**REPORT**

THIẾT KẾ VÀ THỰC HIỆN HỆ THỐNG ĐẾM SỐ BƯỚC CHÂN

**Design and Implementation of a Pedometer**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Họ và tên (Full name)** | **Mã SV (ID)** | **Đóng góp (Contribution)** |
| Thành viên 1 (Member 1) | Trần Đình Nam | 20020697 | Toàn bộ Project và Báo cáo. |
| Tên/Địa chỉ Report trên Github | <https://github.com/TranNamCHY/Embeded_Project.git> | | |

.

|  |
| --- |
| **Từ khóa (Keywords)** |
| External reference clock: nguồn clock tham chiếu bên ngoài.  Internal reference clock: nguồn clock tham chiếu bên trong.  Segment: các led đơn tạo thành led 7 đoạn.  Backplane pin: chân dùng để tạo ra dạng sóng trong LCD.  Frontplane pin: chân dùng để điều khiển các segment.  Phase: Pha tín hiệu.  Master: Thiết bị chủ.  Slave: Thiết bị tớ. |

Mục lục (Table of Contents)

[Mục lục (Table of Contents) 2](#_Toc136984069)

[1. Giới thiệu (Introduction) 3](#_Toc136984070)

[2. Yêu cầu đối với thiết kế (Requirements) 3](#_Toc136984071)

[2.1. Yêu cầu đối với thiết kế 3](#_Toc136984072)

[2.2. Đặc tả kỹ thuật (Specification) 4](#_Toc136984073)

[3. Thực hiện hệ thống (Implementation) 4](#_Toc136984074)

[3.1. Kiến trúc phần cứng (Hardware Architecture) 4](#_Toc136984075)

[3.1.1. Khối xử lý trung tâm 6](#_Toc136984076)

[3.1.2. Cảm biến đầu vào 8](#_Toc136984077)

[3.1.3. Các LED 9](#_Toc136984078)

[3.1.4. Switch 9](#_Toc136984079)

[3.1.5. LCD 10](#_Toc136984080)

[3.2. Lập trình phần mềm 12](#_Toc136984081)

[3.2.1. Thiết lập Clock cho hệ thống 12](#_Toc136984082)

[3.2.2. Khởi tạo và thiết lập LCD 13](#_Toc136984083)

[3.2.3. Khởi tạo và thiết lập các LED 16](#_Toc136984084)

[3.2.4. Khởi tạo và thiết lập ngắt cho Systick Timer 16](#_Toc136984085)

[3.2.5. Khởi tạo và thiết lập ngắt cho SW1 và SW3 18](#_Toc136984086)

[3.2.6. Thiết lập module I2C 19](#_Toc136984087)

[3.2.7. Chương trình điều khiển (Hàm main()) 22](#_Toc136984088)

[4. Kiểm chứng (Validation) 24](#_Toc136984089)

[5. Kết luận (Conclusion) 24](#_Toc136984090)

[Danh sách Hình (List of Figures) 27](#_Toc136984091)

[Danh sách Bảng (List of Tables) 28](#_Toc136984092)

[References 29](#_Toc136984093)

# Giới thiệu (Introduction)

*(Introduction to the motivation, Objectives, and main Contents of the project)*

**Mục tiêu của dự án:** Vận dụng các kiến thức, kỹ năng đã được học đểthiết kế và thực thi một bộ đếm số bước chân. Hệ thống được thiết kế để có thể thực hiện bằng các mô-đun cảm biến gia tốc và các ngoại vi có sẵn trên bo mạch FRDM-KL46z. Hệ thống đọc dữ liệu từ cảm biến, xử lý để phát hiện được bước chân của người dùng sau đó đẩy dữ liệu về số bước chân nên màn hình LCD trên bo mạch.

# Yêu cầu đối với thiết kế (Requirements)

## Yêu cầu đối với thiết kế

**Yêu cầu thiết kế:**

* *Chức năng:* Đếm số bước chân khi người sử dụng thiết bị di chuyển và hiển thị kết quả trên màn LCD.
* *Các đầu vào:*
  + Hệ thống sử dụng cảm biến gia tốc MMA8451 để đo số bước chân.
  + SW1 để chuyển đổi bộ đếm bước chân hoạt động giữa trạng thái hoạt động và trạng thái dừng. Khi bộ đếm dừng hoạt động, nếu bấm SW1 bộ đếm chuyển sang trạng thái hoạt động, và ngược lại. Khi chuyển từ trạng thái dừng sang trạng thái hoạt động, bộ đếm tiếp tục đếm từ số bước chân đang hiển thị trên LCD.
  + SW2 để xóa (reset) kết quả đếm của bộ đếm về không.
* *Các đầu ra:*
  + Hệ thống có 2 lối ra trạng thái: LED xanh nhấp nháy với tần số 1Hz khi bộ đếm hoạt động; tắt khi hệ thống dừng hoạt động. LED đỏ sáng khi hệ thống dừng hoạt động; tắt khi hệ thống hoạt động.
  + LCD: Hiện thị số bước chân đã đi của người dùng.
* Bộ đếm sử dụng một timer để xác định khoảng thời gian nhấp nháy LED xanh.

## Đặc tả kỹ thuật (Specification)

* Hệ thống bao gồm các module màn hình LCD, cảm biến gia tốc MMA8451Q, I2C và các LED và Switch cần được thiết lập một cách phù hợp để có thể hoạt động chính xác.
* Vi xử lý đọc dữ liệu từ cảm biến MMA8451Q thông qua module I2C, đồng thời để active và thiết lập trạng thái hoạt động của cảm biến cũng phải thông qua giao tiếp I2C. Các quy trình trong việc đọc và viết dữ liệu nên cảm biến, cũng như cách thiết lập các thanh ghi phù hợp với mode hoạt động mong muốn được quy định trong datasheet của cảm biến.
* Switch 1 và 3 cần được cấp clock và thiết lập ở chế độ sử dụng ngắt để điều khiển hoạt động của hệ thống.
* LED xanh và đỏ được cấp clock và được thiết lập ở chế độ GPIO.
* Bộ định thời Systick timer được thiết lập với nguồn clock lấy từ clock của vi xử lý – SystemCoreclock.
* Chương trình điều khiển chính của hệ thống sẽ thiết lập hoạt động của các module, khởi tạo các biến cần thiết, thực hiện đọc dữ liệu từ cảm biến bằng cách thực hiện quá trình đọc theo chuẩn I2C, sử dụng dữ liệu đọc được với thuật toán phát hiện bước chân, cập nhật giá trị số bước chân mới để hiển thị trở lại màn hình LCD. Ngoài ra nó cũng phải thay đổi được hoạt động của hệ thống, của các module dựa trên các tín hiệu ngắt nhận được từ Switch 1,3 và Systick Timer.

# Thực hiện hệ thống (Implementation)

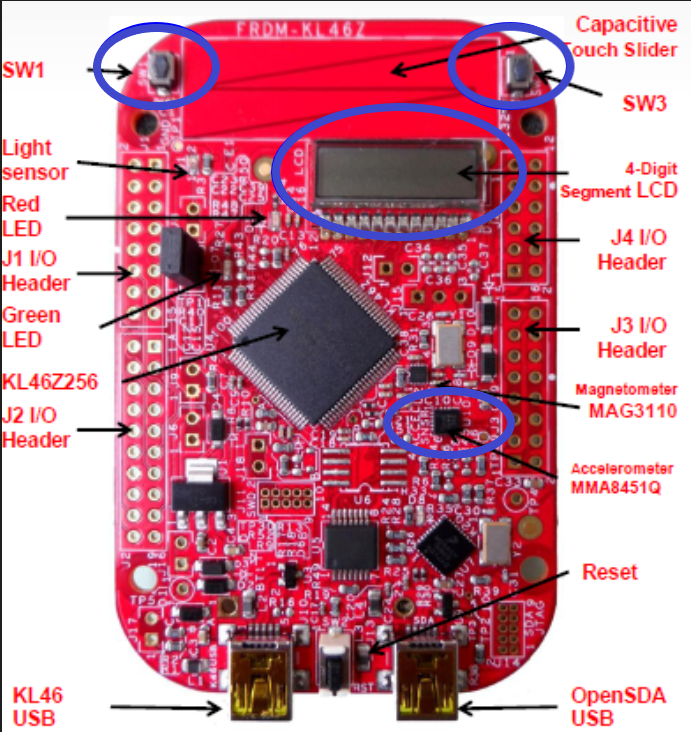
## Kiến trúc phần cứng (Hardware Architecture)

Miêu tả chức năng của các ngoại vi có sẵn trên bo mạch FRDM-KL46Z được sử dụng để thực hiện hệ thống đếm bước chân (Bảng 1);

Bảng 1: Mô tả các ngoại vi được sử dụng.

