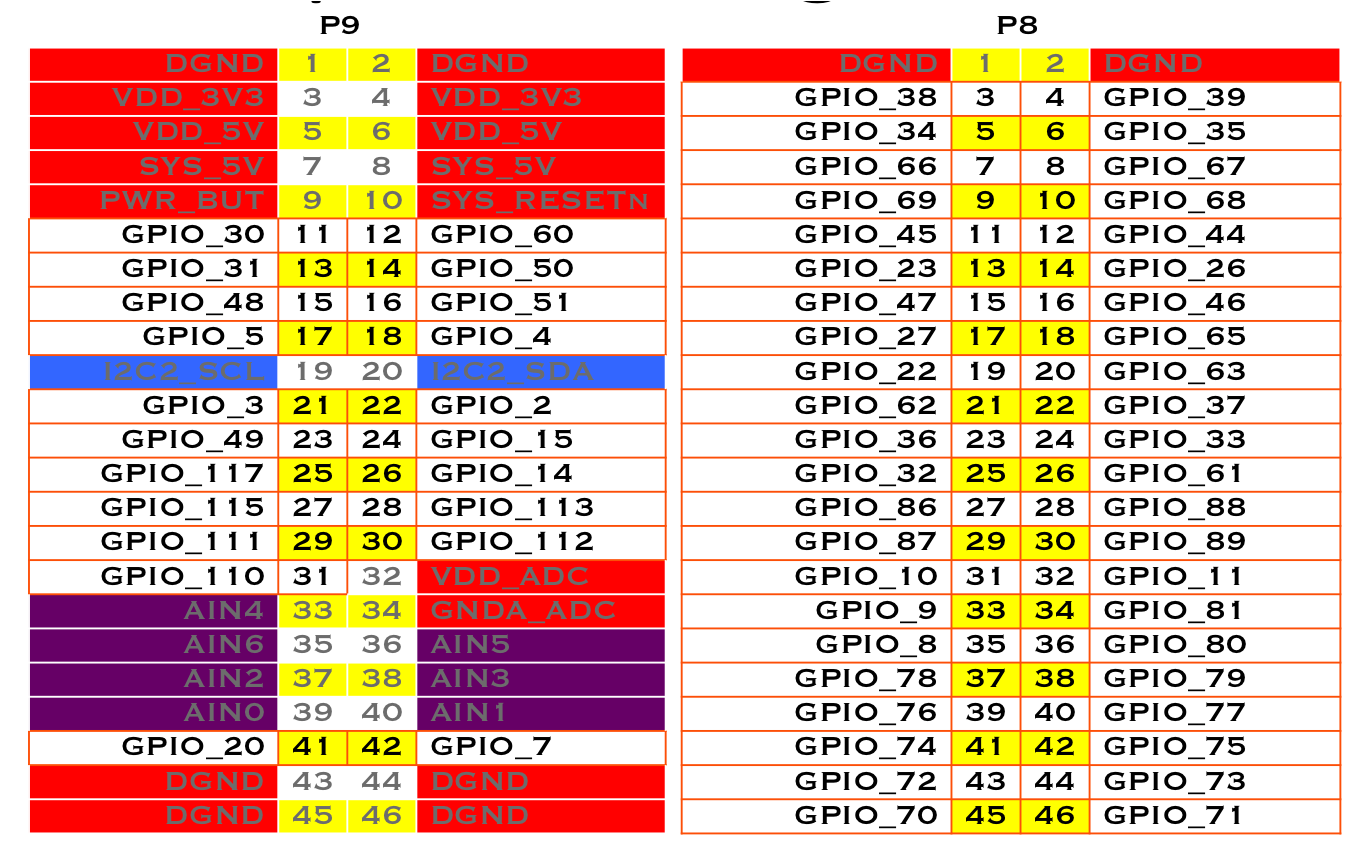


Ví dụ về khai báo và sử dụng mutex:



1. **Lập trình điều khiển tín hiệu vào ra với GPIO**

Bộ vi xử lý AM335x trên kit BeagleBone Black có 4 bộ điều khiển vào ra GPIO, được đặt tên lần lượt là gpio0[], gpio1[], gpio2[] và gpio3[]. Tương ứng với mỗi bộ điều khiển này sẽ có một số thanh ghi được sử dụng cho mục đích điều khiển. Mỗi bộ điều khiển sẽ đáp ứng cho 32 chân vào ra GPIO. Mỗi thanh ghi 32-bit có một chức năng riêng biệt: cấu hình chân (vào/ra, điện trở kéo lên…), điều khiển, đặt/xóa các trạng thái của chân. Các bit tương ứng của các thanh ghi được sử dụng để điều khiển một chân GPIO. Các chức năng được chọn thông qua các thanh ghi và các chân GPIO riêng biệt được chọn bởi các bít của thanh ghi. Trên hệ điều hành Linux, các thao tác làm việc trực tiếp với thanh ghi được thực hiện bởi nhân Linux (Linux kernel) thông qua trình điều khiển thiết bị (device driver). Chúng ta có thể điều khiển các cổng GPIO thông qua giao diện SysFS. Trên Linux, các chân GPIO được đánh số theo thứ tự tăng dần. Ví dụ: gpio/gpio0/ trên Linux tương ứng với bít 0 trên cổng GPIO 0, gpio/gpio1 tương ứng với bit 1 trên cổng GPIO0 và gpio/gpio32/ tương ứng với bit 0 trên cổng GPIO1.



Hình 21: Các chân vào ra khả dụng trên BeagleBone Black (65 chân).

Quan sát thư mục /sys/class/gpio/ ta thấy mỗi chân GPIO có 1 thư mục riêng, chúng được sử dụng để kiểm soát chân đó. Thư mục đó chỉ tồn tại khi chúng ta ghi số thứ tự của chân GPIO vào thư mục hệ thống /sys/class/gpio/export.

root@beaglebone:~# cd /sys/class/gpio/

root@beaglebone:/sys/class/gpio# ls

export gpio60 gpiochip0 gpiochip32 gpiochip64 gpiochip96 unexport

root@beaglebone:/sys/class/gpio# echo 60 > export

root@beaglebone:/sys/class/gpio# ls

export gpio60 gpiochip0 gpiochip32 gpiochip64 gpiochip96 unexport

Tùy vào chức năng mở rộng của từng chân ta sẽ có số lượng các tệp tin trong thư mục là khác nhau. Tuy nhiên, thư mục của mỗi chân GPIO sẽ có 3 tệp tin được sử dụng cho mục đích vào ra như trong Bảng 4.

Bảng 4: Các tệp tin cơ bản trong thư mục SysFS của GPIO.

|  |  |
| --- | --- |
| **Tệp tin** | **Chức năng** |
| direction | Bằng cách ghi giá trị ***low*** hoặc ***high*** vào tệp tin này, chân GPIO sẽ có chức năng tương ứng là vào hoặc ra |
| value | Nếu direction là in, ta sẽ đọc được giá trị của chân GPIO này là 1 hoặc 0 tương ứng với mức điện áp tương ứng trên chân đầu vào là 3.3V hoặc 0V  Nếu direction là out, khi ghi giá trị 1 hoặc 0 vào tệp tin này sẽ đặt mức điện áp trên chân đầu ra tương ứng là 3.3V và 0V. |
| active\_low | Nếu ghi 1 vào tệp tin này, ý nghĩa của giá trị value sẽ bị đảo ngược:  0 tương ứng với mức điện áp 3.3V còn 1 tương ứng với mức điện áp 0V |

Ví dụ về thao tác vào ra trên chân GPIO trên Linux thông qua bash shell:

* Đặt chân gpio60 (bit 28 của cổng vào ra GPIO 1) làm đầu ra:

echo high > /sys/class/gpio/gpio60/direction

* Ghi giá trị 1 ra chân gpio60:

echo 1 > /sys/class/gpio/gpio60/value

* Đặt chân gpio60 làm đầu vào:

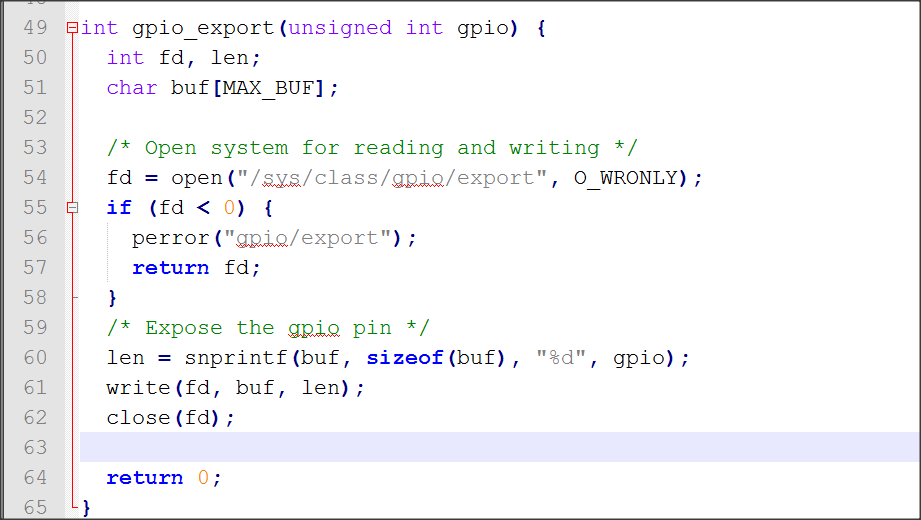
echo low > /sys/class/gpio/gpio48/direction

* Đọc giá trị đầu vào trên chân gpio60:

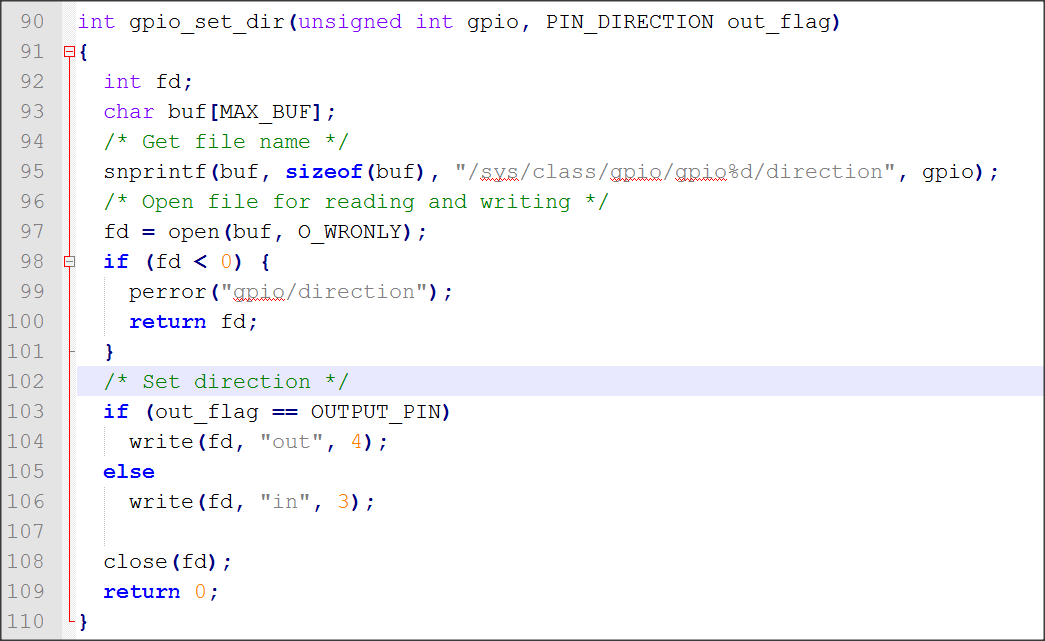
cat /sys/class/gpio/gpio60/value

Chúng ta cũng có thể thực hiện các thao tác trên bằng ngôn ngữ lập trình C thông qua các hàm gọi hệ thống (*system call*):

