

BÁO CÁO ĐỒ ÁN MÔN HỌC

MÔN VI ĐIỀU KHIỂN

HK2 - NĂM HỌC: 2024-2025

ĐỒ ÁN

TỦ GỬI HÀNH LÝ (COIN LOCKER)

LỚP: 23DTV- CLC 2

Nhóm : 3

STT	Họ tên thành viên	MSSV
1	PHẠM TUẤN KHANH	23207025
2	PHẠM MINH THẮNG	23207027
3	TRẦN THÀNH THỊNH	23207029
4	NGUYỄN THANH AN	23207034

BẢNG THỐNG KÊ CẬP NHẬT TÀI LIỆU KỸ THUẬT

SỐ LẦN CẬP NHẬT BÁO CÁO VI ĐIỀU KHIỂN					
Phiên bản	Mô tả	Thành viên	Ngày	Xác nhận (Trưởng nhóm)	Ngày xác nhận
1.0	Tạo tài liệu	N.T.An	29/03/25	N.T.An	29/03/25
2.0	Cập nhật lần 1	N.T.An	30/03/25	N.T.An	30/03/25
3.0	Cập nhật lần 2	N.T.An	31/03/25	N.T.An	31/03/25
4.0	Cập nhật lần 3	P.T.Khanh	06/04/25	N.T.An	06/04/25
5.0	Cập nhật lần 4	T.T.Thịnh	19/04/25	N.T.An	20/04/25
6.0	Cập nhật lần 5	P.M.Thắng	20/04/25	N.T.An	21/04/25
7.0	Cập nhật lần 6	N.T.An	21/04/25	N.T.An	21/04/25
8.0	Cập nhật lần 7	P.T.Khanh	21/04/25	N.T.An	21/04/25
9.0	Cập nhật lần 8	T.T.Thịnh	21/04/25	N.T.An	21/04/25
10.0	Cập nhật lần 9	P.M.Thắng	21/04/25	N.T.An	21/04/25
11.0	Cập nhật lần 10	P.T.Khanh	21/04/25	N.T.An	21/04/25
12.0	Cập nhật lần 11	N.T.An	21/04/25	N.T.An	21/04/25
13.0	Cập nhật lần 12	T.T.Thịnh	24/04/25	N.T.An	24/04/25
14.0	Cập nhật lần 13	T.T.Thịnh	24/04/25	N.T.An	24/04/25
15.0	Cập nhật lần 14	T.T.Thịnh	24/04/25	N.T.An	24/04/25
16.0	Cập nhật lần 15	N.T.An	26/04/25	N.T.An	26/04/25
Vi điều khiển.DTV_CLC					

Bảng 1 Thống kê cập nhật tài liệu kỹ thuật

MỤC LỤC

DANH MỤC BẢNG.....	5
DANH MỤC HÌNH.....	5
DANH MỤC CODE.....	6
PHẦN I: GIỚI THIỆU ĐÒ ÁN	7
PHẦN II: FLOWCHART	9
PHẦN III: MÔ HÌNH MÔ PHỎNG	10
1.MÔ HÌNH PROTEUS.....	10
2. MÔ HÌNH NGOẠI VI	10
2.1 Pad 4x4	10
2.2 Khóa chốt điện Solenoid Lock LY-03.....	11
2.3 LCD TFT Touch Screen 2.8 inch ILI9341 SPI Interface	12
2.4 ESP8266 NodeMCU Lua V3 CH340	13
2.5 RFID NFC 13.56MHz RC522.....	13
2.6 Endstop Switch.....	14
PHẦN IV: MÔ HÌNH THỰC TẾ.....	15
1. HỆ THỐNG.....	15
1.1 Code hệ thống	15
a. Maincode	15
b. Giải thích main code.....	33
1.2 Code SPI ILI9341, RC522	36
a. RC522.h.....	36
b. Code RC522.c	38
c. Giải thích RC522.c	47
d. Sử dụng RC522 trong main code	50
e. Code ILI934.h	51
f. Code ILI934.c.....	52
g. Giải thích ILI9341.c.....	56
g. Các driver của ILI9341.....	58
1.3. Code UART ESP32	66
a. Code Arduino IDE.....	66
b. Giải thích Arduino code của Esp8266.....	68
c. Code google script.....	70
d. Giải thích Google Apss Script Code.....	70
e. Google script	72
f. Google sheet	72
1.4. Cấu hình hệ thống.....	73
a. Cấu hình sơ đồ chân STM32 (IOC).....	73
b. Cấu hình chân STM32.....	74
c. Cấu hình Clock STM32.....	76
2. MÔ HÌNH TỦ NHỰA.....	77
2.1. Bản vẽ	77
a. Mục đích thiết kế.....	77
b. Quy trình thiết kế.....	77
c. Thông số kỹ thuật	77
2.2. Vẽ và in 3D	78

VI ĐIỀU KHIỂN

a. Thân tủ.....	78
b. Cửa tủ.....	79
c. Tay nắm cửa	79
PHẦN V: TỦ THỰC TẾ	80
1. TỦ VÀ HOẠT ĐỘNG CỦA TỦ THỰC TẾ	80
2. HỆ THỐNG GOOGLE SHEET	81
3. LINH KIEN SỬ DỤNG.....	81
BÁO CÁO NHIỆM VỤ	82
ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH NHIỆM VỤ	83
KẾT LUẬN.....	84

DANH MỤC BẢNG

BẢNG 1 THỐNG KÊ CẬP NHẬT TÀI LIỆU KỸ THUẬT	2
BẢNG 2 LINH KIỆN.....	81
BẢNG 3 BÁO CÁO NHIỆM VỤ.....	82
BẢNG 4: ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH NHIỆM VỤ CỦA THÀNH VIÊN.....	83

DANH MỤC HÌNH

HÌNH 1.1: COIN LOCKER.....	8
HÌNH 1.2: COIN LOCKER.....	8
HÌNH 2: FLOWCHART	9
HÌNH 3.1: SƠ ĐỒ MẠCH PROTEUS.....	10
HÌNH 3.2: PAD 4X4	11
HÌNH 3.3: SƠ ĐỒ MẠCH SOLENOID	11
HÌNH 3.4: KHÓA CHÓT SOLENOID	12
HÌNH 3.5: LCD TFT TOUCH SCREEN 2.8 INCH ILI9341 SPI INTERFACE (MẶT TRƯỚC)	12
HÌNH 3.6: LCD TFT TOUCH SCREEN 2.8 INCH ILI9341 SPI INTERFACE (MẶT SAU)	13
HÌNH 3.7: ESP8266 NODEMCU LUA V3 CH340.....	13
HÌNH 3.8: RFID NFC 13.56MHZ RC522	14
HÌNH 3.9: ENDSTOP SWITCH	14
HÌNH 4.1 GOOGLE SHEET.....	73
HÌNH 4.2: SƠ ĐỒ CHÂN STM32.....	73
HÌNH 4.3: CẤU HÌNH CHÂN GPIO	74
HÌNH 4.4: CẤU HÌNH CHÂN GPIO	74
HÌNH 4.5: CẤU HÌNH CHÂN GPIO	74
HÌNH 4.6: CẤU HÌNH SPI 1	75
HÌNH 4.7: CẤU HÌNH SPI 2	75

HÌNH 4.8: CẤU HÌNH UART	76
HÌNH 4.8: CẤU HÌNH CLOCK.....	76
HÌNH 5.1: PHẦN MỀM AUTODESK FUSION	77
HÌNH 5.2: KHỚP LẮP TỦ NHỰA	78
HÌNH 5.3: THÂN TỦ	78
HÌNH 5.4: CÁNH CỦA TỦ.....	79
HÌNH 5.5: TAY NẮM CỦA.....	79
HÌNH 6.1: TỦ THỰC TẾ.....	80
HÌNH 6.2: MÃ QR XEM VIDEO	80
HÌNH 6.3: GOOGLE SHEET HIỆN TRẠNG THÁI	81
HÌNH 7: BIỂU ĐỒ TRÒN THỜI GIAN HOÀN THÀNH CÔNG VIỆC.....	82

DANH MỤC CODE

CODE 1: MAIN CODE	33
CODE 2: RFID-RC522.H.....	38
CODE 3: RFID-RC522.C.....	47
CODE 4: ILI934-FONTS.H	51
CODE 5: ILI9341_GFX.H	52
CODE 6: ILI9341_GFX.C.....	56
CODE 7: ILI9341_STM32_DRIVER.H	59
CODE 8: ILI9341_STM32_DRIVER.C.....	65
CODE 9: ESP8266	68
CODE 10: GOOGLE APPS SCRIPT CODE.....	70

PHẦN I: GIỚI THIỆU ĐỒ ÁN

Giới Thiệu Đồ Án: Hệ Thống Tủ Gửi Hành Lý Thông Minh tại Ga Tàu Điện Ngầm và Trung Tâm Thương Mại

- Tại Nhật Bản, hệ thống tủ gửi hành lý tự động (*Coin locker*) là một dịch vụ phổ biến, đặc biệt ở các nhà ga, trung tâm mua sắm và điểm du lịch. Những tủ này giúp hành khách và khách hàng gửi đồ đạc tạm thời một cách nhanh chóng, an toàn và tiện lợi, giảm bớt gánh nặng khi di chuyển.

- Các tủ gửi có nhiều kích thước khác nhau và thường được mở khóa bằng chìa cơ, mã số, thẻ IC (Suica, PASMO) hoặc mã QR. Một số hệ thống hiện đại còn hỗ trợ thanh toán điện tử và đặt trước qua ứng dụng. Nhờ vào sự tự động hóa, bảo mật cao và dễ sử dụng, tủ gửi hành lý đã trở thành một phần không thể thiếu trong đời sống đô thị Nhật Bản, mang lại trải nghiệm thoải mái hơn cho cả người dân và du khách.

- Dựa trên mô hình này, đồ án tập trung phát triển một hệ thống tủ thông minh sử dụng vi điều khiển **STM32F103C8T6**, kết hợp *công nghệ RFID, bàn phím ma trận, khóa chốt điện tử*. Hệ thống không lưu thông tin thẻ RFID mà chỉ tạo mã code cá nhân, giúp người dùng mở tủ sau này. Ngoài ra, hệ thống có cơ chế giới hạn số lần nhập sai mã, sau 3 lần nhập sai, tủ sẽ bị khóa tạm thời trong 5 phút, đảm bảo tính bảo mật cao.

- Hệ thống được thiết kế nhằm tối ưu hóa trải nghiệm người dùng, đảm bảo tính bảo mật, dễ sử dụng và linh hoạt, phù hợp với cả nhà ga và trung tâm thương mại. Đặc biệt, trong môi trường trung tâm thương mại, hệ thống giúp khách hàng thoải mái di chuyển, mua sắm mà không cần mang theo hành lý cồng kềnh. Đây là một giải pháp hiện đại, giúp nâng cao hiệu quả quản lý và cải thiện sự tiện lợi trong việc gửi hành lý tự động.

VI ĐIỀU KHIỂN

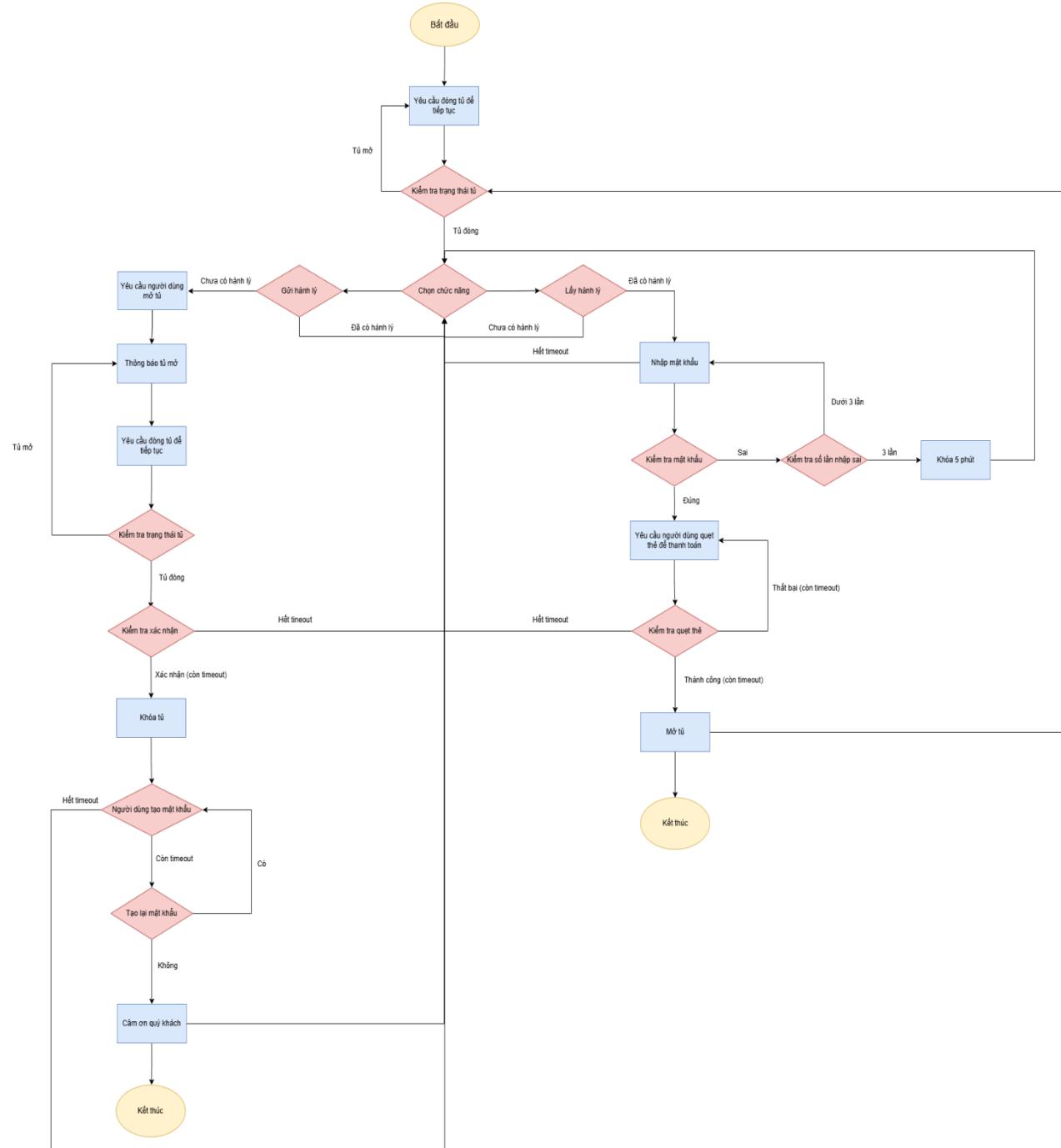


HÌNH 1.1: Coin locker



HÌNH 1.2: Coin locker

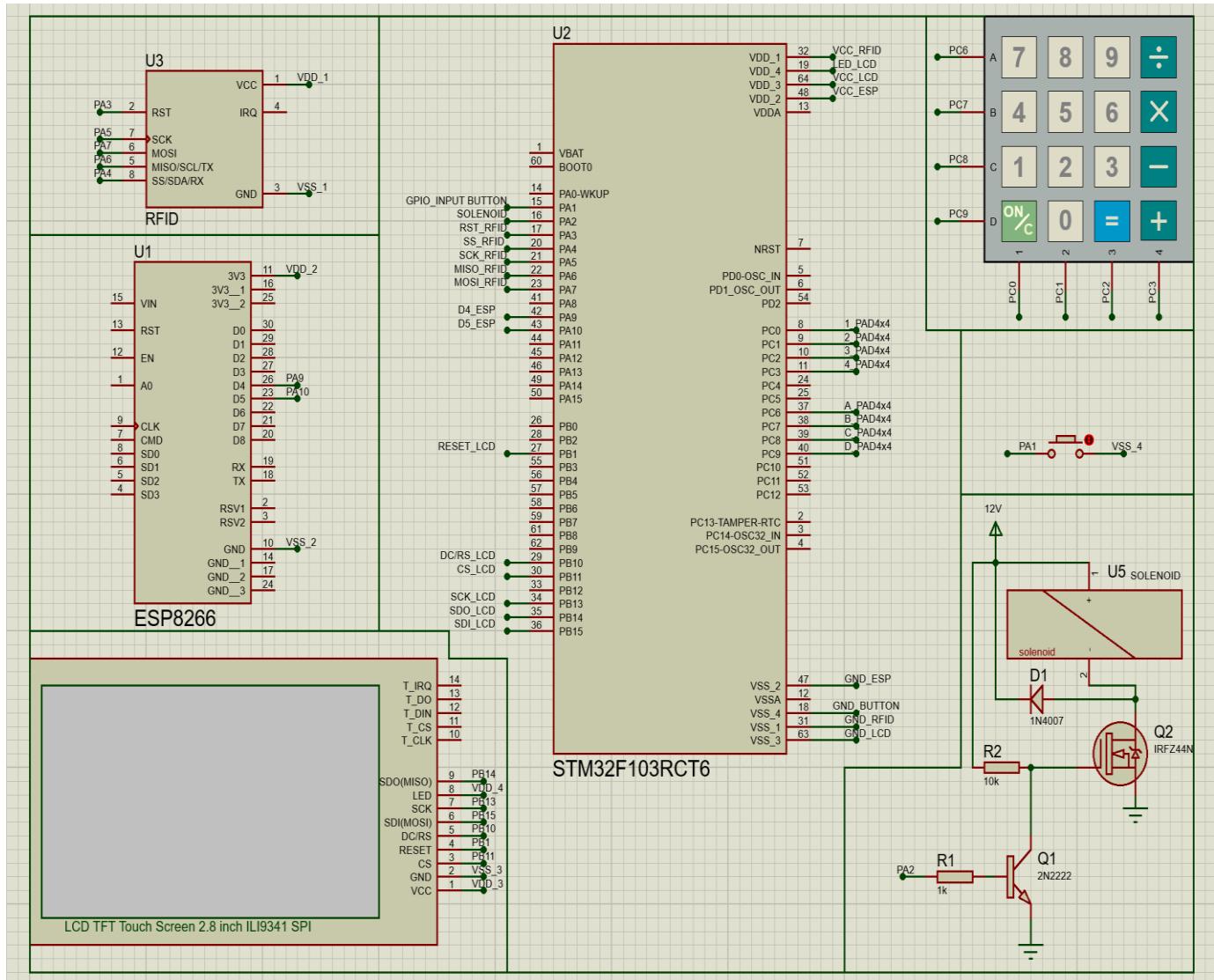
PHẦN II: FLOWCHART



HÌNH 2: Flowchart

PHẦN III: MÔ HÌNH MÔ PHỎNG

1. Mô hình Proteus

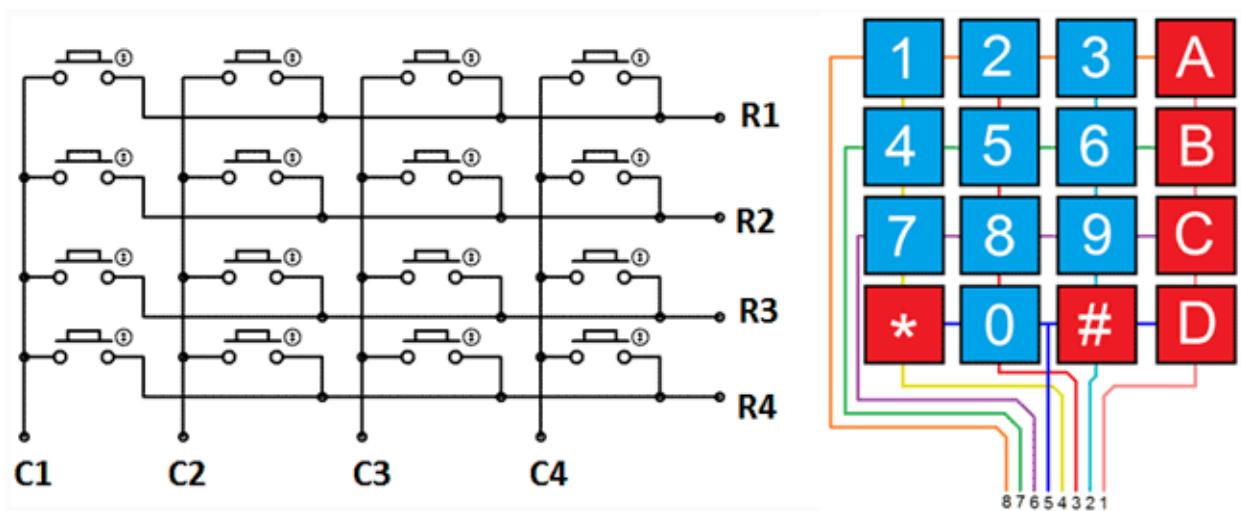


HÌNH 3.1: Sơ đồ mạch Proteus

2. Mô hình ngoại vi

2.1 Pad 4x4

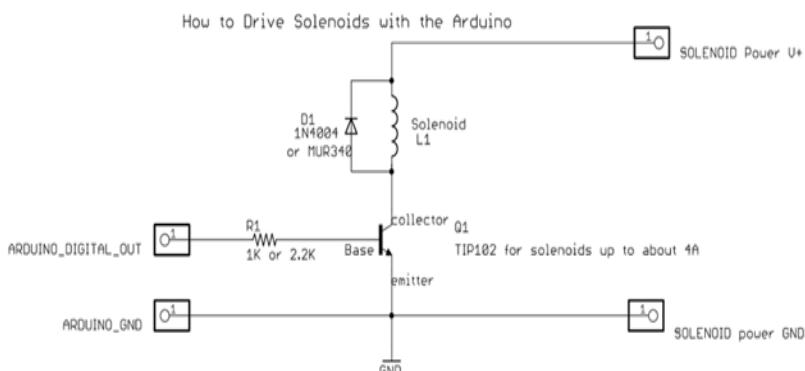
- Module bàn phím ma trận 4x4 loại phím nhựa cứng.
- Độ dài cáp: 88mm.
- Nhiệt độ hoạt động 0 ~ 70oC.
- Đầu nối ra 8 chân.
- Kích thước bàn phím 65 x 64mm



HÌNH 3.2: Pad 4x4

2.2 Khóa chốt điện Solenoid Lock LY-03

- Điện áp sử dụng: 12 / 24VDC (tùy chọn).
- Dòng điện tiêu thụ: 0.8A.
- Công suất tiêu thụ: 9.6W
- Sử dụng Solenoid từ.
- Tốc độ phản ứng: < 1s.
- Thời gian kích liên tục: < 10s
- Đi kèm gá chốt khóa.