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên ngoại vi** | **Chức năng** |
| 1 | SW1 | Start/Stop bộ đếm |
| 2 | SW3 | Reset giá trị số đếm về 0 khi bộ đếm đang ở trạng thái Start |
| 3 | Red led | Tắt khi bộ đếm ở trạng thái Start, Sáng khi bộ đếm ở trạng thái Stop |
| 4 | Green led | Nháy khi bộ đếm ở trạng thái Start, không nháy khi bộ đếm ở trạng thái Stop |
| 5 | LCD | Hiển thị giá trị thập phân là số bước chân mà người dùng đã đi |
| 6 | MMA8451Qs | Thu thập giá trị gia tốc , gửi về cho vi xử lý |

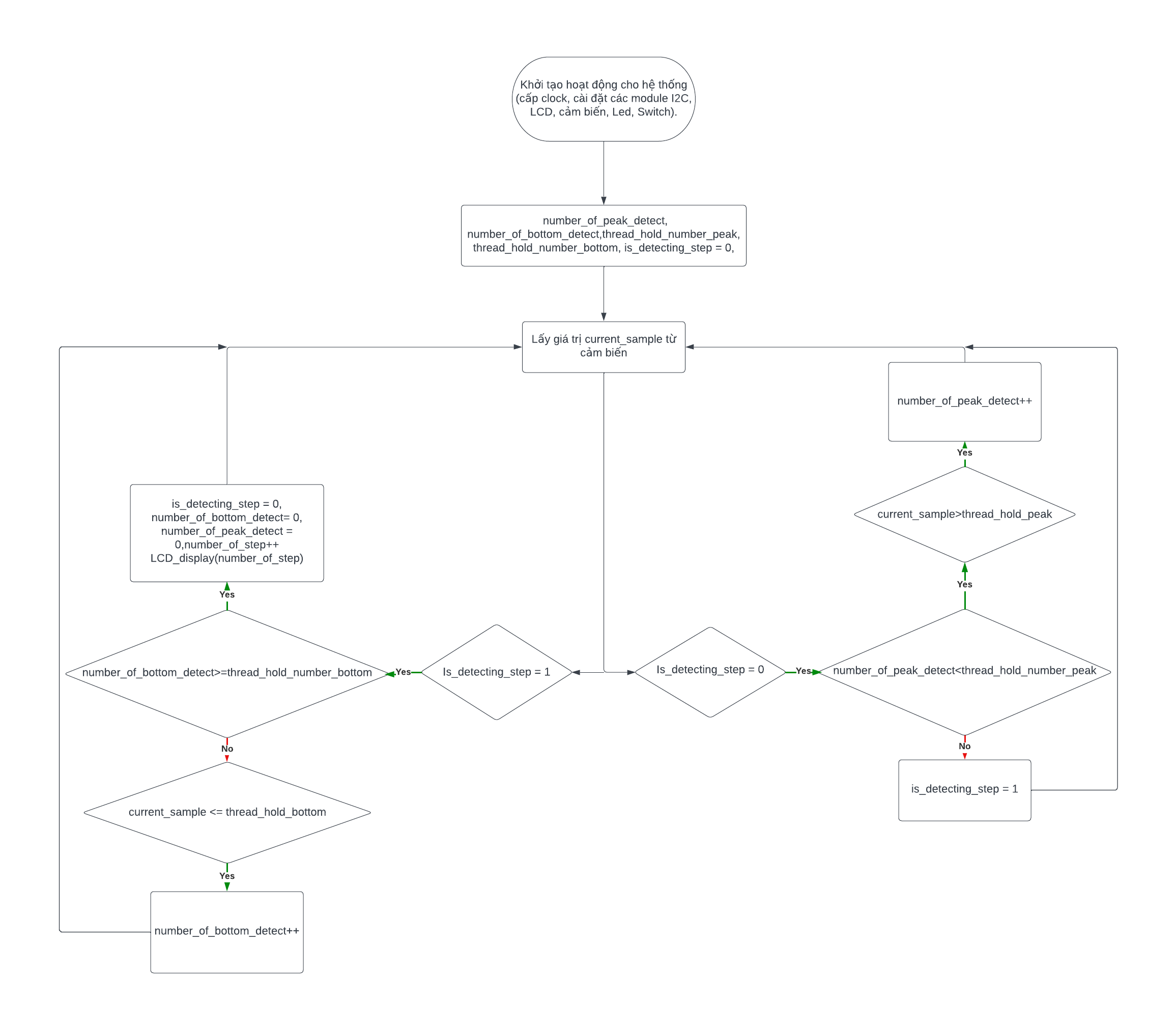
Các mô-đun phần cứng trên bo mạch FRDM-KL46Z được sử dụng để thực hiện bộ điều khiển như sau:



Hình 1. Các Module sử dụng ở trên bo mạch.

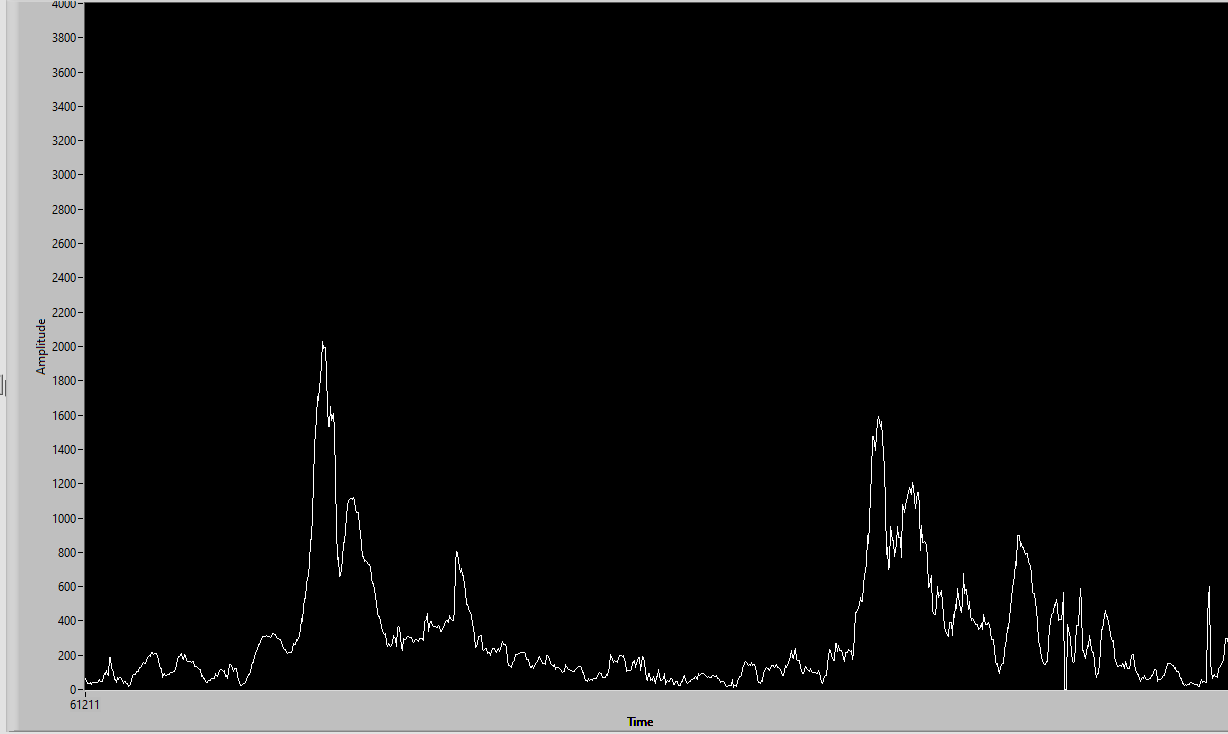
### Khối xử lý trung tâm

Ta có lưu đồ thuật toán của chương trình điều khiển:



Hình 2. Lưu đồ thuật toán của chương trình điều khiển.

