REPORT

THIẾT KẾ VÀ THỰC HIỆN HỆ THỐNG PHÁT HIỆN NGÃ

Ver 0.2

19/04/2024

	Họ và tên (Full name)	**Mã SV (ID)**	**Đóng góp (Contribution)**
Thành viên 1 (Member 1)	Hoàng Trung Hiệp	21020907	I2C, MMA8451, Systick, Switch, LED
Thành viên 2 (Member 2)	Lê Đình Huy	21020914	SLCD, I2C, LED, Systick, Switch
Tên/Địa chỉ Repo trên Github	Link mã nguồn		

Tóm tắt (Abstract - from 5 to 10 lines)
Thiết kế một dự án hệ thống nhúng phát hiện ngã sử dụng cảm biến gia tốc MMA8451Q và module LCD 7 đoạn hiển thị 4 chữ số. Nút bấm SW1 được sử dụng để chuyển đổi hệ thống giữa trạng thái hoạt động và trạng thái dừng. Khi hệ thống dừng hoạt động, nếu bấm SW1 bộ đếm chuyển sang trạng thái hoạt động, và ngược lại. Nút bấm SW2 để xóa trạng thái hệ thống về trạng thái bình thường. Dữ liệu về cảm biến gia tốc được xử lý và hiển thị trạng thái của hệ thống (1 – Ngã, 0 – Bình thường)

Từ khóa (Keywords)

Hướng dẫn (Guide)
Sinh viên điền vào báo cáo theo mẫu đính kèm. Sinh viên điền các mục: • Thông tin sinh viên, mã số sinh viên • Mục *Đóng Góp* điền các công việc đã làm tương ứng của từng sinh viên. • Tên/Địa chỉ Repo trên Github Ngoại trừ phần thông tin sinh viên, mã số sinh viên và tên/địa chỉ Repo trên Github ở đầu, sinh viên cần hoàn thành các phần nội dung (theo các mục đã được gợi ý – nhưng không hạn chế) trong phần báo cáo để mô tả các công việc nhóm đã thực hiện và các kết quả đã đạt được. Sinh viên làm theo nhóm chỉ cần **1 sinh viên đại diện nộp 1 bản báo cáo, sửa tên file thành tên của các thành viên trong nhóm (viết có dấu).** Sinh viên nộp lại báo cáo này trước khi tới trình bày kết quả, **muộn nhất trước ngày thi hết môn một ngày**. **Ngày thi, SV cần mang máy tính laptop và sản phẩm để chạy demo!** **Lưu ý: Nghiêm cấm mọi hình thức copy bài (bao gồm cả report và mã nguồn) của nhau. Nếu phát hiện sự giống nhau giữa 2 bài thì tùy mức độ mà có thể sẽ bị trừ điểm hoặc chia lấy điểm trung bình làm điểm của project.**

Document History

Version	Time	Revised by	Description
V0.1	15/4/2024	Nguyễn Kiêm Hùng	Original Version
V0.2	19/04/2024	Nguyễn Kiêm Hùng	Modified the format of report

Mục lục (Table of Contents)

1. Giới thiệu (Introduction)

Mục tiêu của dự án: Vận dụng các kiến thức, kỹ năng đã được học để thiết kế và thực thi một hệ thống nhúng phát hiện ngã. 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. Sinh viên cũng có thể sử dụng một bo mạch tương đương (dùng vi xử lý ARM) để thực hiện dự án.

2. Yêu cầu đối với thiết kế (Requirements)

2.1. Yêu cầu đối với thiết kế

Yêu cầu thiết kế:

- *Chức năng:* phát hiện người ngã khi người sử dụng thiết bị bị ngã và hiển thị kết quả trên màn LCD.
- *Các đầu vào:*
    - Hệ thống sử dụng cảm biến gia tốc MMA8451 để phát hiện ngã.
    - SW1 để chuyển đổi hệ thống giữa trạng thái hoạt động và trạng thái dừng. Khi hệ thống 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, hệ thống tiếp tục giám sát trạng thái của người dùng và hiển thị trên LCD.
    - SW2 để xóa (reset) trạng thái hệ thống về trạng thái bình thường.
- *Các đầu ra:*
    - Hệ thống có 2 lối ra trạng thái: LED xanh nhấp nháy với tần số 1Hz khi hệ thống hoạt động; tắt khi hệ thống dừng hoạt động. LED đỏ nháy ở tần số 2Hz khi hệ thống phát hiện trạng thái ngã; tắt khi hệ thống ở trạng thái bình thường.
    - LCD: Hiện thị trạng thái của hệ thống: 0 – bình thường, 1 - ngã
- Bộ đếm sử dụng một timer để xác định khoảng thời gian nhấp nháy các LED.

2.2. Đặc tả kỹ thuật (Specification)

2.2.1. *Mô tả chung*

Dự án phát hiện ngã sử dụng cảm biến gia tốc MMA8451, module LCD để hiển thị, 2 đèn LED (xanh và đỏ) và 2 nút bấm (SW1 và SW2). Hệ thống sẽ giám sát chuyển động của người sử dụng và phát hiện khi có hiện tượng ngã, sau đó hiển thị thông báo lên LCD và kích hoạt đèn LED.