HÌNH 3.3: Sơ đồ mạch solenoid



Hình 3.4: Khóa chốt Solenoid

2.3 LCD TFT Touch Screen 2.8 inch ILI9341 SPI Interface

- Model: LCD TFT Touch Screen 2.8 inch ILI9341 SPI Interface
- Điện áp sử dụng: 3.3~5VDC
- Điện áp giao tiếp: TTL 3.3~5VDC
- IC Driver hiển thị: ILI9341 giao tiếp SPI.
- Cỡ màn hình: 2.8 inch
- Độ phân giải: 240 x 320 pixels
- IC Driver cảm ứng: XPT2046 giao tiếp SPI
- Tích hợp khe thẻ nhớ SD giao tiếp SPI.
- Kích thước hiển thị: 46(W) X 65(H)mm
- Kích thước toàn màn hình: 86 x 50 mm



HÌNH 3.5: LCD TFT Touch Screen 2.8 inch ILI9341 SPI Interface (mặt trước)



HÌNH 3.6: LCD TFT Touch Screen 2.8 inch ILI9341 SPI Interface (mặt sau)

2.4 ESP8266 NodeMCU Lua V3 CH340

- IC chính: ESP8266 Wifi SoC.
- Phiên bản firmware: NodeMCU Lua
- Chip nạp và giao tiếp UART: CH340
- GPIO tương thích hoàn toàn với firmware Node MCU.
- Cáp nguồn: 5VDC MicroUSB hoặc Vin.
- GIPO giao tiếp mức 3.3VDC
- Tích hợp Led báo trạng thái, nút Reset, Flash.
- Tương thích hoàn toàn với trình biên dịch Arduino.
- Kích thước: 59 x 32mm

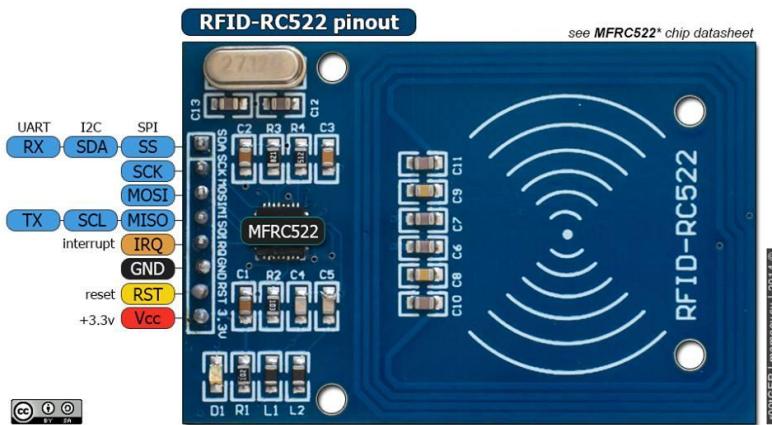


HÌNH 3.7: ESP8266 NodeMCU Lua V3 CH340

2.5 RFID NFC 13.56MHz RC522

- Nguồn sử dụng: 3.3VDC
- Dòng điện: 13~26mA
- Tần số hoạt động: 13.56Mhz
- Khoảng cách hoạt động: 0~60mm (mifare1 card)
- Chuẩn giao tiếp: SPI
- Tốc độ truyền dữ liệu: tối đa 10Mbit/s

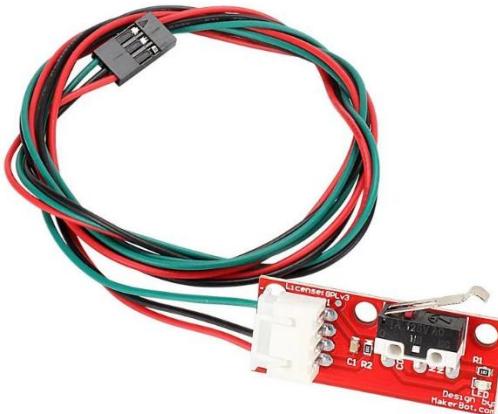
- Các loại card RFID hỗ trợ: mifare1 S50, mifare1 S70, mifare UltraLight, mifare Pro, mifare Desfire
- Kích thước: 40mm × 60mm



HÌNH 3.8: RFID NFC 13.56MHz RC522

2.6 Endstop Switch

- Điện áp sử dụng: 5VDC
- Tích hợp đèn led báo tín hiệu.



HÌNH 3.9: Endstop Switch

PHẦN IV: MÔ HÌNH THỰC TẾ

1. HỆ THỐNG

1.1 Code hệ thống

a. Maincode

```
/* USER CODE BEGIN Header */
/*
*****
* @file      : main.c
* @brief     : Main program body
*****
* @attention
*
* Copyright (c) 2025 STMicroelectronics.
* All rights reserved.
*
* This software is licensed under terms that can be found in the LICENSE file
* in the root directory of this software component.
* If no LICENSE file comes with this software, it is provided AS-IS.
*
*****
*/
/* USER CODE END Header */

/* Includes -----*/
#include "main.h"
#include "rc522.h"
#include "stdio.h"
#include "string.h"
#include "ILI9341_GFX.h"
#include "ILI9341_STM32_Driver.h"
/* Private includes -----*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define -----*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -----*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables -----*/
SPI_HandleTypeDef hspi1;
SPI_HandleTypeDef hspi2;
DMA_HandleTypeDef hdma_spi2_tx;

/* USER CODE BEGIN PV */
```

```

uint8_t status;
uint8_t str[MAX_LEN]; // Max_LEN = 16
uint8_t sNum[5];
#define COLOR_BACKGROUND BLACK
#define COLOR_BUTTON NAVY
#define COLOR_BUTTON_HIGHLIGHT BLUE
#define COLOR_TEXT WHITE
#define COLOR_ERROR RED
#define COLOR_SUCCESS GREEN
#define COLOR_LOCKER_EMPTY GREEN
#define COLOR_LOCKER_OCCUPIED RED
#define COLOR_WARNING YELLOW
/* USER CODE END PV */

/* Private function prototypes -----
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_SPI1_Init(void);
static void MX_SPI2_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code -----*/
/* USER CODE BEGIN 0 */
void key(void);
void locker(void);
void quetthe(void);
void lock(void);
void unlock(void);
void guihanhly(void);
void confirm(void);
void taocode(void);
void xincamon(void);
void unlock(void);
void sosanhpass(void);
void dongcua(void);
void layhanhly(void);
void taolaipassword();
uint32_t timeout = 0;
uint8_t i = 0, k = 0, dacohanhly = 0, count = 1, close = 0;
uint8_t f=1;
char keyPressed = 0;
char keystack[5], Password[5], sosanh[5];
uint8_t from_locker = 0;

/* USER CODE END 0 */

/**
 * @brief The application entry point.
 * @retval int
 */
int main(void)
{
    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */

    /* MCU Configuration-----*/
    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
}

```

```

HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_DMA_Init();
MX_SPI1_Init();
MX_SPI2_Init();
/* USER CODE BEGIN 2 */
    MFRC522_Init();
    ILI9341_Init();
    // Lcd_PortType ports[] = { D4_GPIO_Port, D5_GPIO_Port, D6_GPIO_Port, D7_GPIO_Port };
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
    while (1) {
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
    if (close == 0) {
        dongua();
    } else {
        locker();
    }
}
/* USER CODE END 3 */
}

/**
 * @brief System Clock Configuration
 * @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_HSE;
    RCC_OscInitStruct.HSEState = RCC_HSE_ON;
    RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
    RCC_OscInitStruct.HSISState = RCC_HSI_ON;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
    RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
    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 SPI1 Initialization Function
 * @param None
 * @retval None
 */
static void MX_SPI1_Init(void)
{
    /* USER CODE BEGIN SPI1_Init 0 */

    /* USER CODE END SPI1_Init 0 */

    /* USER CODE BEGIN SPI1_Init 1 */

    /* USER CODE END SPI1_Init 1 */
    /* SPI1 parameter configuration*/
    hspi1.Instance = SPI1;
    hspi1.Init.Mode = SPI_MODE_MASTER;
    hspi1.Init.Direction = SPI_DIRECTION_2LINES;
    hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
    hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
    hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
    hspi1.Init.NSS = SPI_NSS_SOFT;
    hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_8;
    hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
    hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
    hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
    hspi1.Init.CRCPolynomial = 10;
    if (HAL_SPI_Init(&hspi1) != HAL_OK)
    {
        Error_Handler();
    }
    /* USER CODE BEGIN SPI1_Init 2 */

    /* USER CODE END SPI1_Init 2 */
}

/**
 * @brief SPI2 Initialization Function
 * @param None
 * @retval None
 */
static void MX_SPI2_Init(void)
{
    /* USER CODE BEGIN SPI2_Init 0 */

    /* USER CODE END SPI2_Init 0 */
}

```

```

/* USER CODE BEGIN SPI2_Init 1 */

/* USER CODE END SPI2_Init 1 */
/* SPI2 parameter configuration*/
hspi2.Instance = SPI2;
hspi2.Init.Mode = SPI_MODE_MASTER;
hspi2.Init.Direction = SPI_DIRECTION_2LINES;
hspi2.Init.DataSize = SPI_DATASIZE_8BIT;
hspi2.Init.CLKPolarity = SPI_POLARITY_LOW;
hspi2.Init.CLKPhase = SPI_PHASE_1EDGE;
hspi2.Init.NSS = SPI_NSS_SOFT;
hspi2.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_2;
hspi2.Init.FirstBit = SPI_FIRSTBIT_MSB;
hspi2.Init.TIMode = SPI_TIMODE_DISABLE;
hspi2.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
hspi2.Init.CRCPolynomial = 10;
if (HAL_SPI_Init(&hspi2) != HAL_OK)
{
    Error_Handler();
}
/* USER CODE BEGIN SPI2_Init 2 */

/* USER CODE END SPI2_Init 2 */

}

/**
 * Enable DMA controller clock
 */
static void MX_DMA_Init(void)
{

/* DMA controller clock enable */
__HAL_RCC_DMA1_CLK_ENABLE();

/* DMA interrupt init */
/* DMA1_Channel5_IRQHandler interrupt configuration */
HAL_NVIC_SetPriority(DMA1_Channel5_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(DMA1_Channel5_IRQn);

}

/**
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static 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_GPIOC_CLK_ENABLE();
__HAL_RCC_GPIOD_CLK_ENABLE();
__HAL_RCC_GPIOA_CLK_ENABLE();
__HAL_RCC_GPIOB_CLK_ENABLE();

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOA, Solenoid_Pin|RST_Pin|SS_Pin, GPIO_PIN_RESET);

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(LCD_RST_GPIO_Port, LCD_RST_Pin, GPIO_PIN_SET);

```

```

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

/*Configure GPIO pin Output Level*/
HAL_GPIO_WritePin(GPIOC, C1_Pin|C2_Pin|C3_Pin|C4_Pin, GPIO_PIN_RESET);

/*Configure GPIO pins : R1_Pin R2_Pin R3_Pin R4_Pin */
GPIO_InitStruct.Pin = R1_Pin|R2_Pin|R3_Pin|R4_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_PULLDOWN;
HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

/*Configure GPIO pin : Input_Hanhtrinh_Pin */
GPIO_InitStruct.Pin = Input_Hanhtrinh_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(Input_Hanhtrinh_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pins : Solenoid_Pin RST_Pin SS_Pin */
GPIO_InitStruct.Pin = Solenoid_Pin|RST_Pin|SS_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/*Configure GPIO pins : LCD_RST_Pin LCD_DC_Pin LCD_CS_Pin PB3
PB4 PB6 PB7 PB8 */
GPIO_InitStruct.Pin = LCD_RST_Pin|LCD_DC_Pin|LCD_CS_Pin|GPIO_PIN_3
    |GPIO_PIN_4|GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_8;
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);

/*Configure GPIO pins : C1_Pin C2_Pin C3_Pin C4_Pin */
GPIO_InitStruct.Pin = C1_Pin|C2_Pin|C3_Pin|C4_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

/*Configure GPIO pin : PB5 */
GPIO_InitStruct.Pin = GPIO_PIN_5;
GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */
void drawTitle(const char *text) {
    ILI9341_DrawFilledRectangleCoord(0, 0, 239, 30, COLOR_BUTTON);
    ILI9341_DrawText(text, FONT4, 20, 8, COLOR_TEXT, COLOR_BUTTON);
}
void drawUnlockedPadlock(uint8_t x, uint8_t y, uint8_t color) {
    // Vẽ thân khóa (hình vuông 30x30)
    ILI9341_DrawFilledRectangleCoord(x, y, x + 30, y + 30, color);

    // Vẽ vòng khóa mở lệch phải (bán nguyệt)
    ILI9341_DrawHollowCircle(x + 20, y - 10, 15, color);
}

```

```

// Xóa phần dưới của vòng khóa để tạo hiệu ứng "mờ"
ILI9341_DrawFilledRectangleCoord(x + 10, y - 10, x + 35, y, COLOR_BACKGROUND);
}

void drawLockedPadlock(uint8_t x, uint8_t y, uint8_t color) {
    // Vẽ thân khóa (hình chữ nhật)
    ILI9341_DrawFilledRectangleCoord(x, y + 15, x + 40, y + 50, color);

    // Vẽ vòng khóa (hình bán nguyệt)
    ILI9341_DrawHollowCircle(x + 20, y + 15, 15, color);
    ILI9341_DrawFilledRectangleCoord(x + 5, y + 15, x + 35, y + 20, color); // Nối vòng khóa với thân khóa

    // Vẽ chốt khóa (đường doc ở giữa ô khóa)
    ILI9341_DrawFilledRectangleCoord(x + 18, y + 30, x + 22, y + 45, COLOR_BACKGROUND);
}

void drawLuggageIcon(uint16_t x, uint16_t y, uint16_t color) {
    // Thân vali (hình chữ nhật)
    ILI9341_DrawFilledRectangleCoord(x, y, x + 20, y + 25, color);

    // Tay cầm vali (hình chữ nhật nhỏ trên cùng)
    ILI9341_DrawFilledRectangleCoord(x + 5, y - 5, x + 15, y, color);

    // Bánh xe vali (hai hình tròn nhỏ)
    ILI9341_DrawFilledCircle(x + 3, y + 27, 2, color);
    ILI9341_DrawFilledCircle(x + 17, y + 27, 2, color);
}

void drawHandHoldingLuggage(uint16_t x, uint16_t y, uint16_t color) {
    // Tay cầm (hình chữ nhật nhỏ, giả lập ngón tay)
    ILI9341_DrawFilledRectangleCoord(x, y, x + 20, y + 6, color); // Ngón tay cầm

    // Tay người (vẽ hình oval nhỏ tượng trưng)
    ILI9341_DrawFilledCircle(x + 10, y - 5, 6, color);

    // Tay nắm vali (thanh ngang nối tay và vali)
    ILI9341_DrawHLine(x + 5, y + 6, 10, color);

    // Thân vali
    ILI9341_DrawFilledRectangleCoord(x, y + 10, x + 20, y + 35, color);

    // Bánh xe vali
    ILI9341_DrawFilledCircle(x + 4, y + 37, 2, color);
    ILI9341_DrawFilledCircle(x + 16, y + 37, 2, color);
}

void drawMainScreen() {
    ILI9341_FillScreen(COLOR_BACKGROUND);
    drawTitle("TU GUI HANH LY");

    // Viền phân cách giữa 2 phần
    ILI9341_DrawHLine(20, 150, 200, COLOR_TEXT);

    // Nút "Gửi hành lý" (trên)
    ILI9341_DrawFilledRectangleCoord(20, 80, 220, 140, COLOR_BUTTON);
    ILI9341_DrawHollowRectangleCoord(20, 80, 220, 140, COLOR_TEXT); // Viền sáng
    drawLuggageIcon(30, 95, COLOR_TEXT); // Vẽ icon vali
    ILI9341_DrawText("A. Gui hanh ly", FONT4, 70, 105, COLOR_TEXT, COLOR_BUTTON);

    // Nút "Lấy hành lý" (dưới)
    ILI9341_DrawFilledRectangleCoord(20, 160, 220, 220, COLOR_BUTTON_HIGHLIGHT);
    ILI9341_DrawHollowRectangleCoord(20, 160, 220, 220, COLOR_TEXT); // Viền sáng
    drawHandHoldingLuggage(30, 175, COLOR_TEXT); // Vẽ icon tay cầm vali
    ILI9341_DrawText("B. Lay hanh ly", FONT4, 70, 185, COLOR_TEXT, COLOR_BUTTON_HIGHLIGHT);
}

void drawCloseDoorScreen() {
    ILI9341_FillScreen(COLOR_BACKGROUND);
}

```

```

drawTitle("DONG CUA");

// Hiển thị thông báo yêu cầu đóng cửa
ILI9341_DrawFilledRectangleCoord(20, 80, 220, 120, COLOR_BUTTON);
ILI9341_DrawHollowRectangleCoord(20, 80, 220, 120, COLOR_TEXT);
ILI9341_DrawText("Vui lòng đóng cửa", FONT4, 40, 95, COLOR_TEXT, COLOR_BUTTON);

// Hiển thị trạng thái "Đang chờ đóng cửa"
ILI9341_DrawFilledRectangleCoord(20, 140, 220, 180, COLOR_BUTTON_HIGHLIGHT);
ILI9341_DrawHollowRectangleCoord(20, 140, 220, 180, COLOR_TEXT);
ILI9341_DrawText("Đang chờ...", FONT4, 80, 155, COLOR_TEXT, COLOR_BUTTON_HIGHLIGHT);
}

void drawDoorLockedScreen() {
    ILI9341_FillScreen(COLOR_BACKGROUND);
    drawTitle("TÙ DA KHOA");

    // Hiển thị thông báo "Tù đã khóa thành công"
    ILI9341_DrawFilledRectangleCoord(35, 70, 249, 110, COLOR_BUTTON_HIGHLIGHT);
    ILI9341_DrawHollowRectangleCoord(35, 70, 249, 110, COLOR_TEXT);
    ILI9341_DrawText("Tù đã khóa thành công", FONT4, 40, 85, COLOR_TEXT,
COLOR_BUTTON_HIGHLIGHT);

    // Vẽ ô khóa đã khóa
    drawLockedPadlock(135, 140, COLOR_TEXT);
}

void drawTimeoutScreen() {
    ILI9341_FillScreen(COLOR_BACKGROUND);
    drawTitle("HẾT THOI GIAN");

    // Hiển thị thông báo chính
    ILI9341_DrawText("Thao tac qua thoi gian!", FONT4, 60, 80, COLOR_ERROR,
COLOR_BACKGROUND);

    // Vẽ đồng hồ cát biểu thị thời gian hết hạn
    uint16_t clock_x = 150, clock_y = 140;
        ILI9341_DrawHollowCircle(clock_x, clock_y, 20, COLOR_TEXT); // Viền đồng hồ
        ILI9341_DrawFilledRectangleCoord(clock_x - 2, clock_y - 10, clock_x + 2, clock_y + 5,
COLOR_TEXT); // Kim phút
        ILI9341_DrawFilledRectangleCoord(clock_x - 5, clock_y - 2, clock_x + 10, clock_y + 2,
COLOR_TEXT); // Kim giờ
        ILI9341_DrawHollowRectangleCoord(clock_x - 10, clock_y + 20, clock_x + 10, clock_y + 30,
COLOR_TEXT); // Chân đồng hồ
    // Thông báo quay lại màn hình chính
    ILI9341_DrawText("Quay về màn hình chính...", FONT4, 50, 190, COLOR_WARNING,
COLOR_BACKGROUND);

    HAL_Delay(3000); // Đợi 3 giây trước khi quay lại màn hình chính
}

void drawLockerOpenWarning() {
    ILI9341_FillScreen(COLOR_BACKGROUND);
    drawTitle("CANH BAO!");

    // Hiển thị thông báo chính
    ILI9341_DrawText("Tú đang mở!", FONT4, 90, 70, COLOR_ERROR, COLOR_BACKGROUND);
    ILI9341_DrawText("Hay đóng tú trước khi sử dụng.", FONT4, 30, 110, COLOR_ERROR,
COLOR_BACKGROUND);

    // Vẽ ô khóa mở
    uint16_t lockX = 130, lockY = 170; // Tọa độ trung tâm ô khóa
    drawUnlockedPadlock(lockX, lockY, COLOR_ERROR);

    // Hiệu ứng cảnh báo (sóng tín hiệu đỏ)
    ILI9341_DrawHollowCircle(lockX + 20, lockY + 10, 25, COLOR_ERROR);
    ILI9341_DrawHollowCircle(lockX + 20, lockY + 10, 35, COLOR_ERROR);
}

```

```

}

void drawLockerOpenScreen2() {
    ILI9341_FillScreen(COLOR_BACKGROUND);
    drawTitle("YEU CAU NGUOI DUNG");
    // Hiển thị thông báo
    ILI9341_DrawText("An C de mo tu", FONT4, 90, 80, COLOR_WARNING, COLOR_BACKGROUND);

    // Vẽ cửa (hình chữ nhật có viền)
    uint16_t door_x1 = 120, door_y1 = 120;
    uint16_t door_x2 = 200, door_y2 = 200;

    ILI9341_DrawFilledRectangleCoord(door_x1, door_y1, door_x2, door_y2, COLOR_WARNING); // Cửa chính
    ILI9341_DrawHollowRectangleCoord(door_x1, door_y1, door_x2, door_y2, COLOR_TEXT); // Viền cửa

    // Vẽ khoảng trống cửa mở
    ILI9341_DrawFilledRectangleCoord(door_x2 - 10, door_y1, door_x2, door_y2, COLOR_BACKGROUND);

    // Viền cho khoảng trống cửa mở
    ILI9341_DrawHollowRectangleCoord(door_x2 - 10, door_y1, door_x2, door_y2, COLOR_TEXT);

    // Vẽ tay nắm cửa (hình chữ nhật nhỏ có viền)
    uint16_t handle_x1 = 130, handle_y1 = 150;
    uint16_t handle_x2 = 140, handle_y2 = 160;

    ILI9341_DrawFilledRectangleCoord(handle_x1, handle_y1, handle_x2, handle_y2,
    COLOR_BACKGROUND);
    ILI9341_DrawHollowRectangleCoord(handle_x1, handle_y1, handle_x2, handle_y2, COLOR_TEXT);
}

void drawLockerOpenWarning2() {
    ILI9341_FillScreen(COLOR_BACKGROUND);
    drawTitle("GUI HANH LY");

    // Hiển thị thông báo chính
    ILI9341_DrawText("Tu dang mo!", FONT4, 90, 70, COLOR_TEXT, COLOR_BACKGROUND);
    ILI9341_DrawText("Dong tu de tiep tuc", FONT4, 70, 110, COLOR_TEXT, COLOR_BACKGROUND);

    // Vẽ ô khóa mở
    uint16_t lockX = 130, lockY = 170; // Toa độ trung tâm ô khóa
    drawUnlockedPadlock(lockX, lockY, COLOR_TEXT);
}

void drawConfirmScreen() {
    ILI9341_FillScreen(COLOR_BACKGROUND);
    drawTitle("XAC NHAN");

    // Hiển thị thông báo
    ILI9341_DrawText("An phim D de confirm", FONT4, 60, 100, COLOR_TEXT, COLOR_BACKGROUND);

    // Vẽ ô chứa chữ CONFIRM
    uint16_t box_x1 = 80, box_y1 = 150; // Góc trên trái
    uint16_t box_x2 = 200, box_y2 = 200; // Góc dưới phải

    // Vẽ nền và viền
    ILI9341_DrawFilledRectangleCoord(box_x1, box_y1, box_x2, box_y2, COLOR_BUTTON);
    ILI9341_DrawHollowRectangleCoord(box_x1, box_y1, box_x2, box_y2, COLOR_TEXT); // Viền sáng

    // Chữ CONFIRM
    ILI9341_DrawText("CONFIRM", FONT4, 100, 170, COLOR_TEXT, COLOR_BUTTON);
}

void drawCardSuccess() {
}

