Embedded Systems Summative

0.1

Generated by Doxygen 1.10.0

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 2

File Documentation

2.1 BUZZER.c

```
/*
     ################################
     Principle Author:

     Tommy Peach, 100358179

     File name:

     BUZZER.c

     Description:

     BUZZER.c initialises the buzzer, and includes functions for setting and resetting the pin for the buzzer

     ################################
 */

#include <stdio.h>
#include "stm32f7xx_hal.h"
#include "stm32f7xx_hal_gpio.h"
#include "buzzer.h"

GPIO_InitTypeDef gpio_buzz;

// Function to initialise the buzzer
void init_buzzer(){
    __HAL_RCC_GPIOH_CLK_ENABLE(); // GPIO port H

    gpio_buzz.Mode = GPIO_MODE_OUTPUT_PP;
    gpio_buzz.Pull = GPIO_NOPULL;
    gpio_buzz.Speed = GPIO_SPEED_HIGH;
    gpio_buzz.Pin = GPIO_PIN_6;     // Pin 6

    HAL_GPIO_Init(GPIOH, &gpio_buzz); // Initialise the pin

}
// Function to turn on the buzzer
void buzz_on(){
        HAL_GPIO_WritePin(GPIOH, gpio_buzz.Pin, GPIO_PIN_SET);
}

// Function to turn off the buzzer
void buzz_off(){
    HAL_GPIO_WritePin(GPIOH, gpio_buzz.Pin, GPIO_PIN_RESET);
}
```

2.2 buzzer.h

```
/*
     ################################
     Principle Author:

     Tommy Peach, 100358179
```

```

    File name:

    buzzer.h

    Description:

    buzzer.h is used to include the buzzer.c definitions inside other source files.

    ################################
 */
void init_buzzer();
void buzz_off();
void buzz_on();
```