2.2.2. *Mô tả phần cứng*

- Bộ vi xử lý: ARM Cortex M0+
- Cảm biến gia tốc MMA8451:
- Module LCD: Hiển thị thông tin về trạng thái của hệ thống.
- LED Xanh: Chỉ báo trạng thái bình thường.
- LED Đỏ: Chỉ báo trạng thái ngã.
- Nút bấm SW1: Được sử dụng để chuyển đổi hệ thống giữa trạng thái hoạt động và trạng thái dừng
- Nút bấm SW2: Được sử dụng để xóa (reset) trạng thái hệ thống về trạng thái bình thường

3. Thực hiện hệ thống (Implementation)

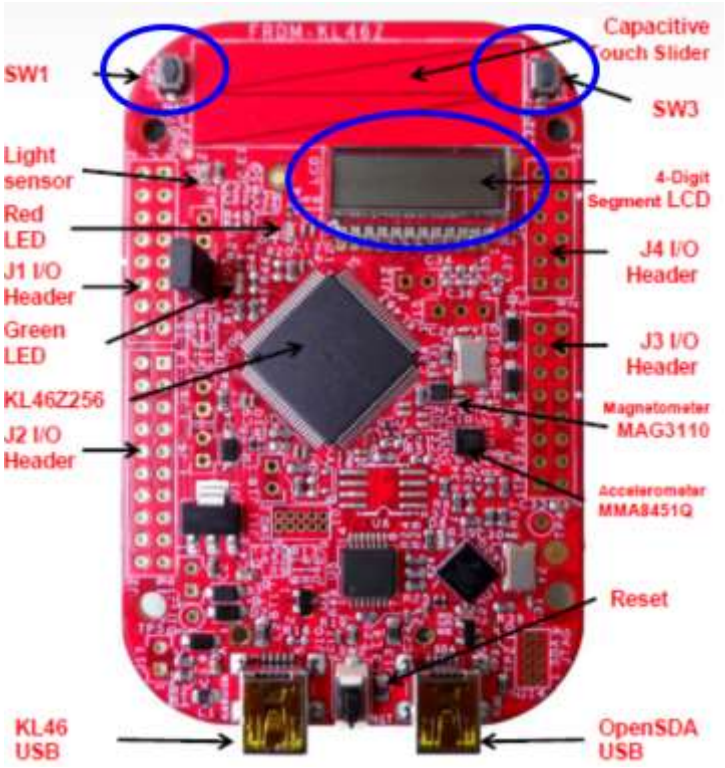
3.1. Kiến trúc phần cứng (Hardware Architecture)

Miêu tả chức năng của các ngoại vi có sẵn trên bo mạch FRDM-KL46Z được sử dụng để thực hiện hệ thống (**Error! Reference source not found.**):

Bảng 1: Mô tả các ngoại vi được sử dụng

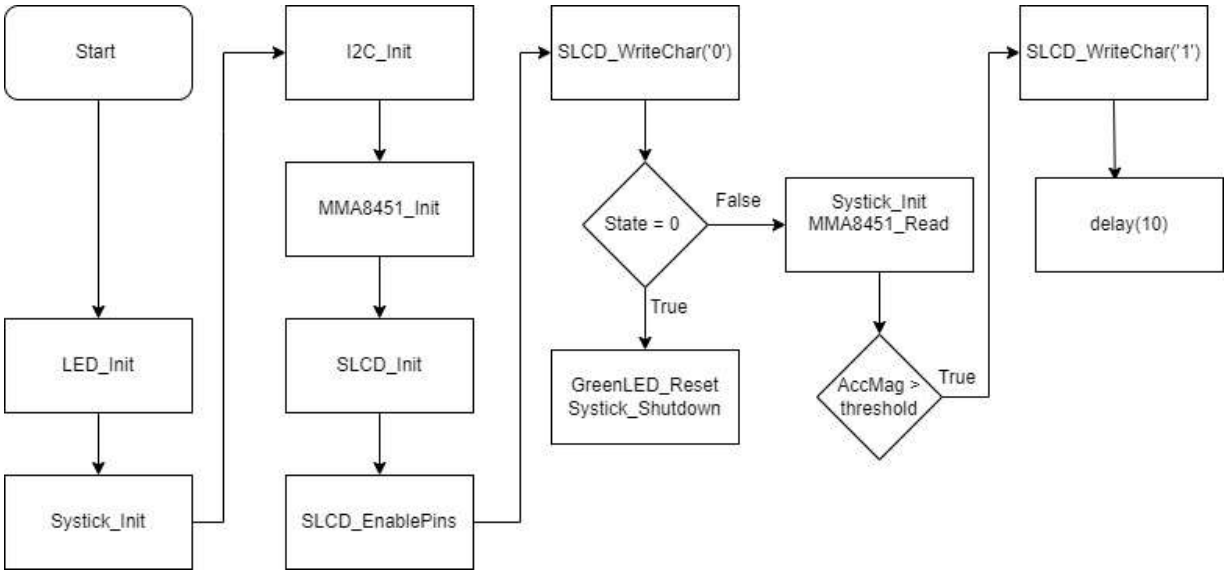
STT	Tên ngoại vi	Chức năng
1	SW1	Chuyển đổi hệ thống giữa trạng thái hoạt động và dừng
2	SW2	Xóa (reset) trạng thái hệ thống về trạng thái bình thường
3	LED xanh	Hiển thị trạng thái bình thường
4	LED đỏ	Hiển thị trạng thái ngã
5	MMA8451Q	Cảm biến gia tốc 3 trục
6	LCD-S401M16KR	LED 7 đoạn hiển thị 4 chữ số