```

```

ILI9341_FillScreen(COLOR_BACKGROUND);
drawTitle("QUET THE");

// Chữ thông báo (dịch phải)
ILI9341_DrawText("Quét the thanh cong!", FONT4, 50, 100, COLOR_SUCCESS,
COLOR_BACKGROUND);

// Biểu tượng thẻ (dịch phải)
uint16_t card_x = 120 - 50 + 30; // Dịch phải 20 pixel
uint16_t card_y = 160 - 25;
ILI9341_DrawFilledRectangleCoord(card_x, card_y, card_x + 100, card_y + 50, COLOR_TEXT);
ILI9341_DrawFilledRectangleCoord(card_x + 5, card_y + 5, card_x + 90, card_y + 15,
COLOR_BACKGROUND);

// Hiệu ứng sóng NFC (dịch phải)
uint16_t nfc_x = 120 + 30; // Dịch phải 20 pixel
uint16_t nfc_y = card_y + 70;
ILI9341_DrawHollowCircle(nfc_x, nfc_y, 10, COLOR_SUCCESS);
ILI9341_DrawHollowCircle(nfc_x, nfc_y, 15, COLOR_SUCCESS);
ILI9341_DrawHollowCircle(nfc_x, nfc_y, 20, COLOR_SUCCESS);
}

void drawCardFail() {
    ILI9341_FillScreen(COLOR_BACKGROUND);
    drawTitle("QUET THE");

    // Chữ thông báo lỗi
    ILI9341_DrawText("Quét the that bai!", FONT4, 80, 90, COLOR_ERROR, COLOR_BACKGROUND);

    // Vẽ thẻ
    uint16_t card_x = 110 ; // Dịch phải 20 pixel
    uint16_t card_y = 140 - 25;
    ILI9341_DrawFilledRectangleCoord(card_x, card_y, card_x + 100, card_y + 50, COLOR_TEXT);
    ILI9341_DrawFilledRectangleCoord(card_x + 5, card_y + 5, card_x + 90, card_y + 15,
COLOR_BACKGROUND);

    // Vẽ dấu X đơn giản (bằng 2 đường chéo)
    ILI9341_DrawText("X", FONT4, 155, 170, COLOR_ERROR, COLOR_BACKGROUND);
    ILI9341_DrawText("Vui long quet the lai", FONT4, 80, 90, COLOR_ERROR, COLOR_BACKGROUND);
}

void drawPaymentScreen() {
    ILI9341_FillScreen(COLOR_BACKGROUND);
    drawTitle("THANH TOAN");

    // Hướng dẫn quét thẻ (dịch phải)
    ILI9341_DrawText("Vui long quet the de thanh toan", FONT4, 20, 100, COLOR_TEXT,
COLOR_BACKGROUND);

    // Biểu tượng thẻ (dịch phải)
    uint16_t card_x = 120 - 50 + 40;
    uint16_t card_y = 160 - 25;
    ILI9341_DrawFilledRectangleCoord(card_x, card_y, card_x + 100, card_y + 50, COLOR_TEXT);
    ILI9341_DrawFilledRectangleCoord(card_x + 5, card_y + 5, card_x + 90, card_y + 15,
COLOR_BACKGROUND);

    // Hiệu ứng sóng NFC (dịch phải)
    uint16_t nfc_x = 120 + 35;
    uint16_t nfc_y = card_y + 70;
    ILI9341_DrawHollowCircle(nfc_x, nfc_y, 10, COLOR_TEXT);
    ILI9341_DrawHollowCircle(nfc_x, nfc_y, 15, COLOR_TEXT);
}

void drawCardTimeout() {
    ILI9341_FillScreen(COLOR_BACKGROUND);
    drawTitle("QUET THE");
}

```

```

// Hiển thị thông báo
ILI9341_DrawText("Het thoigian quet the!", FONT4, 50, 90, COLOR_ERROR,
COLOR_BACKGROUND);

uint16_t clock_x = 140, clock_y = 160;
ILI9341_DrawHollowCircle(clock_x, clock_y, 20, COLOR_TEXT); // Viền đồng hồ
ILI9341_DrawFilledRectangleCoord(clock_x - 2, clock_y - 10, clock_x + 2, clock_y + 5, COLOR_TEXT); //

Kim phút ILI9341_DrawFilledRectangleCoord(clock_x - 5, clock_y - 2, clock_x + 10, clock_y + 2, COLOR_TEXT); //
Kim giờ ILI9341_DrawHollowRectangleCoord(clock_x - 10, clock_y + 20, clock_x + 10, clock_y + 30,
COLOR_TEXT); // Chân đồng hồ

}

void drawEnterPasswordScreen() {
    ILI9341_FillScreen(COLOR_BACKGROUND);
    drawTitle("NHAP MAT KHAU");

    // Hiển thị hướng dẫn
    ILI9341_DrawText("Nhập mat khau (5 số):", FONT4, 50, 80, COLOR_TEXT, COLOR_BACKGROUND);

    // Vẽ các ô nhập mật khẩu
    for (int i = 0; i < 5; i++) {
        ILI9341_DrawHollowRectangleCoord(60 + i * 40, 130, 90 + i * 40, 170, COLOR_TEXT);
    }
}

void updatePasswordDisplay(uint8_t index, char digit) {
    if (index < 5) { // Chỉ cập nhật khi index hợp lệ (0-4)
        char str[2] = {digit, '\0'}; // Chuyển số thành chuỗi (để vẽ)

        // Tính vị trí x để căn giữa trong ô 30px (kích thước chữ là 15px)
        int xPos = 60 + index * 40 + (30 - 15) / 2;

        // Tính vị trí y để căn giữa trong ô 40px (kích thước chữ là 19px)
        int yPos = 150; // Căn giữa trong ô 130 đến 170 (40px chiều cao)

        // Vẽ số tại ô nhập mật khẩu tương ứng
        ILI9341_DrawText(str, FONT4, xPos, yPos, COLOR_TEXT, COLOR_BACKGROUND);
    }
}

void drawConfirmResetPasswordScreen() {
    ILI9341_FillScreen(COLOR_BACKGROUND); // Xóa màn hình, tô nền tối
    drawTitle("TAO LAI MAT KHAU"); // Tiêu đề trên cùng

    // Hiển thị câu hỏi xác nhận
    ILI9341_DrawText("Ban co muon tao lai?", FONT4, 60, 80, COLOR_TEXT, COLOR_BACKGROUND);
    ILI9341_DrawText("password khong?", FONT4, 80, 110, COLOR_TEXT, COLOR_BACKGROUND);

    // Vẽ khung viền cho lựa chọn *YES
    ILI9341_DrawHollowRectangleCoord(50, 160, 130, 190, COLOR_BUTTON_HIGHLIGHT);
    ILI9341_DrawText("* YES", FONT4, 65, 165, COLOR_BUTTON_HIGHLIGHT,
COLOR_BACKGROUND);

    // Vẽ khung viền cho lựa chọn #NO
    ILI9341_DrawHollowRectangleCoord(150, 160, 230, 190, COLOR_TEXT);
    ILI9341_DrawText("# NO", FONT4, 170, 165, COLOR_TEXT, COLOR_BACKGROUND);
}

void drawThankYouScreen() {
    ILI9341_FillScreen(COLOR_BACKGROUND); // Xóa màn hình, tô nền tối

    // Vẽ đường kẻ trang trí trên & dưới
    ILI9341_DrawFilledRectangleCoord(40, 60, 250, 70, COLOR_TEXT);
}

```

```

    ILI9341_DrawFilledRectangleCoord(40, 180, 250, 190, COLOR_TEXT);

    // Hiển thị tiêu đề cảm ơn
    ILI9341_DrawText("CAM ON QUY KHACH", FONT4, 50, 90, COLOR_TEXT,
    COLOR_BACKGROUND);

    // Hiển thị dòng chữ nhỏ hơn
    ILI9341_DrawText("DA SU DUNG DICH VU", FONT4, 45, 130, COLOR_TEXT,
    COLOR_BACKGROUND);

    HAL_Delay(2000); // Hiển thi 2 giây rồi quay lại màn hình chính
}

void drawIncorrectPasswordAttemptScreen(int count) {
    ILI9341_FillScreen(COLOR_BACKGROUND); // Tô nền tối
    drawTitle("SAI MAT KHAU !!!");

    // Tạo chuỗi hiển thi số lần nhập sai
    char text[50];
    sprintf(text, "Ban da nhap sai %d lan", count); // Tạo thông báo

    // Hiển thi thông báo lên màn hình
    ILI9341_DrawText(text, FONT4, 40, 100, RED, COLOR_BACKGROUND); // Vi trí hiển thi thông báo
    HAL_Delay(2000);
}

void drawLockerOpenScreen() {
    ILI9341_FillScreen(COLOR_BACKGROUND);
    drawTitle("CUA MO!");

    // Hiển thi thông báo
    ILI9341_DrawText("VUI LONG LAY HANH LY", FONT4, 50, 80, COLOR_WARNING,
    COLOR_BACKGROUND);

    // Vẽ cửa (hình chữ nhật có viền)
    uint16_t door_x1 = 120, door_y1 = 120;
    uint16_t door_x2 = 200, door_y2 = 200;

    ILI9341_DrawFilledRectangleCoord(door_x1, door_y1, door_x2, door_y2, COLOR_WARNING); // Cửa chính
    ILI9341_DrawHollowRectangleCoord(door_x1, door_y1, door_x2, door_y2, COLOR_TEXT); // Viền cửa

    // Vẽ khoảng trống cửa mở
    ILI9341_DrawFilledRectangleCoord(door_x2 - 10, door_y1, door_x2, door_y2, COLOR_BACKGROUND);

    // Viền cho khoảng trống cửa mở
    ILI9341_DrawHollowRectangleCoord(door_x2 - 10, door_y1, door_x2, door_y2, COLOR_TEXT);

    // Vẽ tay nắm cửa (hình chữ nhật nhỏ có viền)
    uint16_t handle_x1 = 130, handle_y1 = 150;
    uint16_t handle_x2 = 140, handle_y2 = 160;

    ILI9341_DrawFilledRectangleCoord(handle_x1, handle_y1, handle_x2, handle_y2,
    COLOR_BACKGROUND);
    ILI9341_DrawHollowRectangleCoord(handle_x1, handle_y1, handle_x2, handle_y2, COLOR_TEXT);
}

void drawLockerEmptyScreen() {
    ILI9341_FillScreen(COLOR_BACKGROUND); // Xóa toàn bộ màn hình về màu nền
    drawTitle("THONG BAO"); // Tiêu đề ở trên cùng

    // Nội dung chính
    ILI9341_DrawText("Tu chua co hanh ly", FONT4, 20, 80, COLOR_TEXT, COLOR_BACKGROUND);
    ILI9341_DrawText("Vui long chon", FONT4, 20, 110, COLOR_TEXT, COLOR_BACKGROUND);
}

```

```

    ILI9341_DrawText("Gui hành lý (A)", FONT4, 20, 140, COLOR_BUTTON_HIGHLIGHT,
    COLOR_BACKGROUND);

    // Vẽ khung hình minh họa đơn giản
    ILI9341_DrawHollowRectangleCoord(10, 70, 230, 170, COLOR_TEXT); // khung chữ
}

void dongcua() {
    i = 0; //Trang thái 0
    timeout=0;
    HAL_Delay(50);
    drawCloseDoorScreen();

    // Chờ cho đến khi công tắc hành trình báo tủ đã đóng
    while (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_1) != 0) {
        HAL_Delay(500); // Kiểm tra lại sau mỗi 500ms
    }

    drawDoorLockedScreen();
    HAL_Delay(300);
    // Khi tủ đã đóng, khóa lại
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, 0);
    close = 1;
    f=1;
    locker(); // Quay lại menu chính
}

void locker() {
    keyPressed = 0; //Đảm bảo không có kí tự đè
    HAL_Delay(50);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, 0); // Đảm bảo khóa tủ lúc bắt đầu
    i = 1;

    if(f==1) drawMainScreen();
    f=0;
    // Kiểm tra trạng thái cửa trước khi cho phép chọn chức năng
    if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_1) == 1) {
        drawLockerOpenWarning();
        HAL_Delay(300);
        dongcua();
        return;
    }
    key();
    HAL_Delay(500);
    if (keyPressed == 65) { // A = Gửi hành lý
        HAL_Delay(100);
        lock();
    } else if (keyPressed == 66) { // B = Lấy hành lý
        keyPressed=0;
        HAL_Delay(100);
        unlock();
    }
}

```

```

        }

void lock() {
    i = 2;
    timeout=0;
    drawLockerOpenScreen2();
    timeout = HAL_GetTick() + 100000; // Timeout sau 100 giây
    //PA1 la mach hanh trinh
    while (HAL_GetTick() < timeout-90000) {
        key();
        // Kiểm tra xem tủ đã có hành lý chưa
        if (dacohanhly == 1) {
            ILI9341_FillScreen(COLOR_BACKGROUND);
            drawTitle("CANH BAO!!");
            ILI9341_DrawText("Tu da co hanh ly. Thoat sau 5s", FONT4, 15, 80,
COLOR_WARNING, COLOR_BACKGROUND);
            uint8_t clock_x = 150, clock_y = 140;
            ILI9341_DrawHollowCircle(clock_x, clock_y, 20, COLOR_TEXT); // Viền đồng hồ
            ILI9341_DrawFilledRectangleCoord(clock_x - 2, clock_y - 10, clock_x + 2, clock_y + 5,
COLOR_TEXT); // Kim phút
            ILI9341_DrawFilledRectangleCoord(clock_x - 5, clock_y - 2, clock_x + 10, clock_y + 2,
COLOR_TEXT); // Kim giờ
            ILI9341_DrawHollowRectangleCoord(clock_x - 10, clock_y + 20, clock_x + 10, clock_y +
30, COLOR_TEXT); // Chân đồng hồ
            HAL_Delay(5000);
            f=1;
            locker();
            return;
        }
        // Chờ nhập phím C để gửi hành lý
    }
    else {
        if (keyPressed == 67) {
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, 1); //Mở chốt solenoid
            HAL_Delay(50);
            guihanhly();
        }
    }
}

void guihanhly() {
    i = 3;
    close = 0;
    timeout=0;
    timeout = HAL_GetTick() + 100000; // Timeout sau 100 giây
    //PA1 la mach hanh trinh
    while (HAL_GetTick() < timeout) {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, 1); //chắc chắn rằng tủ đang mở
        drawLockerOpenWarning2();
        HAL_Delay(2000); // chống nhảy qua trạng thái i4 quá sớm nếu khách chưa kịp mở tủ
        if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_1) == 0) {
            close = 1;
            confirm();
        }
    }
    HAL_Delay(50); // Kiểm tra lại sau mỗi 2050
}

// Sau 100 giây, thử lại quy trình tủ
locker();
}

```

```

void confirm() {
    drawConfirmScreen();
    i = 4;
    timeout=0;
    timeout = HAL_GetTick() + 100000;
    while (HAL_GetTick() < timeout) {
        keyPressed = 0; //Chắc chắn không có kí tự đè
        key();
        if (keyPressed == 68) { //Nhấn D để confirm rằng đã gửi hành lý xong
            if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_1) == 0) { //Công tắc hành trình đóng
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, 0); //Tủ chốt lại
                close = 1;
                taocode();
            } else {
                guihanhly();
            }
        }
        HAL_Delay(10);
    }
    locker();
}

void quetthe() {
    i = 5;
    timeout = HAL_GetTick() + 100000;
    uint8_t str[5];
    const uint8_t expectedID[5] = { 254, 226, 48, 3, 47 }; // Thẻ đúng

    drawPaymentScreen();

    while (HAL_GetTick() < timeout) {
        // Kiểm tra nếu đọc thẻ thành công
        if (MFRC522_Request(PICC_REQIDL, str) == MI_OK) //Kiểm tra xem có thẻ nào trong vùng quét
            không. Nếu có, thông tin lưu vào str.
        && MFRC522_Anticoll(str) == MI_OK) { //Đọc số seri (UID) của thẻ để tránh trùng lắp.
            //Nếu cả hai hàm trả về MI_OK, nghĩa là đọc thẻ thành công, thì tiếp tục.
            memcpy(sNum, str, 5); //Sao chép 5 byte số seri UID vào biến sNum.

            // Kiểm tra nếu thẻ hợp lệ
            if (memcmp(str, expectedID, 5) == 0) { //Nếu 5 byte trong str trùng với expectedID, nghĩa
                là thẻ hợp lệ.
                drawCardSuccess();
                layhanhly();
                return; //Thẻ hợp lệ, thoát hàm ngay
            } else {
                i = 16;
                drawCardFail();
            }
        }
        HAL_Delay(500); //Tránh spam UART
    }
    locker(); // Chỉ khóa nếu hết thời gian mà không có thẻ hợp lệ
}

void taocode() {
    i = 6;
    k = 0;
    timeout=0;
    keyPressed = 0;
    drawEnterPasswordScreen();
    timeout = HAL_GetTick() + 100000;
    HAL_Delay(500);
}

```

```

while (HAL_GetTick() < timeout) {
    keyPressed=0;
    printf("Password: %d%d%d%d\n", keystack[0], keystack[1], keystack[2],
           keystack[3], keystack[4]);
    key();
    if (keyPressed != 0) {
        keystack[k] = keyPressed; // Lưu giá trị vào vị trí k
        updatePasswordDisplay(k, keystack[k]);
        k++; // Tăng k để lần nhập sau lưu vào vị trí mới
        keyPressed = 0; //reset lại để tạo vòng lặp mới
        HAL_Delay(500);
        if (k == 5) {
            for (int j = 0; j < 5; j++) {
                Password[j] = keystack[j]; //gán Password = keystack theo thứ tự j
                keystack[j] = 0; //reset keystack để chuẩn bị cho chương trình kế tiếp
            }
            taolaipassword();
        }
    }
}
locker();
}

void taolaipassword() {
    dacohanhly = 0;
    timeout=0;
    i = 7;
    k = 0;
    drawEnterPasswordScreen();
    for (int a = 0; a < 5; a++) {
        updatePasswordDisplay(a, Password[a]);
    }
    printf("Password: %d%d%d%d\n", Password[0], Password[1], Password[2],
           Password[3], Password[4]);
    drawConfirmResetPasswordScreen();
    printf("Ban co muon tao lai password khong? *Yes #No"); // 42 la * 35 la # theo ASCII
    keyPressed = 0;
    timeout = HAL_GetTick() + 100000;
    while (HAL_GetTick() < timeout) {
        key();
        if (keyPressed == 35) {
            xincamon(); //Nhấn # NO
        }
        if (keyPressed == 42) {
            taocode(); //Nhấn * YES
        }
    }
    xincamon();
}

void xincamon() {
//ien thi da thuc hien thanh cong tro ve trang thai ban dau sau 10s

    i = 8;
    timeout=0;
    timeout = HAL_GetTick() + 3000;
    drawThankYouScreen();
    while (HAL_GetTick() < timeout) {
        dacohanhly = 1;
    }
    HAL_Delay(2000);
    f=1;
    locker();
}

```

```

}

void unlock() {
    timeout=0;
    updatePasswordDisplay(k, keystack[k]);
    HAL_Delay(100);
    keyPressed = 0; //Đảm bảo không có kí tự đè
    k = 0;
    i = 9;
    if(dacohanhlly==0){
        drawLockerEmptyScreen();
        HAL_Delay(3000);
        f=1;
        locker();
    }
    if(dacohanhlly==1){
        drawEnterPasswordScreen();
        timeout = HAL_GetTick() + 100000;
        while (HAL_GetTick() < timeout) {
            printf("Hay nhap password");
            printf("Password: %d%d%d%d%d\n", keystack[0], keystack[1], keystack[2],
                   keystack[3], keystack[4]);
            key();
            if (keyPressed != 0) {
                keystack[k] = keyPressed; // Lưu giá trị vào vị trí j
                updatePasswordDisplay(k, keystack[k]);
                k++; // Tăng j để lần nhập sau lưu vào vị trí mới
                keyPressed = 0; // Reset giá trị sau khi lưu
                HAL_Delay(500);
                if (k == 5) {
                    for (int j = 0; j < 5; j++) {
                        sosanh[j] = keystack[j];
                        keystack[j] = 0;
                        k = 0;
                    }
                    sosanhpass();
                }
            }
        }
        printf("Het thoi gian nhap password");
        f=1;
        locker();
    }
}

void sosanhpass() {
    i = 10;
    for (int t = 0; t < 5; t++) {
        if (sosanh[t] != Password[t]) {
            //printf("Ban da nhap sai %d lan password", count);
            drawIncorrectPasswordAttemptScreen(count);
            count++;

            if (count > 3) {
                printf("Ban da nhap qua 3 lan password he thong se bi khoa trong 5p");
                ILI9341_DrawText(" ", FONT4, 40, 100, RED,
COLOR_BACKGROUND);
                ILI9341_DrawText("Ban da nhap sai qua 3 lan", FONT4, 40, 100, RED,
COLOR_BACKGROUND);
                ILI9341_DrawText("He thong bi khoa trong 5 phut", FONT4, 40, 120, RED,
COLOR_BACKGROUND);
                HAL_Delay(30000); // Khóa tam 5 phút
                count = 1; // Reset bộ đếm sau khi khóa
                locker(); //Trở về trang thái ban đầu
            }
        }
    }
}

```

```

        }

        unlock(); // Cho phép nhập lại
    }

// Nếu vòng lặp chay hết mà không có lỗi -> Mở tủ

quetthe();

}

void layhanhly() {
    i = 11;
    count = 0;
    close = 0;
    timeout=0;
    timeout = HAL_GetTick() + 1000;
    dacohanhly = 0;
    drawLockerOpenScreen();
    HAL_Delay(3000);
    while (HAL_GetTick() < timeout) {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, 1);
    }
    f=1;
    locker(); // Hết thời gian thì quay về locker, chờ dù có không đóng cửa thì cũng đã có hàm đóng cửa để backup
}
void key() {

    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_9, 1);
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, 0);
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, 0);
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, 0);
    if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_3) == 1) {
        keyPressed = 68; //ASCII value of D
    } else if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_2) == 1) {
        keyPressed = 67; //ASCII value of C
    } else if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_1) == 1) {
        keyPressed = 66; //ASCII value of B
    } else if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_0) == 1) {
        keyPressed = 65; //ASCII value of A
    }
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_9, 0);
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, 1);
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, 0);
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, 0);
    if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_3) == 1) {
        keyPressed = 35; //ASCII value of #
    } else if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_2) == 1) {
        keyPressed = 57; //ASCII value of 9
    } else if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_1) == 1) {
        keyPressed = 54; //ASCII value of 6
    } else if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_0) == 1) {
        keyPressed = 51; //ASCII value of 3
    }

    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_9, 0);
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, 0);
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, 1);
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, 0);
    if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_3) == 1) {
        keyPressed = 48; //ASCII value of 0
    } else if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_2) == 1) {
        keyPressed = 56; //ASCII value of 8
    }
}

```

```

        } else if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_1) == 1) {
            keyPressed = 53; //ASCII value of 5
        } else if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_0) == 1) {
            keyPressed = 50; //ASCII value of 2
        }
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_9, 0);
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, 0);
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, 0);
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6, 1);
        if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_3) == 1) {
            keyPressed = 42; //ASCII value of *
        } else if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_2) == 1) {
            keyPressed = 55; //ASCII value of 7
        } else if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_1) == 1) {
            keyPressed = 52; //ASCII value of 4
        } else if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_0) == 1) {
            keyPressed = 49; //ASCII value of 1
        }
    }

/* USER CODE END 4 */

/**
 * @brief This function is executed in case of error occurrence.
 * @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 */
}

#endif 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 */

```

CODE 1: MAIN CODE

b. Giải thích main code

- **Luồng chính: Hàm locker()**

Đây là "menu chính" sau khi hệ thống khởi động.