2.3 GLCDTOUCH.c

```
/*
    ################################
    Principle Author:

    Tommy Peach, 100358179

    File name:

    GLCDTOUCH.C

    Description:

    GLCDTOUCH.C handles logic surrounding unlocking, resetting and setting the pin for the system. It also handles transitions between screens defined in Screens.c.
    GLCDTOUCH.C also handles 2FA for the UART/USART bluetooth communication.
  ################################
 */
        #include "stm32f7xx_hal.h"
        #include "GLCD_Config.h"
        #include "Board_GLCD.h"
        #include "Board_Touch.h"
        #include <string.h>
        #include "buzzer.h"
        #include "keypad.h"
        #include "screens.h"
        #include "led_main.h"
        #include "servo.h"
        #include "tracking.h"
        #include "led_main.h"
        #include "interrupt.h"
        #define wait_delay HAL_Delay
        extern GLCD_FONT GLCD_Font_16x24; // Externally including font for GLCD text
        int pin[8] = {1,2,3,4,5,6,7,8}; // Pin array that holds up to 8 integers, used for multiple functions. Preset for testing purposes.
        char status[6] = "Locked";        // Current status for the system, set to locked and is displayed on the home screen
        int attempts = 0;                 // Records the amount of attempts the user has made to enter the currently set pin
        extern int ScreenState;           // Externally included variable from Screens.c, this variable is used to check the current screen the user is interacting with
        TOUCH_STATE tsc_state;            // Stores the current TOUCH_STATE
        UART_HandleTypeDef Bluetooth_UARTHandle;  // Handle for enabling UART bluetooth communication with HC-05
        uint8_t RxBuffer;                 // Buffer for holding received data from bluetooth communication


#ifdef __RTX
extern uint32_t os_time;
uint32_t HAL_GetTick(void) {
    return os_time;
}
#endif
// Function for initialising UART, it initialises RX and TX ports
void init_uart(void){  // Initialising UART capabilities
    GPIO_InitTypeDef GPIO_InitStructure;

    // Initialising RX port for D0
    GPIO_InitStructure.Pin = GPIO_PIN_7;             // Pin 7
    GPIO_InitStructure.Mode = GPIO_MODE_AF_PP;       // Push Pull config
    GPIO_InitStructure.Alternate = GPIO_AF8_USART6; // Initialising USART6 as it is mapped to pin D0 on board
    GPIO_InitStructure.Speed = GPIO_SPEED_HIGH;
    GPIO_InitStructure.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStructure);      // Initialising the pin on port C
    // Initialising TX port for D1
```

```c
    GPIO_InitStructure.Pin = GPIO_PIN_6;            // PIN 6
    GPIO_InitStructure.Mode = GPIO_MODE_AF_PP;      // Push Pull config
    HAL_GPIO_Init(GPIOC, &GPIO_InitStructure);      // Initialising the pin on port C
    // Configuring the bluetooth handle
    Bluetooth_UARTHandle.Instance        = USART6;
    Bluetooth_UARTHandle.Init.BaudRate   = 9600;    // Setting BaudRate to 9600, this is because the chip I bought online recommends/requires 9600
    Bluetooth_UARTHandle.Init.WordLength = UART_WORDLENGTH_8B;
    Bluetooth_UARTHandle.Init.StopBits   = UART_STOPBITS_1;
    Bluetooth_UARTHandle.Init.Parity     = UART_PARITY_NONE;
    Bluetooth_UARTHandle.Init.HwFlowCtl  = UART_HWCONTROL_NONE;
    Bluetooth_UARTHandle.Init.Mode       = UART_MODE_TX_RX;
    HAL_UART_Init(&Bluetooth_UARTHandle);           // Initialising the handle
}


void SystemClock_Config(void) {  // Configuring System Clock
    RCC_OscInitTypeDef RCC_OscInitStruct;
    RCC_ClkInitTypeDef RCC_ClkInitStruct;
    /* Enable Power Control clock */
    __HAL_RCC_PWR_CLK_ENABLE();
    /* The voltage scaling allows optimizing the power
    consumption when the device is clocked below the
    maximum system frequency. */
    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
    /* Enable HSE Oscillator and activate PLL
    with HSE as source */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
    RCC_OscInitStruct.HSEState = RCC_HSE_ON;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
    RCC_OscInitStruct.PLL.PLLM = 25;
    RCC_OscInitStruct.PLL.PLLN = 336;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
    RCC_OscInitStruct.PLL.PLLQ = 7;
    HAL_RCC_OscConfig(&RCC_OscInitStruct);
    /* Select PLL as system clock source and configure
    the HCLK, PCLK1 and PCLK2 clocks dividers */
    RCC_ClkInitStruct.ClockType = 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_DIV4;
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
    HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5);
}

// Function to delay the system for 1 second.
void delay(){
    HAL_Delay(1000);
}

// 2FA verification method, allows user to respond on compatible bluetooth device to confirm unlock function
bool verify(){
    init_uart();  // Initialising uart
    RxBuffer = 0; // Resetting RxBuffer each time to avoid potential issues
    GLCD_DrawString(56, 100, "Please confirm on device.");

    HAL_UART_Transmit(&Bluetooth_UARTHandle, (uint8_t *)"Confirm", 7, HAL_MAX_DELAY);  // Prints "Confirm" on connected bluetooth terminal
    HAL_UART_Receive(&Bluetooth_UARTHandle, &RxBuffer, 1, HAL_MAX_DELAY);              //  Takes response from bluetooth terminal and stores in RxBuffer


    if (RxBuffer != 'y'){ // If the buffer is not equal to 'y'
        return false;     // Verification failed
    }
    // If verified
    buzz_on();  // buzzer on
    delay();    // prolong buzzing
    buzz_off(); // turn buzzer off
    return true;// return successful

}
// Unlock function, checks a temporary array filled with Membrane Keypad inputs and compares to the current stored pin
bool unlock(){
    int attempt[8];  // attempt array, this stores the attempt that the user will input on the membrane keypad
    int i = 0;       // Loop variable which is incremented
