ĐẠI HỌC QUỐC GIA, THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN - ĐIỆN TỬ


# THIẾT KẾ HỆ THỐNG NHÚNG

---

## BÁO CÁO BÀI TẬP LỚN

# BASIC MUSIC

---

GVHD: THẦY BÙI QUỐC BẢO
danh sách thành viên : Võ Phùng Ngọc Khôi 2311710
Trần Ngọc Khiêm 2311570

TP Hồ Chí Minh, Tháng 11 Năm 2025

MỤC LỤC

## I.TỔNG QUAN VỀ DỰ ÁN

Thiết kế một máy chơi nhạc cơ bản có mười hai nút nhấn tám nút nhấn tương ứng phát ra tám nốt nhạc là đồ rê mi pha son la si đố, còn bốn nút còn lại là một nút khởi động, nút start/ stop, nút chuyển chế độ, nút chuyển bản nhạc phát hoặc chọn nơi để lưu vào, thiết bị có ba chế độ hoạt động thứ nhất là chế độ chơi nhạc khi nhấn một trong các nút bấm thì phát ra tiếng tương ứng với các nút nhấn, nhấn nút chuyển chế độ để chuyển chế độ ghi âm sau đó nhấn thêm một lần nữa thì sẽ là chế độ phát nhạc thêm 1 lần nữa sẽ là chế độ chơi nhạc ban đầu, còn nút nhấn chuyển bản nhạc thì chuyển từ một đến 10 bản nhạc. Khi ở chế độ ghi âm bài nhạc nhấn nút start để bắt đầu chế độ ta nhấn một trong tám nút nhấn sau đó sau khi nhấn xong bài nhạc cần lưu thì nhấn tiếp nút start để dừng ghi âm. Còn ở chế độ phát thì chọn bản nhạc cần phát sau đó nhấn nút start để phát và nhấn tiếp nút start để dừng phát. Màn hình LCD thì hiển thị chế độ hiện tại và bài nhạc đang chọn. LCD Hiển thị “xin chào” khi mới khởi động.

### 1. thiết kế

LCD giao tiếp bốn bit và các chân điều khiển là RS RW E nên cần thêm bảy chân gpio

Phát xung pwm để phát ra passive buzzer để phát ra tiếng cần một chân gpio

Có tổng cộng mười hai nút nhấn nên cần mười hai chân gpio

Chọn vi xử lý có ít nhất hai mươi chân gpio

Chọn kit phát triển stm32f103c8t6, chân pb3 đến pb 9 sử dụng điều khiển LCD, chân pa8 pa9 pa10 pa11 pb12 pb13 pb14 pb15 pa2 pa3 pb1 pb0 để điều khiển 12 nút nhấn, và chân pa1 sử dụng timer2 để xuất xung pwm ra loa. Lcd chọn lcd loại 16x2 có sử dụng biến trở để điều chỉnh độ sang của chữ hiển thị, loa chọn passive buzzer để có thể điều chỉnh âm thanh phát ra dựa vào tần số, mười hai nút nhấn chọn nút nhấn bốn chân. Cấp nguồn bằng adaptor 12 vol thay cho bin sử dụng các tụ và 2 ic hạ áp từ 12v xuống 5v và từ 5v sang 3v3.

## II.SYSTEM REQUIREMENT

### 1. Functional  requirement

#### 1.1. Nút nhấn có khả năng hoạt động tốt

FR1.1.1. Thời gian delay kể từ khi nhấn nút nhấn đến khi phát ra nốt nhạc không quá 50 ms

FR1.1.2. Thời gian trễ từ khi nhả nút nhấn đến khi ngừng phát ra nốt nhạc không quá 1s

#### 1.2. Loa hoạt động tốt

FR1.2.1. Loa có khả năng phát ra nốt nhạc với tần số tương ứng lệch không quá 2%.

#### 1.3 Thiết bị có khả năng lưu trữ bản nhạc được chơi

FR1.3.1. Lưu trữ được tối đa 10 bài nhạc (mỗi bài tối đa 64 nốt nhạc)

FR1.3.2. Có khả năng chuyển sang chế độ lưu trữ nhạc

FR1.3.3. Có khả năng chọn slot nhạc được lưu trữ

FR1.3.4. Có khả năng tùy chọn việc bắt đầu và dừng ghi

#### 1.4 Thiết bị có khả năng phát bản nhạc được lưu trữ hoặc ghi

FR1.4.1. Có khả năng chuyển sang chế độ phát nhạc

FR1.4.2. Có khả năng chọn slot nhạc được lưu để phát

FR1.4.3. Có khả năng tùy chọn việc bắt đầu và dừng phát

#### 1.5 Thiết bị có khả năng hiển thị thông tin trên màn hình LCD

FR1.5.1. Thiết bị có màn hình LCD 16x2

FR1.5.2. Độ sáng màn hình LCD điều chỉnh được

FR1.5.3. Màn hình LCD hiển thị “hello” sau khi nhấn nút Power

FR1.5.4. Màn hình LCD hiển thị tên chế độ đang hoạt động

**2.Non functional requirement**

NFR2.1. Kích thước thiết bị phải nhỏ hơn 30x10x5cm

NFR2.2. Thiết bị phải nhẹ hơn 0.5kg

NFR2.4. Thiết bị phải dễ dàng trong việc sử dụng

## III.USECASE MODELLING

### 1. Usecase table

| **Use case ID** | **Description** |
|---|---|
| **UC1** | Chơi nhạc (normal) |
| **UC2** | Ghi bản nhạc (record) |
| **UC3** | Phát bản nhạc được lưu trữ(playback) |