* Chương trình bắt đầu với việc khởi tạo hoạt động cho hệ thống, bao gồm việc thiết lập clock, cài đặt các mode hoạt động cho các module của hệ thống.
* Sau khi hoàn thành việc thiết lập các module, chương trình khởi tạo các biến cho quá trình đọc và sử lý dữ liệu từ cảm biến để phát hiện bước chân của người dùng:
* Biến number\_of\_peak\_detect và number\_of\_bottom\_detect được dùng để đếm số lượng đỉnh và số lượng đáy của giá trị gia tốc đọc được từ cảm biến, 2 biến này được gán giá trị 0 khi khởi tạo.
* Biến thread\_hold\_number\_bottom và thread\_hold\_number\_peak dùng để xác định giá trị ngưỡng mà khi vượt quá thì sẽ làm thay đổi luồng điều khiển, 2 biến này được gán giá trị mặc định là 20 và 150.
* Biến thread\_hold\_bottom và thread\_hold\_peak là biến dùng để xác định ngưỡng mà khi gia tốc đọc được lớn hơn thread\_hold\_peak hoặc nhỏ hơn thread\_hold\_bottom thì nó được coi là 1 giá trị đỉnh hoặc 1 giá trị đáy.
* Biến is\_detecting\_step dùng để điều khiển luồng thực thi của chương trình, biến này được gán giá trị 1 khi mà number\_of\_peak\_detect lớn hơn thread\_hold\_number\_peak, và được gán lại về 0 khi đã phát hiện được bước chân của người dùng
* Giải thích rõ hơn về thuật toán phát hiện số bước chân:

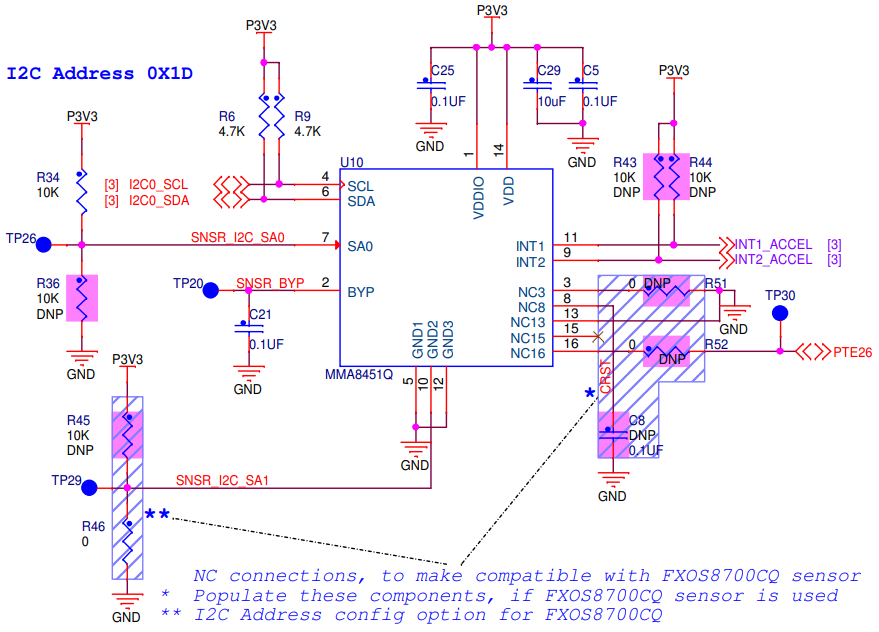


Hình 3. Giá trị gia tốc theo đọc được từ cảm biến sau 2 bước chân của người dùng

Ta nhận thấy rằng sau mỗi bước chân thì giá trị của gia tốc sẽ tạo thành các đỉnh sóng như ở trên hình, để phát hiện được số bước chân thì ta cần phải phát hiện ra được các đỉnh sóng như vậy, thuật toán chia ra làm 2 giai đoạn:

* Giai đoạn 1: Phát hiện đỉnh sóng: Hệ thống liên tục đọc dữ liệu từ cảm biến, khi phát hiện ra giá trị hiện tại lớn hơn giá trị ngưỡng của đỉnh sóng (thread\_hold\_peak), nó tăng thêm một cho giá trị biến đếm số lượng đỉnh sóng đã phát hiện được (number\_of\_peak\_detect). Khi số lượng đỉnh sóng phát hiện được lớn hơn giá trị ngưỡng (thread\_hold\_number\_peak), hệ thống chuyển sang Giai đoạn 2 (is\_detecting\_step = 1).
* Giai đoạn 2: Phát hiện đáy sóng: Khi phát hiện ra giá trị đầu vào nhỏ hơn ngưỡng của đáy sóng (thread\_hold\_bottom), nó tăng thêm một cho giá trị biến đếm số lượng đáy sóng đã phát hiện được (number\_of\_bottom\_detect). Khi số lượng đáy sóng lớn hơn giá trị ngưỡng (thread\_hold\_number\_bottom), hệ thống sẽ coi như là đã phát hiện được bước chân của người dùng, nó cật nhập lại giá trị số bước chân (number\_of\_step), hiển thị lại giá trị số bước chân mới nên LCD và sau đó reset lại giá trị các biến đếm và biến điều khiển (is\_detecting\_step = 0), quay trở lại giai đoạn 1 – Phát hiện đỉnh sóng.

### Cảm biến đầu vào

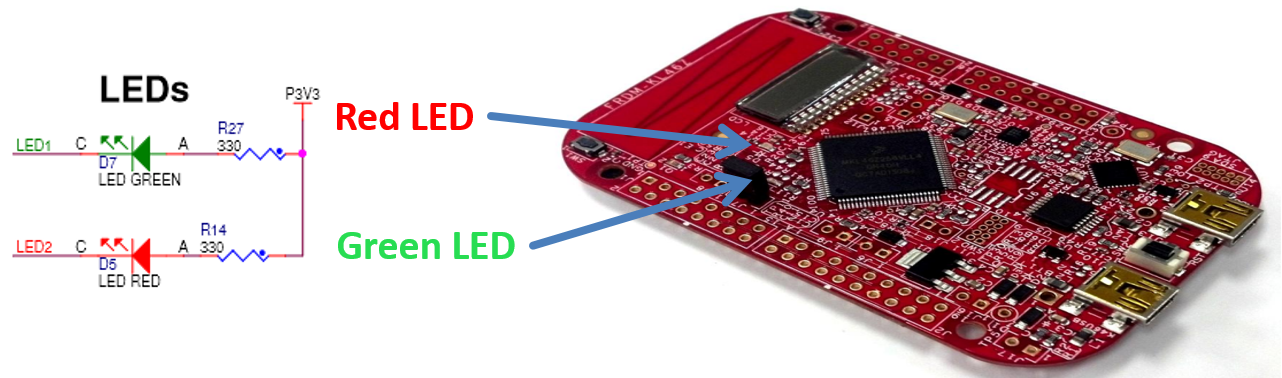


Hình 4. Cấu trúc của cảm biến đầu vào.

|  |  |
| --- | --- |
| **MMA8451Q** | **KL46Z Pin** |
| SDA | PTE25 |
| SCL | PTE24 |
| INT1\_ACCEL | PTC5 |
| INT2\_ACCEL | PTD1 |

Bảng 1. Kết nối I2C trên bo mạch FRDM-KL46Z.