1. Xác nhận tủ đã đóng (bằng công tắc hành trình GPIO_PIN_1)
 2. Nếu chưa đóng, gọi dongcua() để ép người dùng đóng tủ.
 3. Nếu tủ đã đóng → đợi người dùng chọn:
 - Nhấn A (65): Gửi hành lý → gọi lock()
 - Nhấn B (66): Lấy hành lý → gọi unlock()
-

▪ **Gửi hành lý – Hàm lock()**

1. Hiển thị màn hình mở tủ để gửi đồ.
 2. Nếu phát hiện dacohanhly == 1 → cảnh báo tủ đã có hành lý → thoát về locker().
 3. Nếu người dùng nhấn C (67):
 - Mở chốt điện từ (solenoid) qua GPIO_PIN_2 = 1
 - Gọi guihanhly() để tiếp tục quy trình gửi.
-

▪ **Hàm guihanhly() – Mở tủ để bỏ đồ**

1. Tủ mở (chốt vẫn giữ ở mức HIGH)
 2. Kiểm tra công tắc hành trình GPIO_PIN_1:
 - Nếu người dùng đóng tủ → gọi confirm() để xác nhận
 - Nếu hết 100 giây mà không đóng → về lại locker()
-

▪ **Hàm confirm() – Xác nhận đã gửi hành lý**

1. Hiển thị yêu cầu nhấn D (68) để xác nhận.
 2. Nếu người dùng xác nhận:
 - Kiểm tra lại công tắc hành trình
 - Nếu đóng → hạ chốt (GPIO_PIN_2 = 0), gọi taocode() để nhập mật khẩu
-

▪ **Hàm taocode() – Nhập mã PIN**

1. Nhập 5 ký tự → lưu vào Password[]
-

2. Sau đó gọi taolaipassword() để xác nhận lại mật khẩu

3. Cho phép người dùng nhấn:

- * (42) để tạo lại mã
- # (35) để xác nhận hoàn tất → xincamon()

▪ **Hàm xincamon() – Cảm ơn và kết thúc gửi hành lý**

1. Đặt dacohanhly = 1 để đánh dấu tủ đã có hành lý
 2. Sau 5 giây → trả lại locker()
-

▪ **Hàm unlock() – Lấy hành lý**

1. Nếu dacohanhly == 0 → báo tủ trống
 2. Nếu có hành lý:
 - Yêu cầu người dùng nhập mã PIN
 - So sánh với Password[] qua sosanhpass()
-

▪ **Hàm sosanhpass() – So sánh mã PIN**

1. Nếu sai → tăng biến count
 2. Nếu sai quá 3 lần → khóa hệ thống trong 5 phút (HAL_Delay(30000))
 3. Nếu đúng → gọi quetthe() để xác minh thẻ RFID
-

▪ **Hàm quetthe() – Kiểm tra thẻ RFID**

1. So sánh UID thẻ với expectedID
 2. Nếu đúng → gọi layhanhly()
 3. Nếu sai → hiển thị thông báo lỗi
-

▪ **Hàm layhanhly() – Mở tủ cho người dùng lấy hành lý**

1. Mở chốt trong vài giây

2. Đặt dacohanhly = 0 vì đã lấy đồ
 3. Trở lại locker()
-

- **Hàm dongcua() – Đảm bảo tủ đã đóng**
 1. Chờ công tắc hành trình báo tủ đã đóng (GPIO_PIN_1 == 0)
 2. Sau đó khóa chốt điện từ (GPIO_PIN_2 = 0)
 3. Trở về locker()
-

- **Hàm key() – Đọc bàn phím ma trận 4x4**
 1. Lần lượt kích hoạt từng hàng
 2. Đọc từng cột để xác định phím nhấn
 3. Gán keyPressed bằng mã ASCII tương ứng (ví dụ: A=65, * = 42)

1.2 Code SPI ILI9341, RC522

a. RC522.h

```
#include "stm32f1xx_hal.h"

#define uchar unsigned char
#define uint unsigned int
extern SPI_HandleTypeDef hspi1;

//Maximum length of the array
#define MAX_LEN 16

#define HSPI_INSTANCE &hspi1
#define MFRC522_CS_PORT GPIOA
#define MFRC522_CS_PIN  GPIO_PIN_4
#define MFRC522_RST_PORT GPIOA
#define MFRC522_RST_PIN  GPIO_PIN_3

// MFRC522 commands. Described in chapter 10 of the datasheet.
#define PCD_IDLE 0x00 // no action, cancels current command execution
#define PCD_AUTHENT 0x0E // performs the MIFARE standard authentication as a reader
#define PCD_RECEIVE 0x08 // activates the receiver circuits
#define PCD_TRANSMIT 0x04 // transmits data from the FIFO buffer
#define PCD_TRANSCEIVE 0x0C // transmits data from FIFO buffer to antenna and automatically
activates the receiver after transmission
#define PCD_RESETPHASE 0x0F // resets the MFRC522
#define PCD_CALCCRC 0x03 // activates the CRC coprocessor or performs a self-test

// Commands sent to the PICC.
#define PICC_REQIDL 0x26 // REQuest command, Type A. Invites PICCs in state IDLE to go to
READY and prepare for anticollision or selection. 7 bit frame.
#define PICC_REQALL 0x52 // Wake-UP command, Type A. Invites PICCs in state IDLE and HALT to
go to READY(*) and prepare for anticollision or selection. 7 bit frame.
```

```

#define PICC_ANTICOLL 0x93 // Anti collision>Select, Cascade Level 1
#define PICC_SELECTTAG 0x93 // Anti collision>Select, Cascade Level 2
#define PICC_AUTHENT1A 0x60 // Perform authentication with Key A
#define PICC_AUTHENT1B 0x61 // Perform authentication with Key B
#define PICC_READ 0x30 // Reads one 16 byte block from the authenticated sector of the PICC. Also used for MIFARE Ultralight.
#define PICC_WRITE 0xA0 // Writes one 16 byte block to the authenticated sector of the PICC. Called "COMPATIBILITY WRITE" for MIFARE Ultralight.
#define PICC_DECREMENT 0xC0 // Decrement the contents of a block and stores the result in the internal data register.
#define PICC_INCREMENT 0xC1 // Increments the contents of a block and stores the result in the internal data register.
#define PICC_RESTORE 0xC2 // Reads the contents of a block into the internal data register.
#define PICC_TRANSFER 0xB0 // Writes the contents of the internal data register to a block.
#define PICC_HALT 0x50 // HALT command, Type A. Instructs an ACTIVE PICC to go to state HALT.

// Success or error code is returned when communication
#define MI_OK 0
#define MI_NOTAGERR 1
#define MI_ERR 2

// MFRC522 registers. Described in chapter 9 of the datasheet.
// Page 0: Command and Status
#define Reserved00 0x00
#define CommandReg 0x01
#define CommIEnReg 0x02
#define DivIEnReg 0x03
#define CommIrqReg 0x04
#define DivIrqReg 0x05
#define ErrorReg 0x06
#define Status1Reg 0x07
#define Status2Reg 0x08
#define FIFODataReg 0x09
#define FIFOLevelReg 0x0A
#define WaterLevelReg 0x0B
#define ControlReg 0x0C
#define BitFramingReg 0x0D
#define CollReg 0x0E
#define Reserved01 0x0F

//Page 1: Command
#define Reserved10 0x10
#define ModeReg 0x11
#define TxModeReg 0x12
#define RxModeReg 0x13
#define TxControlReg 0x14
#define TxAutoReg 0x15
#define TxSelReg 0x16
#define RxSelReg 0x17
#define RxThresholdReg 0x18
#define DemodReg 0x19
#define Reserved11 0x1A
#define Reserved12 0x1B
#define MifareReg 0x1C
#define Reserved13 0x1D
#define Reserved14 0x1E
#define SerialSpeedReg 0x1F

//Page 2: Configuration
#define Reserved20 0x20
#define CRCResultRegH 0x21
#define CRCResultRegL 0x22
#define Reserved21 0x23
#define ModWidthReg 0x24

```

```

#define Reserved22      0x25
#define RFCfgReg        0x26
#define GsNReg          0x27
#define CWGsPReg         0x28
#define ModGsPReg        0x29
#define TModeReg         0x2A
#define TPrescalerReg   0x2B
#define TReloadRegH     0x2C
#define TReloadRegL     0x2D
#define TCounterValueRegH 0x2E
#define TCounterValueRegL 0x2F
//Page 3: Test Registers
#define Reserved30      0x30
#define TestSel1Reg      0x31
#define TestSel2Reg      0x32
#define TestPinEnReg    0x33
#define TestPinValueReg  0x34
#define TestBusReg       0x35
#define AutoTestReg     0x36
#define VersionReg       0x37
#define AnalogTestReg   0x38
#define TestDAC1Reg     0x39
#define TestDAC2Reg     0x3A
#define TestADCReg      0x3B
#define Reserved31      0x3C
#define Reserved32      0x3D
#define Reserved33      0x3E
#define Reserved34      0x3F

// Functions for manipulating the MFRC522
void MFRC522_Init(void);
uchar MFRC522_Request(uchar reqMode, uchar *TagType);
uchar MFRC522_Anticoll(uchar *serNum);
uchar MFRC522_SelectTag(uchar *serNum);
uchar MFRC522_Auth(uchar authMode, uchar BlockAddr, uchar *Sectorkey, uchar *serNum);
uchar MFRC522_Write(uchar blockAddr, uchar *writeData);
uchar MFRC522_Auth(uchar authMode, uchar BlockAddr, uchar *Sectorkey, uchar *serNum);
uchar MFRC522_Read(uchar blockAddr, uchar *recvData);
void MFRC522_Halt(void);

```

CODE 2: RFID-RC522.h

b. Code RC522.c

```

#include "rc522.h"

/*
 * Function Name: RC522_SPI_Transfer
 * Description: A common function used by Write_MFRC522 and Read_MFRC522
 * Input Parameters: data - the value to be written
 * Returns: a byte of data read from the module
 */
uint8_t RC522_SPI_Transfer(uchar data)
{
    uchar rx_data;
    HAL_SPI_TransmitReceive(HSPI_INSTANCE,&data,&rx_data,1,100);

    return rx_data;
}

/*
 * Function Name: Write_MFRC522
 * Function Description: To a certain MFRC522 register to write a byte of data
 * Input Parameters: addr - register address; val - the value to be written

```

```

/* Return value: None
*/
void Write_MFRC522(uchar addr, uchar val)
{
    /* CS LOW */
    HAL_GPIO_WritePin(MFRC522_CS_PORT,MFRC522_CS_PIN,GPIO_PIN_RESET);

    // even though we are calling transfer frame once, we are really sending
    // two 8-bit frames smooshed together-- sending two 8 bit frames back to back
    // results in a spike in the select line which will jack with transactions
    // - top 8 bits are the address. Per the spec, we shift the address left
    // 1 bit, clear the LSb, and clear the MSb to indicate a write
    // - bottom 8 bits are the data bits being sent for that address, we send them
    RC522_SPI_Transfer((addr<<1)&0x7E);
    RC522_SPI_Transfer(val);

    /* CS HIGH */
    HAL_GPIO_WritePin(MFRC522_CS_PORT,MFRC522_CS_PIN,GPIO_PIN_SET);
}

/*
* Function Name: Read_MFRC522
* Description: From a certain MFRC522 read a byte of data register
* Input Parameters: addr - register address
* Returns: a byte of data read from the module
*/
uchar Read_MFRC522(uchar addr)
{
    uchar val;

    /* CS LOW */
    HAL_GPIO_WritePin(MFRC522_CS_PORT,MFRC522_CS_PIN,GPIO_PIN_RESET);

    // even though we are calling transfer frame once, we are really sending
    // two 8-bit frames smooshed together-- sending two 8 bit frames back to back
    // results in a spike in the select line which will jact with transactions
    // - top 8 bits are the address. Per the spec, we shift the address left
    // 1 bit, clear the LSb, and set the MSb to indicate a read
    // - bottom 8 bits are all 0s on a read per 8.1.2.1 Table 6
    RC522_SPI_Transfer(((addr<<1)&0x7E) | 0x80);
    val = RC522_SPI_Transfer(0x00);

    /* CS HIGH */
    HAL_GPIO_WritePin(MFRC522_CS_PORT,MFRC522_CS_PIN,GPIO_PIN_SET);

    return val;
}

/*
* Function Name: SetBitMask
* Description: Set RC522 register bit
* Input parameters: reg - register address; mask - set value
* Return value: None
*/
void SetBitMask(uchar reg, uchar mask)
{
    uchar tmp;
    tmp = Read_MFRC522(reg);
    Write_MFRC522(reg, tmp | mask); // set bit mask
}

/*
* Function Name: ClearBitMask

```

```

/* Description: clear RC522 register bit
 * Input parameters: reg - register address; mask - clear bit value
 * Return value: None
 */
void ClearBitMask(uchar reg, uchar mask)
{
    uchar tmp;
    tmp = Read_MFRC522(reg);
    Write_MFRC522(reg, tmp & (~mask)); // clear bit mask
}

/*
 * Function Name: AntennaOn
 * Description: Open antennas, each time you start or shut down the natural barrier between the transmitter should be at least 1ms interval
 * Input: None
 * Return value: None
 */
void AntennaOn(void)
{
    Read_MFRC522(TxControlReg);
    SetBitMask(TxControlReg, 0x03);
}

/*
 * Function Name: AntennaOff
 * Description: Close antennas, each time you start or shut down the natural barrier between the transmitter should be at least 1ms interval
 * Input: None
 * Return value: None
 */
void AntennaOff(void)
{
    ClearBitMask(TxControlReg, 0x03);
}

/*
 * Function Name: MFRC522_Reset
 * Description: Reset RC522
 * Input: None
 * Return value: None
 */
void MFRC522_Reset(void)
{
    Write_MFRC522(CommandReg, PCD_RESETPHASE);
}

/*
 * Function Name: MFRC522_Init
 * Description: Initialize RC522
 * Input: None
 * Return value: None
 */
void MFRC522_Init(void)
{
    HAL_GPIO_WritePin(MFRC522_CS_PORT,MFRC522_CS_PIN,GPIO_PIN_SET);
    HAL_GPIO_WritePin(MFRC522_RST_PORT,MFRC522_RST_PIN,GPIO_PIN_SET);
    MFRC522_Reset();

    //Timer: TPrescaler*TreloadVal/6.78MHz = 24ms
    Write_MFRC522(TModeReg, 0x8D);      //Tauto=1; f(Timer) = 6.78MHz/TPreScaler
    Write_MFRC522(TPrescalerReg, 0x3E); //TModeReg[3..0] + TPrescalerReg
    Write_MFRC522(TReloadRegL, 30);
}

```

```

        Write_MFRC522(TReloadRegH, 0);

        Write_MFRC522(TxAutoReg, 0x40);           // force 100% ASK modulation
        Write_MFRC522(ModeReg, 0x3D);             // CRC Initial value 0x6363

        AntennaOn();
    }

/*
 * Function Name: MFRC522_ToCard
 * Description: RC522 and ISO14443 card communication
 * Input Parameters: command - MF522 command word,
 *                   sendData--RC522 sent to the card by the data
 *                   sendLen--Length of data sent
 *                   backData--Received the card returns data,
 *                   backLen--Return data bit length
 * Return value: the successful return MI_OK
 */
uchar MFRC522_ToCard(uchar command, uchar *sendData, uchar sendLen, uchar *backData, uint *backLen)
{
    uchar status = MI_ERR;
    uchar irqEn = 0x00;
    uchar waitIRq = 0x00;
    uchar lastBits;
    uchar n;
    uint i;

    switch (command)
    {
        case PCD_AUTHENT:           // Certification cards close
        {
            irqEn = 0x12;
            waitIRq = 0x10;
            break;
        }
        case PCD_TRANSCEIVE: // Transmit FIFO data
        {
            irqEn = 0x77;
            waitIRq = 0x30;
            break;
        }
        default:
            break;
    }

    Write_MFRC522(CommIEnReg, irqEn|0x80);      // Interrupt request
    ClearBitMask(CommIrqReg, 0x80);              // Clear all interrupt request bit
    SetBitMask(FIFOLevelReg, 0x80);               // FlushBuffer=1, FIFO Initialization

    Write_MFRC522(CommandReg, PCD_IDLE);         // NO action; Cancel the current command

    // Writing data to the FIFO
    for (i=0; i<sendLen; i++)
    {
        Write_MFRC522(FIFODataReg, sendData[i]);
    }

    // Execute the command
    Write_MFRC522(CommandReg, command);
    if (command == PCD_TRANSCEIVE)
    {
        SetBitMask(BitFramingReg, 0x80);          // StartSend=1, transmission of data starts
    }
}

```

```

// Waiting to receive data to complete
    i = 2000; // i according to the clock frequency adjustment, the operator M1 card maximum waiting time 25ms
do
{
    //CommIrqReg[7..0]
    //Set1 TxIRq RxIRq IdleIRq HiAlerIRq LoAlertIRq ErrIRq TimerIRq
    n = Read_MFRC522(CommIrqReg);
    i--;
}
while ((i!=0) && !(n&0x01) && !(n&waitIRq));

ClearBitMask(BitFramingReg, 0x80);                                //StartSend=0

if (i != 0)
{
    if(!(Read_MFRC522(ErrorReg) & 0x1B))//BufferOvfl Collerr CRCErr ProtecolErr
    {
        status = MI_OK;
        if (n & irqEn & 0x01)
        {
            status = MI_NOTAGERR;
        }
    }

    if (command == PCD_TRANSCEIVE)
    {
        n = Read_MFRC522(FIFOLevelReg);
        lastBits = Read_MFRC522(ControlReg) & 0x07;
        if (lastBits)
        {
            *backLen = (n-1)*8 + lastBits;
        }
        else
        {
            *backLen = n*8;
        }
    }

    if (n == 0)
    {
        n = 1;
    }
    if (n > MAX_LEN)
    {
        n = MAX_LEN;
    }
}

// Reading the received data in FIFO
for (i=0; i<n; i++)
{
    backData[i] = Read_MFRC522(FIFODataReg);
}
}
else
{
    status = MI_ERR;
}

//SetBitMask(ControlReg,0x80);      //timer stops
//Write_MFRC522(CommandReg, PCD_IDLE);

return status;
}

```

```

/*
 * Function Name: MFRC522_Request
 * Description: Find cards, read the card type number
 * Input parameters: reqMode - find cards way
 *   TagType - Return Card Type
 *   0x4400 = Mifare_UltraLight
 *   0x0400 = Mifare_One(S50)
 *   0x0200 = Mifare_One(S70)
 *   0x0800 = Mifare_Pro(X)
 *   0x4403 = Mifare_DESFire
 * Return value: the successful return MI_OK
 */
uchar MFRC522_Request(uchar reqMode, uchar *TagType)
{
    uchar status;
    uint backBits; // The received data bits

    Write_MFRC522(BitFramingReg, 0x07); //TxLastBists = BitFramingReg[2..0]

    TagType[0] = reqMode;
    status = MFRC522_ToCard(PCD_TRANSCEIVE, TagType, 1, TagType, &backBits);

    if ((status != MI_OK) || (backBits != 0x10))
    {
        status = MI_ERR;
    }

    return status;
}

/*
 * Function Name: MFRC522_Anticoll
 * Description: Anti-collision detection, reading selected card serial number card
 * Input parameters: serNum - returns 4 bytes card serial number, the first 5 bytes for the checksum byte
 * Return value: the successful return MI_OK
 */
uchar MFRC522_Anticoll(uchar *serNum)
{
    uchar status;
    uchar i;
    uchar serNumCheck=0;
    uint unLen;

    Write_MFRC522(BitFramingReg, 0x00); //TxLastBists = BitFramingReg[2..0]

    serNum[0] = PICC_ANTICOLL;
    serNum[1] = 0x20;
    status = MFRC522_ToCard(PCD_TRANSCEIVE, serNum, 2, serNum, &unLen);

    if (status == MI_OK)
    {
        //Check card serial number
        for (i=0; i<4; i++)
        {
            serNumCheck ^= serNum[i];
        }
        if (serNumCheck != serNum[1])
        {
            status = MI_ERR;
        }
    }

    return status;
}

```

```

}

/*
 * Function Name: CalulateCRC
 * Description: CRC calculation with MF522
 * Input parameters: pIndata - To read the CRC data, len - the data length, pOutData - CRC calculation results
 * Return value: None
 */
void CalulateCRC(uchar *pIndata, uchar len, uchar *pOutData)
{
    uchar i, n;

    ClearBitMask(DivIrqReg, 0x04);                                //CRCIrq = 0
    SetBitMask(FIFOLevelReg, 0x80);                                //Clear the FIFO pointer

    //Writing data to the FIFO
    for (i=0; i<len; i++)
    {
        Write_MFRC522(FIFODataReg, *(pIndata+i));
    }
    Write_MFRC522(CommandReg, PCD_CALCCRC);

    //Wait CRC calculation is complete
    i = 0xFF;
    do
    {
        n = Read_MFRC522(DivIrqReg);
        i--;
    }
    while ((i!=0) && !(n&0x04));                                //CRCIrq = 1

    //Read CRC calculation result
    pOutData[0] = Read_MFRC522(CRCResultRegL);
    pOutData[1] = Read_MFRC522(CRCResultRegH);
}

/*
 * Function Name: MFRC522_SelectTag
 * Description: election card, read the card memory capacity
 * Input parameters: serNum - Incoming card serial number
 * Return value: the successful return of card capacity
 */
uchar MFRC522_SelectTag(uchar *serNum)
{
    uchar i;
    uchar status;
    uchar size;
    uint recvBits;
    uchar buffer[9];

    //ClearBitMask(Status2Reg, 0x08);                                //MFCrypto1On=0

    buffer[0] = PICC_SELECTTAG;
    buffer[1] = 0x70;
    for (i=0; i<5; i++)
    {
        buffer[i+2] = *(serNum+i);
    }
    CalulateCRC(buffer, 7, &buffer[7]);
    status = MFRC522_ToCard(PCD_TRANSCEIVE, buffer, 9, buffer, &recvBits);

    if ((status == MI_OK) && (recvBits == 0x18))
    {
        size = buffer[0];
    }
}