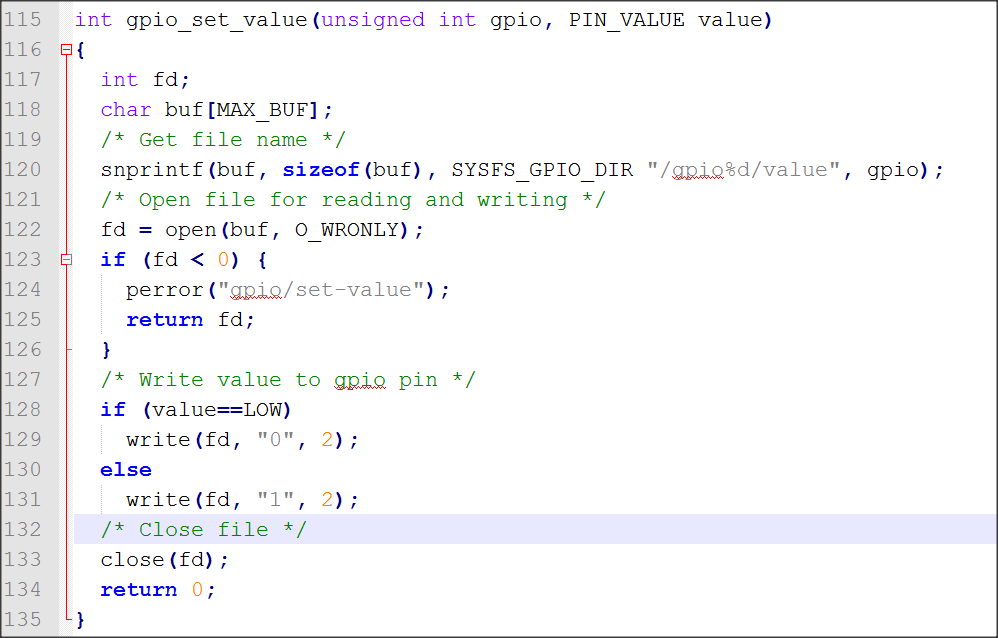
* Tạo thư mục của chân gpio để điều khiển:



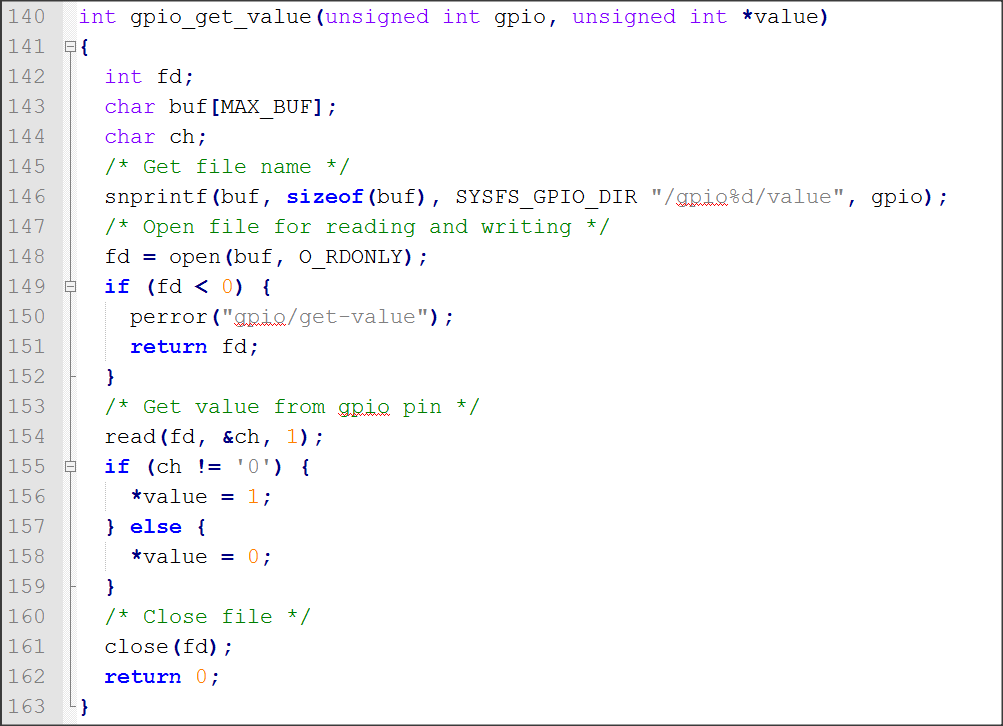
* Thiết lập hướng vào/ra:



* Ghi giá trị ra chân GPIO:



* Đọc giá trị từ chân GPIO:



1. **Lập trình giao tiếp I2C**

BeagleBone Black có 3 giao diện I2C:

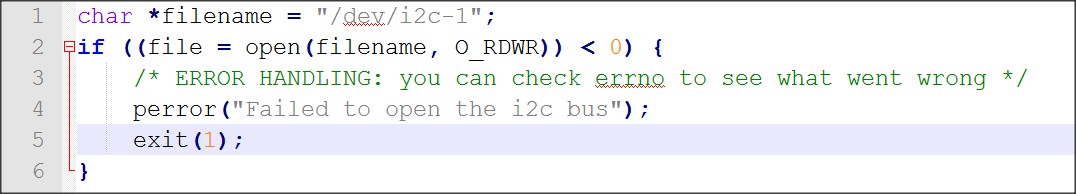
* I2C0: có địa chỉ 0x44E0\_B000. Các chân giao tiếp không được đưa ra trên cổng mở rộng.
* I2C1: có địa chỉ 0x4802\_A000. Các chân 17, 18 (và 24, 26) của cổng mở rộng P9.
* I2C2: có địa chỉ 0x4819\_C000. Các chân 19, 20 (và 21, 22) của cổng mở rộng P9.

Trong nội dung luận văn này sử dụng 2 giao diện I2C để giao tiếp với các cảm biến. Tuy nhiên chỉ có I2C0 và I2C2 được cấu hình để hoạt động ở chế độ mặc định và các chân của I2C0 đưa ra cổng mở rộng. Do đó chúng ta cần cấu hình I2C1 để nó có thể hoạt động. Việc này có thể được thực hiện thông qua device tree trên Linux.

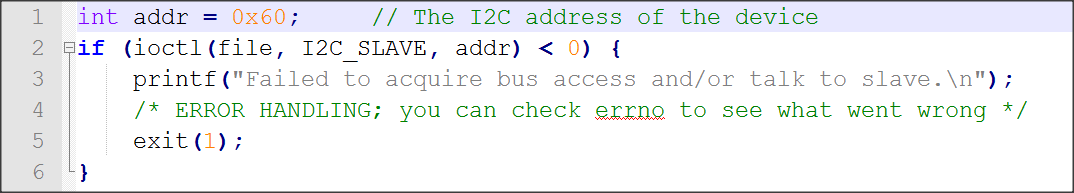


Tương tự như GPIO, chúng ta lập trình giao tiếp I2C bằng ngông ngữ C thông qua các hàm “system call”.

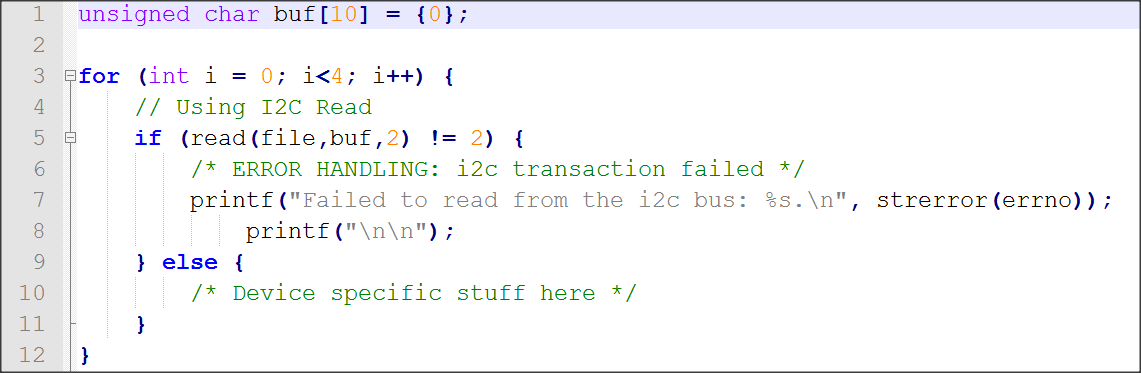
Mở tệp tin hệ thống của giao diện I2C:



Khởi tạo mô đun I2C đề giao tiếp với thiết bị ngoại vi:



Đọc dữ liệu từ thiết bị ngoại vi thông qua giao tiếp I2C:



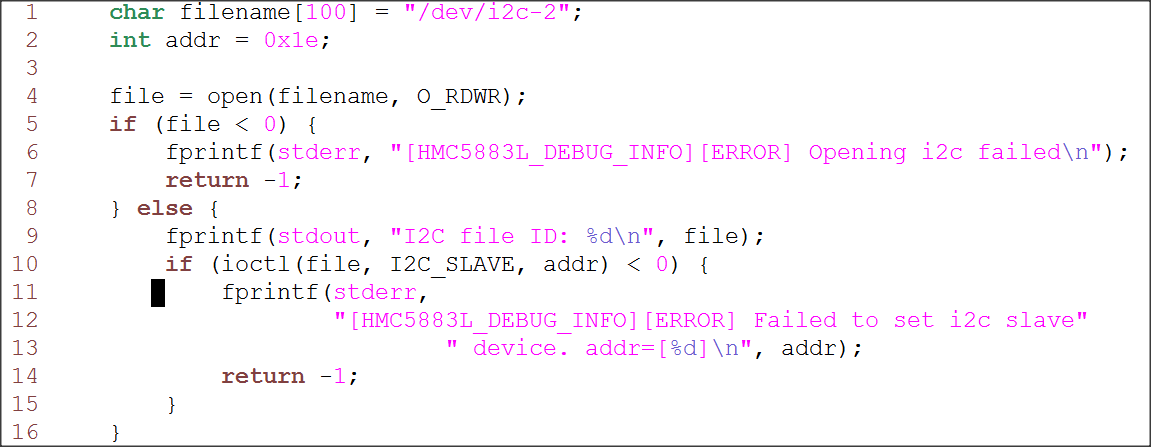
* + - 1. **Tìm hiểu và giao tiếp với các cảm biến**

1. **Cảm biến la bàn số HMC5883L**

Mô đun cảm biến la bàn số trong hệ thống này sử dụng chip cảm biến HMC5883L. Cảm biến HMC5883L giao tiếp với vi điều khiển thông qua giao tiếp I2C với chức năng “slave”, địa chỉ slave là 0x1E. Tốc độ truyền dữ liệu ở chế độ chuẩn là 100kbps hoặc 400kbps được định nghĩa trong tài liệu kỹ thuật của chuẩn I2C. Cấu hình bus I2C gồm 8-bit địa chỉ/dữ liệu và 1-bit ACK. Vì HMC5883L giao tiếp qua I2C nên nó sử dụng 2 dây đề giao tiếp với vi điều khiển: 1 cho dây cho xung đồng hồ điều khiển (SCL) và 1 dây cho việc truyền dữ liệu (SDA).