    int spacing = 37;// This is used to space the asteriks inline with the drawn _'s.
    while(i<8){
        int Membrane = getInput();  // Getting input from the membrane keypad
        if (ScreenState != 2){      // If its not the unlock screen, break. The reason being is because this function should not run on any other screens
            break;
        }

        Touch_GetState(&tsc_state); // Getting the touch state
        if(tsc_state.pressed){
            CheckCoords(tsc_state.x, tsc_state.y);      // If CheckCoords is true, the ScreenState will change and the function will break based on the above if conditional
            tsc_state.pressed = 0;                              // I'm resetting the touch state here, just to avoid potential errors further into the program
        }

        if(Membrane!=-1){                   // If the membrane detects a valid input
            spacing+=42;
            GLCD_DrawString(spacing, 150, "*");     // Draw '*' for each input
            attempt[i] = Membrane;          // store the input into index i of temporary array attempt
            i++;                            // increment
            delay();                        // delay inputs using the delay function, prevents rapid inputs from one press of button
        }
    }


    for (i = 0; i < 8; i++){
        if (attempt[i] != pin[i]){          // Checking the array of inputs against current stored pin
            return false;                   // If one attempt is wrong, return false. Makes it more efficient as the worst case time complexity is (O(n)).
        }
    }
    if (verify() != true){                  // If the unlock is not verified return false
        return false;
    }
    return true;                            // If all loops continue, return true

}
/*
 * Confirm function, this function takes in a temporary attempt array from a previous function and checks for discrepancies between the two arrays.
 *
 */
bool confirm(int attempt[]){
    int temp_pin[8];    // Temporary pin array to store confirmation attempt
    int i = 0;          // Int that will be used to increment loops
    int spacing = 37;   // Used for spacing asteriks that will be printed each input

    while(i<8){
        int Membrane = getInput();  // Getting keypad input
        Touch_GetState(&tsc_state); // Getting the current touch state

        if(tsc_state.pressed){
            CheckCoords(tsc_state.x, tsc_state.y); // CheckCoords to check for back button press
            tsc_state.pressed = 0;                          // reset tsc_state to avoid potential errors
        }

        if (ScreenState != 5 && ScreenState != 6){      // If the screen is not on the confirmation screens, break
            break;
        }

        if(Membrane!=-1){                   // If valid Membrane Keypad input
            spacing+=42;
            GLCD_DrawString(spacing, 150, "*");
            temp_pin[i] = Membrane;         // store the current membrane input into temporary array
            i++;                            // increment i
            delay();                        // delay
        }
    }

    for (i =0; i < 8; i++){
        if (attempt[i] != temp_pin[i]){             // Checking the inputs against the array passed into the confirm function
            return false;                   // returns false if any input does not equal the array elements passed in the function
        }
    }
    return true;                            // If all loops finish, return true
}

bool resetSetPin(){
    int temp_pin[8];    // Storing a temporary input
    int i = 0;          // Int to increment loops
    int spacing = 37;   // Int used to apply spacing to asteriks inputted

    while(i<8){
        int Membrane = getInput();

        if (ScreenState != 4){  // If the user is not on the resetSetScreen
            return false;
        }
```

```c
        Touch_GetState(&tsc_state);

        if(tsc_state.pressed){
            CheckCoords(tsc_state.x, tsc_state.y);
            tsc_state.pressed = 0;

        }
        if(Membrane!=-1){
            spacing+=42;
            GLCD_DrawString(spacing, 150, "*");
            temp_pin[i] = Membrane;
            i++;
            delay();
        }
    }

    GLCD_ClearScreen(); // Clearing the screen for the next screen
    ScreenState = 5;    // Setting the ScreenState to be the ScreenState of the confirm screen, essential for not immediately returning false
    confirmScreen();

    while (ScreenState == 5){       // While user is on the confirm screen
        if (confirm(temp_pin)){     // If confirm returns ture
            for (i = 0; i < 8; i++){
            pin[i] = temp_pin[i];   // Store new pin into pin global variable
            }
            return true;            // return true
        } else {
            if (ScreenState != 5){  // If not on the confirm screen, return false
                return false;
            }
            GLCD_ClearScreen();     // Clear the screen
            confirmScreen();        // Go back to the confirm screen, this will also call the confirm function again
            GLCD_DrawString(72, 100, "Incorrect, try again.");     // Giving the user feedback
        }
    }
}

bool setPin(){
    int temp_pin[8]; // Temporary pin to store input
    int i = 0;
    int spacing = 37;

    while(i<8){
        int Membrane = getInput();

        Touch_GetState(&tsc_state);

        if(tsc_state.pressed){
            CheckCoords(tsc_state.x, tsc_state.y);
            tsc_state.pressed = 0;
        }

        if(Membrane!=-1){
            spacing+=42;
            GLCD_DrawString(spacing, 150, "*");

            temp_pin[i] = Membrane; // storing the inputs into the temporary array
            i++;                    // Incrementing loop
            delay();
        }