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



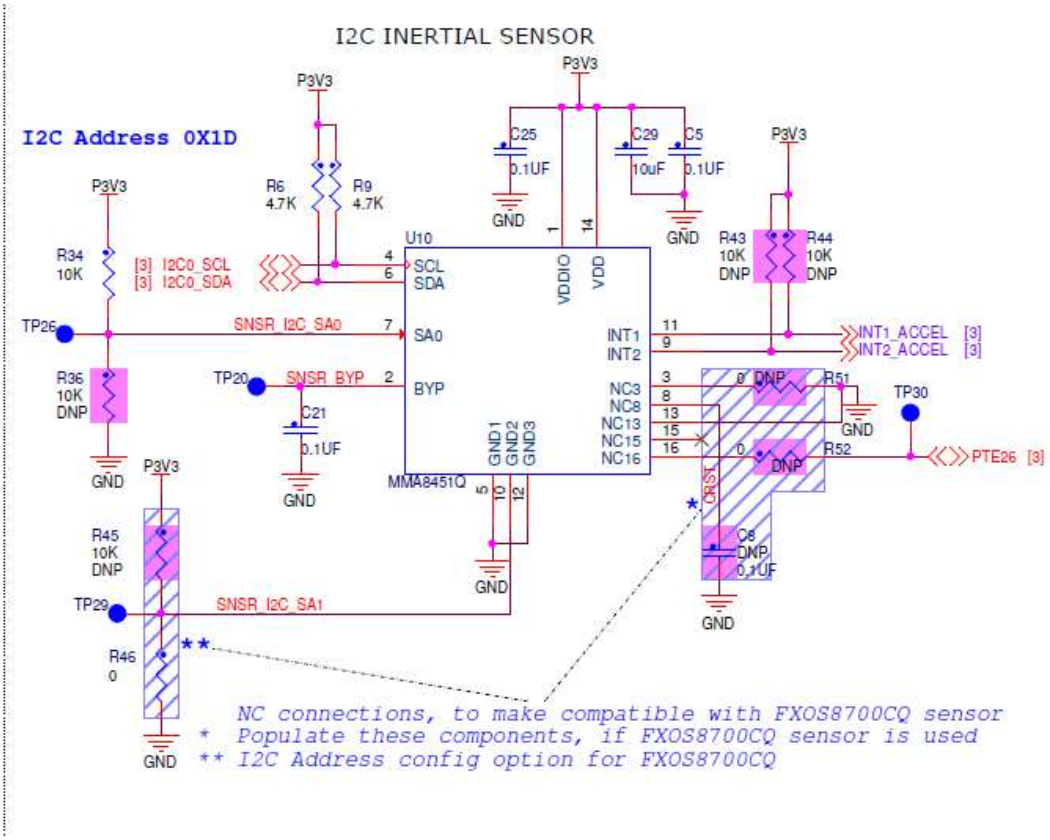
Hình 1: Giao diện ghép nối I/O của đơn vị Sorting Unit.

3.1.1. *Khối xử lý trung tâm*



Hình 2: Khối xử lý trung tâm

3.1.2. *Cảm biến đầu vào*

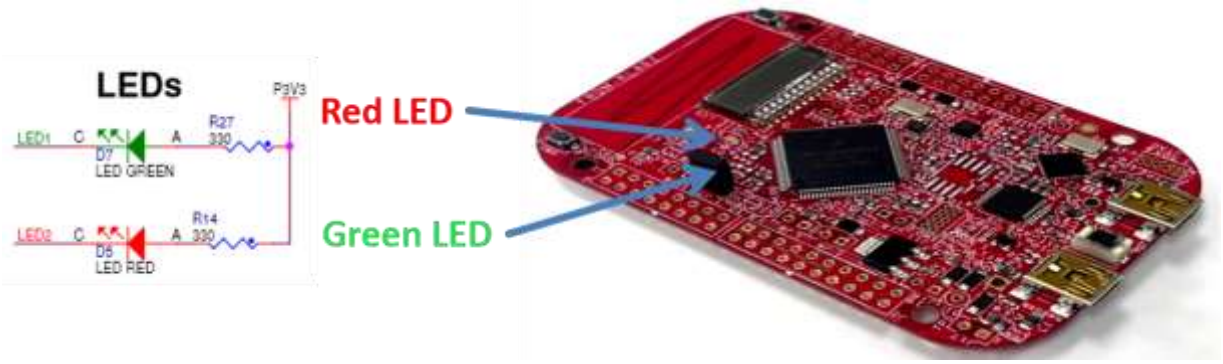


Hình 3: Cấu trúc của cảm biến đầu vào.