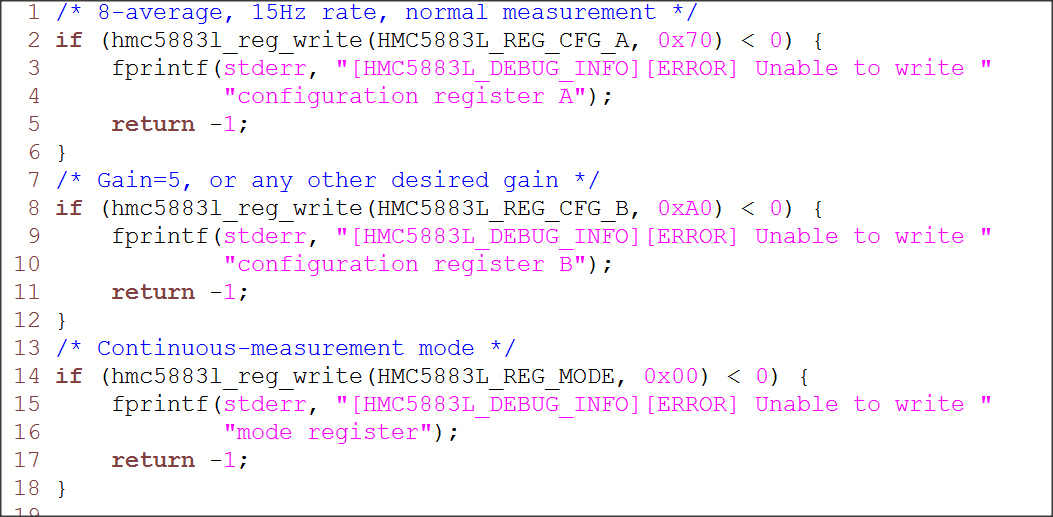
Các thao tác giao tiếp với cảm biển la bàn số bao gồm:

* Khởi tạo cảm biến:

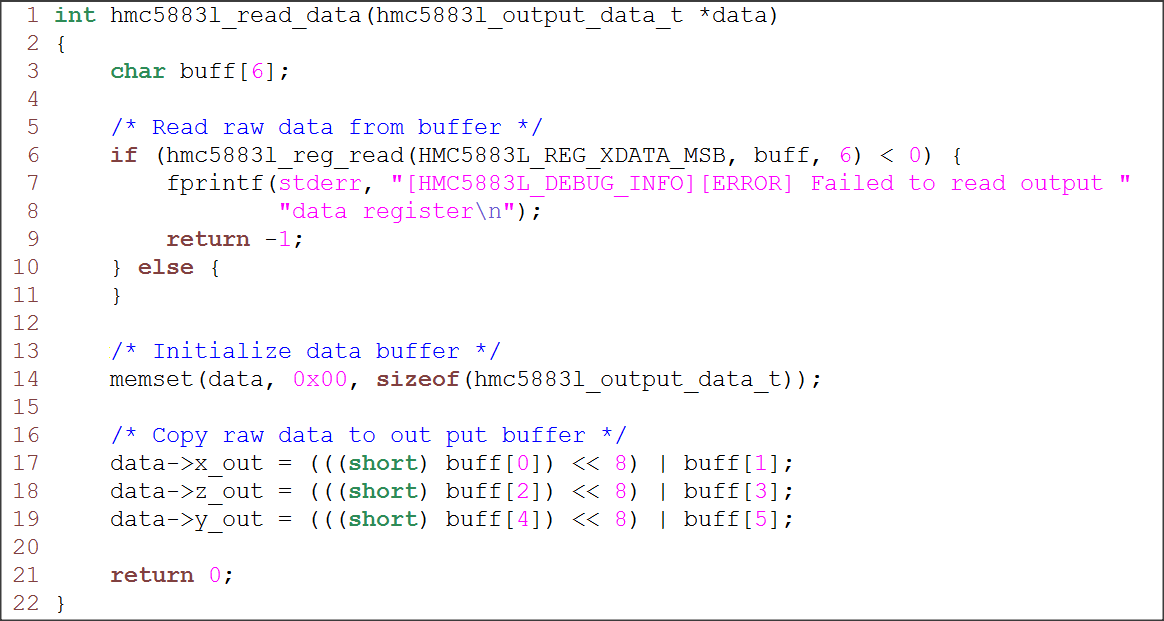


* Cấu hình cảm biến:

HMC5883L được khởi tạo đề hoạt động ở chế độ đo liên tục, tốc độ lấy mẫu là 15Hz, số lần lấy mẫu là 8.



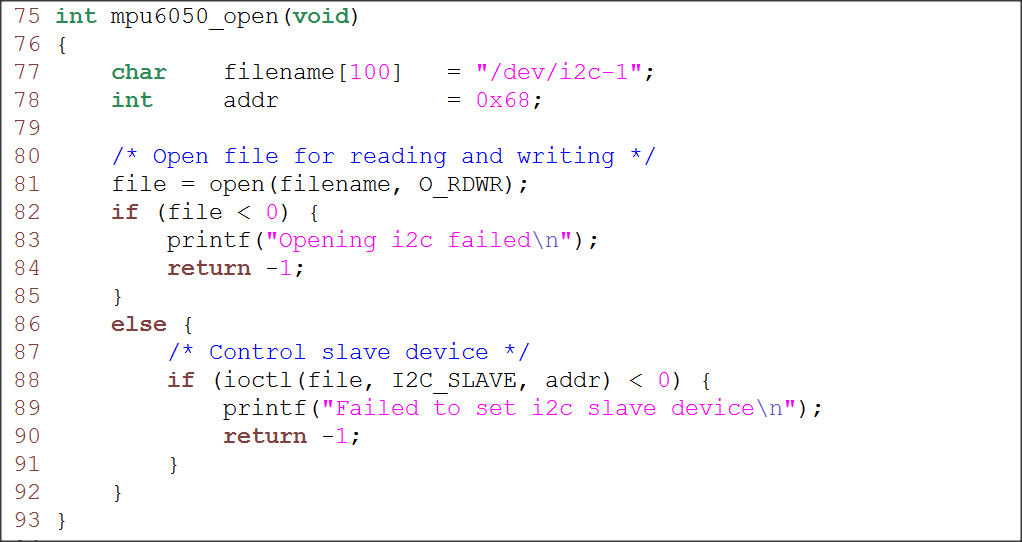
* Đọc giá trị từ cảm biến:



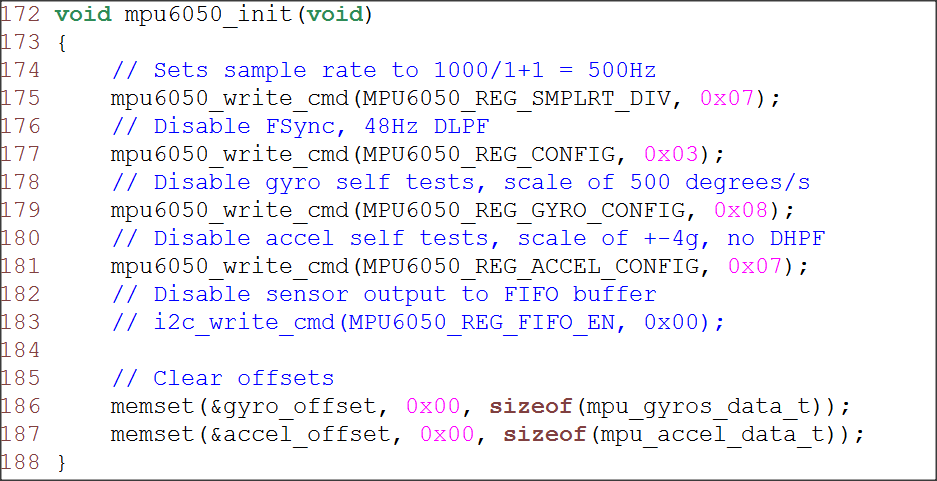
1. **Cảm biến IMU MPU6050.**

Cảm biến MPU6050 sử dụng I2C đề giao tiếp với bộ vi điều khiển. Trong hệ thống này MPU6050 hoạt động ở chế độ slave với địa chỉ là 0x68. Các thao tác giao tiếp với cảm biến bao gồm:

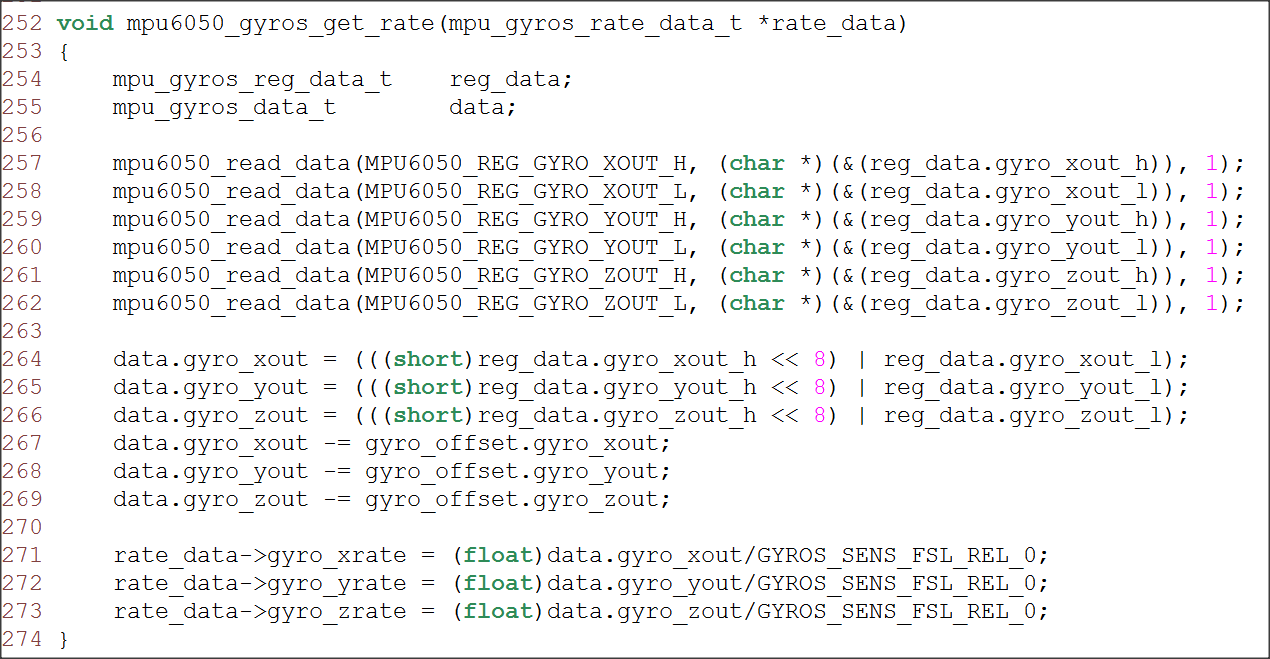
Khởi tạo cảm biến:



Cấu hình:



Đọc các giá trị vận tốc góc từ cảm biến:



* + - 1. **Điều khiển động cơ bước**

Mỗi động cơ bước 28BYJ-48 được điều khiển thông qua mô đun ULN2003. Đầu vào của mô đun này gồm 4 chân điều khiển: IN1, IN2, IN3, IN4 được nối đến 4 chân GPIO của BeagleBone Black. Trong hệ thống này, các động cơ được điều khiển ở chế độ nửa bước với mỗi bước là 2.81250. Ta suy ra được thứ tự cấp xung trên các chân GPIO để điều khiển động cơ bước như sau:

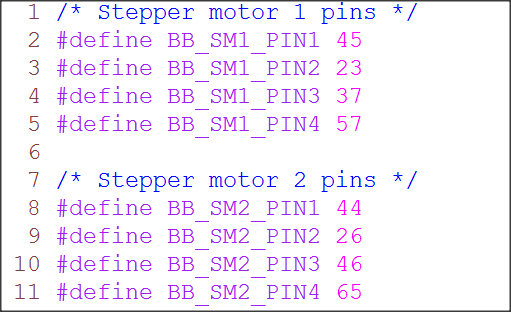
Bảng 5: Thứ tự cấp xung điều khiển

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Chân điều khiển | Các bước điều khiển | | | | | | | |
| B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 |
| IN1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| IN2 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| IN3 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| IN4 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

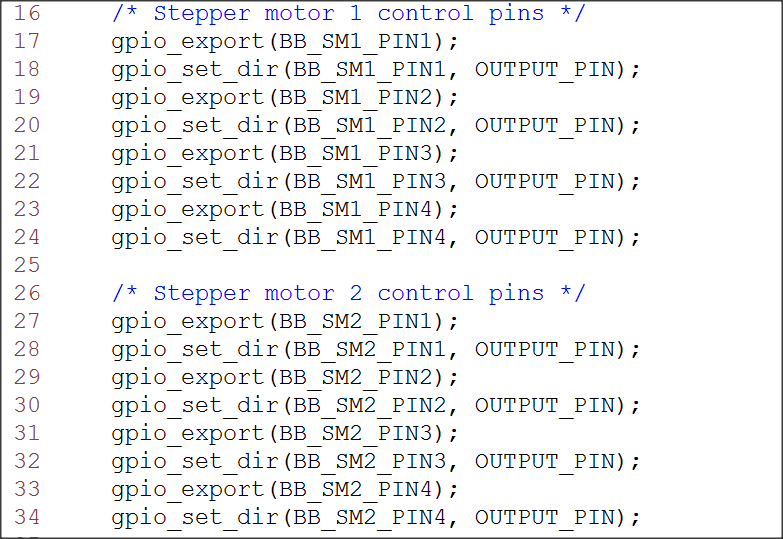
**Các thao tác điều khiển động cơ bước gồm:**