### 2. Usecases

| | |
|---|---|
| Use case name | Chơi nhạc bình thường(normal) |
| Use case id | UC1 |
| Scope | Hệ thống chơi nhạc sử dụng các nút nhấn |
| Primary actor | Người dùng |

| | |
|---|---|
| Stakeholders and interests | Người dùng mong muốn chơi nhạc, điều khiển bằng các nút nhấn |
| Preconditions | Thiết bị được cấp nguồn<br>Hệ thống ở chế độ chơi nhạc bình thường, không ở trong chế độ ghi hay phát bản nhạc<br>Các nút nhấn hoạt động tốt<br>Loa hoạt động tốt |
| Postconditions | Thiết bị phát ra nốt nhạc tương ứng với nút nhấn được nhấn |
| Main flow of event | 1.Người dùng nhấn nút Power để khởi động<br>2. Màn hình LCD hiển thị "Xin chào"<br>3. Màn hình LCD hiển thị "chế độ: normal"<br>4. Người dùng nhấn nút nhấn để chơi nhạc<br>5. Hệ thống phát âm thanh ra loa theo nút nhấn. |
| Alternative Flow | Không có phím nào được nhấn sau 30s |
| Exception Flow | Nếu loa không hoạt động → hệ thống vẫn nhận nút nhấn nhưng không có âm thanh. |
| Includes | Khởi động hệ thống<br>Hiển thị LCD |
| Extends | none |
| Special Requirements | 1. Hệ thống phải phản hồi nút nhấn trong vòng 50ms<br>2. Các nút nhấn phải được chống rung |

| | |
|---|---|
| Assumptions | Người dùng biết được quy trình sử dụng và chức năng các nút nhấn |
| Notes | Đây là chế độ mặc định khi khởi động. |
| Author | Võ Phùng Ngọc Khôi và Trần Ngọc Khiêm |
| Dates | 2\10\2025 |

| | |
|---|---|
| Use case name | Ghi bản nhạc(record) |
| Use case id | UC2 |
| Scope | Hệ thống ghi lại bản nhạc |
| Primary actor | Người dùng |
| Stakeholders and interests | Người dùng: mong muốn lưu trữ bài nhạc để có thể phát lại |
| Preconditions | Thiết bị được cấp nguồn<br>Hệ thống ở chế độ ghi bản nhạc, không ở trong chế độ chơi nhạc bình thường hay phát bản nhạc |

| | |
|---|---|
| | Các nút nhấn hoạt động tốt<br>Loa hoạt động tốt |
| Postconditions | Thiết bị lưu trữ chính xác bản nhạc vừa chơi |
| Main flow of event | 1.Người dùng nhấn nút Power để khởi động<br>2. Màn hình LCD hiển thị "Xin chào".<br>3. Người dùng nhấn nút Change mode để chuyển sang mode ghi nhạc<br>4. Màn hình LCD hiển thị "chế độ: record" và slot nhạc đang chọn<br>5. Người dùng nhấn nút Change slot để chọn slot nhạc.<br>6. Người dùng nhấn nút Start/stop để bắt đầu ghi.<br>7. Người dùng đánh bản nhạc cần ghi<br>8. Hệ thống vừa phát âm thanh, vừa lưu lại bản nhạc.<br>9. Người dùng nhấn nút nút Start/stop để dừng ghi nhạc |
| Alternative Flow | none |
| Exception Flow | Người dùng nhấn nút Power khi đang ghi → bản nhạc không được lưu.<br>Bộ nhớ đầy → hệ thống báo lỗi trên LCD. |
| Includes | Khởi động hệ thống<br>Hiển thị LCD |

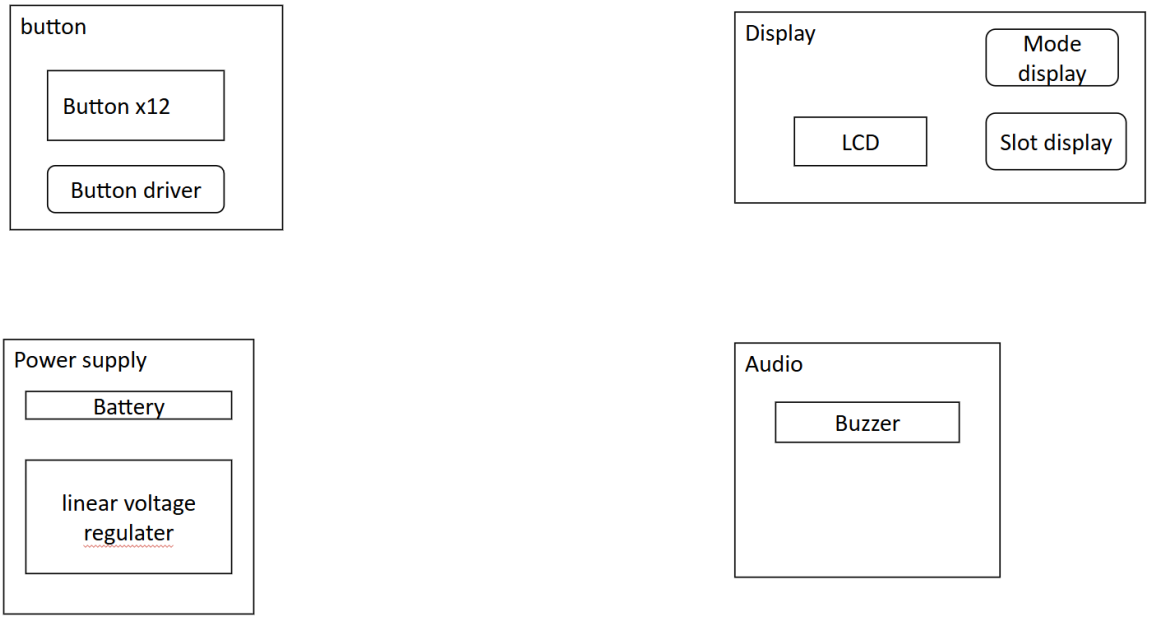
| | |
|---|---|
| Extends | Có thể mở rộng thêm “chọn vị trí lưu” hoặc “đặt tên bản nhạc”. |
| Special Requirements | Ghi được tối thiểu 64 nốt nhạc. |
| Assumptions | none |
| Notes | Mỗi lần chỉ ghi được một bản nhạc tại một slot nhớ. |
| Author | Võ Phùng Ngọc Khôi và Trần Ngọc Khiêm |
| Dates | 2\10\2025 |

| | |
|---|---|
| Use case name | Phát bản nhạc được ghi lại(playback) |
| Use case id | UC3 |
| Scope | Hệ thống phát bản nhạc được lưu trữ trong bộ nhớ |
| Primary actor | Người dùng |
| Stakeholders and interests | Người dùng: mong muốn phát bài nhạc đã ghi, hệ thống phải phát chính xác bản nhạc được lưu |
| preconditions | Thiết bị được cấp nguồn<br>Hệ thống ở chế độ phát bản nhạc, không trong chế độ chơi nhạc bình thường hay ghi bản nhạc |