```

```

        }

    else
    {
        size = 0;
    }

    return size;
}

/*
* Function Name: MFRC522_Auth
* Description: Verify card password
* Input parameters: authMode - Password Authentication Mode
*                   0x60 = A key authentication
*                   0x61 = Authentication Key B
*                   BlockAddr--Block address
*                   Sectorkey--Sector password
*                   serNum--Card serial number, 4-byte
* Return value: the successful return MI_OK
*/
uchar MFRC522_Auth(uchar authMode, uchar BlockAddr, uchar *Sectorkey, uchar *serNum)
{
    uchar status;
    uint recvBits;
    uchar i;
    uchar buff[12];

    //Verify the command block address + sector + password + card serial number
    buff[0] = authMode;
    buff[1] = BlockAddr;
    for (i=0; i<6; i++)
    {
        buff[i+2] = *(Sectorkey+i);
    }
    for (i=0; i<4; i++)
    {
        buff[i+8] = *(serNum+i);
    }
    status = MFRC522_ToCard(PCD_AUTHENT, buff, 12, buff, &recvBits);

    if ((status != MI_OK) || (!(Read_MFRC522(Status2Reg) & 0x08)))
    {
        status = MI_ERR;
    }

    return status;
}

/*
* Function Name: MFRC522_Read
* Description: Read block data
* Input parameters: blockAddr - block address; recvData - read block data
* Return value: the successful return MI_OK
*/
uchar MFRC522_Read(uchar blockAddr, uchar *recvData)
{
    uchar status;
    uint unLen;

    recvData[0] = PICC_READ;
    recvData[1] = blockAddr;
    CalulateCRC(recvData,2, &recvData[2]);
    status = MFRC522_ToCard(PCD_TRANSCEIVE, recvData, 4, recvData, &unLen);
}

```

```

if ((status != MI_OK) || (unLen != 0x90))
{
    status = MI_ERR;
}

return status;
}

/*
* Function Name: MFRC522_Write
* Description: Write block data
* Input parameters: blockAddr - block address; writeData - to 16-byte data block write
* Return value: the successful return MI_OK
*/
uchar MFRC522_Write(uchar blockAddr, uchar *writeData)
{
    uchar status;
    uint recvBits;
    uchar i;
    uchar buff[18];

    buff[0] = PICC_WRITE;
    buff[1] = blockAddr;
    CalulateCRC(buff, 2, &buff[2]);
    status = MFRC522_ToCard(PCD_TRANSCEIVE, buff, 4, buff, &recvBits);

    if ((status != MI_OK) || (recvBits != 4) || ((buff[0] & 0x0F) != 0x0A))
    {
        status = MI_ERR;
    }

    if (status == MI_OK)
    {
        for (i=0; i<16; i++)           //Data to the FIFO write 16Byte
        {
            buff[i] = *(writeData+i);
        }
        CalulateCRC(buff, 16, &buff[16]);
        status = MFRC522_ToCard(PCD_TRANSCEIVE, buff, 18, buff, &recvBits);

        if ((status != MI_OK) || (recvBits != 4) || ((buff[0] & 0x0F) != 0x0A))
        {
            status = MI_ERR;
        }
    }

    return status;
}

/*
* Function Name: MFRC522_Halt
* Description: Command card into hibernation
* Input: None
* Return value: None
*/
void MFRC522_Halt(void)
{
    uint unLen;
    uchar buff[4];

    buff[0] = PICC_HALT;
    buff[1] = 0;
    CalulateCRC(buff, 2, &buff[2]);
}

```

```

    MFRC522_ToCard(PCD_TRANSCEIVE, buff, 4, buff,&unLen);
}

```

CODE 3: RFID-RC522.c

c. Giải thích RC522.c

▪ **RC522_SPI_Transfer**

- **Mục đích:** Truyền và nhận dữ liệu qua giao thức SPI với module RC522.
 - **Chi tiết:** Dữ liệu được truyền qua SPI và một byte phản hồi được trả về từ module.
 - **Đối số:** data - dữ liệu gửi đi.
 - **Trả về:** Byte dữ liệu đọc được từ module RC522.
-

▪ **Write_MFRC522**

- **Mục đích:** Ghi một byte dữ liệu vào một thanh ghi của module RC522.
 - **Chi tiết:**
 - Địa chỉ thanh ghi addr được dịch sang một giá trị phù hợp để chỉ định một địa chỉ ghi trong giao thức SPI.
 - Sau đó, giá trị val sẽ được ghi vào thanh ghi này.
 - Điều khiển CS (chip select) để đảm bảo rằng module chỉ nhận lệnh từ vi điều khiển.
-

▪ **Read_MFRC522**

- **Mục đích:** Đọc một byte dữ liệu từ một thanh ghi của module RC522.
 - **Chi tiết:**
 - Tương tự như Write_MFRC522, nhưng lệnh này đọc dữ liệu từ thanh ghi và trả lại giá trị.
 - Điều khiển CS và gửi yêu cầu đọc đến module RC522.
-

▪ **SetBitMask và ClearBitMask**

- **Mục đích:** Thiết lập hoặc xóa một bit trong thanh ghi của module RC522.

- **Chi tiết:** SetBitMask sẽ đặt các bit mong muốn (sử dụng phép OR với giá trị hiện tại của thanh ghi), trong khi ClearBitMask xóa các bit mong muốn (sử dụng phép AND với giá trị phủ định của bit cần xóa).
-

▪ **AntennaOn và AntennaOff**

- **Mục đích:** Bật hoặc tắt anten của module RC522.
 - **Chi tiết:** Điều khiển việc bật/tắt anten để thực hiện các thao tác đọc thẻ.
-

▪ **MFRC522_Reset**

- **Mục đích:** Reset module RC522.
 - **Chi tiết:** Gửi lệnh reset đến module RC522.
-

▪ **MFRC522_Init**

- **Mục đích:** Khởi tạo module RC522.
 - **Chi tiết:** Cấu hình các thanh ghi của module để sẵn sàng cho các thao tác đọc và ghi. Cấu hình các bộ đếm, thời gian chờ, và bật anten.
-

▪ **MFRC522_ToCard**

- **Mục đích:** Thực hiện các giao tiếp với thẻ RFID.
- **Chi tiết:** Gửi lệnh và dữ liệu đến thẻ RFID thông qua RC522 và nhận phản hồi từ thẻ.
- **Các tham số:** command là lệnh cần thực hiện (như xác thực thẻ, truyền dữ liệu, v.v.), sendData là dữ liệu gửi đi, backData là nơi lưu dữ liệu nhận lại.

▪ **MFRC522_Request**

- **Mục đích:** Tìm kiếm thẻ RFID trong vùng đọc của module RC522.
 - **Chi tiết:** Gửi lệnh yêu cầu thẻ và nhận lại loại thẻ. Lệnh này sẽ tìm kiếm thẻ RFID trong phạm vi và xác định loại thẻ (Mifare UltraLight, Mifare S50, v.v.).
-

▪ **MFRC522_Anticoll**

- **Mục đích:** Phát hiện và đọc số serial của thẻ RFID.

- **Chi tiết:** Thực hiện kiểm tra chống va chạm (anti-collision) để đảm bảo rằng chỉ có một thẻ được phát hiện tại một thời điểm.
-

▪ **CalulateCRC**

- **Mục đích:** Tính toán giá trị CRC cho một chuỗi dữ liệu.
 - **Chi tiết:** Tính toán và trả về giá trị CRC cho một mảng dữ liệu, được sử dụng để kiểm tra tính toàn vẹn của dữ liệu khi giao tiếp với thẻ.
-

▪ **MFRC522_SelectTag**

- **Mục đích:** Lựa chọn thẻ RFID và đọc dung lượng bộ nhớ của nó.
 - **Chi tiết:** Lựa chọn một thẻ và đọc dung lượng bộ nhớ của thẻ sau khi xác thực.
-

▪ **MFRC522_Auth**

- **Mục đích:** Xác thực thẻ RFID với mật khẩu.
 - **Chi tiết:** Xác thực bằng cách sử dụng một trong các phương thức xác thực A hoặc B cho các khối bộ nhớ của thẻ.
-

▪ **MFRC522_Read**

- **Mục đích:** Đọc dữ liệu từ một khối bộ nhớ trên thẻ RFID.
 - **Chi tiết:** Gửi yêu cầu đọc từ một khối bộ nhớ của thẻ và trả lại dữ liệu.
-

▪ **MFRC522_Write**

- **Mục đích:** Ghi dữ liệu vào một khối bộ nhớ trên thẻ RFID.
 - **Chi tiết:** Gửi dữ liệu vào một khối bộ nhớ của thẻ RFID và xác nhận việc ghi thành công.
-

▪ **MFRC522_Halt**

- **Mục đích:** Đưa thẻ RFID vào trạng thái ngủ (hibernation).
- **Chi tiết:** Gửi lệnh để đưa thẻ RFID vào trạng thái ngừng hoạt động, tiết kiệm năng lượng.

d. Sử dụng RC522 trong main code

```

void quetthe() {
    i = 5;
    timeout = HAL_GetTick() + 100000;
    uint8_t str[5];
    const uint8_t expectedID[5] = { 254, 226, 48, 3, 47 }; // Thẻ đúng

    drawPaymentScreen();

    while (HAL_GetTick() < timeout) {
        // Kiểm tra nếu đọc thẻ thành công
        if (MFRC522_Request(PICC_REQIDL, str) == MI_OK) // Kiểm tra xem có thẻ nào trong vùng
            quét không. Nếu có, lưu thông tin vào str.
        && MFRC522_Anticoll(str) == MI_OK) { //Đọc số seri (UID) của thẻ để tránh trùng lặp.
            // Nếu cả hai hàm trả về MI_OK, nghĩa là đọc thẻ thành công, thì tiếp tục.
            memcpy(sNum, str, 5); //Sao chép 5 byte số seri UID vào biến sNum.

            // Kiểm tra nếu thẻ hợp lệ
            if (memcmp(str, expectedID, 5) == 0) { //Nếu 5 byte trong str trùng với expectedID, nghĩa là
                thẻ hợp lệ.
                drawCardSuccess();
                layhanhly();
                return; // Thẻ hợp lệ, thoát hàm ngay
            } else {
                i = 16;
                drawCardFail();
            }
        }

        HAL_Delay(500); // Tránh spam UART
    }
    locker(); // Chí khóa nếu hết thời gian mà không có thẻ hợp lệ
}

```

- **MFRC522_Request(PICC_REQIDL, str):**
 - Đây là hàm kiểm tra xem có thẻ RFID nào trong vùng quét không. Nếu có, nó lưu thông tin thẻ vào mảng str.
 - PICC_REQIDL là lệnh yêu cầu thẻ trong chế độ chuẩn, giúp phát hiện các thẻ đang ở gần.

-
- **MFRC522_Anticoll(str):**
 - Hàm này đọc số seri (UID) của thẻ RFID để tránh trùng lặp. Nó giúp xác định thẻ một cách duy nhất.

- Nếu thành công, hàm này trả về MI_OK và số seri của thẻ sẽ được lưu vào str.
-

▪ **Kiểm tra UID:**

- Mảng expectedID chứa một dãy 5 byte đại diện cho một UID hợp lệ của thẻ.
 - Hàm memcmp so sánh UID của thẻ (str) với expectedID để xác nhận thẻ hợp lệ.
 - Nếu thẻ hợp lệ (so sánh thành công), hàm drawCardSuccess() được gọi và chương trình tiếp tục với layanhly().
 - Nếu thẻ không hợp lệ, hàm drawCardFail() được gọi để hiển thị thông báo lỗi.
-

▪ **HAL_Delay(500):**

- Chờ 500ms trước khi tiếp tục vòng lặp, nhằm tránh việc quét liên tục và tạo ra thông báo lỗi không cần thiết.
-

▪ **Kết thúc:**

- Nếu không có thẻ hợp lệ được quét trong khoảng thời gian nhất định, hàm locker() được gọi để quay lại trạng thái ban đầu.
-

d. Code ILI934.h

```
#ifndef __FONTS_H__
#define __FONTS_H__

#include <stdint.h>

#define FONT1  Arial_Narrow8x12
#define FONT2  Arial_Narrow10x13
#define FONT3  Arial_Narrow12x16
#define FONT4  Arial_Narrow15x19

extern const uint8_t Arial_Narrow8x12[];
extern const uint8_t Arial_Narrow10x13[];
extern const uint8_t Arial_Narrow12x16[];
extern const uint8_t Arial_Narrow15x19[];

#endif // __FONTS_H__
```

CODE 4: ILI934-fonts.h

```
#ifndef ILI9341_GFX_H
#define ILI9341_GFX_H

#include "stm32f1xx_hal.h"
#include "fonts.h"

#define HORIZONTAL_IMAGE    0
#define VERTICAL_IMAGE       1

void ILI9341_DrawHollowCircle(uint16_t X, uint16_t Y, uint16_t radius, uint16_t color);
void ILI9341_DrawFilledCircle(uint16_t X, uint16_t Y, uint16_t radius, uint16_t color);
void ILI9341_DrawHollowRectangleCoord(uint16_t X0, uint16_t Y0, uint16_t X1, uint16_t Y1, uint16_t color);
void ILI9341_DrawFilledRectangleCoord(uint16_t X0, uint16_t Y0, uint16_t X1, uint16_t Y1, uint16_t color);
void ILI9341_DrawChar(char ch, const uint8_t font[], uint16_t X, uint16_t Y, uint16_t color, uint16_t bgcolor);
void ILI9341_DrawText(const char* str, const uint8_t font[], uint16_t X, uint16_t Y, uint16_t color, uint16_t bgcolor);
void ILI9341_DrawImage(const uint8_t* image, uint8_t orientation);

#endif
```

CODE 5: ILI9341_GFX.h

e. Code ILI934.c

```
#include "ILI9341_STM32_Driver.h"
#include "ILI9341_GFX.h"

/* imprecise small delay */
STATIC_INLINE void DelayUs(volatile uint32_t us)
{
    us *= (SystemCoreClock / 1000000);
    while (us--);
}

void ILI9341_DrawHollowCircle(uint16_t X, uint16_t Y, uint16_t radius, uint16_t color)
{
    int x = radius-1;
    int y = 0;
    int dx = 1;
    int dy = 1;
    int err = dx - (radius << 1);

    while (x >= y)
    {
        ILI9341_DrawPixel(X + x, Y + y, color);
        ILI9341_DrawPixel(X + y, Y + x, color);
        ILI9341_DrawPixel(X - y, Y + x, color);
        ILI9341_DrawPixel(X - x, Y + y, color);
        ILI9341_DrawPixel(X - x, Y - y, color);
        ILI9341_DrawPixel(X - y, Y - x, color);
        ILI9341_DrawPixel(X + y, Y - x, color);
        ILI9341_DrawPixel(X + x, Y - y, color);

        if (err <= 0)
        {
            y++;
            err += dy;
            dy += 2;
        }

        if (err > 0)
        {
            x--;
            dx += 2;
        }
    }
}
```

```

        err += (-radius << 1) + dx;
    }
}

void ILI9341_DrawFilledCircle(uint16_t X, uint16_t Y, uint16_t radius, uint16_t color)
{
    int x = radius;
    int y = 0;
    int xChange = 1 - (radius << 1);
    int yChange = 0;
    int radiusError = 0;

    while (x >= y)
    {
        for (int i = X - x; i <= X + x; i++)
        {
            ILI9341_DrawPixel(i, Y + y,color);
            ILI9341_DrawPixel(i, Y - y,color);
        }

        for (int i = X - y; i <= X + y; i++)
        {
            ILI9341_DrawPixel(i, Y + x,color);
            ILI9341_DrawPixel(i, Y - x,color);
        }

        y++;
        radiusError += yChange;
        yChange += 2;

        if (((radiusError << 1) + xChange) > 0)
        {
            x--;
            radiusError += xChange;
            xChange += 2;
        }
    }
}

void ILI9341_DrawHollowRectangleCoord(uint16_t X0, uint16_t Y0, uint16_t X1, uint16_t Y1, uint16_t color)
{
    uint16_t xLen = 0;
    uint16_t yLen = 0;
    uint8_t negX = 0;
    uint8_t negY = 0;
    float negCalc = 0;

    negCalc = X1 - X0;
    if(negCalc < 0) negX = 1;
    negCalc = 0;

    negCalc = Y1 - Y0;
    if(negCalc < 0) negY = 1;

    //DRAW HORIZONTAL!
    if(!negX)
    {
        xLen = X1 - X0;
    }
    else
    {
        xLen = X0 - X1;
    }
}

```

```

        }

    ILI9341_DrawHLine(X0, Y0, xLen, color);
    ILI9341_DrawHLine(X0, Y1, xLen, color);

    //DRAW VERTICAL!
    if(!negY)
    {
        yLen = Y1 - Y0;
    }
    else
    {
        yLen = Y0 - Y1;
    }

    ILI9341_DrawVLine(X0, Y0, yLen, color);
    ILI9341_DrawVLine(X1, Y0, yLen, color);

    if((xLen > 0)||(yLen > 0))
    {
        ILI9341_DrawPixel(X1, Y1, color);
    }
}

void ILI9341_DrawFilledRectangleCoord(uint16_t X0, uint16_t Y0, uint16_t X1, uint16_t Y1, uint16_t color)
{
    uint16_t xLen = 0;
    uint16_t yLen = 0;
    uint8_t negX = 0;
    uint8_t negY = 0;
    int32_t negCalc = 0;
    uint16_t X0True = 0;
    uint16_t Y0True = 0;

    negCalc = X1 - X0;
    if(negCalc < 0) negX = 1;
    negCalc = 0;

    negCalc = Y1 - Y0;
    if(negCalc < 0) negY = 1;

    if(!negX)
    {
        xLen = X1 - X0;
        X0True = X0;
    }
    else
    {
        xLen = X0 - X1;
        X0True = X1;
    }

    if(!negY)
    {
        yLen = Y1 - Y0;
        Y0True = Y0;
    }
    else
    {
        yLen = Y0 - Y1;
        Y0True = Y1;
    }

    ILI9341_DrawRectangle(X0True, Y0True, xLen, yLen, color);
}

```

```

void ILI9341_DrawChar(char ch, const uint8_t font[], uint16_t X, uint16_t Y, uint16_t color, uint16_t bgcolor)
{
    if ((ch < 31) || (ch > 127)) return;

    uint8_t fOffset, fWidth, fHeight, fBPL;
    uint8_t *tempChar;

    fOffset = font[0];
    fWidth = font[1];
    fHeight = font[2];
    fBPL = font[3];

    tempChar = (uint8_t*)&font[((ch - 0x20) * fOffset) + 4]; /* Current Character = Meta + (Character Index * Offset) */

    /* Clear background first */
    ILI9341_DrawRectangle(X, Y, fWidth, fHeight, bgcolor);

    for (int j=0; j < fHeight; j++)
    {
        for (int i=0; i < fWidth; i++)
        {
            uint8_t z = tempChar[fBPL * i + ((j & 0xF8) >> 3) + 1]; /* (j & 0xF8) >> 3, increase one by 8-bits */
            uint8_t b = 1 << (j & 0x07);
            if ((z & b) != 0x00)
            {
                ILI9341_DrawPixel(X+i, Y+j, color);
            }
        }
    }
}

void ILI9341_DrawText(const char* str, const uint8_t font[], uint16_t X, uint16_t Y, uint16_t color, uint16_t bgcolor)
{
    uint8_t charWidth; /* Width of character */
    uint8_t fOffset = font[0]; /* Offset of character */
    uint8_t fWidth = font[1]; /* Width of font */

    while (*str)
    {
        ILI9341_DrawChar(*str, font, X, Y, color, bgcolor);

        /* Check character width and calculate proper position */
        uint8_t *tempChar = (uint8_t*)&font[((*str - 0x20) * fOffset) + 4];
        charWidth = tempChar[0];

        if(charWidth + 2 < fWidth)
        {
            /* If character width is smaller than font width */
            X += (charWidth + 2);
        }
        else
        {
            X += fWidth;
        }

        str++;
    }
}

void ILI9341_DrawImage(const uint8_t* image, uint8_t orientation)
{
}

```

```

if(orientation == SCREEN_HORIZONTAL_1)
{
    ILI9341_SetRotation(SCREEN_HORIZONTAL_1);
    ILI9341_SetAddress(0,0,ILI9341_SCREEN_WIDTH,ILI9341_SCREEN_HEIGHT);
}
else if(orientation == SCREEN_HORIZONTAL_2)
{
    ILI9341_SetRotation(SCCREEN_HORIZONTAL_2);
    ILI9341_SetAddress(0,0,ILI9341_SCREEN_WIDTH,ILI9341_SCREEN_HEIGHT);
}
else if(orientation == SCREEN_VERTICAL_2)
{
    ILI9341_SetRotation(SCCREEN_VERTICAL_2);
    ILI9341_SetAddress(0,0,ILI9341_SCREEN_HEIGHT,ILI9341_SCREEN_WIDTH);
}
else if(orientation == SCREEN_VERTICAL_1)
{
    ILI9341_SetRotation(SCCREEN_VERTICAL_1);
    ILI9341_SetAddress(0,0,ILI9341_SCREEN_HEIGHT,ILI9341_SCREEN_WIDTH);
}

uint32_t counter = 0;
for(uint32_t i = 0; i < ILI9341_SCREEN_WIDTH*ILI9341_SCREEN_HEIGHT*2/BURST_MAX_SIZE;
i++)
{
    ILI9341_WriteBuffer((uint8_t*)(image + counter), BURST_MAX_SIZE);
    counter += BURST_MAX_SIZE;

    /* DMA Tx is too fast, It needs some delay */
    DelayUs(1);
}
}

```

CODE 6: *ILI9341_GFX.c*

f. Giải thích *ILI9341.c*

- **Delay Function (DelayUs):**

```

STATIC_INLINE void DelayUs(volatile uint32_t us)
{
    us *= (SystemCoreClock / 1000000);
    while (us--);
}

```

- Hàm này tạo ra một độ trễ nhỏ, có thể sử dụng để đồng bộ hóa khi truyền dữ liệu hoặc vẽ hình. Thời gian trễ được tính theo hệ thống đồng hồ của vi điều khiển (SystemCoreClock), chia cho 1 triệu để chuyển sang micro giây.

- **Hàm vẽ hình tròn rỗng (ILI9341_DrawHollowCircle):**

```

void ILI9341_DrawHollowCircle(uint16_t X, uint16_t Y, uint16_t radius, uint16_t color)
{
    // Vẽ hình tròn rỗng với tâm (X, Y) và bán kính radius
}

```

- Hàm này sử dụng thuật toán "Midpoint Circle" để vẽ hình tròn rỗng. Các điểm của hình tròn được vẽ xung quanh tâm (X, Y) với bán kính radius và màu sắc color. Các pixel được vẽ ở 8 điểm đối xứng nhau qua các trục của hình tròn.

- **Hàm vẽ hình tròn đặc (ILI9341_DrawFilledCircle):** void ILI9341_DrawFilledCircle(uint16_t X, uint16_t Y, uint16_t radius, uint16_t color)

```
{
    // Vẽ hình tròn đặc với tâm (X, Y) và bán kính radius
}
```

- Hàm này vẽ hình tròn đặc, tức là không chỉ vẽ biên mà còn vẽ toàn bộ vùng trong hình tròn. Thuật toán vẽ được dựa trên "Midpoint Circle" nhưng thay vì vẽ các điểm ở biên, nó vẽ tất cả các điểm giữa tâm và biên.

- **Hàm vẽ hình chữ nhật rỗng (ILI9341_DrawHollowRectangleCoord):**

```
void ILI9341_DrawHollowRectangleCoord(uint16_t X0, uint16_t Y0, uint16_t X1, uint16_t Y1,
uint16_t color)
{
    // Vẽ hình chữ nhật rỗng với các điểm góc trái (X0, Y0) và phải (X1, Y1)
}
```

- Hàm này vẽ một hình chữ nhật rỗng (chỉ vẽ biên ngoài). Các đường biên ngang và dọc của hình chữ nhật được vẽ từ các điểm (X0, Y0) đến (X1, Y1). Hệ thống xử lý sự âm của chiều dài và chiều rộng của hình chữ nhật.