    }

    GLCD_ClearScreen();             // Clearing the screen
    ScreenState = 6;                // Setting the screen to the confirm screen for setting a new pin
    setPinConfirmScreen();
    while (ScreenState == 6){       // avoids the function from carrying over to the next screen
        if (confirm(temp_pin)){
            for (i = 0; i < 8; i++){
                pin[i] = temp_pin[i];   // if confirmed, store in global pin array
            }
            return true;
        } else {
            if (ScreenState != 6){   // If not on correct screen, return false
                return false;
            }
            GLCD_ClearScreen();      // Clearing the screen, this gets rid of previous asteriks
            setPinConfirmScreen();   // Go to the same screen
            GLCD_DrawString(72, 100, "Incorrect, try again."); // User feedback
        }
    }
}

bool resetPin(){
    int attempt[8];
    int i = 0;
    int spacing = 37;

    while(i<8){
        int Membrane = getInput();
        if (ScreenState != 3){      // If not on the reset screen, return false
            break;
        }

        Touch_GetState(&tsc_state);
        if(tsc_state.pressed){
                CheckCoords(tsc_state.x, tsc_state.y);
                tsc_state.pressed = 0;
        }

        if(Membrane!=-1){
            spacing+=42;
            GLCD_DrawString(spacing, 150, "*");
            attempt[i] = Membrane;
            i++;
            delay();
        }
    }

    for (i = 0; i < 8; i++){
        if (attempt[i] != pin[i]){
            return false;
        }
    }

    return true;
}

// This function will reset the attempts global variable attempts and return the value. (Returning the value seemed to work whereas if I did not the return the value it did not work.)
int attemptsReset(){
    attempts = 0;
    return attempts;
}
// Main function for changing between screens and calling the relevant functions for those screens
void changeScreen(int a){

    GLCD_SetFont(&GLCD_Font_16x24); // Setting the font for all the text
    photoresistor_main();           // Calls function from photoresistor.c. This function controls dark mode vs light mode
    GLCD_ClearScreen();             // Clearing the screen
    delay();                        // Delaying, reduces screen flickering

    switch(a){
        case 0:                     // a = 0
            ScreenState = 0;        // ScreenState = 0, which is representative of the setPin screen
            setPinScreen();

            while (ScreenState == 0 || ScreenState == 6){  // While at the setPinScreen or the setPinConfirm screen
                if (setPin()){                             // Calling the setPin function
                    changeScreen(1);                    // if it returns true, change to homescreen
                }
            }
            break;
        case 1:
            ScreenState = 1;         // Represents home screen
            hscreen();               // home screen
            GLCD_DrawString(192, 0*24, "Locked"); // Printing status for home screen
            break;

        case 2:
            ScreenState = 2;        // Represents unlock screen
            unlockScreen();
            attemptsReset();        // Resetting the attempts at the start, this prevents attempts carrying over from other screens or previous attempts

            while (ScreenState == 2){
                GLCD_ClearScreen();
                unlockScreen();     // This is essential, as it will reset asteriks from previous attempts
                attempts++;
                timed_delay(attempts); // timed_delay is defined in interrupt.c

                if (unlock()){      // if the unlock function returns true
                    changeScreen(4); // change to unlocked screen
                }
            }
            break;
        case 3:
            ScreenState = 3;    // Represents reset Screen
            attemptsReset();    // Resetting attempts to avoid carry over

            resetScreen();

            while (ScreenState == 3 || ScreenState == 4 || ScreenState==5){ //  While on resetScreen
or any of the confirm screens associated with resetting pin
                GLCD_ClearScreen();
                resetScreen();
                attempts++;
                timed_delay(attempts);

                if (resetPin()){
                    GLCD_ClearScreen();
                    ScreenState=4;          // represents screen to set after resetting

                    resetSetScreen();       // screen to set pin after resetting
                    if(resetSetPin()) {
                        changeScreen(1); // return to home screen
                    }
                }

            }

            break;

        case 4:
            ScreenState = 7;    // Represents unlocked screen
            opened_angle();     // calls opened_angle() from servo.c, which turns servo by 90 degrees
            HAL_Delay(3000);    // delay, allows time for user to open the door
            while (HAL_GPIO_ReadPin(GPIOI, GPIO_PIN_3) == GPIO_PIN_SET){   // This turns out to be the
opposite, whenever the tracking sensor does not detect something, the pin for the sensor is set
                turnOnGreen();  // from led_main.c, turns led green
                lockedScreen();
                HAL_Delay(3000); // delay for 3 seconds
                if (HAL_GPIO_ReadPin(GPIOI, GPIO_PIN_3) != GPIO_PIN_SET){   // If the tracking sensor
detects something, break out of while loop
                    break;
                }
            }
            closed_angle();     // servo.c function, turns servo 90 degrees opposite of what it was
just turned
            turnOnRed();        // turn on red led
            changeScreen(1); // go to home screen
            break;

        default:
            break;
    }
}