* Cấu hình các chân điều khiển:

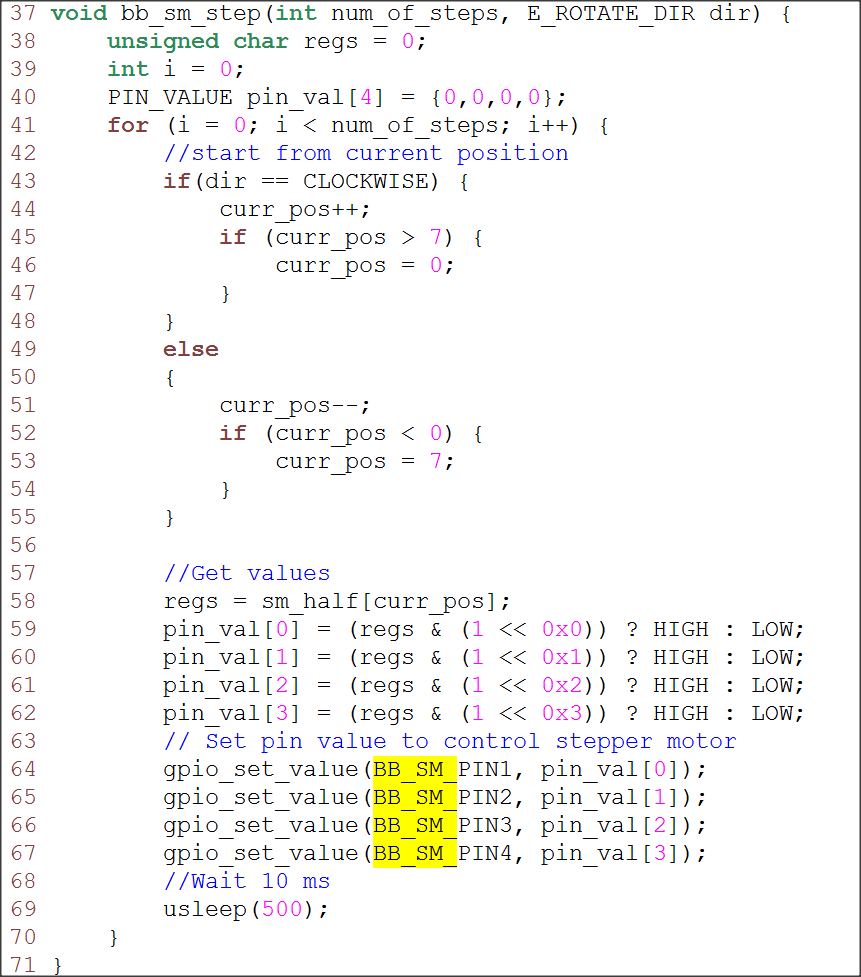
Từ hình 5, ta có danh sách các chân GPIO trên BeagleBone Black được sử dụng để điều khiển 2 động cơ bước



Vì các chân GPIO này đều được dùng dể điều khiển nên chúng được thiết lập ở chế độ ra (OUTPUT\_PIN).



* Điều khiển động cơ quay 1 số bước tính từ vị trí hiện tại theo hướng xác định:



* + 1. **Xử lý dữ liệu thu thập từ các cảm biến**

Dữ liệu thu được từ các cảm biến đều là dữ liệu ở dạng thô (raw) không thể sử dụng trực tiếp cho mục đích điều khiển. Để sử dụng dữ liệu này cho mục đích điều khiển, ta phải đưa chúng về dạng dữ liệu có ý nghĩa hơn [20, 21]. Phần này sẽ diễn đạt cách xử lý dữ liệu đọc được từ các cảm biến IMU và la bàn số và đưa chúng về dạng được mô tả trong phần nội dung.

1. **Tính góc nghiêng () và góc chúc ()**

Giả sử  là các dữ liệu về ra tốc đọc được từ cảm biến MPU6050 theo các phương X, Y, Z như trong phần III.2.2.2 b). Từ những dữ liệu này chúng ta cần tính ra các góc quay nghiêng, góc chúc theo công thức (9).

1. **Tính góc lệch D**

Với X, Y, Z là các giá trị đọc được từ cảm biến la bàn số, cùng với các góc nghiêng (), góc chúc () được tính ở phần a). Ta tính được các giá trị Xh, Yh (là các thành phần của X và Y trên mặt phẳng song song với mặt phẳng tiếp tuyến) theo công thức . Đưa các giá trị Xh, Yh vào công thức ta thu được góc lệch D tương ứng.

* + 1. **Nâng cao độ chính xác của dữ liệu đọc được từ cảm biến**

1. **Sử dụng các biện pháp căn chỉnh**

Để tăng độ tin cậy của đầu ra từ cảm biến gia tốc, giá trị đầu ra của cảm biến cần thực hiện căn chỉnh bằng cách đo độ lệch của cảm biến khi cảm biến được đặt trên mặt phẳng nằm ngang. Khi đó giá trị thật của các thành phần đầu ra từ cảm biến bằng hiệu số giữa giá trị đọc từ cảm biến trừ đi giá trị lệch đã đo được. Các bước xác định độ lệch bao gồm:

* Đặt cảm biến lên một mặt phẳng nằm ngang (ở độ chính xác nhất có thể).
* Đo các giá trị đầu ra của cảm biến một số lần nhất định.
* Lấy trung bình các giá trị đo được.

Khi đó độc lệch chính là các giá trị trung bình của đầu ra. Để chính xác hơn, ta thực hiện phép đo này một số lần và tính giá trị trung bình của chúng (trong hệ thống này, số lần thực hiện phép đo là 1000 lần).

Đối với cảm biến la bàn số, để tăng độ chính xác của đầu ra, ta cộng thêm hoặc trừ đi một giá trị góc độ lệch thích hợp tùy thuộc vào vị trí địa lý của hệ thống. Ví dụ: tại Hà Nội góc lệch này có giá trị -1° 24', tại Cẩm Phả, Quảng Ninh là -1° 35'.

1. **Sử dụng bộ lọc để nâng cao tính tin cậy.**

Cảm biến IMU (ở đây là MPU6060) có 6-bậc tự do bao gồm 3 thành phần cảm biến gia tốc và 3 thành phần cảm biến con quay hồi chuyển. Về mặt lý thuyết, IMU có thể đo chính xác và vị trí của vật thể mà nó gắn vào [22, 23], điều này có nghĩa là nếu đo được chúng, chúng ta có thể sử dụng trực tiếp các giá trị đó cho mục đích điều khiển. Tuy nhiên, về mặt thực tế những giá trị đo được từ cảm biến không phải lúc nào cũng đủ chính xác. Do đó, những dữ liệu này cần được xử lý trước khi chúng được sử dụng trong hệ thống.

Chúng ta sử dụng cảm biến gia tốc và cảm biến con quay hồi chuyển cho cùng một mục đích: xác định vị trí góc của các đối tượng. Cảm biến con quay hồi chuyển thực hiện được điều này bằng cách tổng hợp các vận tốc góc theo thời gian. Để xác định được vị trí góc bằng cảm biến gia tốc, chúng ta xác định giá của vector trọng lực được đo bởi cảm biến này. Khi đó các giá trị về góc có thể dễ dàng tính toán dựa trên công thức và . Tuy nhiên, trong cả 2 trường hợp đều xảy ra một vấn đề lớn là dữ liệu từ các cảm biến đều có độ sai lệch nhất định nếu không được xử lý bằng các bộ lọc số.

* **Vấn đề của cảm biến gia tốc**

Cảm biến gia tốc đo tất cả các lực tác động lên đối tượng. Ngoài vector trọng lực còn có nhiều lực khác tác động lên đối tượng [24]. Mỗi lực đó dù nhỏ cũng tác động lên đối tượng gây ra những sai lệch trong phép đo. Các thành phần dữ liệu đo được từ cảm biến gia tốc ổn định trong một khoảng thời gian dài. Do đó chúng ta có thể xử dụng bộ lọc thông thấp cho các thành phần dữ liệu này.

* **Vấn đề của cảm biến con quay hồi chuyển**

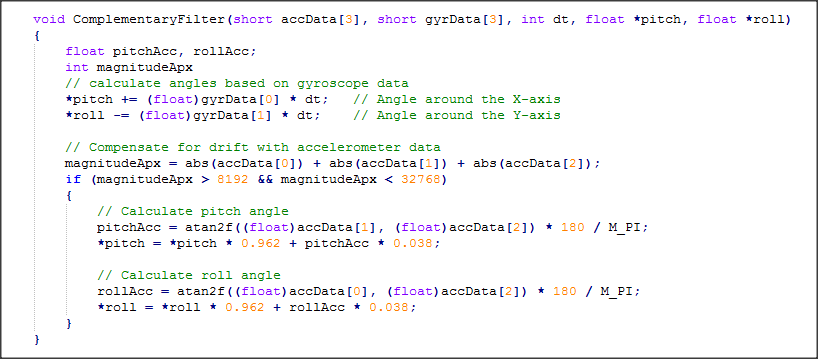
Các phép đo dữ liệu từ còn quay hồi chuyển không chịu tác dụng từ các lực bên ngoài. Tuy nhiên, nó thực hiện phép đo bằng cách tích hợp các vận tốc góc theo thời gian. Do đó phép đo có thể bị trôi và cảm biến con quay hồi chuyển không trở về vị trí 0 khi hệ thống trở lại vị trí ban đầu của nó. Dữ liệu từ cảm biến con quay hồi chuyển chỉ đáng tin cậy ở ngắn hạn, nó sẽ bị trôi sau một khoảng thời gian đủ dài.

* **Bộ lọc bù**

Bộ lọc bù kết hợp ưu điểm của 2 loại cảm biến trên để đưa ra dữ liệu chính xác về vị trí. Về mặt ngắn hạn, chúng sử dụng dữ liệu từ cảm biến con quay hồi chuyển vì nó rất chính xác và không chịu ảnh hưởng bởi các lực bên ngoài. Về lâu dài, các dữ liệu từ cảm biến gia tốc được sử dụng vì chúng không bị trôi. Phương trình của bộ lọc được mô tả ở công thức :



Sau đây là một ví dụ về bộ lọc bù xử lý dữ liệu từ MPU6050 IMU được viết bằng ngôn ngữ C:



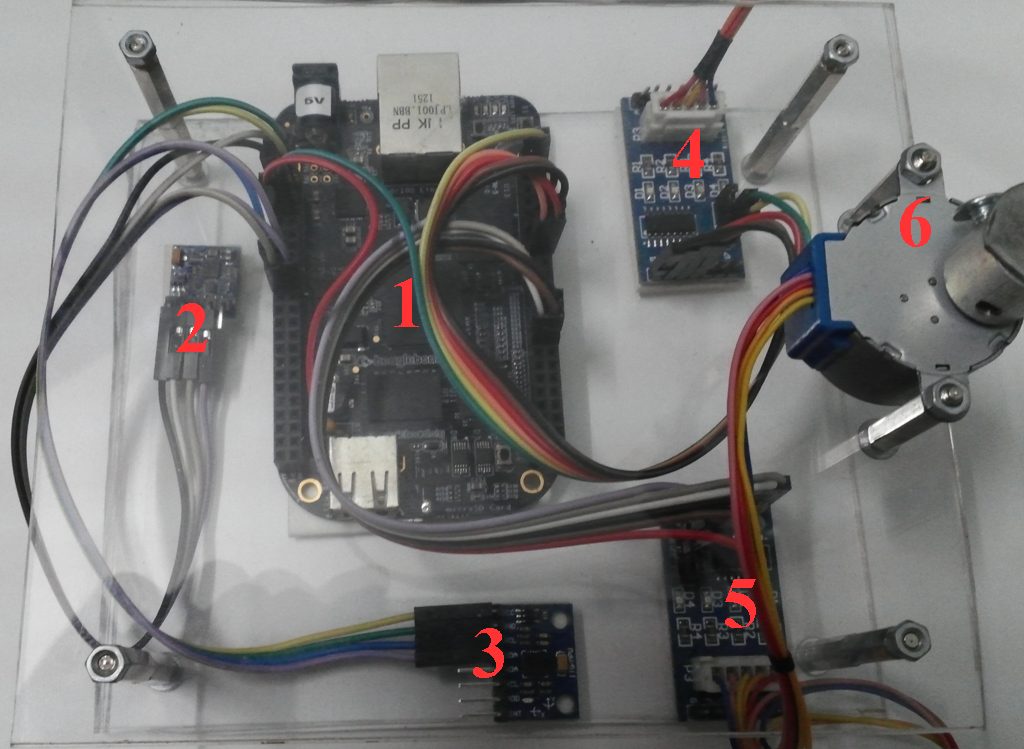
Hàm "*ComplementaryFilter*" phải được sử dụng trong một vòng lặp vô hạn. Mỗi lần lặp giá trị của các góc chúc (θ) và nghiêng (ϕ) phải được cập nhật từ dữ liệu của con quay hồi chuyển. Các bộ lọc cũng thực hiện việc kiểm tra độ lớn của vector được đo bởi IMU. Nếu các giá trị này quá nhỏ hoặc quá lớn, đồng nghĩa với sự xáo trộn trong phép đo. Khi đó chúng ta bỏ qua những giá trị này. Các góc chúc (θ) và nghiêng (ϕ) được cập nhật bằng cách lấy 98% của giá trị hiện tại cộng thêm 2% của góc tính theo dữ liệu từ cảm biến gia tốc. Điều này sẽ đảm bảo rằng các phép đo sẽ không bị trôi trong 1 khoảng thời gian dài, đồng thời đạt được độ chính xác trong ngắn hạn.