### Các LED

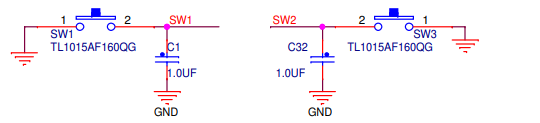


Hình 5. Các LED trên bo mạch FRDM-KL46Z.

|  |  |
| --- | --- |
| **LED** | **KL46Z Pin** |
| Red | PTE29 |
| Green | PTD5 |

Bảng 2. Kết nối các LED trên bo mạch KL46Z.

### Switch

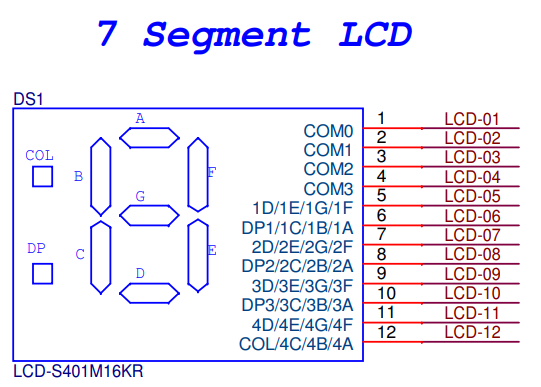


Hình 6. Cấu trúc của SW1,SW3

|  |  |
| --- | --- |
| **Switch** | **KL46Z Pin** |
| SW1 | PTC3 |
| SW3 | PTC12 |

Bảng 3. Kết nối các Switch trên bo mạch KL46Z.

### LCD



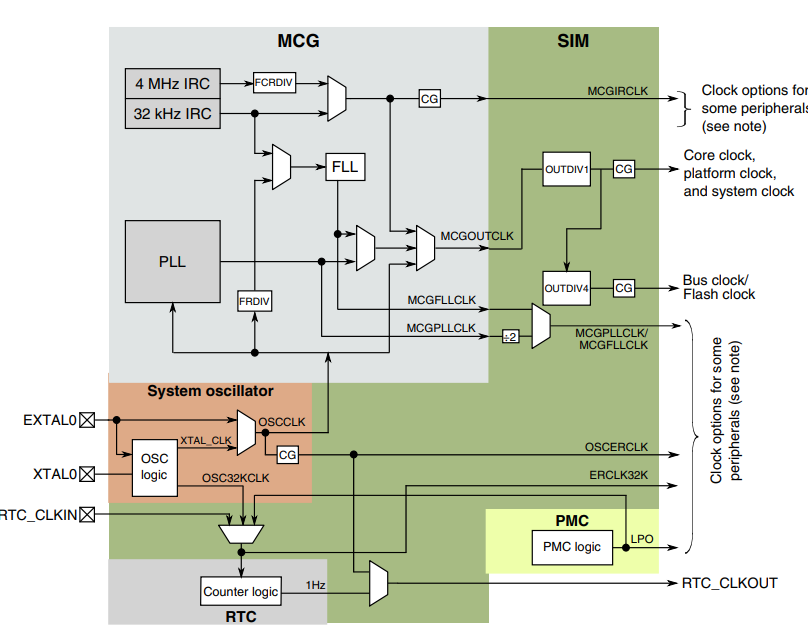
Hình 7. Cấu trúc của LCD.

|  |  |  |
| --- | --- | --- |
| **s401 Pin** | **KL46 LCD Pin** | **Pin Name** |
| 1 | LCD\_P40 (COM 0) | PTD0 |
| 2 | LCD\_P52 (COM 1) | PTE4 |
| 3 | LCD\_P19 (COM 2) | PTB23 |
| 4 | LCD\_P18 (COM 3) | PTB22 |
| 5 | LCD\_P37 | PTC17 |
| 6 | LCD\_P17 | PTB21 |
| 7 | LCD\_P7 | PTB7 |
| 8 | LCD\_P8 | PTB8 |
| 9 | LCD\_P53 | PTE5 |
| 10 | LCD\_P38 | PTC18 |
| 11 | LCD\_P10 | PTB10 |
| 12 | LCD\_P11 | PTB11 |

Bảng 4. Kết nối các chân của LCD trên bo mạch KL46Z.

## Lập trình phần mềm

### Thiết lập Clock cho hệ thống



Hình 8. Sơ đồ khối các module clock của hệ thống.

Việc thiết lập clock cho hệ thống bao gồm việc thiết lập các module OSC, MCG và SIM:

* Module OSC: Ta enable clock OSCERCLK, thiết lập chế độ disable OSCERCLK khi vi điều khiển ở Stop Mode, các bit điều khiển giá trị Capacitor Load để ở disable, đồng thời với module MCG, ta chọn nguồn External reference clock từ Oscillator, có giá trị 8Mhz, chế độ hoạt động của Oscillator ở mode High Frequency, Low power.
* Module MCG và SIM:
* MCGOUTCLK: được chọn từ PLL, nguồn clock của PLL được lấy từ module OSC với giá trị 8Mhz, được đưa qua 1 bộ chia tần với giá trị được đặt là chia 2, và 1 bộ chia tần nữa với giá trị được đặt là nhân 24, khi đó giá trị của MCGOUTCLK sẽ là (8/2)\*24 = 96(Mhz), giá trị của 2 bộ chia tần OUTDIV1 và OUTDIV2 được đặt là 2 và 2, đầu ra clock của 2 bộ lần lượt là 48Mhz và 24Mhz.
* MCGIRCK: được chọn từ nguồn Slow internal reference clock (IRC) có giá trị 32Khz.
* MCGPLLCK/MCGFLLCLK: được chọn từ VCO của PLL, đi qua một bộ chia đôi tần số, có giá trị : 96/2 = 48(Mhz).
* LPO: được chọn làm nguồn clock cho module LCD, có giá trị 1Khz.

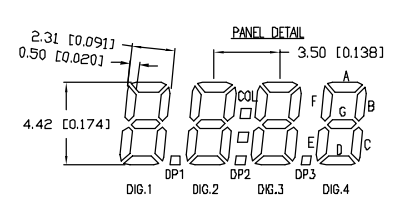
### Khởi tạo và thiết lập LCD

Việc thiết lập hoạt động của LCD bao gồm việc cấp clock cho module, thiết lập các thông số liên quan đến các chế độ hoạt động và thiết lập các chân backplane, frontplane của LCD theo datasheet của bo mạch:

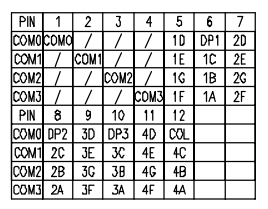
* Cấp clock cho module: Ta sử dụng thanh ghi SCGC5 của module SIM với bit SLCD để cấp clock cho module.
* Thanh ghi GCR: Thanh ghi điều khiển các thông số liên quan đến các chức năng của LCD khi hoạt động, 1 số bit quan trọng ta quan tâm đến như:
* VSUPPLY: chọn nguồn nuôi cho LCD từ VLL3 được tải từ VD.
* ALTSOURCEL: chọn nguồn clock của LCD từ nguồn clock thay thế 1-MCGIRCK.
* SOURCE: chọn nguồn clock thay thế (alternate clock).
* DUTY: lựa chọn duty cycle cho LCD, vì ta có 4 dạng sóng nên ta set 011.
* Thanh ghi AR: Thanh ghi điều khiển một số mode hoạt động đặc biệt của LCD như: Blink mode, Blank display, ta khởi tạo giá trị 0 cho tất cả các bit của thanh ghi này.
* Thanh ghi Back Plane Enable register gồm 2 thanh ghi giúp ta thiết lập mode hoạt động của chân tương ứng là backplane hay frontplane, bit 1 thì chân được set là backplane pin,bit 0 chân là fronplane pin, ta dựa theo datasheet của Board để set các chân ở mode hoạt động phù hợp.

Dựa vào bảng kết nối chân, các chân 40,52,19,18 của LCD sẽ được set là backplane pin, tương ứng với các Waveform: A, B, C, D, các chân còn lại là frontplane pin, các chân có số thứ tự nhỏ hơn hoặc bằng 31 sẽ được set mode hoạt động ở thanh ghi BPEN0, với các chân có số thứ tự lớn hơn 31 thì được thực hiện ở thanh ghi BPEN1, với bit tương ứng với chân bằng: Số thứ tự chân – 32.