Bảng 2: Kết nối các chân của cảm biến MMA8451Q trên bo mạch KL46Z

MMA8451Q	KL46
SCL	PTE25/TPM0_CH1/**I2C0_SDA**
SDA	PTE24/TPM0_CH0/**I2C0_SCL**
INT1_ACCEL	*PTC5/LLWU_P9*
INT2 _ACCEL	PTD1 (shared with INT2_MAG)

3.1.3. *Các LED*

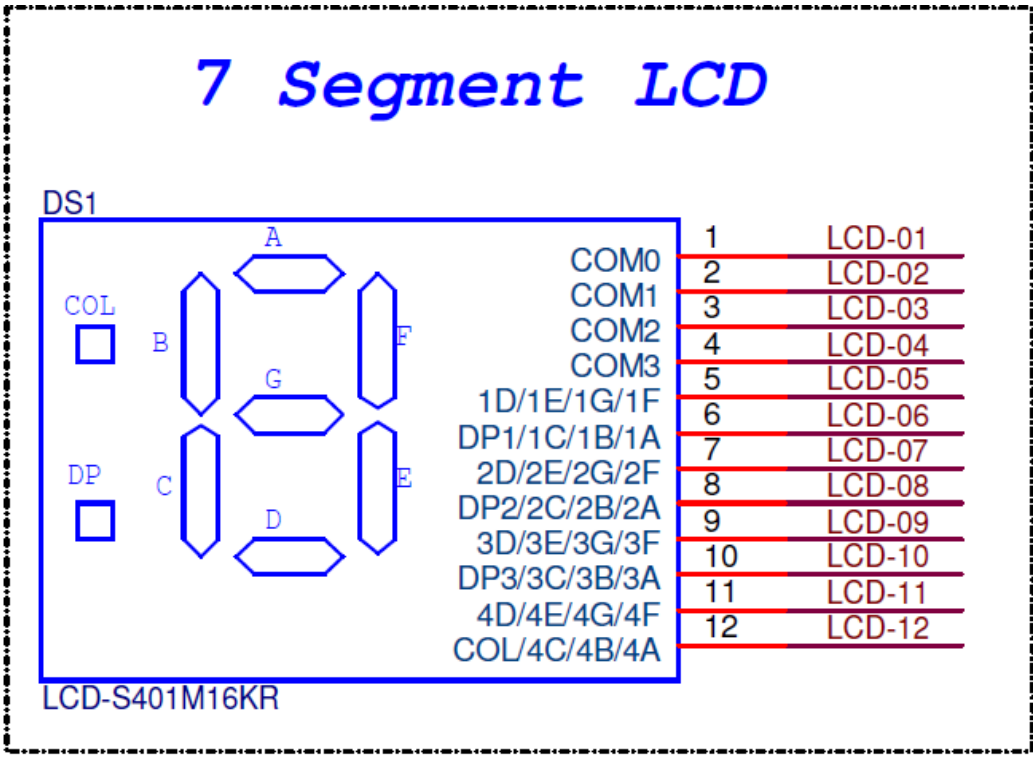


Hình 4: Các LED trên bo mạch FRDM-KL46Z.

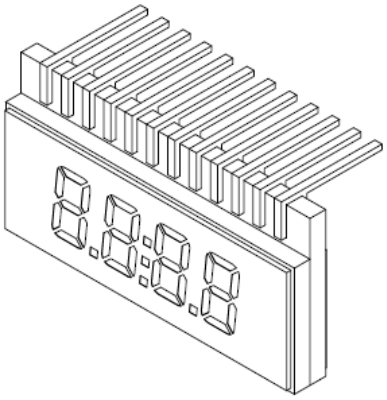
Bảng 3: Kết nối các LED trên bo mạch KL46Z

LED	KL46Z Pin
Red	PTE29
Green	PTD5

3.1.4. *LCD*



Hình 5: Các chân LCD 7 đoạn trên bo mạch KL46Z



LCD CHARACTERISTICS	
FLUID TYPE	TN
BACKGROUND COLOR	GRAY
VIEWING DIRECTION	6:00
DISPLAY MODE	REFLECTIVE
OPERATING VOLTAGE	3.0V AC
OPERATING TEMPERATURE	0°C TO 50°C
STORAGE TEMPERATURE	0°C TO 50°C
DRIVING METHOD	1/4 DUTY, 1/3 BIAS
CONNECTION TYPE	12 PINS

PIN	1	2	3	4	5	6	7	8	9	10	11	12
COM0	COM0	/	/	/	1D	DP1	2D	DP2	3D	DP3	4D	COL
COM1	/	COM1	/	/	1E	1C	2E	2C	3E	3C	4E	4C
COM2	/	/	COM2	/	1G	1B	2G	2B	3G	3B	4G	4B
COM3	/	/	/	COM3	1F	1A	2F	2A	3F	3A	4F	4A

s401 pin	KL46 LCD Pin
1	LCD_P40 (COM0)
2	LCD_P52 (COM1)
3	LCD_P19 (COM2)
4	LCD_P18 (COM3)
5	LCD_P37
6	LCD_P17
7	LCD_P7
8	LCD_P8
9	LCD_P53
10	LCD_P38
11	LCD_P10
12	LCD_P11