# KẾT QUẢ

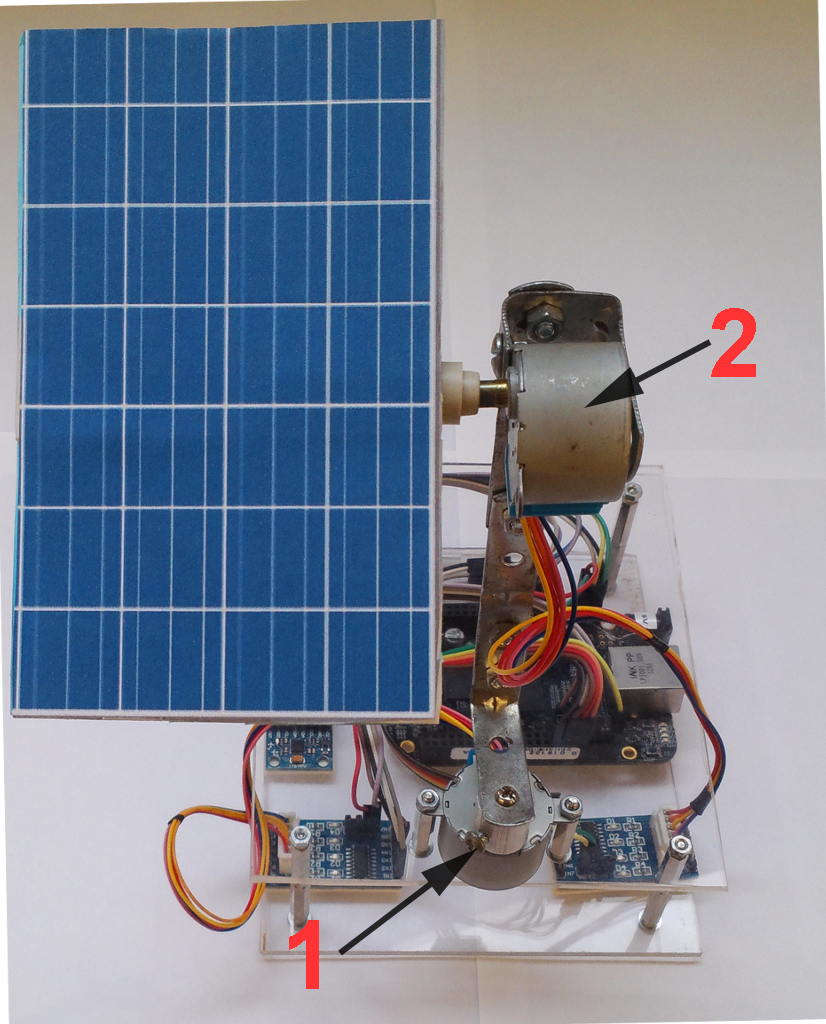
* 1. **Mô hình phần cứng thực tế**

Mô hình thực tế của hệ thống được đưa ra trong Hình 21. Trong đó:

* 1 – Kit BeagleBone Black Rev C.
* 2 – Mô đun cảm biến la bàn số GY-271.
* 3 – Mô đun cảm biến IMU GY-521.
* 4, 5 – Mô đun điều khiển động cơ bước ULN2003.
* 6- Động cơ bước 28BYJ-48.



Hình 22: Các thành phần của hệ thống thực tế



Hình 23: Mô hình hệ thống bám mặt trời thực tế

Từ Hình 22 ta thấy hệ thống thử nghiệm có 2 động cơ bước, chúng được dùng để quay tấm năng lượng mặt trời đến hướng chiếu sáng của mặt trời. Động cơ số 1 được dùng đề điều khiển tấm năng lượng mặt trời quay theo phương ngang, góc quay của động cơ này tùy thuộc vào độ lệch giữa góc đo được từ cảm biến la bàn số so với góc phương vị của mặt trời. Động cơ số 2 được dùng để quay tấm năng lượng theo phương thằng đứng, góc quay này phụ thuộc vào góc nâng của mặt trời, góc nâng thay đổi với tốc độ 150/h và đạt được góc 900 vào lúc 12h trưa.

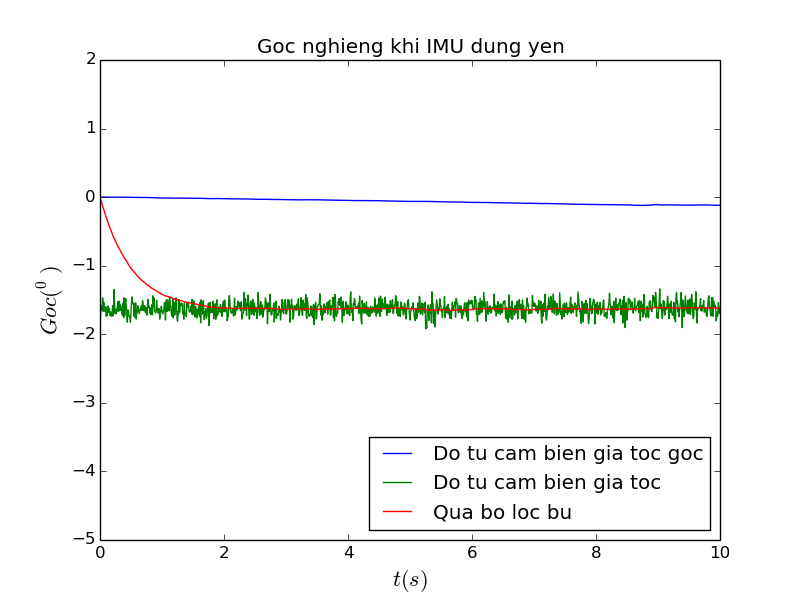
* 1. **Nhận xét các kết quả thu được**

Phần này mô tả kết quả của việc đọc dữ liệu từ các cảm biến và xử lý những dữ liệu này để đưa ra các phương án điều khiển cho hệ thống.

1. **Đọc và xử lý dữ liệu từ IMU**

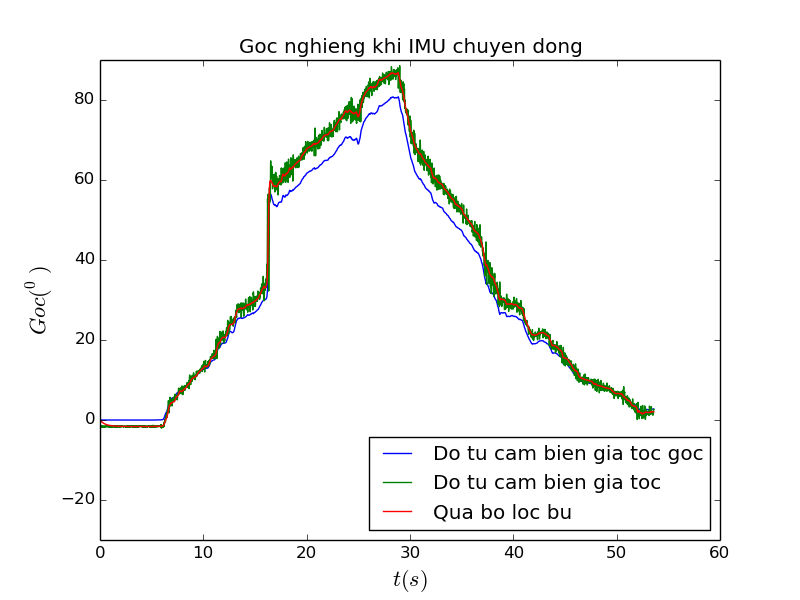
Hình 23 và Hình 24 so sánh các góc nghiêng đo được theo 3 cách khác nhau:

* Đường màu xanh nước biển thể hiện góc nghiêng được tính toán từ thành phần con quay hồi chuyển của IMU.
* Đường màu xanh lá cây thể hiện góc nghiêng được tính theo thành phần gia tốc của IMU.
* Đường màu đỏ thể hiện góc nghiêng được tính theo cả 2 thành phần trên và được đưa qua bộ lọc bù.



Hình 24: Đồ thị so sánh các góc nghiêng khi IMU đứng yên

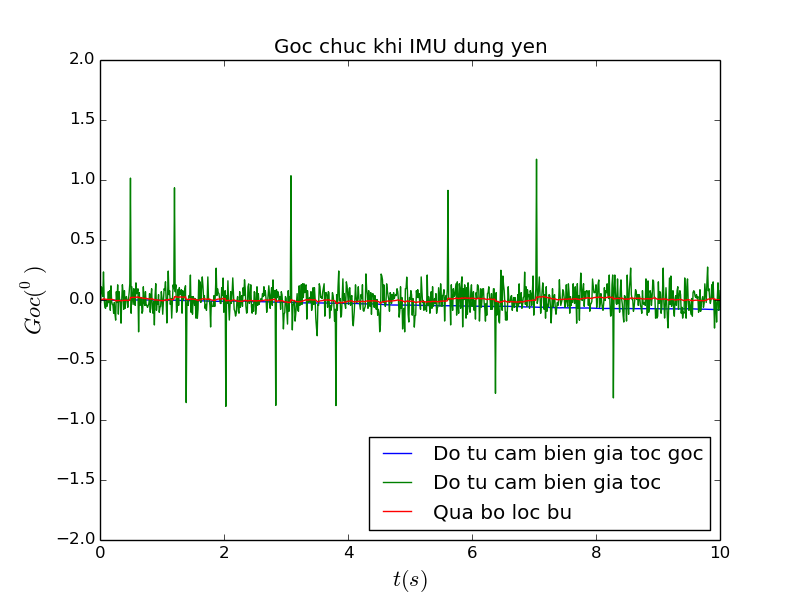
Từ Hình 24 ta thấy rằng dữ liệu đo được khi IMU đứng yên là cố định theo thời gian. Góc đo được theo thành phần cảm biến gia tốc có sự chênh lệch so với gốc 0, điều này dẫn đến sự chênh lệch của góc đo được từ bộ lọc bù. Có nhiều nguyên nhân gây ra sự sai lệch này: sai số của cảm biến, thao tác căn chỉnh chưa tốt...Tuy nhiên, giá trị chênh lệch này là rất nhỏ (<10), do đó đối với hệ thống không đòi hỏi độ chính xác quá cao thì giá trị này là chấp nhận được.