| | |
|---|---|
| | Các nút nhấn hoạt động tốt<br>Loa hoạt động tốt<br>Đã có ít nhất một bản nhạc được ghi( hoặc lưu) |
| postconditions | Bản nhạc được phát chính xác ra loa |
| Main flow of event | 1.Người dùng nhấn nút Power để khởi động<br>2. Màn hình LCD hiển thị “Xin chào”<br>3. Người dùng nhấn nút “change mode” chuyển đổi mode để chuyển sang mode phát nhạc<br>5. Màn hình LCD hiển thị “chế độ: playback”<br>6. Người dùng nhấn nút “change slot” để chọn bản nhạc cần phát<br>7. Người dùng nhấn nút start để bắt đầu phát nhạc<br>8. Hệ thống phát nhạc qua loa.<br>9. Người dùng có thể nhấn nút stop để dừng phát |
| Alternative Flow | Người dùng có thể chọn bản nhạc khác để phát ngay sau khi dừng. |
| Exception Flow | Nếu chưa có bản nhạc nào trong bộ nhớ → hệ thống báo “Không có dữ liệu”.<br>Nếu loa hỏng → không phát ra âm thanh. |
| Includes | Khởi động hệ thống<br>Hiển thị LCD |
| Extends | Có thể mở rộng thêm tính năng “phát lặp lại” hoặc “tua nhanh” |

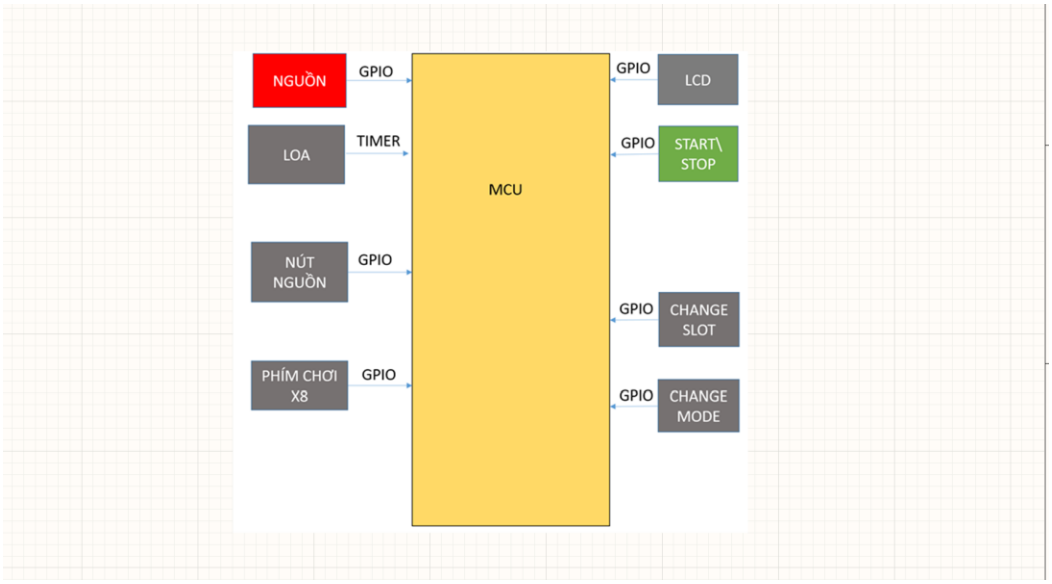
| | |
|---|---|
| Special Requirements | Đảm bảo phát nhạc đúng thứ tự nốt nhạc đã lưu. |
| Assumptions | Người dùng đã ghi ít nhất một bản nhạc trước đó. |
| Notes | Hệ thống nên hiển thị số thứ tự bản nhạc trên LCD. |
| Author | Võ Phùng Ngọc Khôi và Trần Ngọc Khiêm |
| Dates | 2\10\2025 |