- **Hàm vẽ hình chữ nhật đặc (ILI9341_DrawFilledRectangleCoord):**

```
void ILI9341_DrawFilledRectangleCoord(uint16_t X0, uint16_t Y0, uint16_t X1, uint16_t Y1,
uint16_t color)
{
    // Vẽ hình chữ nhật đặc với các điểm góc trái (X0, Y0) và phải (X1, Y1)
}
```

- Hàm này vẽ một hình chữ nhật đặc (với toàn bộ khu vực giữa các điểm góc). Nó vẽ các đường ngang từ X0 đến X1 tại các vị trí Y0 và Y1, và vẽ các cột dọc ở các vị trí tương ứng.

- **Hàm vẽ ký tự (ILI9341_DrawChar):**

```
void ILI9341_DrawChar(char ch, const uint8_t font[], uint16_t X, uint16_t Y, uint16_t color, uint16_t bgcolor)
{
    // Vẽ ký tự trên màn hình tại vị trí (X, Y) với font, màu nền, và màu chữ
}
```

- Hàm này vẽ một ký tự lên màn hình. Các ký tự được vẽ từ font bitmap, trong đó mỗi ký tự được mã hóa theo một bảng font đã cho. Hàm sử dụng các bit trong font bitmap để xác định các pixel cần vẽ.

- **Hàm vẽ chuỗi ký tự (ILI9341_DrawText):**

- void ILI9341_DrawText(const char* str, const uint8_t font[], uint16_t X, uint16_t Y, uint16_t color, uint16_t bgcolor)
- {
- // Vẽ một chuỗi ký tự tại vị trí (X, Y)
- }

- Hàm này vẽ một chuỗi ký tự lên màn hình, sử dụng font bitmap đã cho. Nó lặp qua từng ký tự trong chuỗi và vẽ từng ký tự ở các vị trí xác định, di chuyển sang phải sau khi vẽ mỗi ký tự.

- **Hàm vẽ hình ảnh (ILI9341_DrawImage):**

```
void ILI9341_DrawImage(const uint8_t* image, uint8_t orientation)
{
    // Vẽ hình ảnh lên màn hình với các chế độ xoay khác nhau
}
```

- Hàm này vẽ một hình ảnh lên màn hình. Hình ảnh được truyền dưới dạng một mảng các byte. Hàm hỗ trợ các chế độ xoay màn hình khác nhau và vẽ toàn bộ hình ảnh lên màn hình theo chế độ đã chọn.

g. Các driver của ILI9341

```
#ifndef ILI9341_STM32_DRIVER_H
#define ILI9341_STM32_DRIVER_H

#include "stm32f1xx_hal.h"

extern SPI_HandleTypeDef hspi2;

#define ILI9341_SCREEN_HEIGHT      320
#define ILI9341_SCREEN_WIDTH       320
```

```

/* PIN Configuration */
#define HSPI_INSTANCE          &hspi2
#define LCD_CS_PORT             GPIOB
#define LCD_CS_PIN               GPIO_PIN_11
#define LCD_DC_PORT              GPIOB
#define LCD_DC_PIN               GPIO_PIN_10
#define LCD_RST_PORT             GPIOB
#define LCD_RST_PIN               GPIO_PIN_1

#define BURST_MAX_SIZE           500
#define BLACK                      0x0000
#define NAVY                      0x000F
#define DARKGREEN                 0x03E0
#define DARKCYAN                  0x03EF
#define MAROON                     0x7800
#define PURPLE                     0x780F
#define OLIVE                      0x7BE0
#define LIGHTGREY                  0xC618
#define DARKGREY                   0x7BEF
#define BLUE                       0x001F
#define GREEN                      0x07E0
#define CYAN                       0x07FF
#define RED                        0xF800
#define MAGENTA                    0xF81F
#define YELLOW                     0xFFE0
#define WHITE                      0xFFFF
#define ORANGE                     0xFD20
#define GREENYELLOW                0xAFE5
#define PINK                       0xF81F

#define SCREEN_VERTICAL_1          0
#define SCREEN_HORIZONTAL_1        1
#define SCREEN_VERTICAL_2          2
#define SCREEN_HORIZONTAL_2        3

void ILI9341_WriteCommand(uint8_t cmd);
void ILI9341_WriteData(uint8_t data);
void ILI9341_WriteBuffer(uint8_t *buffer, uint16_t len);
void ILI9341_Reset(void);
void ILI9341_Enable(void);
void ILI9341_Init(void);
void ILI9341_SetRotation(uint8_t rotation);
void ILI9341_SetAddress(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2);
void ILI9341_DrawColor(uint16_t color);
void ILI9341_DrawColorBurst(uint16_t color, uint32_t size);
void ILI9341_FillScreen(uint16_t color);
void ILI9341_DrawPixel(uint16_t x, uint16_t y, uint16_t color);
void ILI9341_DrawRectangle(uint16_t x, uint16_t y, uint16_t width, uint16_t height, uint16_t color);
void ILI9341_DrawHLine(uint16_t x, uint16_t y, uint16_t width, uint16_t color);
void ILI9341_DrawVLine(uint16_t x, uint16_t y, uint16_t height, uint16_t color);
#endif

```

CODE 7: ILI9341_STM32_Driver.h

```

#include "ILI9341_STM32_Driver.h"

volatile uint16_t LCD_HEIGHT = ILI9341_SCREEN_HEIGHT;
volatile uint16_t LCD_WIDTH   = ILI9341_SCREEN_WIDTH;

void HAL_SPI_TxCpltCallback(SPI_HandleTypeDef *hspi)
{
    /* Deselect when Tx Complete */
    if(hspi == HSPI_INSTANCE)

```

```

{
    HAL_GPIO_WritePin(LCD_CS_PORT, LCD_CS_PIN, GPIO_PIN_SET);
}

static void ILI9341_SPI_Tx(uint8_t data)
{
    while(!_HAL_SPI_GET_FLAG(HSPI_INSTANCE, SPI_FLAG_TXE));
    HAL_SPI_Transmit_DMA(HSPI_INSTANCE, &data, 1);
    //HAL_SPI_Transmit(HSPI_INSTANCE, &data, 1, 10);
}

static void ILI9341_SPI_TxBuffer(uint8_t *buffer, uint16_t len)
{
    while(!_HAL_SPI_GET_FLAG(HSPI_INSTANCE, SPI_FLAG_TXE));
    HAL_SPI_Transmit_DMA(HSPI_INSTANCE, buffer, len);
    //HAL_SPI_Transmit(HSPI_INSTANCE, buffer, len, 10);
}

void ILI9341_WriteCommand(uint8_t cmd)
{
    HAL_GPIO_WritePin(LCD_DC_PORT, LCD_DC_PIN, GPIO_PIN_RESET);           //command
    HAL_GPIO_WritePin(LCD_CS_PORT, LCD_CS_PIN, GPIO_PIN_RESET);           //select
    ILI9341_SPI_Tx(cmd);
    //HAL_GPIO_WritePin(LCD_CS_PORT, LCD_CS_PIN, GPIO_PIN_SET);           //deselect
}

void ILI9341_WriteData(uint8_t data)
{
    HAL_GPIO_WritePin(LCD_DC_PORT, LCD_DC_PIN, GPIO_PIN_SET);           //data
    HAL_GPIO_WritePin(LCD_CS_PORT, LCD_CS_PIN, GPIO_PIN_RESET);           //select
    ILI9341_SPI_Tx(data);
    //HAL_GPIO_WritePin(LCD_CS_PORT, LCD_CS_PIN, GPIO_PIN_SET);           //deselect
}

void ILI9341_WriteBuffer(uint8_t *buffer, uint16_t len)
{
    HAL_GPIO_WritePin(LCD_DC_PORT, LCD_DC_PIN, GPIO_PIN_SET);           //data
    HAL_GPIO_WritePin(LCD_CS_PORT, LCD_CS_PIN, GPIO_PIN_RESET);           //select
    ILI9341_SPI_TxBuffer(buffer, len);
    //HAL_GPIO_WritePin(LCD_CS_PORT, LCD_CS_PIN, GPIO_PIN_SET);           //deselect
}

void ILI9341_SetAddress(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2)
{
    uint8_t buffer[4];
    buffer[0] = x1 >> 8;
    buffer[1] = x1;
    buffer[2] = x2 >> 8;
    buffer[3] = x2;

    ILI9341_WriteCommand(0x2A);
    ILI9341_WriteBuffer(buffer, sizeof(buffer));

    buffer[0] = y1 >> 8;
    buffer[1] = y1;
    buffer[2] = y2 >> 8;
    buffer[3] = y2;

    ILI9341_WriteCommand(0x2B);
    ILI9341_WriteBuffer(buffer, sizeof(buffer));

    ILI9341_WriteCommand(0x2C);
}

```

```

void ILI9341_Reset(void)
{
    HAL_GPIO_WritePin(LCD_RST_PORT, LCD_RST_PIN, GPIO_PIN_RESET); //Disable
    HAL_Delay(10);
    HAL_GPIO_WritePin(LCD_CS_PORT, LCD_CS_PIN, GPIO_PIN_RESET); //Select
    HAL_Delay(10);
    HAL_GPIO_WritePin(LCD_RST_PORT, LCD_RST_PIN, GPIO_PIN_SET); //Enable
    HAL_GPIO_WritePin(LCD_CS_PORT, LCD_CS_PIN, GPIO_PIN_SET); //Deselect
}

void ILI9341_Enable(void)
{
    HAL_GPIO_WritePin(LCD_RST_PORT, LCD_RST_PIN, GPIO_PIN_SET); //Enable
}

void ILI9341_Init(void)
{
    ILI9341_Enable();
    ILI9341_Reset();

    //SOFTWARE RESET
    ILI9341_WriteCommand(0x01);
    HAL_Delay(10);

    //POWER CONTROL A
    ILI9341_WriteCommand(0xCB);
    ILI9341_WriteData(0x39);
    ILI9341_WriteData(0x2C);
    ILI9341_WriteData(0x00);
    ILI9341_WriteData(0x34);
    ILI9341_WriteData(0x02);

    //POWER CONTROL B
    ILI9341_WriteCommand(0xCF);
    ILI9341_WriteData(0x00);
    ILI9341_WriteData(0xC1);
    ILI9341_WriteData(0x30);

    //DRIVER TIMING CONTROL A
    ILI9341_WriteCommand(0xE8);
    ILI9341_WriteData(0x85);
    ILI9341_WriteData(0x00);
    ILI9341_WriteData(0x78);

    //DRIVER TIMING CONTROL B
    ILI9341_WriteCommand(0xEA);
    ILI9341_WriteData(0x00);
    ILI9341_WriteData(0x00);

    //POWER ON SEQUENCE CONTROL
    ILI9341_WriteCommand(0xED);
    ILI9341_WriteData(0x64);
    ILI9341_WriteData(0x03);
    ILI9341_WriteData(0x12);
    ILI9341_WriteData(0x81);

    //PUMP RATIO CONTROL
    ILI9341_WriteCommand(0xF7);
    ILI9341_WriteData(0x20);

    //POWER CONTROL,VRH[5:0]
    ILI9341_WriteCommand(0xC0);
    ILI9341_WriteData(0x23);
}

```

```

//POWER CONTROL,SAP[2:0];BT[3:0]
ILI9341_WriteCommand(0xC1);
ILI9341_WriteData(0x10);

//VCM CONTROL
ILI9341_WriteCommand(0xC5);
ILI9341_WriteData(0x3E);
ILI9341_WriteData(0x28);

//VCM CONTROL 2
ILI9341_WriteCommand(0xC7);
ILI9341_WriteData(0x86);

//MEMORY ACCESS CONTROL
ILI9341_WriteCommand(0x36);
ILI9341_WriteData(0x48);

//PIXEL FORMAT
ILI9341_WriteCommand(0x3A);
ILI9341_WriteData(0x55);

//FRAME RATIO CONTROL, STANDARD RGB COLOR
ILI9341_WriteCommand(0xB1);
ILI9341_WriteData(0x00);
ILI9341_WriteData(0x18);

//DISPLAY FUNCTION CONTROL
ILI9341_WriteCommand(0xB6);
ILI9341_WriteData(0x08);
ILI9341_WriteData(0x82);
ILI9341_WriteData(0x27);

//3GAMMA FUNCTION DISABLE
ILI9341_WriteCommand(0xF2);
ILI9341_WriteData(0x00);

//GAMMA CURVE SELECTED
ILI9341_WriteCommand(0x26);
ILI9341_WriteData(0x01);

//POSITIVE GAMMA CORRECTION
ILI9341_WriteCommand(0xE0);
ILI9341_WriteData(0x0F);
ILI9341_WriteData(0x31);
ILI9341_WriteData(0x2B);
ILI9341_WriteData(0x0C);
ILI9341_WriteData(0x0E);
ILI9341_WriteData(0x08);
ILI9341_WriteData(0x4E);
ILI9341_WriteData(0xF1);
ILI9341_WriteData(0x37);
ILI9341_WriteData(0x07);
ILI9341_WriteData(0x10);
ILI9341_WriteData(0x03);
ILI9341_WriteData(0x0E);
ILI9341_WriteData(0x09);
ILI9341_WriteData(0x00);

//NEGATIVE GAMMA CORRECTION
ILI9341_WriteCommand(0xE1);
ILI9341_WriteData(0x00);
ILI9341_WriteData(0x0E);
ILI9341_WriteData(0x14);

```

```

    ILI9341_WriteData(0x03);
    ILI9341_WriteData(0x11);
    ILI9341_WriteData(0x07);
    ILI9341_WriteData(0x31);
    ILI9341_WriteData(0xC1);
    ILI9341_WriteData(0x48);
    ILI9341_WriteData(0x08);
    ILI9341_WriteData(0x0F);
    ILI9341_WriteData(0x0C);
    ILI9341_WriteData(0x31);
    ILI9341_WriteData(0x36);
    ILI9341_WriteData(0x0F);

    //EXIT SLEEP
    ILI9341_WriteCommand(0x11);
    HAL_Delay(100);

    //TURN ON DISPLAY
    ILI9341_WriteCommand(0x29);

    //STARTING ROTATION
    ILI9341_SetRotation(SCREEN_VERTICAL_1);
}

void ILI9341_SetRotation(uint8_t rotation)
{
    ILI9341_WriteCommand(0x36);
    HAL_Delay(1);

    switch(rotation)
    {
        case SCREEN_VERTICAL_1:
            ILI9341_WriteData(0x40|0x08);
            LCD_WIDTH = 320;
            LCD_HEIGHT = 320;
            break;
        case SCREEN_HORIZONTAL_1:
            ILI9341_WriteData(0x20|0x08);
            LCD_WIDTH = 320;
            LCD_HEIGHT = 320;
            break;
        case SCREEN_VERTICAL_2:
            ILI9341_WriteData(0x80|0x08);
            LCD_WIDTH = 320;
            LCD_HEIGHT = 320;
            break;
        case SCREEN_HORIZONTAL_2:
            ILI9341_WriteData(0x40|0x80|0x20|0x08);
            LCD_WIDTH = 320;
            LCD_HEIGHT = 320;
            break;
        default:
            break;
    }
}

uint16_t ILI9341_FixColor(uint16_t color)
{
    return ((color & 0xF800) >> 11) | // Red -> Blue
           ((color & 0x07E0)) | // Green giữ nguyên
           ((color & 0x001F) << 11); // Blue -> Red
}

```

```

void ILI9341_DrawColor(uint16_t color)
{
    color = ILI9341_FixColor(color);
    uint8_t buffer[2] = {color>>8, color};
    ILI9341_WriteBuffer(buffer, sizeof(buffer));
}

void ILI9341_DrawColorBurst(uint16_t color, uint32_t size)
{
    color = ILI9341_FixColor(color);
    uint32_t BufferSize = 0;

    if((size*2) < BURST_MAX_SIZE)
    {
        BufferSize = size;
    }
    else
    {
        BufferSize = BURST_MAX_SIZE;
    }

    HAL_GPIO_WritePin(LCD_DC_PORT, LCD_DC_PIN, GPIO_PIN_SET);
    HAL_GPIO_WritePin(LCD_CS_PORT, LCD_CS_PIN, GPIO_PIN_RESET);

    uint8_t chifted = color>>8;
    uint8_t BurstBuffer[BufferSize];

    for(uint32_t j = 0; j < BufferSize; j+=2)
    {
        BurstBuffer[j] = chifted;
        BurstBuffer[j+1] = color;
    }

    uint32_t SendingSize = size * 2;
    uint32_t SendingInBlock = SendingSize / BufferSize;
    uint32_t RemainderFromBlock = SendingSize % BufferSize;

    if(SendingInBlock != 0)
    {
        for(uint32_t j = 0; j < (SendingInBlock); j++)
        {
            HAL_SPI_Transmit(HSPI_INSTANCE, BurstBuffer, BufferSize, 10);
        }
    }

    HAL_SPI_Transmit(HSPI_INSTANCE, BurstBuffer, RemainderFromBlock, 10);
    HAL_GPIO_WritePin(LCD_CS_PORT, LCD_CS_PIN, GPIO_PIN_SET);
}

void ILI9341_FillScreen(uint16_t color)
{
    ILI9341_SetAddress(0, 0, LCD_WIDTH, LCD_HEIGHT);
    ILI9341_DrawColorBurst(color, LCD_WIDTH*LCD_HEIGHT);
}

void ILI9341_DrawPixel(uint16_t x,uint16_t y,uint16_t color)
{
    if((x >=LCD_WIDTH) || (y >=LCD_HEIGHT)) return;
    color = ILI9341_FixColor(color);
    uint8_t bufferX[4] = {x>>8, x, (x+1)>>8, (x+1)};
    uint8_t bufferY[4] = {y>>8, y, (y+1)>>8, (y+1)};
    uint8_t bufferC[2] = {color>>8, color};

    ILI9341_WriteCommand(0x2A); //ADDRESS
}

```

```

    ILI9341_WriteBuffer(bufferX, sizeof(bufferX));           //XDATA
    ILI9341_WriteCommand(0x2B);                           //ADDRESS
    ILI9341_WriteBuffer(bufferY, sizeof(bufferY));           //YDATA
    ILI9341_WriteCommand(0x2C);                           //ADDRESS
    ILI9341_WriteBuffer(bufferC, sizeof(bufferC));           //COLOR
}

void ILI9341_DrawRectangle(uint16_t x, uint16_t y, uint16_t width, uint16_t height, uint16_t color)
{
    if((x >= LCD_WIDTH) || (y >= LCD_HEIGHT)) return;

    if((x+width-1)>=LCD_WIDTH)
    {
        width=LCD_WIDTH-x;
    }

    if((y+height-1)>=LCD_HEIGHT)
    {
        height=LCD_HEIGHT-y;
    }

    ILI9341_SetAddress(x, y, x+width-1, y+height-1);
    ILI9341_DrawColorBurst(color, height*width);
}

void ILI9341_DrawHLine(uint16_t x, uint16_t y, uint16_t width, uint16_t color)
{
    if((x >= LCD_WIDTH) || (y >= LCD_HEIGHT)) return;

    if((x+width-1)>=LCD_WIDTH)
    {
        width=LCD_WIDTH-x;
    }

    ILI9341_SetAddress(x, y, x+width-1, y);
    ILI9341_DrawColorBurst(color, width);
}

void ILI9341_DrawVLine(uint16_t x, uint16_t y, uint16_t height, uint16_t color)
{
    if((x >= LCD_WIDTH) || (y >= LCD_HEIGHT)) return;

    if((y+height-1)>=LCD_HEIGHT)
    {
        height=LCD_HEIGHT-y;
    }

    ILI9341_SetAddress(x, y, x, y+height-1);
    ILI9341_DrawColorBurst(color, height);
}

```

CODE 8: ILI9341_STM32_Driver.c

1.3. Code UART ESP32

a. Code Arduino IDE

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <SoftwareSerial.h>

SoftwareSerial mySerial(D4, D5); // RX, TX

const char* host = "script.google.com";
const int httpsPort = 443;
String GAS_ID = "AKfybwdISXXQIrqqEH1uKit5ua63TaoQH3l1sE8hlJg4-u-FyOGwtPiayLyQg9SJFrFxOlz";

const char* ssid = "An Nguyễn";
const char* password = "16072005";

// Time and cost management
String receivedStateStr = "";
unsigned long startTime = 0;
unsigned long usageDuration = 0;
float totalCost = 0.0;
const float PRICE_PER_HOUR = 3000.0;
bool isSessionActive = false;

WiFiClientSecure client;

void setup() {
  Serial.begin(115200);
  mySerial.begin(115200);
  delay(500);

  WiFi.begin(ssid, password);
  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }

  Serial.println();
  Serial.print("Connected to: ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  client.setInsecure();
}

void loop() {
  if (mySerial.available() > 0) {
    String input = mySerial.readStringUntil('\n');
    input.trim();
    Serial.println("Received: " + input);
    processInput(input);
  }
}
```

```

}

void processInput(String input) {
    int stateIndex = input.indexOf("State:");
    int passIndex = input.indexOf("Pass:");

    if (stateIndex != -1 && passIndex != -1) {
        receivedStateStr = input.substring(stateIndex + 6, passIndex - 1);
        receivedStateStr.trim();
        String passStr = input.substring(passIndex + 5);
        passStr.trim();

        Serial.println("State: " + receivedStateStr + ", Password: " + passStr);

        if (receivedStateStr == "CLOSE") {
            startTime = millis();
            isSessionActive = true;
            Serial.println("Locker closed - Timer started");
            sendToGoogleSheet(receivedStateStr, passStr, "00:00:00", 0, totalCost);
        }
        else if (receivedStateStr == "OPEN" && isSessionActive) {
            usageDuration = millis() - startTime;
            float hours = usageDuration / 3600000.0;
            float sessionCost = ceil(hours) * PRICE_PER_HOUR;
            totalCost += sessionCost;
            isSessionActive = false;

            Serial.print("Usage duration: ");
            String formattedTime = formatTime(usageDuration);
            Serial.print(formattedTime);
            Serial.print(" - Cost: ");
            Serial.print(sessionCost, 0);
            Serial.println(" VND");

            sendToGoogleSheet(receivedStateStr, passStr, formattedTime, sessionCost, totalCost);
        }
        else if (receivedStateStr == "NEW_PASS") {
            Serial.println("New password set: " + passStr);
            sendToGoogleSheet("PASS_CHANGE", passStr, "N/A", 0, totalCost);
        }
    }

    String formatTime(unsigned long milliseconds) {
        unsigned long seconds = milliseconds / 1000;
        unsigned long minutes = seconds / 60;
        unsigned long hours = minutes / 60;
        seconds %= 60;
        minutes %= 60;

        char buffer[10];
        sprintf(buffer, "%02lu:%02lu:%02lu", hours, minutes, seconds);
        return String(buffer);
    }

    void sendToGoogleSheet(String state, String pass, String duration, float cost, float total) {
        Serial.print("Connecting to ");
        Serial.println(host);

        if (!client.connect(host, httpsPort)) {
            Serial.println("Connection failed");
            return;
        }
    }
}

```