****

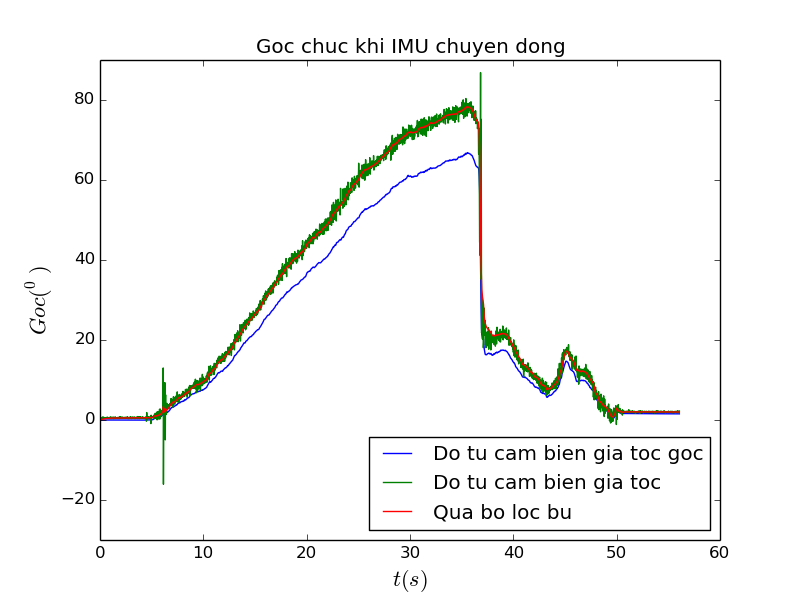
Hình 25: Đồ thị so sánh các góc nghiêng khi IMU chuyển động

Để thử nghiệm độ nhạy của cảm biến IMU và tính chính xác khi thực hiện bộ lọc bù, ta thực hiện việc di chuyển (xoay) cảm biến IMU một cách thủ công và đo đạc các kết quả từ đầu ra của cảm biến và sau khi qua bộ lọc bù theo thời gian. Hình 25 mô tả sự thay đổi góc nghiêng theo thời gian. Trong khoảng 6 giây đầu, cảm biến được giữ đứng yên theo phương nằm ngang, từ giây thứ 7 đến giây thứ 28, cảm biến được dịch chuyển sao cho góc nghiêng tăng dần và từ giây thứ 29 đến giây thứ 54, cảm biến được xoay sao cho góc nghiêng giảm dần. Từ hình 25 ta thấy rằng: góc nghiêng đo được từ đầu ra của con quay hồi chuyển (đường màu xanh nước biển) xuất hiện hiện tượng trôi điểm không sau một khoảng thời gian đủ dài. Góc nghiêng đo được từ đầu ra của cảm biến gia tốc (đường màu xanh lá cây) là ổn định trong 1 khoảng thời gian dài, tuy nhiên trong khoảng thời gian đủ ngắn, các thành phần này có thể thay đổi đột ngột, điều này không thích hợp cho việc điều khiển. Góc nghiêng đo được bằng cách kết hợp 2 thành phần trên và bộ lọc bù (đường màu đỏ) là ổn định và đáng tin cậy, khắc phụ được hạn chế của 2 thành phần cảm biến trên.

Các nhận xét trên là tương tự đối với góc chúc (Hình 26, Hình 27).

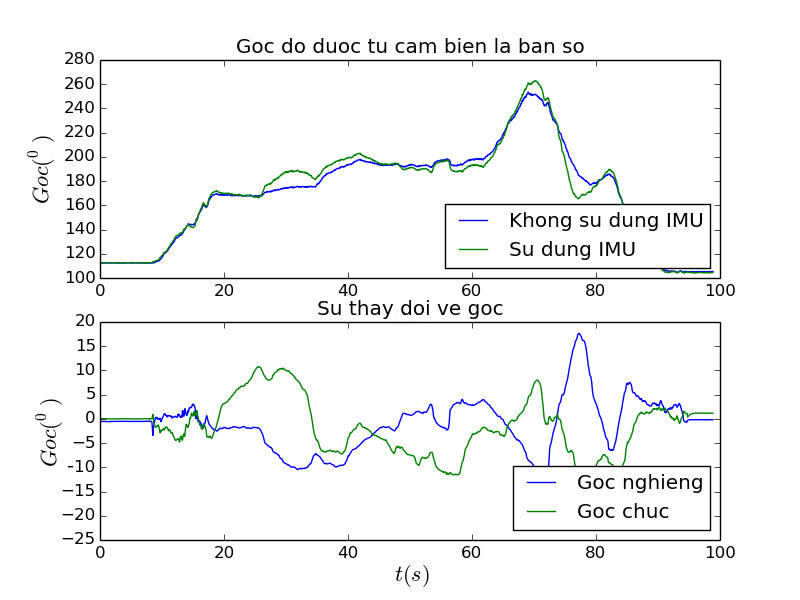


Hình 26: Đồ thị so sánh các góc chúc đo được khi IMU đứng yên

****

Hình 27: Đồ thị so sánh các góc chúc khi IMU chuyển động

1. **Đọc và xử lý dữ liệu từ la bàn số**



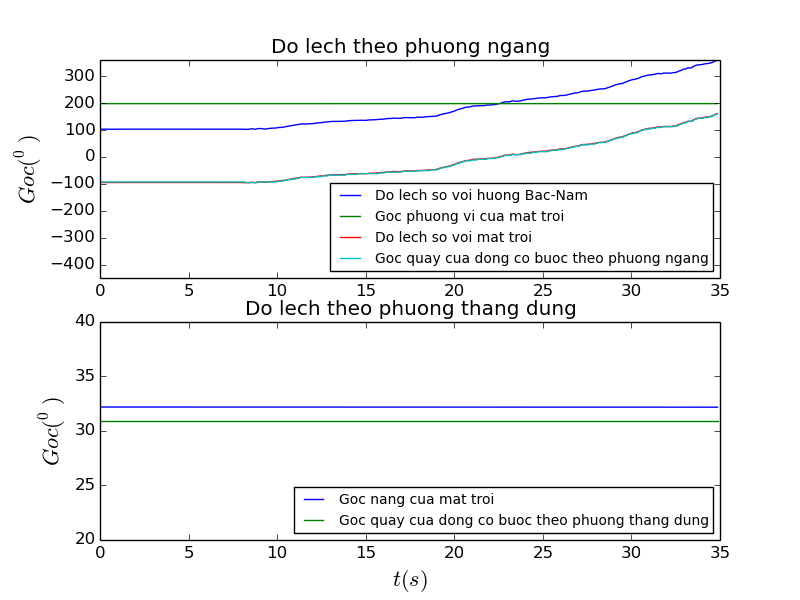
Hình 28: Đọc dữ liệu từ cảm biến la bàn số

Hình 28 mô tả sự thay đổi của góc đo được từ cảm biến la bàn số khi các góc nghiêng và góc chúc thay đổi. Dữ liệu thu được bằng cách đặt 2 cảm biến cố định trên mặt phẳng nằm ngang của vật thể và song song với phương ngang. Sau đó dịch chuyển vật thể sao cho góc nghiêng và góc chúc thay đổi và đo giá trị đầu ra của cảm biến la bàn số. Đường màu xanh nước biển thể hiện góc đo được từ cảm biến la bàn số trong trường hợp không sử dụng IMU, đường màu xanh lá cây thể hiện góc đo được ở trên kết hợp với IMU để bù trừ chênh lệch. Từ đồ thị trên ta thấy rằng, khi các cảm biến được đặt trên mặt phẳng ngang (góc nghiêng và góc chúc bằng 0), các giá trị trong 2 trường hợp là như nhau. Tuy nhiên nếu các góc nghiêng và góc chúc khác không, các giá trị này sẽ khác nhau. Trong trường hợp góc nghiêng và góc chúc đủ lớn, sự chênh lệch là rất đáng kể.

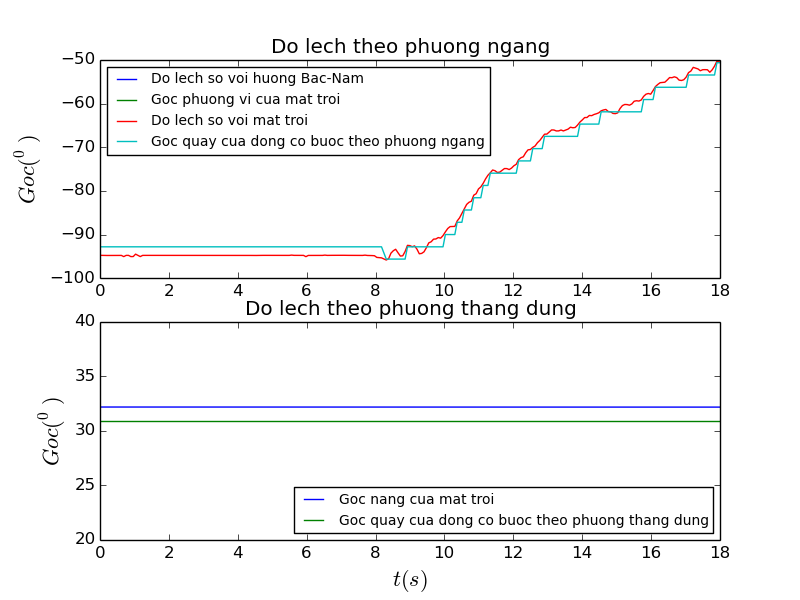
1. **Tính các góc lệch và góc quay của động cơ**

Góc lệch của hệ thống so với mặt trời theo phương ngang được thể hiện bằng đường màu đỏ. Giá trị này được tính bằng hiệu số giữa độ lệch của hệ thống so với hướng bắc-nam (đường màu xanh nước biển) và góc phương vị của mặt trời. Giá trị này có thể âm hoặc dương tùy thuộc vào 2 giá trị tính toán được ở trên. Góc lệch này được sử dụng để điều khiển động cơ số 1 như trong Hình 23.

Góc lệch theo phương thẳng đứng phụ thuộc vào góc nâng của mặt trời. Góc này được thể thiện bằng đường màu xanh nước biển ở phần dưới của đồ thị. Do góc nâng của mặt trời thay đổi với tốc độ (150/h). Nên trong 1 khoảng thời gian ngắn giá trị này là cố định.



Hình 29: Độ lệch về góc của hệ thống so với mặt trời và góc quay của động cơ.



Hình 30: Góc quay của động cơ (phóng to từ hình 29).

Từ hình 30 ta thấy có sự chênh lệch giữa góc lệch của hệ thống so với mặt trời và góc quay thực tế của động cơ. Nguyên nhân của sự chênh lệch này là do động cơ bước được sử dụng trong thệ thống là loại động cơ bước với mỗi bước là 5.6250/bước. Động cơ này được điều khiển ở chế độ nửa bước, do đó góc quay tối thiểu cho mỗi lần quay là 2.81250. Khi đó, góc quay mà động cơ thực hiện là một số nguyên lần của nửa bước (n \* 2.81250). Khi góc lệch của hệ thống so với mặt trời không phải là một số nguyên lần của nửa bước, sự chênh lệch trên sẽ xảy ra. Tuy nhiên, độ chênh lệch này là không đáng kể đối với hệ thống không yêu cầu độ chính xác rất cao.

* 1. **Kết quả chạy thử hệ thống bám mặt trời**

Xem liên kết đến video chạy thử của hệ thống tại phần phụ lục 1.

# KẾT LUẬN

Về mặt nội dung, luận văn này đã nghiên cứu và đưa ra được cơ sở lý thuyết và các thuật toán xử lý cho “*Hệ thống điều khiển bám cho panel lắp đặt pin năng lượng mặt trời*”.

Luận văn này cũng đã xây dựng thành công hệ thống thử nghiệm. Về phần cứng: thiết kế và tích hợp thành công các mô đun phần cứng thành một hệ thống duy nhất với phần xử lý chính là kit BeagleBone Black Rev.C, bộ xử lý này thu thập dữ liệu từ các mô đun cảm biến, xử lý các dữ liệu để và dùng chúng để điều khiển các động cơ bước. Thành phần cảm biến của hệ thống bao gồm 2 mô đun cảm biến: mô đun cảm biến la bàn số GY-271 và mô đun cảm biến IMU GY-521. Mô đun cảm biến la bàn số GY-271 được sử dụng để xác định hướng của tàu. Mô đun cảm biến IMU GY-521 được sử dụng để điều chỉnh độ lệch hướng của tàu khi la bàn nằm nghiêng so với mặt phẳng nằm ngang của trái đất. Bộ lọc bù được sử dụng để kết hợp ưu điểm của 2 thành phần của khối IMU (gia tốc góc và con quay hồi chuyển) để đưa ra dữ liệu chính xác về góc. Thành phần truyền động gồm 2 mô đun điều khiển động cơ bước ULN2003 và 2 động cơ bước 28BYJ-48 được điều khiển ở chế độ nửa bước giúp cho tấm năng lượng có thể xoay tối thiểu 2.81250 trong 1 lần xoay theo phương ngang hoặc phương thẳng đứng. Phần mềm của hệ thống được xây dựng trên nền hệ điều hành Linux với cơ chế xử lý đa luồng, đảm bảo cho hệ thống hoạt động đáp ứng thời gian thực.