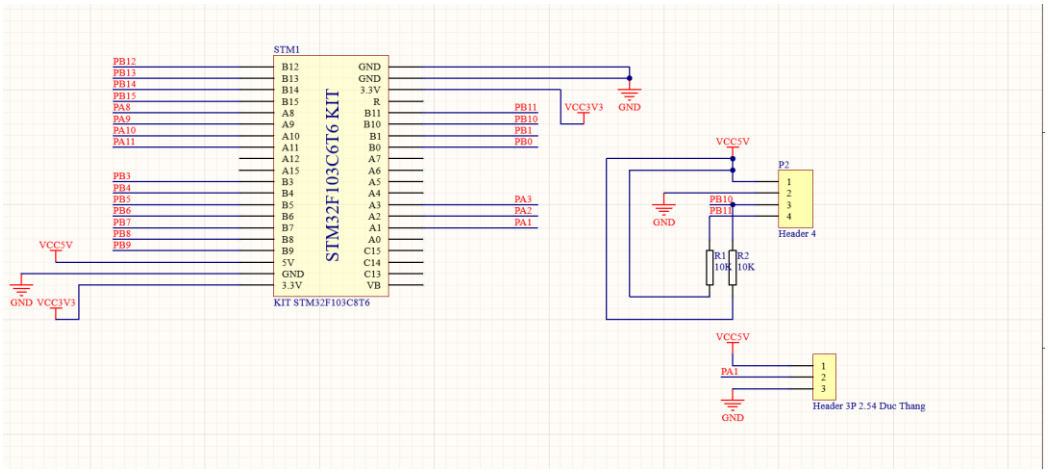
## VI. System architecture

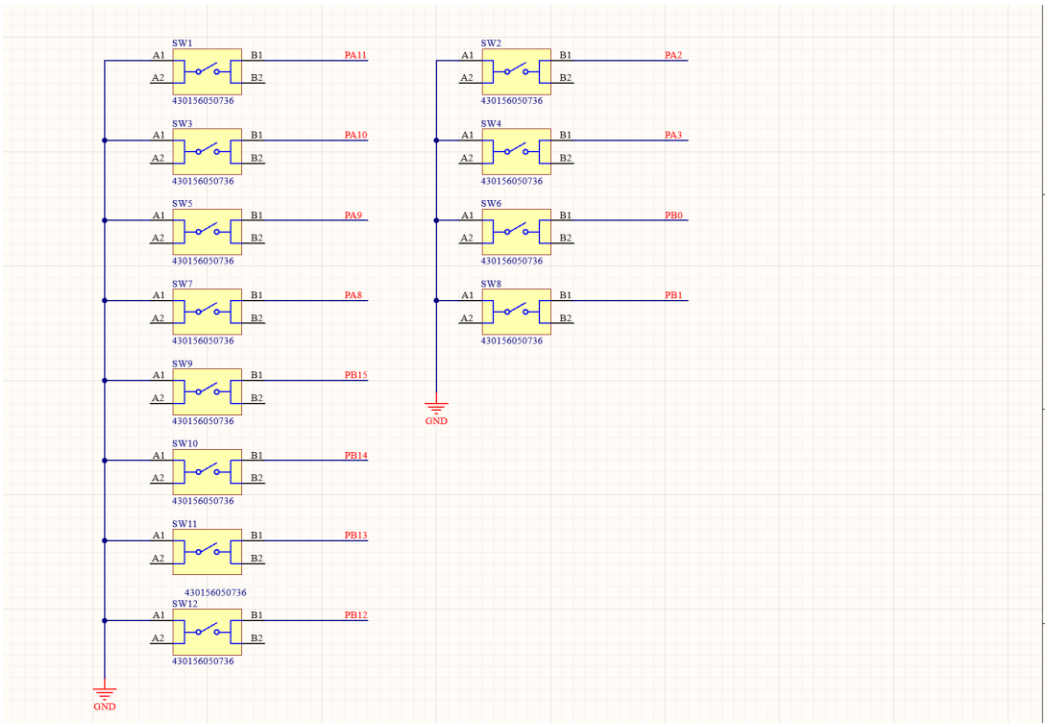


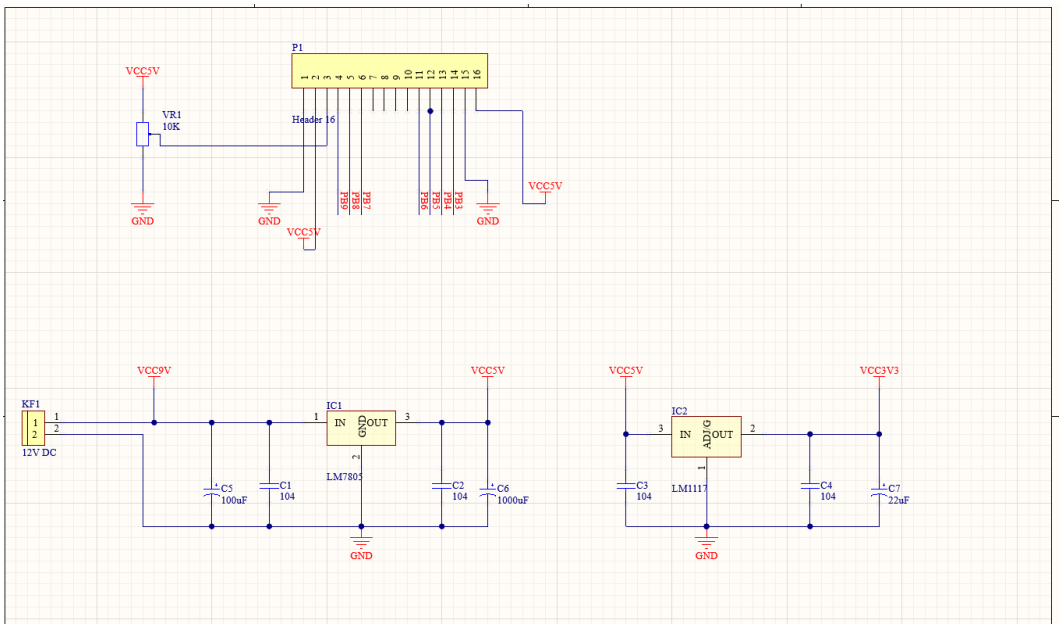

## V.SƠ ĐỒ NGUYÊN LÝ VÀ PCB

### 1. SCHEMATIC

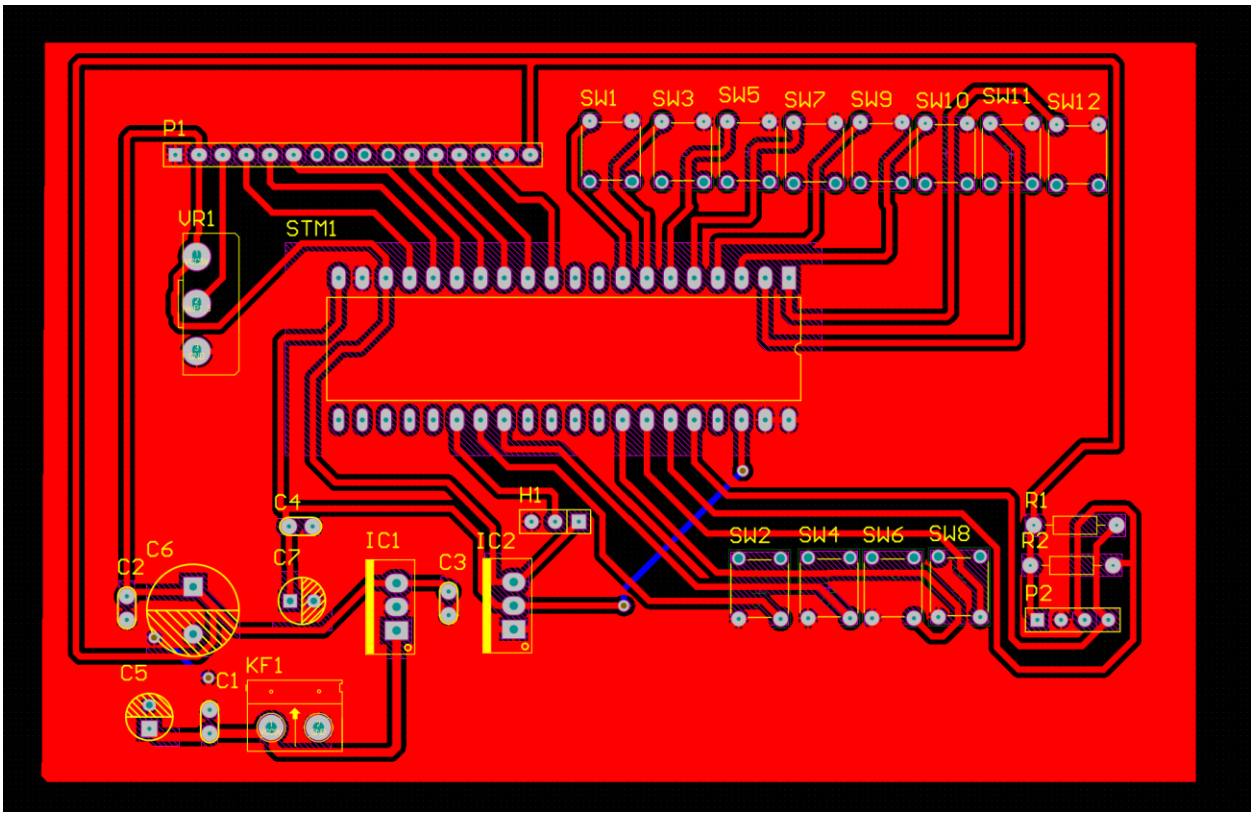

NGUỒN
GPIO
LOA
TIMER
NÚT NGUỒN
GPIO
PHÍM CHƠI X8
GPIO
MCU
GPIO
LCD
GPIO
START\ STOP
GPIO
CHANGE SLOT
GPIO
CHANGE MODE

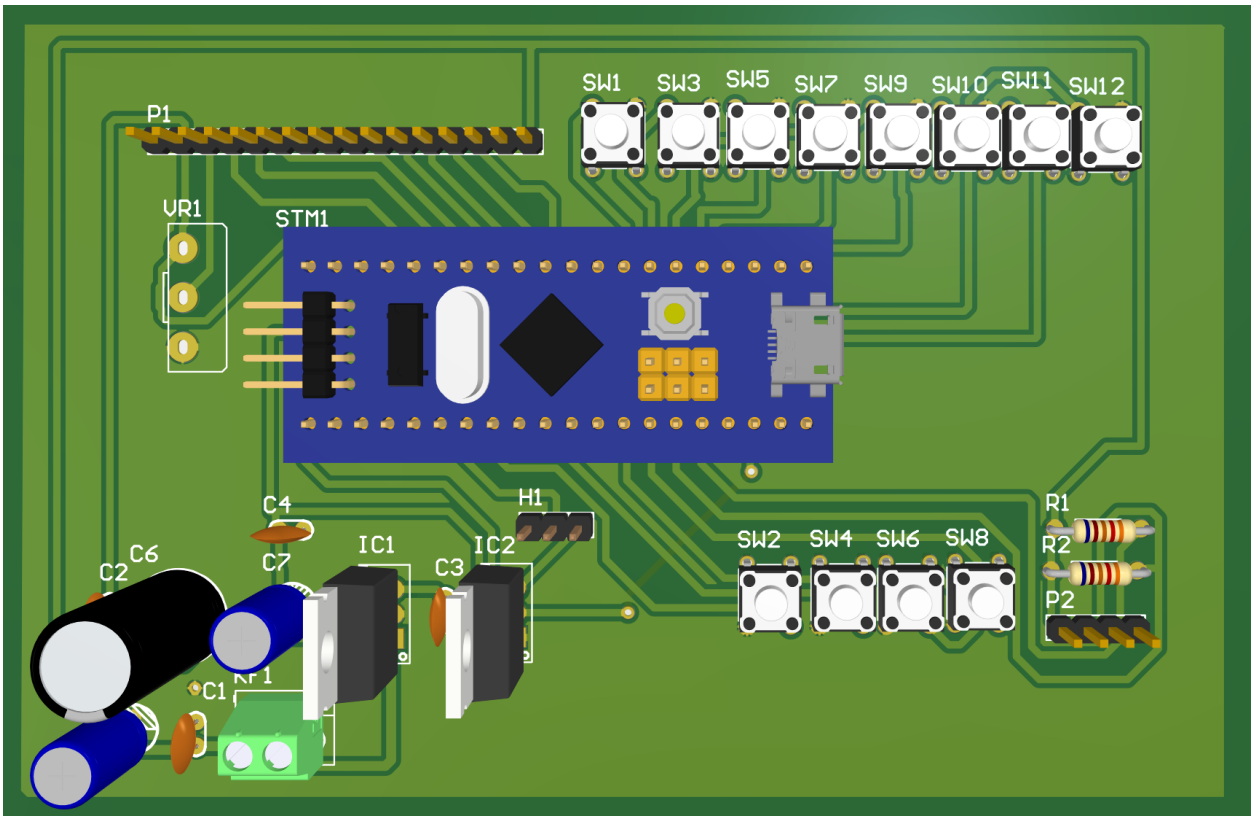

STM1
STM32F103C6T6 KIT
KIT STM32F103C8T6
VCC5V
GND
VCC3V3
P2
Header 4
R1 10K
R2 10K
PA1
Header 3P 2.54 Duc Thang


SW1
SW2
SW3
SW4
SW5
SW6
SW7
SW8
SW9
SW10
SW11
SW12
A1
A2
B1
B2
430156050736
PA11
PA2
PA10
PA3
PA9
PB0
PA8
PB1
PB15
PB14
PB13
PB12
GND


P1
Header 16
VCC5V
VR1
10K
GND
PB7
PB6
PB5
PB4
PB3
VCC9V
KF1
12V DC
IC1
LM7805
IN
GND
OUT
C5
100uF
C1
104
C2
104
C6
1000uF
IC2
ADJG
LM1117
VCC3V3
C3
104
C4
104
C7
22uF