* Thanh ghi Wave form có vai trò trong việc điều khiển việc hiển thị ký tự ở trên màn hình:



Hình 9. Sơ đồ khối LCD.



Hình 10. Bảng trạng thái của các segment tương ứng với từng Phase.

Một chân frontplane sẽ được kết nối với 4 segment của LCD, để điều khiển trạng thái sáng tối của từng segment với mỗi frontplane pin, ta sẽ set mode hoạt động của fronplane pin tương ứng là on/off với mỗi Phase tín hiệu trong một Duty cycle. Ví dụ, để hiển thị số 1, ta cần segment 4B,4C sáng, và tất cả các segment còn lại tắt, như vậy ta sẽ set các chân frontplane từ 5 đến 11 không active ở bất kì Phase tín hiệu nào, với chân 12, để 4C và 4B sáng, nó sẽ được active ở Phase B,C.

* Hàm khởi tạo và điều khiển hoạt động cho LCD:
* Hàm thiết lập chế độ hoạt động của LCD:

**void LCD\_Init(LCD\_Type \*base){**

**SIM->SCGC5 |= SIM\_SCGC5\_SLCD\_MASK; // cap clock cho module LCD**

**base->GCR = 0x8B0004B;**

**// \*\*Note: parameter for setting:**

**// Duty: su dung 4 dang song => 011**

**// LCD Clock Prescaler: LCD frame frequencyLCD\_clock/(((DUTY=3)+1) \* 8 \* (4 + LCLK=1)) \* Y => 001**

**// Source: Select altenate source => 1: MCGIRCLK**

**// LCDEN: LCD Driver disable**

**// LCDSTP: 0: LCD driver, charge pump, resistor bias network, and voltage regulator continue while in Stop mode**

**// LCDDOZE: 0**

**// FFR: 0 standard frame rate**

**// ALTSOURCE: 0 Select Alternate Clock Source 1**

**// 13–12 ALTDIV: 0 Divide factor = 1 (No divide)**

**// FDCIEN: 0 No interrupt request is generated when fault detection is completed**

**// PADSAFE: 0 LCD frontplane and backplane functions enabled**

**// VSUPPLY: 0 Drive VLL3 internally from VDD**

**// CPSEL: 1 LCD charge pump is selected. Resistor network disabled. (The internal 1/3-bias is forced.)**

**// LADJ: 11 Slowest clock source for charge pump (LCD glass capacitance 1000 pF or 500pF or lower if FFR is set )**

**// RVTRIM: 1000**

**// RVEN: Regulated voltage disabled.**

**base->AR = (uint32\_t)0;**

**// BLINK: 0 disable blinking**

**// ALT: 0 normal display mode**

**// BLANK:0 normal or alternate display mode**

**// BMODE: 0 Display blank during the blink period**

**// BRATE: Blink rate = LCD clock/2^(12+0)**

**base->BPEN[0] = 0x000c0000U; // set lcd 19,18 pin as backplane pin**

**base->BPEN[1] = 0x00100100U; // set lcd 40,52 pin as backplane pin**

**base->PEN[0] = 0x000e0d80U;**

**base->PEN[1] = 0x00300160U;**

**// enable all pin of lcd module**

**base->FDCR = 0; // disable fault detection**

**for (uint32\_t regNum = 0; regNum < 16; regNum++)base->WF[regNum] = 0;**

**}**

* Hàm thiết lập chế độ hoạt động của một chân LCD là backplane pin hay frontplane pin:

**void SetBackPlanePhase(LCD\_Type\* base,uint32\_t pinIndx, slcd\_phase\_t phase){**

**base->WF8B[pinIndx]= phase;**

**}**

**void SetFrontPlaneSegments(LCD\_Type \*base, uint32\_t pinIndx, uint8\_t operation){**

**base->WF8B[pinIndx] = operation;**

**}**

### Khởi tạo và thiết lập các LED

* Hàm cấp clock cho PORTD trên module SIM và thiết lập mode hoạt động cho chân 5 cổng D ứng với led xanh:

**void green\_led\_init(){**

**SIM->SCGC5 |= SIM\_SCGC5\_PORTD\_MASK;**

**PORTD->PCR[5] = 1<<8; // dat chan 5 cong D o che do IO**

**PTD->PDDR = 1<<5; // dat chan 5 o che do doc va ghi**

**PTD->PDOR &= ~(1<<5); // set led chan 5 sang**

**}**

* Hàm cáp clock cho PORTE trên module SIM và thiết lập mode hoạt động cho chân 29 cổng E ứng với led đỏ, đồng thời khi hàm này được gọi (chỉ được gọi 1 lần duy nhất trong main) thì nó disable Systick timer để tương ứng với trạng thái đầu tiên của Board là Stop (Led xanh không nháy)

**void red\_led\_init(){**

**SIM->SCGC5 |= SIM\_SCGC5\_PORTE\_MASK;**

**PORTE->PCR[29] |= 1<<8;**

**PTE->PDDR = 1<<29;**

**PTE->PDOR &= ~(1<<29);**

**SysTick->CTRL &= (int32\_t)~(1<<0);**

**}**

### Khởi tạo và thiết lập ngắt cho Systick Timer

* Hàm set up các thanh ghi của Systick Timer với các thông số:
* Thanh ghi Control:
* Bit 0: Gía trị bằng 1-Enable counter
* Bit 1: Giá trị bằng 1-Khi giá trị đếm về 0 thì thực hiện ngắt
* Bit 2: Giá trị bằng 1: Chọn nguồn clock là processor clock
* Thanh ghi ReLoad: Giá trị khởi tạo lại cho bộ đếm mỗi khi đếm đến 0 bằng SystemCoreClock/1000
* Đặt quyền ưu tiên của ngắt với giá trị 127:

**void init\_systick(){**

**NVIC\_SetPriority(SysTick\_IRQn,(1<<7)-1);**

**SysTick->LOAD = SystemCoreClock/1000;**

**SysTick->CTRL = 1<<2 | 1<<1 | 1<<0;**

**}**

* Hàm thực hiện khi xử lý ngắt:
* Sử dụng 2 biến toàn cục là mstick và blink\_mstick, giá trị 2 biến này tăng mỗi khi có ngắt do Systick gây ra, khi giá trị của blink\_mstick vượt quá 1000, hàm sẽ reset giá trị của nó về 0 và gọi hàm toggle\_green\_led() để đảo trạng thái của led xanh, biến mstick dùng cho hàm delay(uint32\_t tick).

**void toggle\_green\_led(){**

**PTD->PDOR^= (uint8\_t)(1<<5);**

**}**

**void SysTick\_Handler (void){**

**if(blink\_mstick>=1000){**

**blink\_mstick = 0;**

**toggle\_green\_led();**

**}**

**mstick++;**

**blink\_mstick++;**

**}**

**void delay(uint32\_t tick){**

**while(mstick < tick);**

**mstick = 0;**

**}**

### Khởi tạo và thiết lập ngắt cho SW1 và SW3

* Hàm khởi tạo ngắt cho SW1 và SW3 xóa tất cả các ngắt hiện tại đang chờ xử lý, thiết lập các chân 3 và 12 cổng C (tương ứng với 2 switch) ở mode: Điện trở treo cao, cho phép sử dụng điện trở trong, GPIO, và ngắt ở sườn giảm. Hàm đặt quyền ưu tiên của ngắt với giá trị 128 (cao hơn ngắt của Systik) và Enable ngắt cho module PORTC\_PORTD