Hình 6: Kết nối các SLCD trên bo mạch KL46Z

3.2. Lập trình phần mềm

3.2.1. *Thiết lập Clock cho hệ thống*

Hệ thống sử dụng clock SCGC5 cho các port B, C, D, E, SLCD, trong đó MMA8451 để giao tiếp bằng I2C PORTE, LED sử dụng PORT D, E, SWITCH là C và sLCD sử dụng PORT B, C, D, E, SLCD

```
SIM->SCGC5 |= SIM_SCGC5_SLCD_MASK
            | SIM_SCGC5_PORTB_MASK
            | SIM_SCGC5_PORTC_MASK
            | SIM_SCGC5_PORTD_MASK
            | SIM_SCGC5_PORTE_MASK;
```

Hình 7: Cấp Clock cho SCGC5

Đối với SCGC4 cần kích hoạt cho I2C:

```
SIM->SCGC4 |= SIM_SCGC4_I2C0_MASK; //clock to I2C0
```

Hình 8: Cấp Clock cho SCGC4

Đối với sLCD cần kích hoạt clock đa chức năng để có thể tạo ra các pha sóng để phân chia và hiện đoạn trên LCD:

```
MCG->C1 = MCG_C1_IRCLKEN_MASK | MCG_C1_IREFSTEN_MASK;
```

Hình 9: Cấp Clock MCG

3.2.2. *Khởi tạo các LED*

```
void LED_Init(void) {
  //Clock port D (Led green)
  SIM->SCGC5 |= (1<<12);
  //Clock port E (Led red)
  SIM->SCGC5 |= (1<<13);

  //GPIO alternative function
  PORTD->PCR[5] |= (1<<8);
  PORTE->PCR[29] |= (1<<8);

  //Set Output mode
  PTD->PDDR |= 1<<5;
  PTE->PDDR |= 1<<29;

  // turn off led red
  PTE->PSOR |= 1<<29;
}
```

Hình 10: Khởi tạo các LED

Đầu tiên bật clock cho cổng D(led xanh), cổng E(led đỏ). Sau đó cấu hình chân PTD[5], PTE[29] để sử dụng chức năng GPIO.Thiết lập chế độ đầu ra để điều khiển các LED.

3.2.3. *Khởi tạo các Switch*

```
void Switch_Init(void) {
  state = 0;
  //Eable clock portC
  SIM->SCGC5 |= (1<<11);

  // SW1
  // Configure GPIO
  PORTC->PCR[3] |= (1<<8);
  //PE: Internal pullup or pulldown resistor is enabled
  PORTC->PCR[3] |= (1<<1);
  //PS: Internal pullup resistor is enabled
  PORTC->PCR[3] |= (1<<0);
  //IRCQ = 1010: Interupt on falling edge
  PORTC->PCR[3] |= (1<<17) | (1<<19);
  //Set input mode
  PTC->PDDR &= (~(uint32_t)(1<<3));

  //SW2
  // Configure GPIO
  PORTC->PCR[12] |= (1<<8);
  //PE: Internal pullup or pulldown resistor is enabled
  PORTC->PCR[12] |= (1<<1);
  //PS: Internal pullup resistor is enabled
  PORTC->PCR[12] |= (1<<0);
  //IRCQ = 1010: Interupt on falling edge
  PORTC->PCR[12] |= (1<<17) | (1<<19);
  //Set input mode
  PTC->PDDR &= (~(uint32_t)(1<<12));

  //PORTC_PORTD_IRQn 31
  NVIC_ClearPendingIRQ(PORTC_PORTD_IRQn);
  NVIC_EnableIRQ(PORTC_PORTD_IRQn);
}
```

Hình 11: Khởi tạo các Switch

Để thiết lập GPIO cho các nút nhấn, đầu tiên bật clock cho PORT C bằng cách đặt bit tương ứng của thanh ghi SIM->SCGC5. Nút nhấn được kích hoạt pull-up, cho phép đọc trạng thái logic. Sau đó thiết lập ngắt trên sườn xuống của cạnh bằng cách cấu hình các bit IRCQ tương ứng. Thiết lập chế độ input cho các nút bấm. Cuối cùng bật ngăt cho PORTC_PORTD.

3.2.4. *Khởi tạo Systick Timer*

```
void Systick_Init(void) {
  SysTick->CTRL |= (1 << 0);
  SysTick->CTRL |= (1 << 1);
  SysTick->CTRL |= (1 << 2);
  SysTick->LOAD = SystemCoreClock/1000;

  NVIC_SetPriority(-1, 15);
}
```

Hình 12: Khởi tạo Systick Timer

Để cấu hình và kích hoạt bộ đếm SysTick, trước tiên, bộ đếm SysTick được bật bằng cách đặt bit đầu tiên (bit 0) trong thanh ghi CTRL của SysTick. Sau đó, kích hoạt yêu cầu ngắt từ SysTick bằng cách đặt bit thứ hai (bit 1) trong thanh ghi CTRL. Tiếp theo, chọn nguồn clock là clock của bộ xử lý bằng cách đặt bit thứ ba (bit 2) trong thanh ghi CTRL. Giá trị tải (LOAD) của SysTick được thiết lập bằng cách chia SystemCoreClock (tần số clock của bộ xử lý) cho 1000, tạo ra ngắt sau mỗi mili giây. Cuối cùng, thiết lập mức độ ưu tiên cho ngắt SysTick bằng cách sử dụng hàm NVIC_SetPriority với tham số -1 và mức ưu tiên 15.