Ưu điểm của hệ thống: hệ thống này là loại có 2 trục (trục dọc và trục ngang). Do đó, nó có thể bám theo hướng mặt trời một cách chính xác tại bất kỳ nơi nào trên trái đất. Hệ thống này theo dõi mặt trời theo cả 2 hướng Đông-Tây, Bắc-Nam, bởi vậy cho năng suất cao hơn so với các hệ thống thụ động và chủ động dùng cảm biến ánh sáng [19]. Hệ thống có thể hoạt động được trên các vật thể di động như tàu biển, ô tô…Hệ thống có thể hoạt động chính xác vào những ngày trời âm u.

Nhược điểm của hệ thống: giá thành cao hơn so với các hệ thống dùng cảm biến ánh sáng; các cảm biến cần phải căn chỉnh trước khi sử đụng; phụ thuộc vào tọa độ của vật thể lắp đặt hệ thống.

# HƯỚNG PHÁT TRIỂN

Hệ thống trên mới triển khai ở mức mô hình thử nghiệm nên còn hạn chế. Bởi vậy, cần có nhiều cải tiến để hệ thống có thể hoàn thiện hơn. Dưới đây là một số hướng cải tiến và phát triển hệ thống:

* Bộ lọc bù có thể xử lý khá tốt các giá trị đầu ra của các cảm biến, đặc biệt là cảm biến IMU. Tuy nhiên, để nâng cao độ tin cậy của các dữ liệu từ các cảm biến, chúng ta có thể áp dụng thuật toán của bộ lọc Kalman vào việc xử lý những dữ liệu này.
* Phân chia xử lý đa luồng tốt hơn để nâng cao việc xử lý thời gian thực của hệ thống.
* Sử dụng định vị toàn cầu để tính toán giờ mặt trời và sai số của la bàn số tại những địa điểm khác nhau mà không cần phải cài đặt thủ công khi hệ thống di chuyển từ địa điểm này sang địa điểm khác.
* Đặt hệ thống trên các tàu dịch chuyển tại các vị trí khác nhau để kiểm nghiệm mức độ chính xác của hệ thống.

**TÀI LIỆU THAM KHẢO**

**Tiếng Việt:**

1. Thanh Nga (2014), “*7 nguồn năng lượng sạch cho tương lai*”, info.net, lược dịch từ nguồn tin từ hãng tin CNBC thuộc Tập đoàn NBCUniversal News Group.
2. Quản Lý Ngành Điện (2009), “*Cơ sở cho việc sản xuất và sử dụng: Năng lượng địa nhiệt*”, Tạp chí Quản Lý Ngành Điện số 10/2009.
3. Thạch Anh (2015), “*Nguồn năng lượng của tương lai”,* Tạp chí PC World VN số 06/2015.

**Tiếng Anh:**

1. National Geographic Education (2012), “*Solar Energy*”, accessed on 14 November 2015, http://education.nationalgeographic.com/encyclopedia/-solar-energy.
2. National Geographic Environment (2007), "*Wind Power*", accessed on 14 November 2015, http://environment.nationalgeographic.com/environment/-global-warming/wind-power-profile.
3. Budhy S., Mauridhi H.P., Mochamad A. (2012), "*Artificial Intelligent based Modeling of Mobile Solar Tracker for a Large Ship*", Graduate School of Industrial Technology, Sepuluh November Technology Institute.
4. Budhy S., Mauridhi H.P., Mochamad A., Takashi H. (2013), “*Advanced Control Of On-Ship Solar Tracker Using Adaptive Wide Range ANFIS*”, International Journal of Innovative Computing, Information Control Volume 9, Number 6, June 2013, pp. 2595-2596.
5. Honeywell Inc. (1995), “*Compass Heading Using Magnetometers*”, Application Note AN203, 7/95 rev. A.
6. Freescale Semiconductor Inc. (2013), “*Tilt Sensing Using a Three-Axis Accelerometer*”, Application Note AN3461 Rev. 6, 03/2013.
7. Caruso M. (1997), "*Applications of Magnetoresistive Sensors in Navigation Systems*", SAE Technical Paper 970602.
8. Shane C. (2007), “*The Balance Filter - A Simple Solution for Integration Accelerometer and Gyroscope Measurements for a Balancing Platform*”, Rev.1: Submitted as a Chief Delphi white paper - June 25, 2007.
9. Christiana H., Stuart B., “*The Sun's Position*”, Photovoltaic (PV) education. Solar Power Lab, ASU, http://www.pveducation.org/pvcdrom/properties-of-sunlight/suns-position.
10. Julius O., Smith III (2007), “*Introduction to digital filters with audio applications*”, September 2007 Edition.
11. John G.P., Dimitris G.M. (1996), “*Digital Signal Processing Principles, Algorithms, and Applications-Third Edition*”.
12. InverseSense Inc. (2011), “*MPU-6000 and MPU-6050 Product Specification Revision 3.1*”, PS-MPU-6000A-00 Rev 3.1, 10/24/2011.
13. InverseSense Inc. (2012), “*MPU-6000 and MPU-6050 Register Map and Descriptions Revision 4.0*”, RM-MPU-6000A-00 Rev 4.0, 03/09/2012.
14. Honeywell Inc. (2013), “*HMC5883L\_3-Axis\_Digital\_Compass\_IC*” Form # 900405 Rev E, February 2013.
15. Hossein M., Alireza K., Arzhang J., Hossein M., Karen A., Ahmad S. (2009), “*A review of principle and sun-tracking methods for maximizing solar systems output*”, Renewable and Sustainable Energy Reviews 13, pp. 1800–1818.
16. Seme S., Štumberger G. (2013), “*Single or dual axis trackers, control systems and electric drive losses for photovoltaic applications*”, Renewable Energy and Power Quality journal, ISSN 2172-038 X, No.11, March 2013.
17. Ha, L. M., Tan, T. D., Long, N. T., Duc, N. D., & Thuy, N. P. (2007). *Errors determination of the MEMS IMU*. Journal of Science VNUH, July, pp. 6-12.
18. Tan, T. D., Ha, L. M., Long, N. T., Thuy, N. P., & Tue, H. H. (2007). *Performance Improvement of MEMS-Based Sensor Applying in Inertial Navigation Systems*. Research-Development and Application on Electronics, Telecommunications and Information Technology, (2), pp. 19-24.
19. Duc-Tan, Tran, Paul Fortier, and Huu-Tue Huynh. "*Design, Simulation, and Performance Analysis of an INS/GPS System using Parallel Kalman Filters* *Structure*." REV Journal on Electronics and Communications, Vol. 1, No. 2, April – June, 2011, ISSN 1859-378X, pp. 88-96.
20. Tan, T. D., Tue, H. H., Long, N. T., Thuy, N. P., & Van Chuc, N. (2006, November). *Designing Kalman filters for integration of inertial navigation system and global positioning system*. In The 10th biennial Vietnam Conference on Radio & Electronics, REV-2006. Hanoi, November 2006, pp. 226-230.
21. Tan, Tran Duc, Nguyen Tien Anh, and Gian Guoc Anh. "*Low-cost Structural Health Monitoring Scheme Using MEMS-based Accelerometers*." Intelligent Systems, Modelling and Simulation (ISMS), 2011 Second International Conference on. IEEE, 2011, pp. 217-220.

**PHỤ LỤC**

1. **Liên kết đến video quay kết quả chạy thử chương trình**

* Video 1: Thử nghiệm tổng thể hệ thống bám mặt trời.

<https://www.youtube.com/watch?v=KcHVw5t02_E>

* Video 2: Chạy thử việc điều khiển động cơ bước.

<https://www.youtube.com/watch?v=jrP_ebA8oCI>

1. **Mã nguồn chương trình chính**

#include "main.h"

#include "config.h"

/\* Mutexs \*/

pthread\_mutex\_t ins\_data\_mutex = PTHREAD\_MUTEX\_INITIALIZER;

pthread\_mutex\_t gps\_data\_mutex = PTHREAD\_MUTEX\_INITIALIZER;

pthread\_mutex\_t compass\_data\_mutex = PTHREAD\_MUTEX\_INITIALIZER;

/\* variables used to store ins datas \*/

#ifdef SYSTEM\_GPS\_DEVICE\_ENABLED

#if (SYSTEM\_GPS\_DEVICE\_ENABLED == STD\_ON)

/\* variables used to store gps data \*/

static nmeaINFO info;

static nmeaPARSER parser;

static nmeaPOS dpos;

#endif

#endif

static float roll, pitch;

static hmc5883l\_output\_data\_t g\_compass\_data;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\* internal functions \*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

static void pabort(const char \*s);

#ifdef SYSTEM\_GPS\_DEVICE\_ENABLED

#if (SYSTEM\_GPS\_DEVICE\_ENABLED == STD\_ON)

static void gps\_recv\_task(void);

#endif

#endif

static void ins\_recv\_task(void);

static void compass\_recv\_task(void);

static void data\_process\_task(void);

// static int date\_of\_year(int day, int month, int year);

static int get\_day\_cnt(void);

static float get\_solar\_direction(void);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// function implementation

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int main(int argc, char \*argv[]) {

#if 0

pthread\_t gps\_thread, ins\_thread, compass\_thread, data\_process\_thread;

#else

pthread\_t ins\_thread, compass\_thread, data\_process\_thread;

#endif

// pthread\_t gps\_thread, ins\_thread;

char msg\_table[4][1024] = { "gps thread is initializing...",

"ins thread is initializing...",

"compass thread is initializing...",

"data processing thread is initializing...", };

/\* initialize new thread for reading and parsing data from gps device \*/

#ifdef SYSTEM\_GPS\_DEVICE\_ENABLED

#if (SYSTEM\_GPS\_DEVICE\_ENABLED == STD\_ON)

if(0 > pthread\_create(&gps\_thread, NULL, (void \*)gps\_recv\_task,

(void\*)msg\_table[0])) {

pabort("Unable to create new thread\n");

}

#endif

#endif

#ifdef SYSTEM\_IMU\_DEVICE\_ENABLED

#if (SYSTEM\_IMU\_DEVICE\_ENABLED == STD\_ON)

/\* initialize new thread for reading and parsing data from ins device \*/

if(0 > pthread\_create(&ins\_thread, NULL, (void \*)ins\_recv\_task,

(void\*)msg\_table[1])) {

pabort("Unable to create new thread\n");

}

#endif

#endif

#ifdef SYSTEM\_COMPASS\_DEVICE\_ENABLED

#if (SYSTEM\_COMPASS\_DEVICE\_ENABLED == STD\_ON)

if (pthread\_create(&compass\_thread, NULL, (void \*) compass\_recv\_task,

(void\*) msg\_table[3]) < 0) {

pabort("Unable to create new thread\n");

}

#endif

#endif

#ifdef SYSTEM\_DATA\_PROCESS\_ENABLED

#if (SYSTEM\_DATA\_PROCESS\_ENABLED == STD\_ON)

/\* initialize new thread for processing data from sensors \*/

if(0 > pthread\_create(&data\_process\_thread, NULL, (void \*)data\_process\_task,

(void \*)msg\_table[4])) {

pabort("Unable to create new thread\n");

}

#endif

#endif

/\* Wait until all thread exit \*/

// pthread\_join(gps\_thread, NULL);

#ifdef SYSTEM\_IMU\_DEVICE\_ENABLED

#if (SYSTEM\_IMU\_DEVICE\_ENABLED == STD\_ON)

pthread\_join(ins\_thread, NULL);

#endif

#endif

#ifdef SYSTEM\_COMPASS\_DEVICE\_ENABLED

#if (SYSTEM\_COMPASS\_DEVICE\_ENABLED == STD\_ON)

pthread\_join(compass\_thread, NULL);

#endif

#endif

pthread\_join(data\_process\_thread, NULL);

return 0;

}

#ifdef SYSTEM\_GPS\_DEVICE\_ENABLED

#if (SYSTEM\_GPS\_DEVICE\_ENABLED == STD\_ON)

/\*

\* @brieft This task get gps data from gps device

\* @detail Get location information (longtitude, latitude) from gps device, convert them to radian

\* values and forward them to data processing task.