**void portc\_portd\_nvic\_init(){**

**NVIC\_ClearPendingIRQ(PORTC\_PORTD\_IRQn);**

**SIM->SCGC5 |= SIM\_SCGC5\_PORTC\_MASK;**

**SIM->SCGC5 |= SIM\_SCGC5\_PORTD\_MASK;**

**PORTC->PCR[3] |= ((uint8\_t)(1<<0)|(uint8\_t)(1<<1)|PORT\_PCR\_IRQC(0xA)|PORT\_PCR\_MUX(1));**

**PORTC->PCR[12] |= ((uint8\_t)(1<<0)|(uint8\_t)(1<<1)|PORT\_PCR\_IRQC(0xA)|PORT\_PCR\_MUX(1));**

**NVIC\_SetPriority(PORTC\_PORTD\_IRQn,(1<<7));**

**NVIC\_EnableIRQ(PORTC\_PORTD\_IRQn);**

**}**

* Hàm thực hiện khi có ngắt: Kiểm tra xem ngắt xuất hiện ở SW1 hay SW3 bằng cách kiểm tra bit 24-ISF của thanh ghi PCR:
* Nếu ngắt từ SW1, ta lật trạng thái của bit 0 thanh ghi Control Systick, kiểm tra trạng thái hiện tại của Board, nếu là Stop thì chuyển sang trạng thái Start, và hiển thị giá trị 0 trên LCD, nếu là Start thì chuyển sang trạng thái Stop, ngừng hiển thị trên LCD. Sau cùng xóa cờ ngắt trẻn SW1 và lật trạng thái của Led red
* Nếu ngắt từ SW3, ta reset giá trị số đếm về 0, hiển thị giá trị 0 trên LCD, sau đó xóa cờ ngắt trên SW3.

**void PORTC\_PORTD\_IRQHandler(){**

**if(PORTC->PCR[3]&(uint32\_t)(1<<24)){**

**SysTick->CTRL ^= (int32\_t)(1<<0);**

**if(state==0){**

**state = 1;**

**MY\_SLCD\_StartDisplay();**

**display\_decimal(0);**

**}**

**else if(state==1){**

**number\_of\_step = 0;**

**state=0;**

**MY\_LCD\_StopDisplay();**

**}**

**PORTC->PCR[3] |= PORT\_PCR\_ISF\_MASK;**

**toggle\_red\_led();**

**}**

**else if(PORTC->PCR[12]&(uint32\_t)(1<<24)){**

**number\_of\_step = 0;**

**display\_decimal(0);**

**PORTC->PCR[12] |= PORT\_PCR\_ISF\_MASK;**

**}**

**}**

### Thiết lập module I2C

Việc thiêt lập module I2C bao gồm việc thiết lập các chân SCLvà SDA ở đúng mode hoạt động, cấp xung clock cho module và thiết lập chế độ hoạt động phù hợp của module dựa trên các thanh ghi điều khiển:

* Hàm I2C\_ConfigurePins thiết lập các chân 24 và 25 PORTE và cấp clock cho PORTE, tương ứng với chân SDA và SCL của module I2C, module hoạt động của 2 chân: Sử dụng trở treo bên trong, 3 bit MUX (bit 8-10) đặt ở chức năng số 2:

**void I2C\_ConfigurePins(void){**

**SIM->SCGC5 = SIM->SCGC5 | SIM\_SCGC5\_PORTE\_MASK; //clock to PTE24 and PTE25 for I2C0**

**PORTE->PCR[24] = (((1u<<10) & ~(1u<<9)) | (1u<<8)|(1u<<0)|(1u<<2)|(1u<<1))&(~(1u<<4))&(~(1u<<6));**

**PORTE->PCR[25] = (((1u<<10) & ~(1u<<9)) | (1u<<8)|(1u<<0)|(1u<<2)|(1u<<1))&(~(1u<<4))&(~(1u<<6));**

**}**

* Hàm Init\_Master cấp clock cho module I2C, đồng thời đặt giá trị nên các thanh ghi để thiết lập hoạt động của module I2C theo ý muốn:
* Thanh ghi Address: Chứa giá trị của địa chỉ module I2C mà ta muốn dùng khi nó hoạt động ở mode Slave, vì ta không dùng chế độ này nên để mặc định là 0x00.
* Thanh ghi Frequency Divider: Dùng để lựa chọn các giá trị thuộc tính như. Baud rate, SDA hold time, SCL start hold time, SCL stop hold time cho phiên truyền, ta lựa chọn giá trị mul bằng 0, giá trị divider là 240.
* Thanh ghi Status: Đây là thanh ghi trạng thái của module, được sử dụng để kiểm tra trạng thái của bus hay của phiên truyền, thường:
* Bit 7: Cho biết quá trình truyền dữ liệu đã hoàn thành chưa, nó được xóa khi ta đọc dữ liệu từ thanh ghi Data trong mode Receive hoặc khi ta ghi dữ liệu vào thanh ghi Date trong mode Transmit.
* Bit 6: Ta không sử dụng.
* Bit 5: Cho biết trạng thái hiện tại của bus có đang rảnh hay không.
* Bit 4: Cho biết có xảy ra xung đột trên bus hay không, xung đột xảy ra khi nhiều hơn 1 thiết bị cùng truyền trên bus, khi đó cơ chế trọng tài sẽ chọn ra chỉ một thiết bị được phép truyền dữ liệu, khi đó các thiết bị còn lại sẽ nhận được bit trạng thái báo có xung đột xảy ra.
* Bit 3: Ta không sử dụng.
* Bit 2: Bit này chỉ ra mode hoạt động Transmit/Receive khi module ở trạng thái Slave, ta không sử dụng.
* Bit 1: Bit này được set khi có ngắt xảy ra, ta sử dụng để kiểm tra quá trình truyền đã hoàn thành hay chưa.
* Bit 0: Kiêm tra xem module nhận được ACK hay NAK Signal sau khi hoàn thành quá trình truyền một byte.
* Thanh ghi Control register 2: Đây là thanh ghi chứa một số mode hoạt động khác của module, ta chỉ quan tâm đến bit 6 giúp ta chọn mode địa chỉ chỉ của slave là 7 bit hay 10bit, do địa chỉ của cảm biến nằm trên boad không quá 7 bit (0x1D) nên ta gán giá trị 0, các bit còn lại cũng để mặc định là 0.
* Thanh ghi Programmable Input Glitch Filter: Đây là thanh ghi cũng chứa một số chức năng đặc biệt khác như: phát hiện tín hiệu stop, stop hold,… ta cũng không dùng đến các chức nang này nên để mặc định là 0x00.
* Thanh ghi Control: Chứa các mode và chức năng đặc biệt quan trọng của module:
* Bit 7: Enable hoặc Disable module, ta gán giá trị 1 để khởi tạo hoạt động của module.
* Bit 6: Enable hoặc Disable ngắt của module, ở trong project này, ta sử dụng phương pháp polling để kiểm tra dữ liệu đã sẵn sàng để đọc chưa nên không cần sử dụng đến ngắt, gán giá trị 0 dể Disable ngắt.
* Bit 5: Lựa chọn chế độ hoạt động của module, khi gán bit này từ 0 nên 1, một tín hiệu Start sẽ được gửi trên bus và module chuyển sang trạng thái Master, khi gán từ 1 về 0, một tín hiệu Stop được gửi và module chuyển về trạng thái Slave, ban đầu ta đặt bit này bằng 0, sau đó gán lại thành 1 để bắt đầu phiên truyền/đọc dữ liệu.
* Bit 4: Lựa chọn giữa 2 mode hoạt động là Transmit hoặc Receive, trong quá trình đọc dữ liệu từ cảm biến, module sẽ linh hoạt chuyển đổi tùy theo quá trình, ta khởi tạo giá trị ban đầu là 0.
* Bit 3: Lựa chọn giữa việc gửi tín hiệu ACK hoặc NAK trên byte dữ liệu nhận được hiện tại hoặc tiếp theo (tùy theo bit FACK thanh ghi SMB), ta sử dụng bit này khi muốn gửi đi NAK kể cả khi đã nhận được dữ liệu, dùng để kết thúc phiên đọc dữ liệu từ cảm biến.
* Bit 2: Bit này luôn được set giá trị 0, gán 1 vào nó sẽ tạo ra tín hiệu Repeat Start trên bus, dùng trong quá trình đọc dữ liệu.
* Bit 1: Sử dụng chế độ Wakeup, đánh thưc MCU từ chế độ Low power khi quá trình gửi địa chỉ hoàn thành với kết quả thành công, ta không sử dụng mode hoạt động này nên gán giá trị 0.
* Bit 0: Sử dụng chức năng DMA: ta không sử dụng nên gán giá trị 0.