## 2.PCB


SW1
SW3
SW5
SW7
SW9
SW10
SW11
SW12
P1
VR1
STM1
C4
H1
R1
C6
C7
IC1
IC2
C3
SW2
SW4
SW6
SW8
R2
C2
P2
C5
C1
KF1


SW1
SW3
SW5
SW7
SW9
SW10
SW11
SW12
P1
VR1
STM1
C4
H1
R1
C6
C7
IC1
IC2
C3
SW2
SW4
SW6
SW8
R2
C2
P2
C1

## VI.CODE

## 1.LCD:

lcd_4bit.c:

```
#include "lcd_4bit.h"
#include <stdio.h>
// Khai báo bien toàn cuc cho LCD_Display
extern uint8_t mode;
extern uint8_t song;

void LCD_Display(void)
{
    char line1[16];
    char line2[16];
    LCD_Clear();
    switch (mode)
    {
        case 1: sprintf(line1, "Mode: Choi nhac"); break;
        case 2: sprintf(line1, "Mode: Ghi am"); break;
        case 3: sprintf(line1, "Mode: Phat nhac"); break;
    }
    LCD_GotoXY(0, 0);
    LCD_Puts(line1);
            switch (mode)
    {
        case 1: sprintf(line2, " "); break;
        case 2: sprintf(line2, "Bai: %d", song); break;
        case 3: sprintf(line2, "Bai: %d", song); break;
```

```
    }
    LCD_GotoXY(1, 0);
    LCD_Puts(line2);
}
void LCD_Send4Bits(uint8_t data)
{
    HAL_GPIO_WritePin(GPIOB, LCD_D4_Pin, (data >> 0) & 0x01);
    HAL_GPIO_WritePin(GPIOB, LCD_D5_Pin, (data >> 1) & 0x01);
    HAL_GPIO_WritePin(GPIOB, LCD_D6_Pin, (data >> 2) & 0x01);
    HAL_GPIO_WritePin(GPIOB, LCD_D7_Pin, (data >> 3) & 0x01);
}

void LCD_Enable(void)
{
    HAL_GPIO_WritePin(GPIOB, LCD_E_Pin, GPIO_PIN_SET);
    HAL_Delay(1);
    HAL_GPIO_WritePin(GPIOB, LCD_E_Pin, GPIO_PIN_RESET);
    HAL_Delay(1);
}

void LCD_Command(uint8_t cmd)
{
    HAL_GPIO_WritePin(GPIOB, LCD_RS_Pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB, LCD_RW_Pin, GPIO_PIN_RESET);
    LCD_Send4Bits(cmd >> 4);
    LCD_Enable();
    LCD_Send4Bits(cmd & 0x0F);
    LCD_Enable();
```

```
    HAL_Delay(2);
}

void LCD_Data(uint8_t data)
{
    HAL_GPIO_WritePin(GPIOB, LCD_RS_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOB, LCD_RW_Pin, GPIO_PIN_RESET);
    LCD_Send4Bits(data >> 4);
    LCD_Enable();
    LCD_Send4Bits(data & 0x0F);
    LCD_Enable();
    HAL_Delay(2);
}

void LCD_Init(void)
{
    HAL_Delay(50);
    LCD_Send4Bits(0x03);
    LCD_Enable();
    HAL_Delay(5);
    LCD_Send4Bits(0x02);
    LCD_Enable();

    LCD_Command(0x28); // 4-bit, 2 dong, font 5x8
    LCD_Command(0x0C); // hien thi, con tro
    LCD_Command(0x06);
    LCD_Command(0x01);
    HAL_Delay(5);
```

```
}
void LCD_Clear(void)
{
   LCD_Command(0x01);
   HAL_Delay(2);
}

void LCD_GotoXY(uint8_t row, uint8_t col)
{
   uint8_t address = (row == 0) ? (0x80 + col) : (0xC0 + col);
   LCD_Command(address);
}
void LCD_Puts(char *str)
{
   while (*str)
   {
      LCD_Data(*str++);
   }
}
```

lcd_4bit.h:

```
#ifndef __LCD_4BIT_H
#define __LCD_4BIT_H
#include "stm32f1xx_hal.h"
// Đ?nh nghia Pin LCD
#define LCD_RS_Pin GPIO_PIN_9
#define LCD_RW_Pin GPIO_PIN_8
#define LCD_E_Pin  GPIO_PIN_7
```

```
#define LCD_D4_Pin GPIO_PIN_6
#define LCD_D5_Pin GPIO_PIN_5
#define LCD_D6_Pin GPIO_PIN_4
#define LCD_D7_Pin GPIO_PIN_3

// Khai báo các hàm LCD
void LCD_Init(void);
void LCD_Command(uint8_t cmd);
void LCD_Data(uint8_t data);
void LCD_Enable(void);
void LCD_Send4Bits(uint8_t data);
void LCD_Clear(void);
void LCD_GotoXY(uint8_t row, uint8_t col);
void LCD_Puts(char *str);
void LCD_Display(void);
#endif /* __LCD_4BIT_H */
```

**2 CẦU HÌNH CHÂN VÀ TIMER**

mcu_config.c:

```
#include "mcu_config.h"
// Đ?nh nghia Handle Timer 2
TIM_HandleTypeDef htim2;
/**
  * @brief System Clock Configuration (CODE G?C)
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
```

```
RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

/** Initializes the RCC Oscillators according to the specified parameters
* in the RCC_OscInitTypeDef structure.
*/
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
RCC_OscInitStruct.HSIState = RCC_HSI_ON;
RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI_DIV2;
RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL16;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
  Error_Handler();
}

/** Initializes the CPU, AHB and APB buses clocks
*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
{
  Error_Handler();
```

```
  }
}

/**
  * @brief TIM2 Initialization Function (CODE G?C)
  * @param None
  * @retval None
  */
void MX_TIM2_Init(void)
{

  /* USER CODE BEGIN TIM2_Init 0 */

  /* USER CODE END TIM2_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  /* USER CODE BEGIN TIM2_Init 1 */

  /* USER CODE END TIM2_Init 1 */
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 127;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 20;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
```

```
if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
{
  Error_Handler();
}
sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
{
  Error_Handler();
}
if (HAL_TIM_PWM_Init(&htim2) != HAL_OK)
{
  Error_Handler();
}
sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
{
  Error_Handler();
}
sConfigOC.OCMode = TIM_OCMODE_PWM1;
sConfigOC.Pulse = 0;
sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
if (HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_2) != HAL_OK)
{
  Error_Handler();
}
/* USER CODE BEGIN TIM2_Init 2 */
```

```
  /* USER CODE END TIM2_Init 2 */

}

/**
  * @brief GPIO Initialization Function (CODE G?C)
  * @param None
  * @retval None
  */
void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  /* USER CODE BEGIN MX_GPIO_Init_1 */

  /* USER CODE END MX_GPIO_Init_1 */

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6
                          |GPIO_PIN_7|GPIO_PIN_8|GPIO_PIN_9, GPIO_PIN_RESET);

  /*Configure GPIO pins : PA2 PA3 PA8 PA9
                           PA10 PA11 */
  GPIO_InitStruct.Pin = GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_8|GPIO_PIN_9
```

```
                    |GPIO_PIN_10|GPIO_PIN_11;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : PB0 PB1 PB12 PB13
                    PB14 PB15 */
  GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_12|GPIO_PIN_13
                    |GPIO_PIN_14|GPIO_PIN_15;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pins : PB3 PB4 PB5 PB6
                    PB7 PB8 PB9 */
  GPIO_InitStruct.Pin = GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6
                    |GPIO_PIN_7|GPIO_PIN_8|GPIO_PIN_9;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /* USER CODE BEGIN MX_GPIO_Init_2 */

  /* USER CODE END MX_GPIO_Init_2 */
}

/**
```

```
  * @brief  This function is executed in case of error occurrence. (CODE G?C)
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}
```