3.2.5. *Thiết lập hàm xử lý cho các ngắt.*

3.2.5.1. Hàm ngắt systick

```
void SysTick_Handler(void) {
  msTicks++;
  LED_tick_green++;

  // 1Hz => 1s between 2 time led green on =>500ms change from turn on to off
  if (LED_tick_green == 500) { //500ms
    PTD->PTOR |= 1 <<5;//Toggle LED xanh
    LED_tick_green = 0;
  }
  if(fall_detected){
    LED_tick_red++;
  }
  // 2Hz => 0.5s between 2 time led green on =>250ms change from turn on to off
  if (fall_detected && LED_tick_red == 250) { // 250ms
    PTE->PTOR |= 1 << 29; // Toggle LED do
    LED_tick_red = 0;
  }
}
```

Hình 13: Hàm ngắt Systick

Hàm này được gọi định kỳ mỗi 1ms . Khi LED_tick_green đếm lên đến 500 (tương ứng với 500ms) thì đảo trạng thái của LED xanh làm cho LED xanh nhấp nháy với tần số là 1Hz. Khi phát hiện ngã fall_detected = 1 thì mới để cho LED_tick_red tăng và khi tăng đến 250(tương ứng với 250ms) thì đảo trạng thái của LED đỏ làm cho LED đỏ nhấp nháy với tần số là 2Hz.

3.2.5.2. Hàm ngắt PortC, PortD

```
void PORTC_PORTD_IRQHandler(void) {
  //Pressed SW1
  if ((PTC->PDIR & (1<<3)) == 0) {
    if (state == 0) {
      state = 1;
    }
    else if (state == 1) {
      state = 0;
    }
  }

  //ISF: Clear interrupt flag
  PORTC->PCR[3] |= (1 << 24);

  //Pressed SW2
  if ((PTC->PDIR & (1<<12)) == 0) {
    state = 0;
    NVIC_SystemReset();
  }

  //ISF: Clear interrupt flag
  PORTC->PCR[12] |= (1<<24);
}
```

Hình 14: Hàm ngắt PortC, PortD

Hàm này kiểm tra các nút bấm, nếu nút 1 được bấm nó sẽ đảo trạng thái của biến state sau đó xóa cờ ngắt. Nếu nút 2 được bấm thì đặt biến state về 0 và khởi động lại hệ thống vi điều khiển.

3.2.6. *Chương trình điều khiển (Hàm main())*

```
int main(void) {
  LED_Init();
  Switch_Init();
  Systick_Init();
  I2C_Init();
  MMA8451_Init();
  delay(1);
  SLCD_Init();
  SLCD_EnablePins();
  SLCD_WriteChar('0');
  fall_detected = 0;

    while (1) {
    if (state ==0) {
      GREEN_LED_Reset();
      RED_LED_Reset();
      Systick_Shutdown();


    } else {
      Systick_Init();
      Data data = MMA8451_Read();
      float accMag = SVM(data);

      if (accMag > threshold) {
        fall_detected = 1;
        SLCD_WriteChar('1'); // Hi?n th? tr?ng thái ngã
      }
      delay(10);
    }
    }
}
```

Hình 15: Hàm main

Để khởi tạo và vận hành hệ thống giao tiếp với các module như cảm biến, LCD và LED bật tắt, trước tiên cần thực hiện các bước khởi tạo trong hàm main. Đầu tiên khởi tạo LED với LED_Init (), khởi tạo nút bấm bằng Switch_Init(), và khởi tạo bộ đếm SysTick bằng Systick_Init(), khởi tạo giao tiếp I2C với I2C_Init() và cảm biến MMA8451 với MMA8451_Init(), khởi tạo LCD với SLCD_Init(), kích hoạt các chân cần thiết của LCD bằng SLCD_EnablePins(), và hiển thị ký tự '0' trên LCD bằng SLCD_WriteChar('0'). Biến fall_detected được đặt bằng 0 để bắt đầu theo dõi trạng thái ngã.

Trong vòng lặp vô hạn while (1), nếu trạng thái state bằng 0 (trạng thái không hoạt động), đèn LED xanh, LED đỏ sẽ tắt bằng cách gọi hàm GREEN_LED_Reset (), RED_LED_Reset (), tắt SysTick bằng Systick_Shutdown(). Nếu state bằng 1 (trạng thái hoạt động), khởi tạo lại SysTick bằng cách gọi Systick_Init (). Sau đó, đọc dữ liệu từ cảm biến MMA8451 bằng cách gọi MMA8451_Read (), tính toán độ lớn của gia tốc bằng hàm SVM (data). Nếu giá trị gia tốc

vượt quá ngưỡng đã định trước (threshold), trạng thái ngã được nhận biết (fall_detected = 1) và hiển thị ký tự '1' trên LCD bằng SLCD_WriteChar ('1').