\*/

static void gps\_recv\_task(void) {

int gps\_fd = -1;

// int file\_fd = -1;

int mSize = 0;

// unsigned int size,i;

unsigned int size;

char rMessage[1024];

char buff[2];

/\* initialize gps device \*/

if (0 > (gps\_fd = gps\_init())) {

perror("GPS device cannot be initialized.");

exit(1);

}

// open output file for writing

// file\_fd = open("gps\_output.txt", O\_RDWR | O\_CREAT | O\_APPEND );

nmea\_zero\_INFO(&info);

nmea\_parser\_init(&parser);

while (1) {

#if 0

if(1 != (size = read(gps\_fd,buff,1))) {

perror("Error was occurred while reading gps device.\n");

// break;

}

#else

if (0 >= (size = read(gps\_fd, buff, 1024))) {

pabort("Error was occurred while reading gps device.\n");

exit(1);

}

#endif

#if 0

/\* Reading output signal from gps device \*/

if('$' == buff[0]) {

i = 0;

rMessage[i] = buff[0];

do {

if(1 != (size = read(gps\_fd, buff, 1))) {

perror("Error while reading character from gps device.\n");

}

if('\0' != buff[0]) {

i++;

rMessage[i] = buff[0];

}

}while(buff[0] != '\n');

/\* Add null terminator to rMessage \*/

rMessage[i+1] = '\n';

}

mSize = i+1;

#endif

/\* Lock dpos while reading data from gps device \*/

pthread\_mutex\_lock(&gps\_data\_mutex);

nmea\_parse(&parser, (const char \*) rMessage, mSize, &info);

nmea\_info2pos(&info, &dpos);

#ifdef NAVIS\_DEBUG\_ENABLE

#if (NAVIS\_DEBUG\_ENABLE == STD\_ON)

#if 0

// Radian

printf("Lat: %f, Lon: %f, Sig: %d, Fix: %d\n", dpos.lat, dpos.lon,

info.sig, info.fix);

#else

// Degree

printf("Lat: %f, Lon: %f, Sig: %d, Fix: %d\n",

nmea\_radian2degree(dpos.lat),

nmea\_radian2degree(dpos.lon),

info.sig,

info.fix);

#endif

#endif

#endif

/\* unlock mutex \*/

pthread\_mutex\_unlock(&gps\_data\_mutex);

// mSize = 0;

}

nmea\_parser\_destroy(&parser);

}

#endif

#endif

void ins\_recv\_task(void) {

#ifdef SYSTEM\_DEBUG\_INFO\_ENABLE

#if (SYSTEM\_DEBUG\_INFO\_ENABLE == STD\_ON)

char mpu\_id[5] = { 0, 0, 0, 0, 0 };

#endif

#endif

float degX, degY, degZ;

mpu\_accel\_rate\_data\_t acc\_data;

mpu\_gyros\_rate\_data\_t gyro\_data;

/\* Open and initialize IMU device \*/

if (0 > mpu6050\_open()) {

pabort("INS device cannot be initialized.");

exit(1);

} else {

mpu6050\_wake\_up();

printf("Open successfully!\n");

/\* Initializing and configuring mpu6050 \*/

mpu6050\_init();

/\* Calibrate gyroscope and accelerometer \*/

// mpu6050\_gyros\_calib();

// mpu6050\_accel\_calib();

}

while (1) {

#ifdef NVS\_IMU\_INFO\_ENABLE

#if (NVS\_IMU\_INFO\_ENABLE == STD\_ON)

mpu6050\_get\_whoami(mpu\_id);

printf("Hello, I'm a MPU6050 device. My ID is %x\n", mpu\_id[0]);

#endif

#endif

mpu6050\_gyros\_get\_rate(&gyro\_data);

mpu6050\_accel\_get\_value(&acc\_data);

#ifdef NAVIS\_DEBUG\_ENABLE

#if (NAVIS\_DEBUG\_ENABLE == STD\_ON)

printf("Gyroscope: %f, %f, %f\n",gyro\_data.gyro\_xrate,

gyro\_data.gyro\_yrate, gyro\_data.gyro\_zrate);

printf("Accelerometer: %f, %f, %f\n",acc\_data.accel\_xrate,

acc\_data.accel\_yrate, acc\_data.accel\_zrate);

#endif

#endif

pthread\_mutex\_lock(&ins\_data\_mutex);

// Process INS output data - T.B.D

roll = atan(acc\_data.accel\_yrate /

sqrt(-acc\_data.accel\_xrate \* -acc\_data.accel\_xrate +

-acc\_data.accel\_zrate \* -acc\_data.accel\_zrate)) \* RAD\_TO\_DEG;

pitch = atan2(-acc\_data.accel\_xrate, acc\_data.accel\_zrate) \* RAD\_TO\_DEG;

degX = atan(acc\_data.accel\_xrate/sqrt(acc\_data.accel\_yrate\* acc\_data.accel\_yrate +

acc\_data.accel\_zrate \* acc\_data.accel\_zrate)) / RAD\_TO\_DEG;

degY = atan(acc\_data.accel\_yrate/sqrt(acc\_data.accel\_xrate\* acc\_data.accel\_xrate +

acc\_data.accel\_zrate \* acc\_data.accel\_zrate)) / RAD\_TO\_DEG;

degZ = atan(sqrt(acc\_data.accel\_xrate\* acc\_data.accel\_xrate +

acc\_data.accel\_xrate \* acc\_data.accel\_xrate)/acc\_data.accel\_zrate) / RAD\_TO\_DEG;

// printf("degreeX: %f\n", degX);

// printf("degreeY: %f\n", degY);

printf("degreeZ: %f\n", degZ);

pthread\_mutex\_unlock(&ins\_data\_mutex);

}

}

static void compass\_recv\_task(void) {

hmc5883l\_output\_data\_t data;

double head = 0.0;

double declinationAngle = 0.0;

/\* Open hmc5883l device \*/

if (hmc5883l\_open() == -1) {

fprintf(stderr, "[COMPASS\_THREAD\_LOG][ERROR] Failed to open hmc5883l"

" device\n");

exit(1);

}

/\* init hmc5883l device (mode, rate ...) \*/

if (hmc5883l\_init() == -1) {

fprintf(stderr,

"[COMPASS\_THREAD\_LOG][ERROR] Failed to init hmc5883l device\n");

exit(1);

}

/\* Wait for internal initialization \*/

usleep(50000);

while (1) {

memset(&data, 0x00, sizeof(hmc5883l\_output\_data\_t));

/\* init hmc5883l device (mode, rate ...) \*/

if (hmc5883l\_read\_data(&data) == -1) {

fprintf(stderr, "[COMPASS\_THREAD\_LOG][ERROR] Failed to read data from compass device\n");

// exit(1);

break;

}

#ifdef SYSTEM\_DEBUG\_INFO\_ENABLE

#if (SYSTEM\_DEBUG\_INFO\_ENABLE == STD\_ON)

printf(

"[COMPASS\_THREAD\_LOG][INFO] Compass data: x=[%05d], y=[%05d], z=[%05d]\n",

data.x\_out, data.y\_out, data.z\_out);

#endif

#endif

pthread\_mutex\_lock(&compass\_data\_mutex);

//#ifdef SYSTEM\_DEBUG\_INFO\_ENABLE

//#if (SYSTEM\_DEBUG\_INFO\_ENABLE == STD\_ON)

head = atan2((double) data.y\_out, (double) data.x\_out);

/\* Calculating compass heading \*/

declinationAngle = 0.0457;

head += declinationAngle;

// Correct for when signs are reversed.

if (head < 0)

head += 2 \* PI;

// Check for wrap due to addition of declination.

if (head > 2 \* PI)

head -= 2 \* PI;

// Convert radians to degrees for readability.

head = head \* 180 / PI;

// head = atan2((double)data.y\_out, (double)data.x\_out) \* (180 / PI) + 180;

/\* print to screen \*/

fprintf(stdout, "[COMPASS\_THREAD\_LOG][INFO] Compass heading: %f\n",

head);

//#endif

//#endif

pthread\_mutex\_unlock(&compass\_data\_mutex);

usleep(67 \* 1000);

}

fprintf(stderr, "[COMPASS\_THREAD\_LOG][INFO] Compass retrieving data"

" thread was terminated!");

}

void data\_process\_task(void) {

float Y\_h, X\_h;

float panel\_dir;

float solar\_dir;

float diff\_angle;

int cnt = 0;

E\_ROTATE\_DIR dir = CLOCKWISE;

//Initialize all variables

X\_h = 0.0;

Y\_h = 0.0;

panel\_dir = 0.0;

solar\_dir = 0.0;

//Init stepper motor

bb\_sm\_init();

while (1) { // Loop forever

//

#if 0

pthread\_mutex\_lock(&compass\_data\_mutex);

pthread\_mutex\_lock(&ins\_data\_mutex);

Y\_h = g\_compass\_data.y\_out \* cos(roll)

- g\_compass\_data.z\_out \* sin(roll);

X\_h = g\_compass\_data.x\_out \* cos(pitch)

+ g\_compass\_data.y\_out \* sin(roll) \* sin(pitch)

- g\_compass\_data.z\_out \* cos(roll) \* sin(pitch);

pthread\_mutex\_unlock(&compass\_data\_mutex);

pthread\_mutex\_unlock(&ins\_data\_mutex);

//Calculate direction of solar panel

if (Y\_h > 0) {

panel\_dir = 90 - (atan(X\_h / Y\_h) \* (180 / PI));

} else if (Y\_h < 0) {

panel\_dir = 270 - (atan(X\_h / Y\_h)) \* (180 / PI);

} else { //Y\_h = 0

if (X\_h < 0) {

panel\_dir = 180;

} else {

panel\_dir = 0;

}

}

// Get solar's direction

solar\_dir = get\_solar\_direction();

// Get diff of angle between solar and solar panel

diff\_angle = panel\_dir - solar\_dir;

// control motor to rotate panel

bb\_sm\_rotate(diff\_angle);

// Update each 1 hour

sleep(3600);

#else

bb\_sm\_rotate((float)10.0, dir);

if(dir == CLOCKWISE)

{

cnt++;

printf("Rotate: %d\n", 10 \*cnt);

if(cnt == 18) {

// cnt = 0;

dir = ANTI\_CLOCKWISE;

}

}

else

{

cnt--;

printf("Rotate: %d\n", 10 \*cnt);

if(cnt == 0) {

dir = CLOCKWISE;

}

}

//usleep(5000);

// if(cnt == 500) {

// break;

// }

#endif

}

printf("Data processing thread was terminated!");

}

static void pabort(const char \*s) {

perror(s);

abort();

}

// Get number of days from 01/Jan to current date

// static int date\_of\_year(int day, int month, int year)

static int get\_day\_cnt(void) {

//--------------------------------------------------------------------------

// Month | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 |

//--------------------------------------------------------------------------

// No of days | 31 | 28 | 31 | 30 | 31 | 30 | 31 | 31 | 30 | 31 | 30 | 31 |

//--------------------------------------------------------------------------

char day\_of\_month[12] =

{ 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

char total\_days = 0;

int year = 0;

int month = 0;

int day = 0;

time\_t t = time(NULL);

struct tm \*tm = localtime(&t);

int i = 0;

// if(((day < 0) || (day > 31)) || ((month < 0) || (month > 12))

// {

// return -1;

// }

// Get current date

year = tm->tm\_year + 1990;

month = tm->tm\_mon + 1;

day = tm->tm\_mday;

// Leap year

if (year % 4) {

day\_of\_month[1] = 29;

}

for (i = 1; i < month; i++) {

total\_days += day\_of\_month[i - 1];

}

total\_days += day;

return total\_days;

}

float get\_solar\_direction(void) {

int day\_cnt = 0;

float altitude\_angle = 0.0;

day\_cnt = get\_day\_cnt();

altitude\_angle = 23.45 \* sin((((float)360.0 \* (284 + day\_cnt)) / 365));

return altitude\_angle;