**void Clear\_flags(){**

**uint32\_t t =(I2C\_S\_ARBL\_MASK|I2C\_S\_IICIF\_MASK)|(I2C\_FLT\_STOPF\_MASK << 8); // lay gia tri co co phat hien tin hieu stop o thanh ghi FLT**

**if(t&(I2C\_FLT\_STOPF\_MASK << 8)){ // neu detect duoc tin hieu stop**

**// xoa co stop neu co o thanh ghi FLT**

**I2C0->FLT |= (uint8\_t)(t >> 8U);**

**}**

**// xoa tat ca cac bit thanh ghi status, clear bit ARBL va set bit interrupt detect**

**I2C0->S = (uint8\_t)t;**

**}**

**void Init\_Master(){**

**SIM->SCGC4 = SIM->SCGC4 | SIM\_SCGC4\_I2C0\_MASK;**

**/\* Reset the module. \*/**

**I2C0->A1 = 0;**

**I2C0->F = 0;**

**I2C0->C1 = 0;**

**I2C0->S = 0xFFU;**

**I2C0->C2 = 0;**

**I2C0->FLT = 0x00U;**

**I2C0->RA = 0;**

**// Disable module I2C**

**I2C0->C1 &= ~(I2C\_C1\_IICEN\_MASK);**

**// Clear Flags**

**Clear\_flags();**

**I2C0->F = I2C\_F\_MULT(0U) | I2C\_F\_ICR(31U);**

**I2C0->C1 = I2C\_C1\_IICEN(1); // Enable module I2C**

**}**

### Chương trình điều khiển (Hàm main())

Chương trình điều khiển sẽ sẽ thiết lập hoạt động của các module dựa trên các hàm đã định nghĩa, đối với cảm biến MMA8451Q, ta sử dụng giao tiếp I2C để điều khiển các thanh ghi của cảm biến, qua đó điều khiển được cảm biến hoạt động theo chế độ mong muốn. Sau khi đã hoàn thành việc thiết lập hoạt động, chương trình việc thực hiện đọc dữ liệu cảm biến, xử lý nó theo thuật toán đã mô tả ở phía trên để phát hiện được bước chân của người dùng, cuối cùng là hiển thị số bước chân nên màn hình LCD:

* Định nghĩa các biến sử dụng:

**int16\_t x,y,z =0;**

**int number\_of\_detect = 0;**

**int is\_detecting\_step=0;**

**int number\_of\_peak\_detect = 0;**

**int thread\_hold\_number\_peak=20;**

**int thread\_hold\_number\_bottom=150;**

**int thread\_hold\_peak = 1000;**

**int thread\_hold\_bottom = 150;**

**int thread\_hold\_**

**float current\_sample =0;**

**float accel=0;**

**uint8\_t rx\_buffer[] = {0,0,0,0,0,0};**

* Thiết lập clock cho hệ thống:

**SetSimSafeDivs();**

**InitOsc0();**

**SetFllExtRefDiv();**

**PLL\_Init();**

**SetInternalRefClkConfig();**

**SetSimConfig();**

**SystemCoreClock = 48000000U;**

* Thiết lập các module:
* Module I2C:

**I2C\_ReleaseBus();**

**I2C\_ConfigurePins();**

**Init\_Master();**

* Module LCD**:**

**LCD\_Init(LCD);**

**SetBackPlanePhase(LCD,40,kSLCD\_PhaseAActivate);**

**SetBackPlanePhase(LCD,52,kSLCD\_PhaseBActivate);**

**SetBackPlanePhase(LCD,19,kSLCD\_PhaseCActivate);**

**SetBackPlanePhase(LCD,18,kSLCD\_PhaseDActivate);**

* Module Switch 1,3:

**portc\_portd\_nvic\_init();**

* Module LED xanh, đỏ:

**green\_led\_init();**

**red\_led\_init();**

* Module Systick Timer:

**init\_systick();**

* Module cảm biến MMA8452Q:

**I2C\_ReadAccelRegs(I2C0,0x1D,0x0D,rx\_buffer,1);**

**uint8\_t databyte =0;**

**I2C\_W riteAccelReg(I2C0,0x1D,0x2AU,databyte);**

**databyte = (uint8\_t)(1<<0)+ (uint8\_t)(1<<4);**

**I2C\_WriteAccelReg(I2C0,0x1D,0x0EU,databyte);**

**databyte = (uint8\_t)(1<<0)+(uint8\_t)(1<<4);**

**I2C\_WriteAccelReg(I2C0,0x1D,0x0F,databyte);**

**databyte = 0x0D;**

**I2C\_WriteAccelReg(I2C0,0x1D,0x2AU,databyte);**

* Đọc dữ liệu cảm biến và thực hiện thuật toán phát hiện bước chân:

**while (1) {**

**I2C\_ReadAccelRegs(I2C0,0x1D,0x01,rx\_buffer,6);**

**x = (int16\_t)(256U\*rx\_buffer[0]|rx\_buffer[1])/4U;**

**y = (int16\_t)(256u\*rx\_buffer[2]|rx\_buffer[3])/4U;**

**z = (int16\_t)(256U\*rx\_buffer[4]|rx\_buffer[5])/4U;**

**x\*=2; y\*=2;**

**accel = (float)x\*x+(float)y\*y+(float)z\*z;**

**current\_sample = sqrt(accel);**

**if(peak\_value<current\_sample)peak\_value = current\_sample;**

**if(is\_detecting\_step ==0&&number\_of\_peak\_detect<thread\_hold\_number\_peak){**

**if(current\_sample>thread\_hold\_peak)number\_of\_peak\_detect++;**

**}**

**elseif(is\_detecting\_step==0&&number\_of\_peak\_detect>=thread\_hold\_number\_peak)is\_detecting\_step = 1;**

**else if(is\_detecting\_step==1){**

**if(current\_sample <= thread\_hold\_bottom&&number\_of\_detect<thread\_hold\_bottom){**

**number\_of\_detect++;**

**}**

**else if(current\_sample >=300)number\_of\_detect = 0;**

**else if(number\_of\_detect>=thread\_hold\_number\_bottom){**

**LCD\_clear();**

**number\_of\_step++;**

**display\_decimal(number\_of\_step);**

**is\_detecting\_step = 0;**

**number\_of\_detect = 0;**

**number\_of\_peak\_detect = 0;**

**}**

**}**

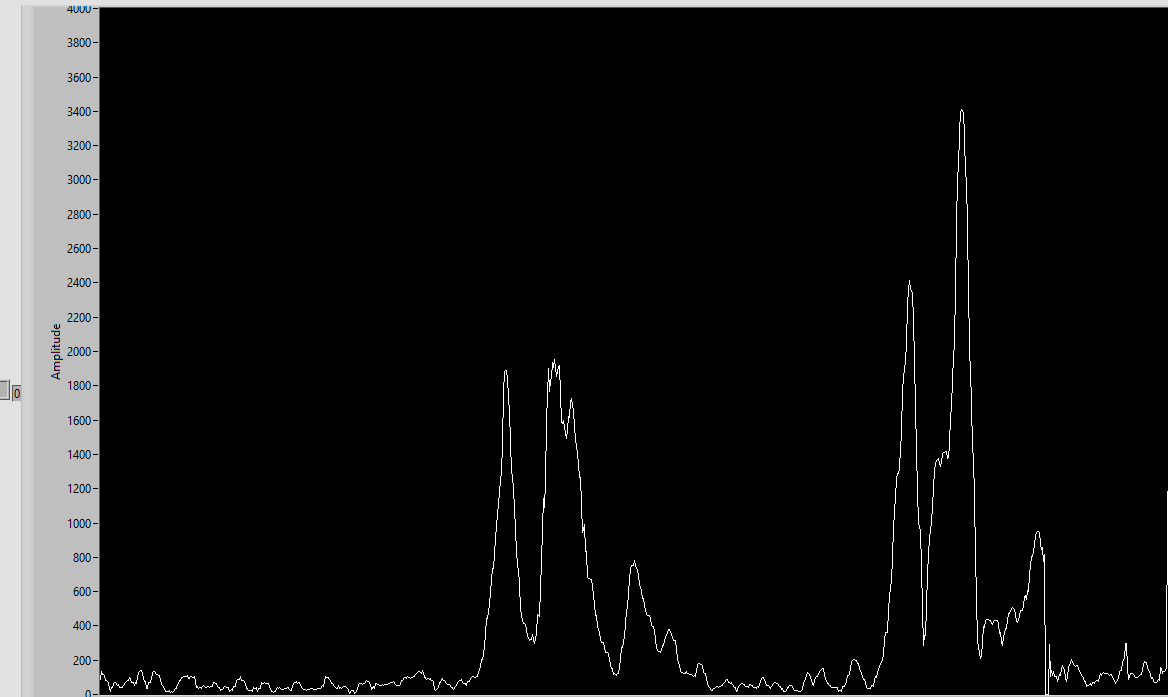
# Kiểm chứng (Validation)