mcu_config.h:

```
#ifndef __MCU_CONFIG_H
#define __MCU_CONFIG_H
#include "stm32f1xx_hal.h"
// Khai báo Handle Timer
extern TIM_HandleTypeDef htim2;
void SystemClock_Config(void);
void Error_Handler(void);
void MCU_Initialize(void);
#endif /* __MCU_CONFIG_H */
```

**3 LƯU BẢN NHẠC**

song_manager.c:

```
#include "song_manager.h"
#include <string.h>
uint16_t song_data[11][MAX_NOTES] = {0};
```

```
uint8_t song_len[11] = {0};
uint16_t record_buf[MAX_NOTES] = {0};
uint8_t record_len = 0;
uint8_t isRecording = 0;
uint8_t isPlaying = 0;
uint8_t play_index = 0;
void SongManager_SaveRecord(uint8_t song_slot)
{
    song_len[song_slot] = record_len;
    for (int i = 0; i < record_len; i++) {
        song_data[song_slot][i] = record_buf[i];
    }
    record_len = 0; // Reset buffer
}
```

song_manager.h:

```
#ifndef __SONG_MANAGER_H
#define __SONG_MANAGER_H
#include "stm32f1xx_hal.h"
#define MAX_NOTES 128
extern uint16_t song_data[11][MAX_NOTES];
extern uint8_t song_len[11];
extern uint16_t record_buf[MAX_NOTES];
extern uint8_t record_len;
extern uint8_t isRecording;
extern uint8_t isPlaying;
extern uint8_t play_index;
void SongManager_SaveRecord(uint8_t song_slot);
#endif /* __SONG_MANAGER_H */
```

**4 ĐỌC NÚT NHẤN PHÁT RA NỐT NHẠC**

keypad_notes.c:

```
#include "keypad_notes.h"

uint32_t ReadNoteKey(void)

{

   if(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_12) == GPIO_PIN_RESET) return NOTE_C4;

   if(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_13) == GPIO_PIN_RESET) return NOTE_D4;

   if(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_14) == GPIO_PIN_RESET) return NOTE_E4;

   if(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_15) == GPIO_PIN_RESET) return NOTE_F4;

   if(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_8) == GPIO_PIN_RESET) return NOTE_G4;

   if(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_9) == GPIO_PIN_RESET) return NOTE_A4;

   if(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_10) == GPIO_PIN_RESET) return NOTE_B4;

   if(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_11) == GPIO_PIN_RESET) return NOTE_C5;


   return 0; // Khong nhan nut nao

}
```

keypad_notes.h:

```
#ifndef __KEYPAD_NOTES_H

#define __KEYPAD_NOTES_H

#include "stm32f1xx_hal.h"

#define NOTE_C4    262

#define NOTE_D4    294

#define NOTE_E4    330

#define NOTE_F4    349

#define NOTE_G4    392

#define NOTE_A4    440

#define NOTE_B4    494

#define NOTE_C5    523
```

```
uint32_t ReadNoteKey(void);
#define IS_BUTTON_PRESSED(PORT, PIN) (HAL_GPIO_ReadPin(PORT, PIN) == GPIO_PIN_RESET)
#endif /* __KEYPAD_NOTES_H */
```

pwm_buzzer.c:

```
#include "pwm_buzzer.h"
#include "mcu_config.h"
void PWM_SetFrequency(uint32_t freq)
{
   if(freq == 0)
   {
      HAL_TIM_PWM_Stop(&htim2, TIM_CHANNEL_2);
      return;
   }
   uint32_t PSC = 72 - 1;       // 72MHz / 72 = 1MHz
   uint32_t ARR = 1000000 / freq;
   htim2.Instance->PSC  = PSC;
   htim2.Instance->ARR  = ARR;
   htim2.Instance->CCR2 = ARR / 2; // Duty 50%
   HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_2);
}
```

pwm_buzzer.h:

```
#ifndef __PWM_BUZZER_H
#define __PWM_BUZZER_H
#include "stm32f1xx_hal.h"
void PWM_SetFrequency(uint32_t freq);
#endif /* __PWM_BUZZER_H */
```