```

String url = "/macros/s/" + GAS_ID + "/exec?state=" + state +
    "&password=" + pass +
    "&duration=" + duration +
    "&cost=" + String((int)cost, DEC) +
    "&total=" + String((int)total, DEC);

Serial.print("Requesting URL: ");
Serial.println(url);

client.print(String("GET ") + url + " HTTP/1.1\r\n" +
    "Host: " + host + "\r\n" +
    "User-Agent: ESP8266LockerSystem\r\n" +
    "Connection: close\r\n\r\n");

while (client.connected()) {
    String line = client.readStringUntil('\n');
    if (line == "\r") {
        break;
    }
}

String response = client.readString();
Serial.println("Response:");
Serial.println(response);
Serial.println("Data sent to Google Sheet");
Serial.println("Closing connection");
}

```

CODE 9: Esp8266

b. Giải thích Arduino code của Esp8266

- **Khởi tạo kết nối**
 - ESP8266 kết nối đến mạng WiFi với tên và mật khẩu đã được cấu hình sẵn (ssid = "Phuong", password = "danang43").
 - Cổng UART phần mềm (SoftwareSerial) được cấu hình trên chân D4 (RX) và D5 (TX) với baud rate 115200.
 - Kết nối HTTPS được thiết lập với chế độ bỏ qua xác thực SSL (client.setInsecure()), giúp ESP8266 dễ dàng kết nối đến dịch vụ Google.

- **Nhận và xử lý dữ liệu UART**
 - ESP8266 liên tục lắng nghe dữ liệu từ cổng UART. Khi nhận được một chuỗi kết thúc bởi \n, chương trình sẽ phân tích nội dung để xác định hành động cần thực hiện. Chuỗi dữ liệu có định dạng như sau:

State:<TRẠNG THÁI> Pass:<MÃ MẬT KHẨU>

- **Các trạng thái xử lý**
 - **State: CLOSE**
 - Khi nhận trạng thái "CLOSE", ESP8266 hiểu rằng tủ vừa được đóng lại.

- Thời gian hiện tại (bằng millis()) được lưu lại để bắt đầu tính thời gian sử dụng.
- Gửi một bản ghi ban đầu lên Google Sheets với thời gian sử dụng là 00:00:00 và chi phí là 0 VND.
- **State: OPEN**
 - Khi nhận trạng thái "OPEN", ESP8266 hiểu rằng tủ đã được mở để lấy hành lý.
 - Thời gian sử dụng được tính bằng hiệu số giữa thời điểm hiện tại và startTime.
 - Chuyển thời gian sử dụng sang đơn vị giờ, sau đó làm tròn lên (ceil) và nhân với đơn giá (3000 VND/giờ) để tính chi phí cho phiên gửi hành lý đó.
 - Cộng dồn tổng chi phí và gửi dữ liệu lên Google Sheets.
- **State: NEW_PASS**
 - Khi nhận trạng thái "NEW_PASS", ESP8266 hiểu rằng người dùng đã thiết lập một mật khẩu mới.
 - Dữ liệu sẽ được gửi lên Google Sheets để ghi lại sự kiện này, giúp quản lý nhật ký hệ thống.

▪ Gửi dữ liệu lên Google Sheets

- Dữ liệu được gửi thông qua yêu cầu HTTP GET đến dịch vụ Google Apps Script với định dạng URL như sau:

```
https://script.google.com/macros/s/<GAS_ID>/exec?state=...&password=...&duration=...&cost=...&total=...
```

Trong đó:

- state: Trạng thái tủ (OPEN, CLOSE, PASS_CHANGE)
- password: Mã người dùng
- duration: Thời gian sử dụng (định dạng hh:mm:ss)
- cost: Chi phí phiên gửi hiện tại
- total: Tổng chi phí từ đầu đến hiện tại
- Sau khi gửi xong, ESP8266 sẽ hiển thị phản hồi từ server trên Serial Monitor để tiện theo dõi.

c. Code google script

```

function doGet(e) {
  Logger.log( JSON.stringify(e) );
  var result = 'Ok';
  if (e.parameter == 'undefined') {
    result = 'No Parameters';
  }
  else {
    var sheet_id = '1uWmTtRVH5vgWxxLdBdyZFUn4c_rcU3I4AXy6yC3oNw4'; // Spreadsheet ID
    var sheet = SpreadsheetApp.openById(sheet_id).getActiveSheet();
    var newRow = sheet.getLastRow() + 1;
    var rowData = [];
    var Curr_Date = new Date();
    rowData[0] = Curr_Date; // Date in column A
    var Curr_Time = Utilities.formatDate(Curr_Date, "Asia/Jakarta", 'HH:mm:ss');
    rowData[1] = Curr_Time; // Time in column B
    for (var param in e.parameter) {
      Logger.log('In for loop, param=' + param);
      var value = stripQuotes(e.parameter[param]);
      Logger.log(param + ':' + e.parameter[param]);
      if(param == 'state'){
        rowData[2] = value; // data in column C
      }
      if(param == 'password'){
        rowData[3] = value; // data in column D
      }
      if(param == 'duration'){
        rowData[4] = value; // data in column E
      }
      if(param == 'cost'){
        rowData[5] = value; // data in column F
      }
      if(param == 'total'){
        rowData[6] = value; // data in column G
      }
    }
    Logger.log(JSON.stringify(rowData));
    var newRange = sheet.getRange(newRow, 1, 1, rowData.length);
    newRange.setValues([rowData]);
  }
  return ContentService.createTextOutput(result);
}
function stripQuotes( value ) {
  return value.replace(/[^"]|[""]$/g, "");
}

```

CODE 10: Google Apps Script Code

d. Giải thích Google Apss Script Code

- **Chức năng tổng quát**

Đoạn script này giúp nhận dữ liệu từ HTTP GET request và ghi vào một bảng tính Google Sheets. Dữ liệu bao gồm: thời gian thực, trạng thái (state), mật khẩu (password), thời lượng (duration), chi phí (cost), và tổng (total).

- **Giải thích từng phần mã**

- **Hàm doGet(e)**

- Hàm này là **endpoint xử lý yêu cầu GET**. Mỗi khi có request đến từ URL chứa tham số, hàm sẽ được gọi.

```
function doGet(e) {
```

- **Logging tham số đầu vào**

- In thông tin của toàn bộ request để phục vụ debug:

```
Logger.log(JSON.stringify(e));
```

- **Kiểm tra sự tồn tại của tham số**

```
if (e.parameter === 'undefined') {
    result = 'No Parameters';
}
```

Nếu không có tham số nào được gửi lên, trả về "No Parameters".

- **Mở bảng tính Google Sheets**

```
var sheet_id = '1uWmTtRVH5vgWxxLdBdyZFUUn4c_rcU3I4AXy6yC3oNw4';
var sheet = SpreadsheetApp.openById(sheet_id).getActiveSheet();
```

- **Tạo dòng dữ liệu mới**

```
var newRow = sheet.getLastRow() + 1;
var rowData = [];
```

- **Ghi thời gian hiện tại**

```
var Curr_Date = new Date();
rowData[0] = Curr_Date; // Cột A
var Curr_Time = Utilities.formatDate(Curr_Date, "Asia/Jakarta", 'HH:mm:ss');
rowData[1] = Curr_Time; // Cột B
```

- **Ghi các tham số truyền vào**

```
for (var param in e.parameter) {  
  
    var value = stripQuotes(e.parameter[param]);  
  
    if(param == 'state')  rowData[2] = value; // Cột C  
  
    if(param == 'password') rowData[3] = value; // Cột D  
  
    if(param == 'duration') rowData[4] = value; // Cột E  
  
    if(param == 'cost')  rowData[5] = value; // Cột F  
  
    if(param == 'total')  rowData[6] = value; // Cột G  
  
}
```

- **Ghi dòng dữ liệu vào sheet**

```
var newRange = sheet.getRange(newRow, 1, 1, rowData.length);  
  
newRange.setValues([rowData]);
```

- **Trả về phản hồi**

```
return ContentService.createTextOutput(result);
```

- **Hàm phụ stripQuotes(value)**

```
function stripQuotes( value ) {  
  
    return value.replace(/^["]|"["]$/g, "");  
  