3.2.7. *Khởi tạo và cấu hình SLCD*

```
void SLCD_Init(void)
{
    SIM->SCGC5 |= SIM_SCGC5_SLCD_MASK
                | SIM_SCGC5_PORTB_MASK
                | SIM_SCGC5_PORTC_MASK
                | SIM_SCGC5_PORTD_MASK
                | SIM_SCGC5_PORTE_MASK;

    PORTC->PCR[20] = 0x00000000; // VLL2
    PORTC->PCR[21] = 0x00000000; // VLL1
    PORTC->PCR[22] = 0x00000000; // VCAP2
    PORTC->PCR[23] = 0x00000000; // VCAP1

    MCG->C1 = MCG_C1_IRCLKEN_MASK | MCG_C1_IREFSTEN_MASK;
    MCG->C2 &= ~MCG_C2_IRCS_MASK;

    LCD->GCR = (LCD_GCR_RVEN_MASK*_LCDRVEN
              | LCD_GCR_RVTRIM(_LCDRVTRIM)
              | LCD_GCR_CPSEL_MASK*_LCDCPSEL
              | LCD_GCR_LADJ(_LCDLOADADJUST)
              | LCD_GCR_VSUPPLY_MASK*_LCDSUPPLY
              | !LCD_GCR_FDCIEN_MASK
              | LCD_GCR_ALTDIV(_LCDALTDIV)
              | !LCD_GCR_LCDDOZE_MASK
              | !LCD_GCR_LCDSTP_MASK
              | !LCD_GCR_LCDEN_MASK
              | LCD_GCR_SOURCE_MASK*_LCDCLKSOURCE
              | LCD_GCR_ALTSOURCE_MASK*_LCDALRCLKSOURCE
              | LCD_GCR_LCLK(_LCDLCK)
              | LCD_GCR_DUTY(_LCDDUTY)
              );

    SLCD_EnablePins();
    LCD->GCR |= LCD_GCR_LCDEN_MASK;
    LCD->AR = LCD_AR_BRATE(_LCDBLINKRATE);
}
```

Hình 16: Cấu hình SLCD

Hàm SLCD_Init được sử dụng để khởi tạo và cấu hình màn hình LCD. Đầu tiên, kích hoạt các clock cho các cổng GPIO (Port B, Port C, Port D, Port E) và LCD bằng cách đặt các bit tương ứng trong thanh ghi SIM->SCGC5.

Sau đó, các chân VLL1, VLL2, VCAP1, VCAP2 của PORTC được cấu hình với các thiết lập mặc định, có thể được sử dụng cho các kết nối mức điện áp và tụ điện liên quan đến LCD
Tiếp theo, kích hoạt nguồn điện tham chiếu nội (Internal Reference) và cho phép tiếp tục hoạt động trong chế độ dừng bằng cách đặt các bit tương ứng trong thanh ghi MCG->C1.
Sau đó, nó cấu hình các thiết lập cho LCD bằng cách đặt các bit tương ứng trong thanh ghi LCD->GCR. Điều này bao gồm việc thiết lập các thông số như điện áp điều chỉnh (Regulated voltage), điện áp điều chỉnh được điều chỉnh (Regulated voltage trim), chọn máy bơm điện (Charge pump), điều chỉnh tải (Load adjust), nguồn cấp (Supply), ngắt phát hiện lỗi (fault detection), chia tần số (Alternate clock divider), tắt LCD trong chế độ dừng (Doze Enable), tắt LCD trong chế độ dừng (Stop), kích hoạt trình điều khiển LCD (Driver Enable) chọn nguồn clock (Clock source), chọn nguồn clock thay thế (Alternate clock source), chia tần số clock (Clock prescaler), và chọn chế độ duty (Duty select)
Cuối cùng, kích hoạt các chân của SLCD và LCD ở chế độ kích hoạt trình điều khiển LCD (Driver Enable) và đặt tốc độ nhấp nháy của LCD (Blink rate)

3.2.8. *Khởi tạo I2C và MMA8451*

3.2.8.1. Khởi tạo I2C

```
void I2C_Init(void) {
    //bus clock is 24/3 = 8MHz
    SIM->CLKDIV1 |= (1u<<17) | (1u<<16);

    //clock to PTE24 and PTE25 for I2C0
    SIM->SCGC5 |= SIM_SCGC5_PORTE_MASK;
    //clock to I2C0
    SIM->SCGC4 |= SIM_SCGC4_I2C0_MASK;

    PORTE->PCR[24] = PORT_PCR_MUX(5);
    PORTE->PCR[25] = PORT_PCR_MUX(5);

    I2C0->F = 0x80; //mult=2h ICR=00h

    I2C0->C1 = 0xB0; //10110000 - module enable, interrupt disable, master, transmit,

    //acknowledge bit sent,repeated start off, wake up off, DMA off
    I2C0->C2 = 0x00;
}
```

Hình 17: Khởi tạo I2C

Hàm I2C_Init() được sử dụng để cấu hình và kích hoạt module I2C trên vi điều khiển. Nó thiết lập tần số bus I2C, chọn chân PTE24 và PTE25 làm chân I2C, và cấu hình các thanh ghi điều khiển để đặt chế độ master và tốc độ truyền dữ liệu.