**5 HÀM CHÍNH**

Main.c:

```
/* USER CODE BEGIN Header */
/**
  * @file           : main.c
  * @brief          : Main program body
  * @attention
  */
/* USER CODE END Header */
#include "main.h"
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
#include "main.h"
#define IS_BUTTON_PRESSED(PORT, PIN) (HAL_GPIO_ReadPin(PORT, PIN) == GPIO_PIN_RESET)
uint8_t start_flag = 0;  // 0: chua chay, 1: dang chay
uint8_t mode = 1;  // 1: Choi nh?c, 2: Ghi am, 3: Ph t nh?c
uint8_t song = 1;
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{

  /* USER CODE BEGIN 1 */
            HAL_Init();
    SystemClock_Config();
```

```
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();
    __HAL_RCC_TIM2_CLK_ENABLE();

    /* =====PA1 PWM TIM2_CH2 ===== */
    GPIO_InitTypeDef gpio = {0};
    gpio.Pin   = GPIO_PIN_1;
    gpio.Mode  = GPIO_MODE_AF_PP;
    gpio.Speed = GPIO_SPEED_FREQ_HIGH;
    HAL_GPIO_Init(GPIOA, &gpio);
    /* ===== KHOI TAO TIMER PWM ===== */
    htim2.Instance = TIM2;
    htim2.Init.Prescaler     = 127;
    htim2.Init.CounterMode   = TIM_COUNTERMODE_UP;
    htim2.Init.Period        = 20;
    htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
    HAL_TIM_PWM_Init(&htim2);

    TIM_OC_InitTypeDef sConfigOC = {0};
    sConfigOC.OCMode     = TIM_OCMODE_PWM1;
    sConfigOC.Pulse      = 0; // 50%
    sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
    HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_2);

    PWM_SetFrequency(0); // ban d?u t?t ti?ng
HAL_Init();
            LCD_Clear();
 SystemClock_Config();
```

```
 MX_GPIO_Init();
  LCD_Init();
  LCD_Clear();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_TIM2_Init();
  while (1)
  {
if (start_flag == 0)
{
    if (IS_BUTTON_PRESSED(GPIOA, GPIO_PIN_2))
    {
        HAL_Delay(100);
        if (IS_BUTTON_PRESSED(GPIOA, GPIO_PIN_2))
        {
            start_flag = 1;
            LCD_Clear();
            LCD_GotoXY(0, 0);
            LCD_Puts("XIN CHAO");
            HAL_Delay(1000);
            LCD_Display();
            while (IS_BUTTON_PRESSED(GPIOA, GPIO_PIN_2));
        }
    }
    HAL_Delay(10);
    continue;
}
if (IS_BUTTON_PRESSED(GPIOB, GPIO_PIN_0))
```

```
{
    HAL_Delay(50);
    if (IS_BUTTON_PRESSED(GPIOB, GPIO_PIN_0))
    {
        PWM_SetFrequency(0);
        isRecording = 0;
        isPlaying = 0;
        mode++;
        if (mode > 3) mode = 1;
        LCD_Display();
        while (IS_BUTTON_PRESSED(GPIOB, GPIO_PIN_0));
    }
}

if (IS_BUTTON_PRESSED(GPIOB, GPIO_PIN_1))
{
    HAL_Delay(50);
    if (IS_BUTTON_PRESSED(GPIOB, GPIO_PIN_1))
    {
        song++;
        if (song > 10) song = 1;
        LCD_Display();
        while (IS_BUTTON_PRESSED(GPIOB, GPIO_PIN_1));
    }
}
if (IS_BUTTON_PRESSED(GPIOA, GPIO_PIN_3))
{
    HAL_Delay(50);
```

```
if (IS_BUTTON_PRESSED(GPIOA, GPIO_PIN_3))
{
    switch (mode)
    {
        case 2:
            if (isRecording)
            {
                isRecording = 0;
                PWM_SetFrequency(0);
                song_len[song] = record_len;
                for (int i = 0; i < record_len; i++) {
                    song_data[song][i] = record_buf[i];
                }
                record_len = 0;
                LCD_Display();
                LCD_GotoXY(1, 0);
                LCD_Puts("Da luu!");
                HAL_Delay(1000);
                LCD_Display();
            }
            else
            {
                isRecording = 1;
                record_len = 0;
                LCD_Display();
            }
            break;
```

```
            case 3:
                isPlaying = !isPlaying;
                PWM_SetFrequency(0);
                if (isPlaying) {
                    play_index = 0;
                }
                LCD_Display();
                break;
        }
        while (IS_BUTTON_PRESSED(GPIOA, GPIO_PIN_3));
    }
}
switch (mode)
{
    case 1:
    {
        uint32_t note = ReadNoteKey();
        if (note != 0)
        {
            PWM_SetFrequency(note);
            HAL_Delay(300);
        }
        else
        {
            PWM_SetFrequency(0);
        }
        break;
    }
```

```
case 2: // GHI AM
{
    uint32_t note = ReadNoteKey();
    if (note != 0)
    {
        PWM_SetFrequency(note);
        HAL_Delay(300);
        if (isRecording)
        {
            if (record_len < MAX_NOTES)
            {
                record_buf[record_len++] = note;


            }
            else
            {
                isRecording = 0;
                song_len[song] = record_len;
                for (int i = 0; i < record_len; i++) {
                    song_data[song][i] = record_buf[i];
                }
                record_len = 0;


                LCD_GotoXY(1, 0);
                LCD_Puts("FULL. Da luu!");
                HAL_Delay(1000);
                LCD_Display();
```

```
            }
        }
    }
    else
    {
        PWM_SetFrequency(0);
    }
    break;
}
case 3:
{
    if (isPlaying)
    {
        if (song_len[song] > 0)
        {
            uint16_t current_note = song_data[song][play_index];
            PWM_SetFrequency(current_note);
            HAL_Delay(300);
            PWM_SetFrequency(0);
            HAL_Delay(50);
            play_index++;
            if (play_index >= song_len[song])
            {
                play_index = 0;
            }
        }
        else
        {
```

```
            isPlaying = 0;
            PWM_SetFrequency(0);
            LCD_GotoXY(1, 0);
            LCD_Puts("Slot trong!");
            HAL_Delay(1000);
            LCD_Display();
        }
      }
      else
      {
        uint32_t note = ReadNoteKey();
        if (note != 0)
        {
          PWM_SetFrequency(note);
          HAL_Delay(300);
        }
        else
        {
          PWM_SetFrequency(0);
        }
      }
      break;
    }
}
HAL_Delay(10);



}
```

```
  /* USER CODE END 3 */
}



#ifdef USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

Main.h:

```
#ifndef __MAIN_H
#define __MAIN_H
#include "stm32f1xx_hal.h"
#include <stdio.h>
// Khai báo các module
#include "lcd_4bit.h"
#include "pwm_buzzer.h"
```

```
#include "keypad_notes.h"

#include "song_manager.h"

#include "mcu_config.h"

extern uint8_t mode;

extern uint8_t song;

void Error_Handler(void);

#endif /* __MAIN_H */
```