Link video kiểm chứng hoạt động của hệ thống:

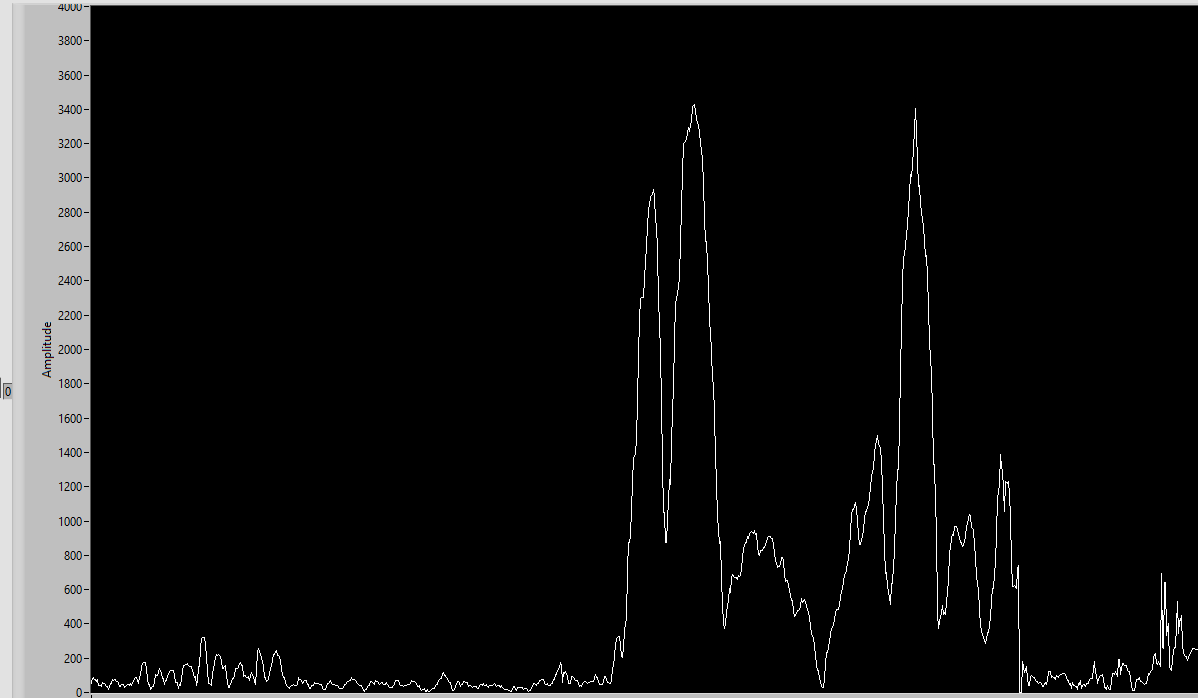
<https://drive.google.com/file/d/1tgF91UvSnRLkXM1yrItJ712wDb9xTFLK/view>

# Kết luận (Conclusion)

Hệ thống có thể phát hiện số bước chân khá chính xác khi người dùng di chuyển với tốc độ trung bình trở xuống, khi di chuyển nhanh, hoặc chạy, hệ thống thường không bắt kịp được số bước chân của người dùng, nguyên nhân là do khi di chuyển nhanh, gia tốc biến thiên nhanh và với giá trị lớn hơn so với khi di chuyển với tốc độ trung bình hoặc chậm. giá trị ngưỡng của đáy sóng lớn hơn, và do di chuyển nhanh nên khoảng thời gian giữa mỗi đỉnh sóng ngắn đi, đồng nghĩa với giá trị thread\_hold\_number\_bottom trên thực tế sẽ không còn phù hợp nữa và khiến cho việc phát hiện bước chân bị thiếu sót:



Hình 11. Giá trị gia tốc khi người dùng đi bộ (2 bước).



Hình 12. Giá trị gia tốc khi người dùng đi nhanh (2 bước).

Ta có thể khắc phục điểm yếu này bằng cách chỉnh giá trị thread\_hold\_number\_bottom xuống và tăng giá trị ngưỡng đáy sóng nên, tuy nhiên trong trường hợp ngược lại việc tăng giá trị ngưỡng có đáy sóng có thể khiển cho việc phát hiện bước chân khi người dùng đi chậm bị thừa.

* Như vậy để cải tiến hệ thống, ta cần phải thêm vào khả năng phát hiện trạng thái di chuyển hiện tại của người dùng để có thể điều khiển các tham số cho quá trình phát hiện bước chân một cách phù hợp.

Danh sách Hình (List of Figures)

[Hình 1. Các Module sử dụng ở trên bo mạch. 6](#_Toc136950150)

[Hình 2. Lưu đồ thuật toán của chương trình điều khiển. 7](#_Toc136950151)

[Hình 3. Giá trị gia tốc theo đọc được từ cảm biến sau 2 bước chân của người dùng 8](#_Toc136950152)

[Hình 4. Cấu trúc của cảm biến đầu vào. 9](#_Toc136950153)

[Hình 5. Các LED trên bo mạch FRDM-KL46Z. 10](#_Toc136950154)

[Hình 6. Cấu trúc của SW1,SW3 10](#_Toc136950155)

[Hình 7. Cấu trúc của LCD. 11](#_Toc136950156)

[Hình 8. Sơ đồ khối các module clock của hệ thống 12](#_Toc136950157)

[Hình 9. Sơ đồ khối LCD 14](#_Toc136950158)

[Hình 10. Bảng trạng thái của các segment tương ứng với từng Phase 14](#_Toc136950159)

[Hình 11. Giá trị gia tốc khi người dùng đi bộ (2 bước) 25](#_Toc136950160)

[Hình 12. Giá trị gia tốc khi người dùng đi nhanh (2 bước) 25](#_Toc136950161)

Danh sách Bảng (List of Tables)

[Bảng 1. Kết nối I2C trên bo mạch FRDM-KL46Z. 10](#_Toc136951737)

[Bảng 2. Kết nối các LED trên bo mạch KL46Z. 10](#_Toc136951738)

[Bảng 3. Kết nối các Switch trên bo mạch KL46Z. 11](#_Toc136951739)

[Bảng 4. Kết nối các chân của LCD trên bo mạch KL46Z. 12](#_Toc136951740)

References

1. <https://octopart.com/lcd-s401m16kr-lumex-21365384?gclid=CjwKCAjwsvujBhAXEiwA_UXnAPQuy_9V59WqfnqQVw_jHzJ9Dk1OjRDDDskWhzcaDy_uORI3HBAtnRoCaUUQAvD_BwE>
2. <https://mcuxpresso.nxp.com/api_doc/dev/210/group__i2c__driver.html>
3. <https://developer.arm.com/documentation/dui0552/a/cortex-m3-peripherals/system-timer--systick>