3.2.8.2. Khởi tạo MMA8451

```
void MMA8451_Init(void) {
  // write bit 1 to CTRL_REG1 to change from standby to active mode.
    I2C_WriteRegister(MMA8451Q_address, CTRL_REG1, 0x1);
}
```

Hình 18: Khởi tạo MMA8451

Đề khởi tạo MMA8451 , ghi 0x01 vào thanh ghi CTRL_REG1 của cảm biến để chuyển từ trạng thái standby sang trạng thái active.

3.2.9. *Tính toán SVM (Signal Vector Magnitude)*

Signal Vector Magnitude (SVM) cung cấp giá trị đo lường về cường độ chuyển động thu được từ cảm biến gia tốc. Cú ngã được xác định bằng cách giám sát sự thay đổi về độ lớn của gia tốc, bởi vì nó sẽ giảm rất nhanh trong quá trình ngã và trước khi va chạm. Khi ngã, vector gia tốc của cơ thể sẽ cùng hướng với vector trọng trường.

Công thức tính SVM: $\sqrt{x^2 + y^2 + z^2}$

```
float SVM(Data data) {
  return sqrt(data.x * data.x + data.y * data.y + data.z * data.z);
}
```

Hình 19: Hàm tính toán SVM

4. Kiểm chứng (Validation)

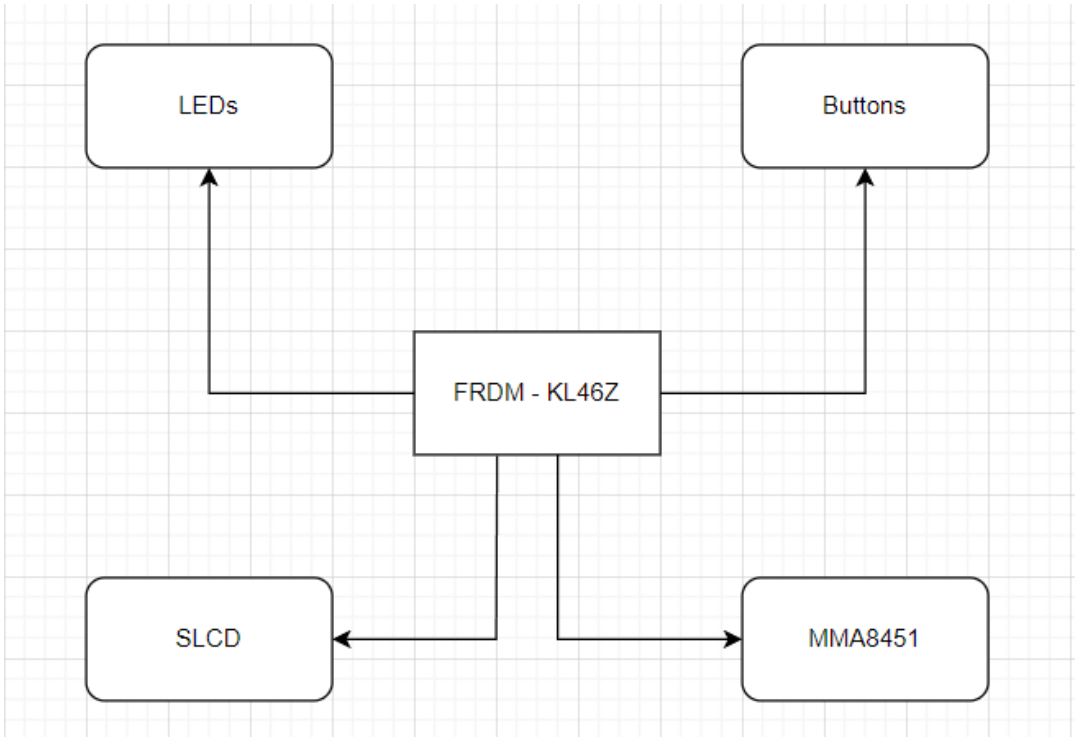
Video demo

5. Kết luận (Conclusion)

Hệ thống đã đáp ứng được một số yêu cầu như:

- Sử dụng được cảm biến MMA8451 để phát hiện ngã
- Các LED đã hoạt động (nháy) theo yêu cầu mong muốn
- Đọc và hiển thị trạng thái của hệ thống trên LCD

Appendix A: Schematic



Hình 20: Tóm tắt sơ đồ mạch

Appendix B: Code

FallDetectionSystem

Danh sách hình (List of Figures)

Danh sách Bảng (List of Tables)

References

[1] KL46 Sub-Family Reference Manual

[2] MMA8451Q Datasheet

[3] FRDM – KL46Z Schematic

[4] LUMEX LCD-S401M16KR Datasheet

[5] A design of Low power wearable system for pre-fall detection (Neeraj R. Rathi)