}
```

- Xóa dấu nháy " hoặc ' đầu và cuối trong chuỗi, để đảm bảo dữ liệu sạch hơn trước khi lưu.

e. Google script

https://script.google.com/u/0/home/projects/1g1iPnL_Wdbb_6M4UFLKLjs_4CJvJWOfLG56hFityhLowju05yZPzgfdh/edit?pli=1

f. Google sheet

VI ĐIỀU KHIỂN

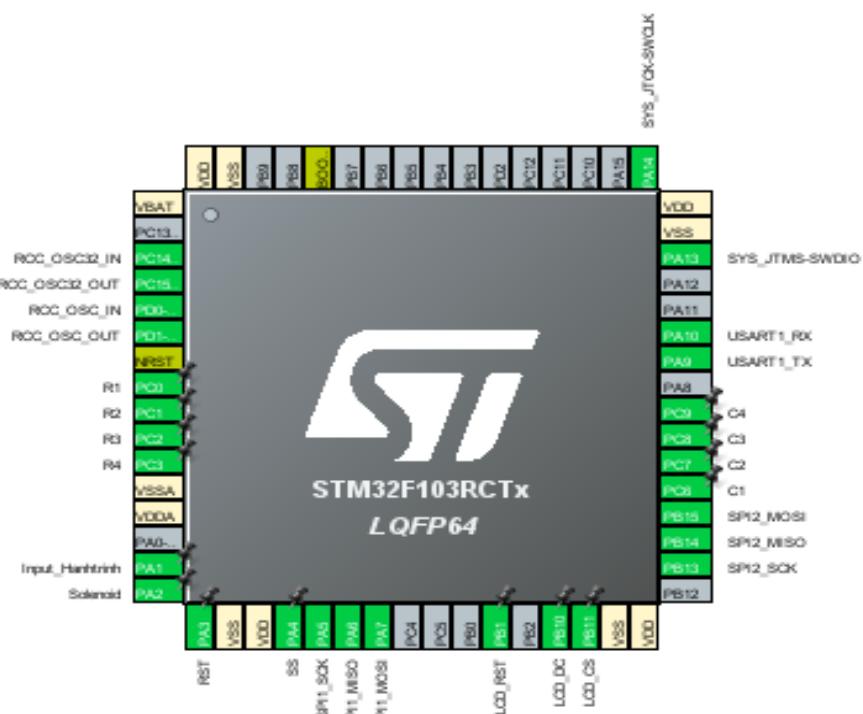
https://docs.google.com/spreadsheets/d/1uWmTtRVH5vgWxxLdBdyZFun4c_rcU3I4AXy6yC3oNw4/edit?gid=0#gid=0

A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	date	time	state	password	duration	cost(VND)	total(VND)						
2	24/04/2025	19:42:41	CLOSE		00:00:00	0	9000						
3	24/04/2025	19:43:15	PASS_CHANGE	12345	N/A	0	9000						
4	24/04/2025	19:43:24	PASS_CHANGE	11111	N/A	0	9000						
5	24/04/2025	19:44:11	OPEN		00:01:29	3000	12000						
6	24/04/2025	19:44:45	CLOSE	11111	00:00:00	0	12000						
7	24/04/2025	19:45:03	OPEN	11111	00:00:18	3000	15000						
8	24/04/2025	19:45:53	CLOSE		00:00:00	0	15000						
9	24/04/2025	19:46:08	PASS_CHANGE	12345	N/A	0	15000						
10	24/04/2025	19:48:27	CLOSE		00:00:00	0	0						
11	25/04/2025	15:51:10	CLOSE		00:00:00	0	0						
12	25/04/2025	15:51:26	PASS_CHANGE	11111	N/A	0	0						
13	25/04/2025	15:51:51	OPEN	11111	00:00:40	3000	3000						
14	25/04/2025	20:10:31	CLOSE		00:00:00	0	0						
15	25/04/2025	20:11:04	PASS_CHANGE	11111	N/A	0	0						
16	25/04/2025	20:11:16	PASS_CHANGE	12345	N/A	0	0						
17	25/04/2025	20:11:39	OPEN	12345	00:01:06	3000	3000						
18													
19													
20													
21													
22													
23													

HÌNH 4.1 Google sheet

1.4. Cấu hình hệ thống

a. Cấu hình sơ đồ chân STM32 (IOC)



HÌNH 4.2: Sơ đồ chân STM32

b. Cấu hình chân STM32

Pin Name	Signal on Pin	GPIO output	GPIO mode	GPIO Pull-up	Maximum o...	User Label	Modified
PA1	n/a	n/a	Input mode	No pull-up a...	n/a	Input_Hanht...	✓
PA2	n/a	Low	Output Pus...	No pull-up a...	Low	Solenoid	✓
PA3	n/a	Low	Output Pus...	No pull-up a...	Low	RST	✓
PA4	n/a	Low	Output Pus...	No pull-up a...	Low	SS	✓
PB1	n/a	High	Output Pus...	No pull-up a...	Low	LCD_RST	✓
PB10	n/a	Low	Output Pus...	No pull-up a...	Low	LCD_DC	✓

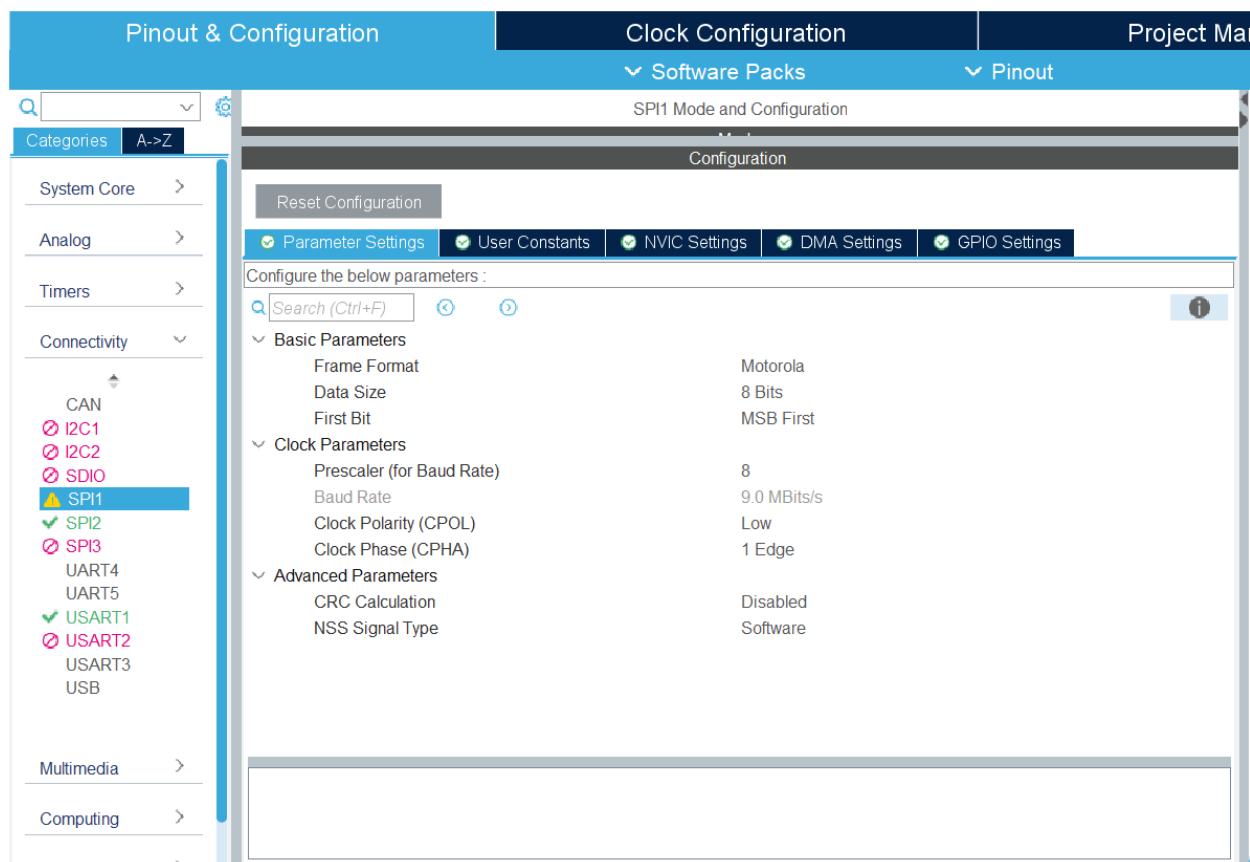
HÌNH 4.3: Cấu hình chân GPIO

Pin Name	Signal on Pin	GPIO output	GPIO mode	GPIO Pull-up	Maximum o...	User Label	Modified
PB11	n/a	Low	Output Pus...	No pull-up a...	Low	LCD_CS	✓
PC0	n/a	n/a	Input mode	Pull-down	n/a	R1	✓
PC1	n/a	n/a	Input mode	Pull-down	n/a	R2	✓
PC2	n/a	n/a	Input mode	Pull-down	n/a	R3	✓
PC3	n/a	n/a	Input mode	Pull-down	n/a	R4	✓
PC6	n/a	Low	Output Pus...	No pull-up a...	Low	C1	✓

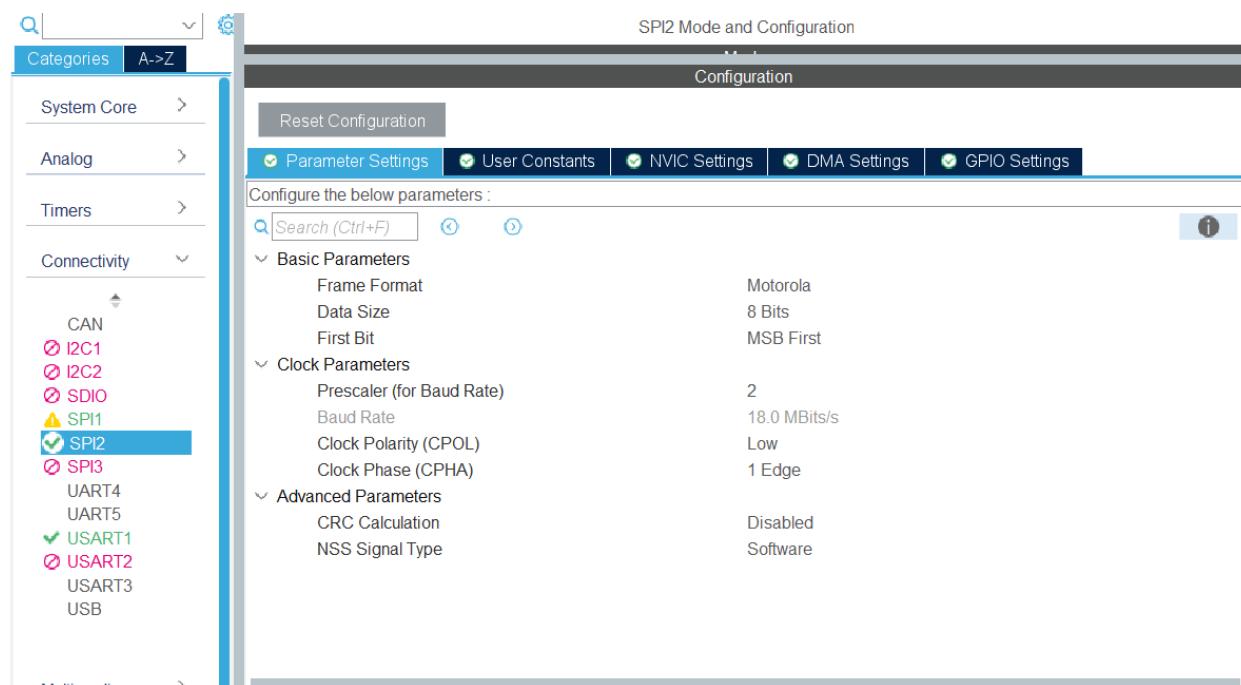
HÌNH 4.4: Cấu hình chân GPIO

Pin Name	Signal on Pin	GPIO output	GPIO mode	GPIO Pull-up	Maximum o...	User Label	Modified
PC2	n/a	n/a	Input mode	Pull-down	n/a	R3	✓
PC3	n/a	n/a	Input mode	Pull-down	n/a	R4	✓
PC6	n/a	Low	Output Pus...	No pull-up a...	Low	C1	✓
PC7	n/a	Low	Output Pus...	No pull-up a...	Low	C2	✓
PC8	n/a	Low	Output Pus...	No pull-up a...	Low	C3	✓
PC9	n/a	Low	Output Pus...	No pull-up a...	Low	C4	✓

HÌNH 4.5: Cấu hình chân GPIO

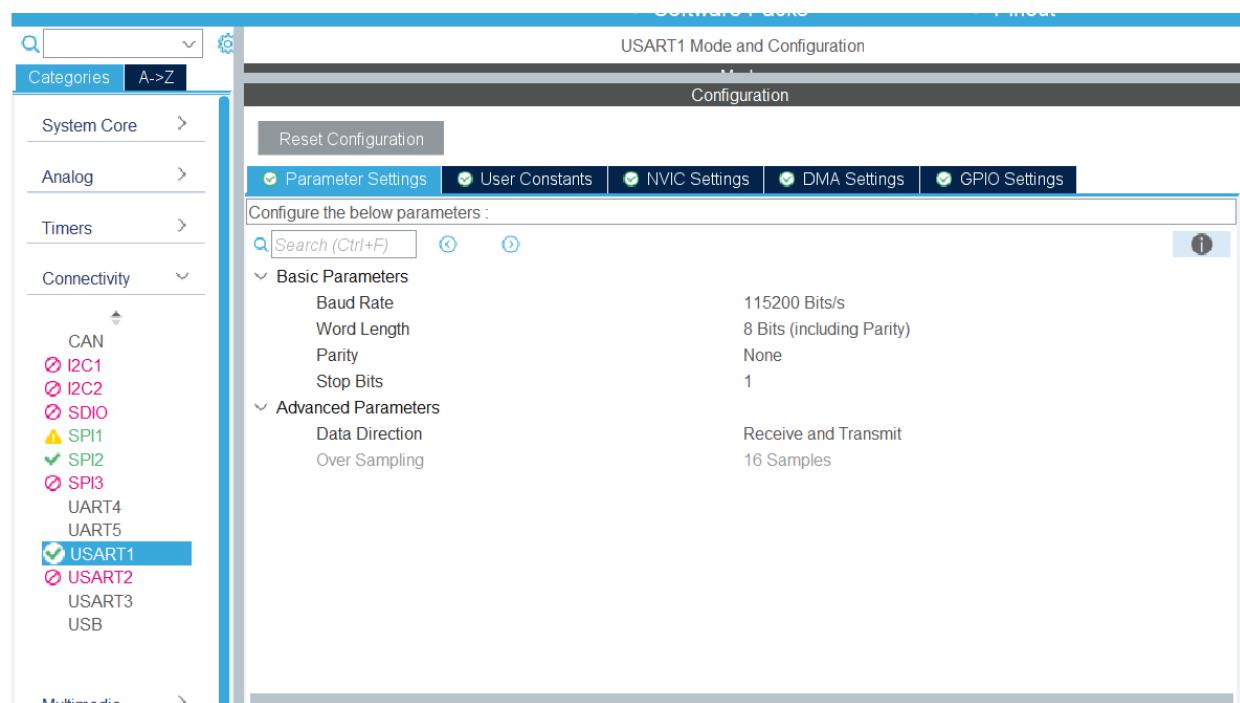


HÌNH 4.6: Cấu hình SPI 1



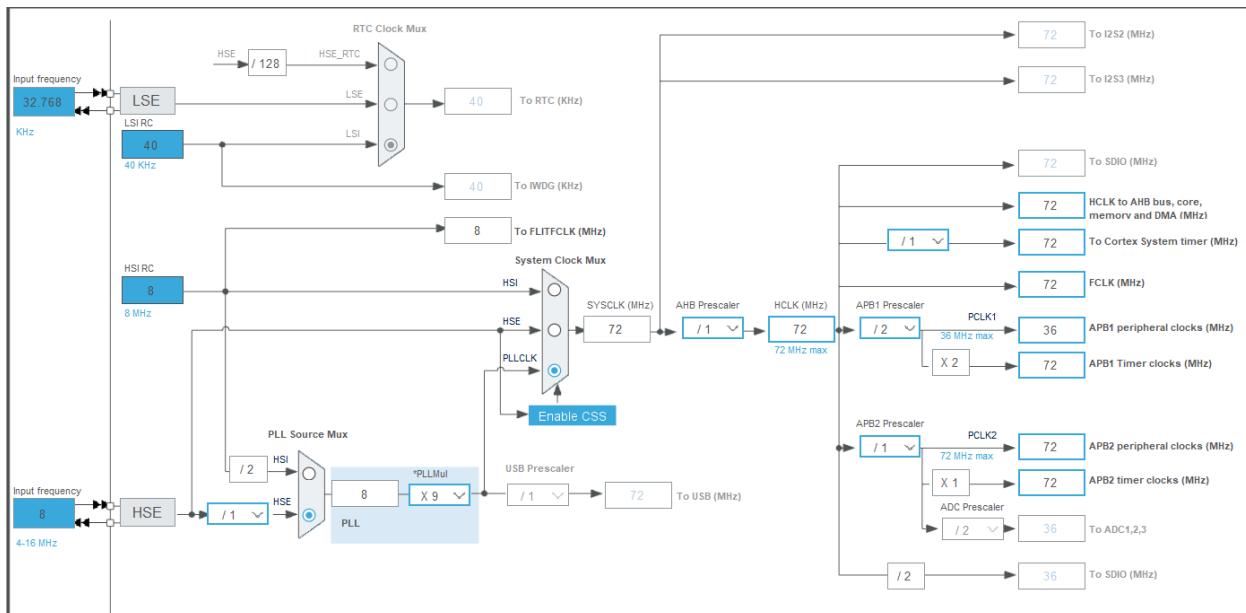
HÌNH 4.7: Cấu hình SPI 2

VI ĐIỀU KHIỂN



HÌNH 4.8: Cấu hình UART

c. Cấu hình Clock STM32



HÌNH 4.8: Cấu hình clock

2. MÔ HÌNH TỦ NHỰA

2.1. Bản vẽ

a. Mục đích thiết kế

Tủ là bản demo của hệ thống tủ gửi hành lý được thiết kế để tích hợp sử dụng công nghệ RFID và khóa điện tử. Mục tiêu là đảm bảo tính tiện lợi, dễ lắp ráp và tương thích với phần cứng vi điều khiển đã thiết kế trước đó.

b. Quy trình thiết kế

- **Phần mềm sử dụng:** Autodesk Fusion 360.



HÌNH 5.1: Phần mềm AUTODESK Fusion

- **Các bước:**

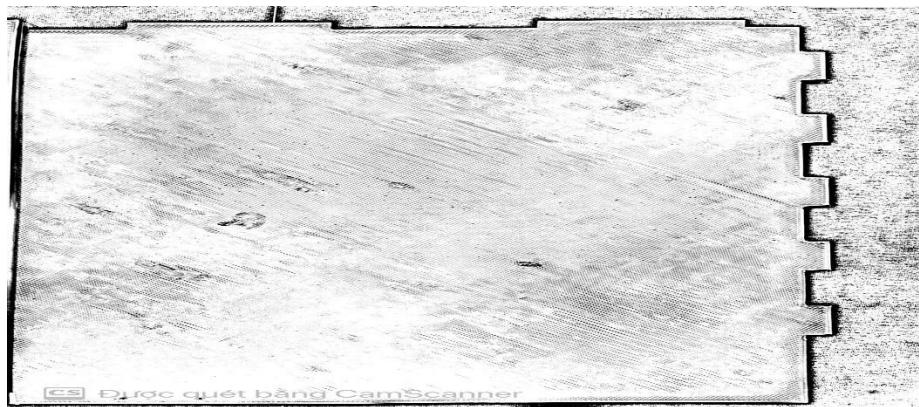
1. Phác thảo kích thước tổng thể.
2. Tạo khối hộp chính (thân tủ).
3. Thiết kế cánh cửa.
4. Thêm tay cầm và các lỗ bắt vít.
5. Khoét các vị trí để gắn cảm biến, khóa điện tử, màn hình...

c. Thông số kỹ thuật

Bộ phận	Kích thước (Dài x Rộng x Cao)	Ghi chú
Thân tủ	30x30x30 cm	Hộp rỗng
Cửa tủ	28.50x18x1 cm	Có bản lề xoay
Tay nắm cửa	16.822x4 cm	Bắt vít hai đầu

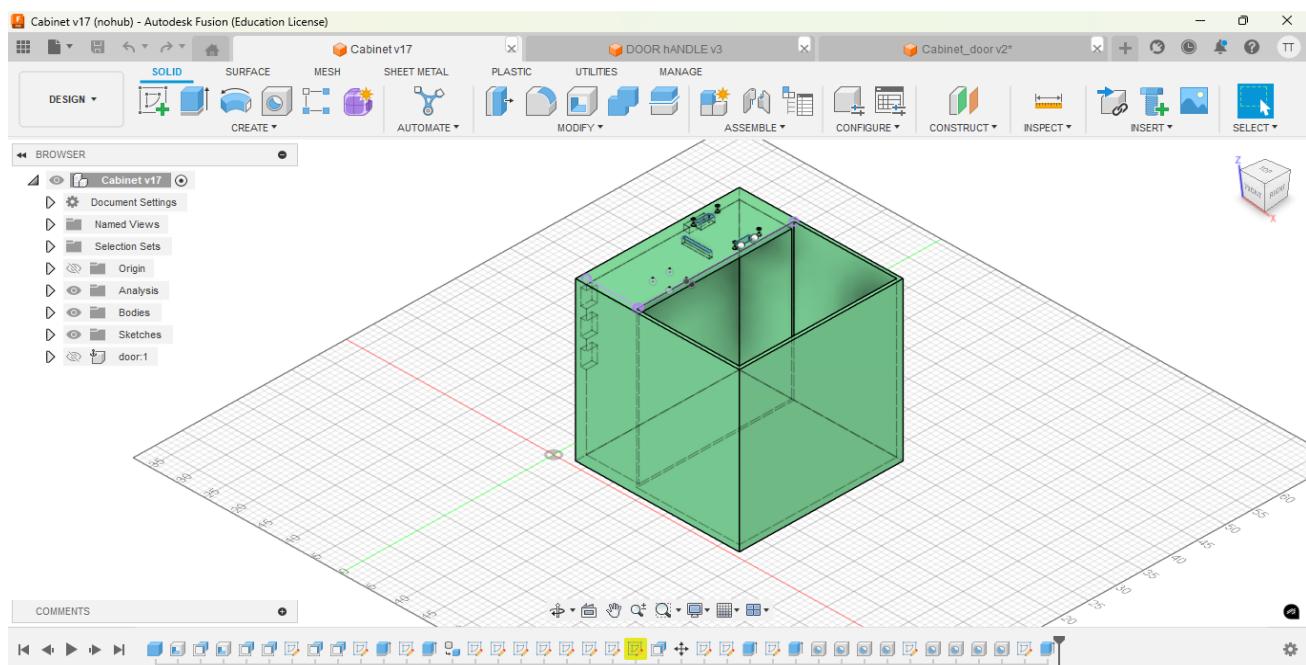
2.2. Vẽ và in 3D

- **Công nghệ:** In 3D FDM
- **Vật liệu:** PLA
- **Layer height:** 0.3mm
- **Infill:** 5%
- **Khối lượng tủ chính:** 1.5kg
- **Tách mô hình:** Tách tủ chính thành 6 mảnh riêng biệt, in ra rồi gắn lại bằng các khớp lắp.



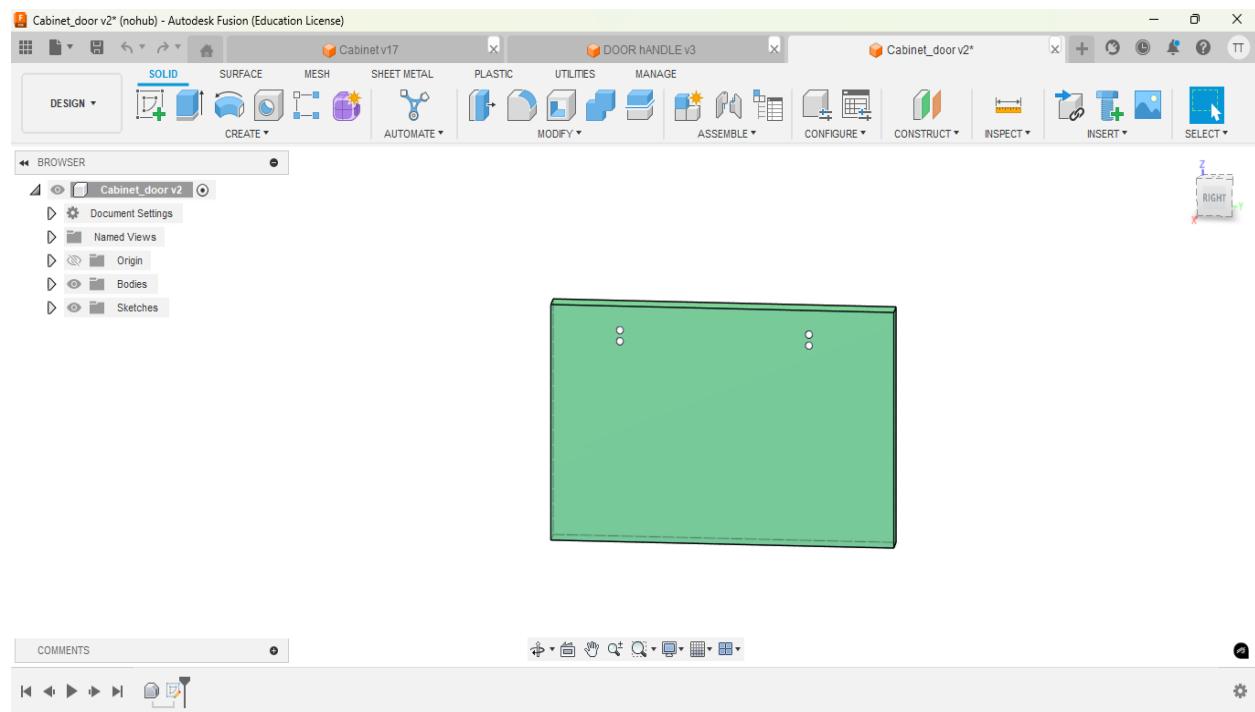
HÌNH 5.2: Khớp lắp tủ nhựa

a. Thân tủ



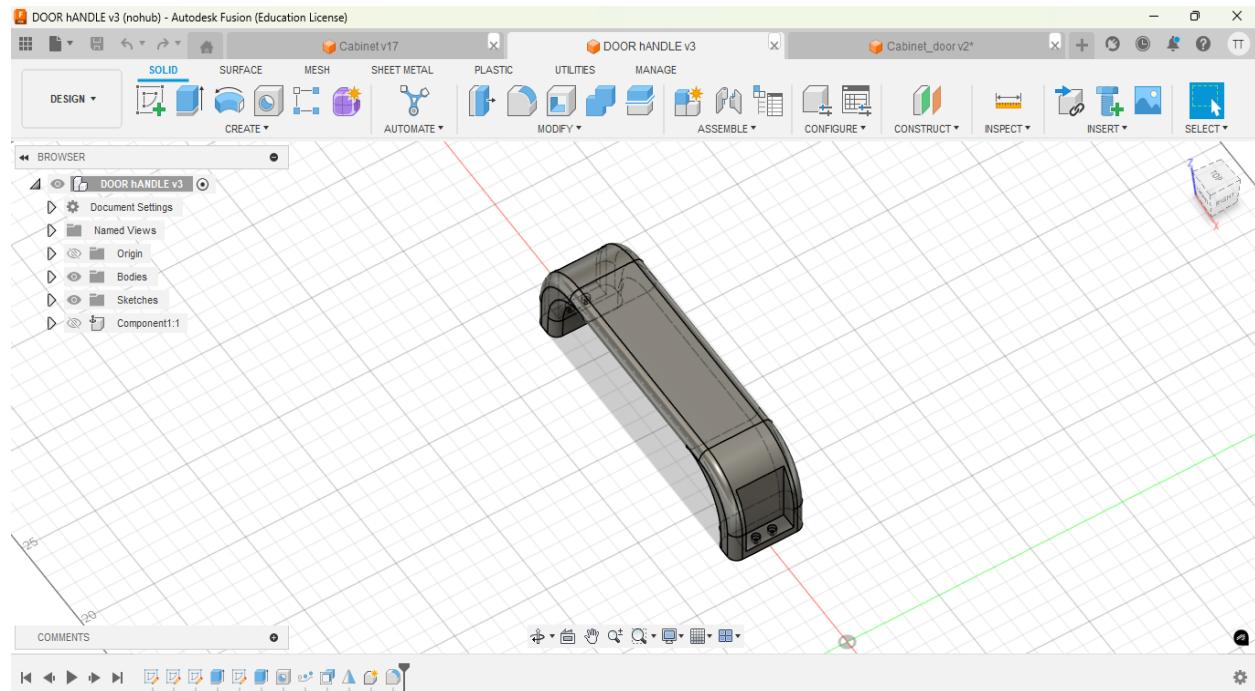
HÌNH 5.3: Thân tủ

b. Cửa tủ



HÌNH 5.4: Cánh cửa tủ

c. Tay nắm cửa



HÌNH 5.5: Tay nắm cửa

PHẦN V: TỦ THỰC TẾ

1. TỦ VÀ HOẠT ĐỘNG CỦA TỦ THỰC TẾ



HÌNH 6.1: Tủ thực tế

QUÉT MÃ QR ĐỂ XEM VIDEO HOẠT ĐỘNG CỦA TỦ



HÌNH 6.2: Mã QR xem video

2. HỆ THỐNG GOOGLE SHEET

E9	A	B	C	D	E	F	G
1	date	time	state	password	duration	cost(VND)	total(VND)
2	24/04/2025	19:42:41	CLOSE		00:00:00	0	9000
3	24/04/2025	19:43:15	PASS_CHANGE	12345	N/A	0	9000
4	24/04/2025	19:43:24	PASS_CHANGE	11111	N/A	0	9000
5	24/04/2025	19:44:11	OPEN	11111	00:01:29	3000	12000
6	24/04/2025	19:44:45	CLOSE	11111	00:00:00	0	12000
7	24/04/2025	19:45:03	OPEN	11111	00:00:18	3000	15000
8	24/04/2025	19:45:53	CLOSE		00:00:00	0	15000
9	24/04/2025	19:46:08	PASS_CHANGE	12345	N/A	0	15000
10	24/04/2025	19:48:27	CLOSE		00:00:00	0	0
11							
12							
13							
14							

HÌNH 6.3: Google sheet hiện trạng thái

3. LINH KIỆN SỬ DỤNG

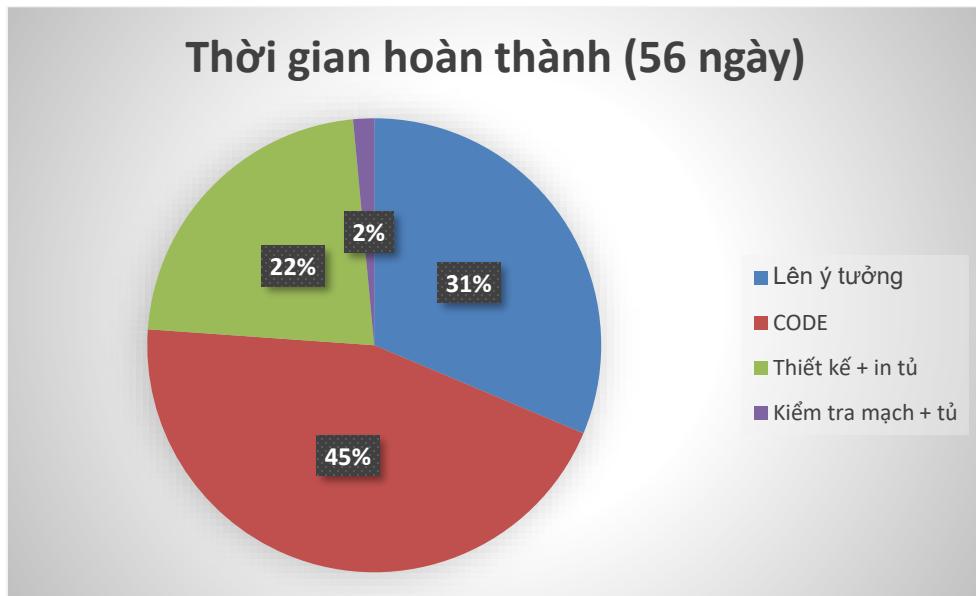
Tên linh kiện	Giá thành	SL	Tổng
Pad 4x4 (Mỏng)	12.000đ	1	12.000đ
Pad 4x4 (Dày)	41.000đ	1	41.000đ
Mạch RFID NFC 13.56MHz RC522	28.000đ	1	28.000đ
Endstop Switch	13.000đ	1	13.000đ
Khóa chốt điện Solenoid Lock LY- 03 12VDC	120.000đ	1	120.000đ
Mạch giảm áp DC-DC Buck XL4005 5A	24.000đ	1	24.000đ
MOSFET IRFZ44N	10.000đ	2	20.000đ
DIODE 1n5819, 1N4007	2.000đ	2	4.000đ
TRANSISTOR 2N2222	2.000đ	2	4.000đ
Nguồn tản nhiệt 12VDC 10A	110.000đ	1	110.000đ
Màn hình TFT 2.8 inches ILI9341	215.000đ	1	215.000đ
DOOR Hinge 69x43x1.2mm	4.000đ	2	8.000đ
Vít M2x10mm, M3x10mm, M4x20mm	38.000đ	-	46.000đ
Keo dán nhựa	11.000đ	1	11.000đ
Tủ nhựa in 3D	200.000đ	1	200.000đ
Dây cắm	32.000đ	2	64.000đ
Tổng	-	-	920.000đ

Bảng 2 Linh kiện

BÁO CÁO NHIỆM VỤ

Nhiệm vụ	Thời gian giao	Thời gian hạn	Người được giao	Trạng thái
Vẽ flowchart	14/03/2025	24/03/2025	P.T.Khanh	Đã hoàn thành
Vẽ mô phỏng	14/03/2025	24/03/2025	P.M.Thắng	Đã hoàn thành
Code thô	17/03/2025	31/03/2025	N.T.An	Đã hoàn thành
Chốt solenoid	17/03/2025	31/03/2025	N.T.An	Đã hoàn thành
Code hiển thị	31/03/2025	10/04/2025	P.T.Khanh	Đã hoàn thành
Code gửi thông tin	10/04/2025	17/04/2025	P.M.Thắng	Đã hoàn thành
Thiết kế tủ	10/04/2025	24/04/2025	T.T.Thịnh	Đã hoàn thành
In tủ nhựa	10/04/2025	24/04/2025	T.T.Thịnh	Đã hoàn thành
Tổng hợp báo cáo	10/04/2025	24/04/2025	T.T.Thịnh	Đã hoàn thành
Kiểm tra mạch	24/04/2025	24/04/2025	N.T.An	Đã hoàn thành
Nộp báo cáo	14/04/2025	24/04/2025	N.T.An	Đã hoàn thành
Thuyết trình thử	24/04/2025	25/04/2025	N.T.An	Đã hoàn thành

Bảng 3 Báo cáo nhiệm vụ



HÌNH 7: Biểu đồ tròn thời gian hoàn thành công việc

ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH NHIỆM VỤ

Họ và tên	MSSV	Nhiệm vụ	Hoàn thành	Điểm
Phạm Tuấn Khanh	23207025	Vẽ flowchart Code hiển thị Kiểm tra mạch	100%	10
Phạm Minh Thắng	23207027	Vẽ mô phỏng Code gửi thông tin Kiểm tra mạch	100%	10
Trần Thành Thịnh	23207029	Thiết kế tủ In tủ nhựa Tổng hợp báo cáo Kiểm tra mạch Nộp báo cáo Làm slide	100%	10
Nguyễn Thành An	23207034	Code thô Chốt solenoid Kiểm tra mạch Thuyết trình thử Làm slide	100%	10

Bảng 4: Đánh giá mức độ hoàn thành nhiệm vụ của thành viên

KẾT LUẬN

Đồ án “Tủ gửi hành lý thông minh – Coin Locker” đã hoàn thành đúng yêu cầu của môn học Vi điều khiển, thể hiện rõ sự tích hợp thành công giữa phần cứng và phần mềm, tạo nên một hệ thống lưu trữ hành lý tiện lợi, an toàn và dễ sử dụng. Hệ thống sử dụng vi điều khiển STM32 kết hợp với các thiết bị ngoại vi như RFID, bàn phím ma trận, màn hình cảm ứng và khóa điện tử đã được triển khai hiệu quả, đáp ứng tốt các chức năng chính như: gửi/lấy hành lý, xác thực password, khóa tạm thời khi nhập sai mật khẩu, thanh toán và gửi dữ liệu thông qua nền tảng Google Sheet.

Thông qua quá trình thực hiện đồ án, nhóm chúng em đã củng cố kiến thức lập trình nhúng, thiết kế mạch điện, giao tiếp các thiết bị ngoại vi, cũng như kỹ năng thiết kế 3D và in ấn mô hình thực tế. Ngoài ra, việc tích hợp giao diện người dùng trực quan, cơ chế thanh toán và bảo mật thông minh là điểm nổi bật, cho thấy tiềm năng ứng dụng rộng rãi của hệ thống trong thực tế tại các nhà ga, trung tâm thương mại và khu du lịch.

Mặc dù đồ án này có ưu điểm nhưng vẫn còn nhiều khuyết điểm. Vì vậy, cần thêm nhiều thời gian nghiên cứu và đầu tư hơn nữa để có thể áp dụng vào cuộc sống thực tế. Tuy nhiên, thông qua quá trình thực hiện, chúng em đã tiếp thu thêm nhiều kiến thức cũng như hiểu biết về môn học Vi điều khiển. Đồng thời, đồ án chắc chắn cũng không tránh khỏi những thiếu sót, nên chúng em rất mong nhận được ý kiến đóng góp.

Chúng em xin chân thành cảm ơn!

NHÓM TRƯỞNG

Nguyễn Thanh An

GIẢNG VIÊN

PGS.TS Lê Đức Hùng