```

2.4 GLCDTOUCH.h

```

/*
     ################################
     Principle Author:

     Tommy Peach, 100358179

     File name:

     GLCDTOUCH.h

     Description:

     GLCDTOUCH.h defines functions that will be used in other source files

     ################################
 */
 #include "stm32f7xx_hal.h"
    #include "GLCD_Config.h"
    #include "Board_GLCD.h"
    #include "Board_Touch.h"
        #include <string.h>
        #include "buzzer.h"
        #include "keypad.h"
        #include "led_main.h"
        #include "servo.h"
        #include "tracking.h"
        #include "interrupt.h"
    #define wait_delay HAL_Delay

        extern GLCD_FONT GLCD_Font_16x24;
        extern UART_HandleTypeDef Bluetooth_UARTHandle;
        extern uint8_t RxBuffer;
        extern int pin[8];
        extern TOUCH_STATE tsc_state;
        extern int ScreenState;
        extern int attempts;


void changeScreen();
_Bool verify();
void delay();
void setPin();
_Bool unlock();
void init_uart(void);
void SystemClock_Config(void);
_Bool resetPin();
_Bool confirm(int attempt[]);
void setPin();
```

2.5 interrupt.c

```
/*
     ################################
     Principle Author:

     Tommy Peach, 100358179

     File name:

     interrupt.c

     Description:

     interrupt.c handles delaying the system when too many incorrect attempts are entered. It also handles the interrupt caused by the shock sensor.

     ################################
 */

#include <stdio.h>
#include "stm32f7xx_hal.h"
#include "stm32f7xx_hal_gpio.h"
#include "Board_GLCD.h"

#include "buzzer.h"
char countdown[1]; // buffer for showing seconds for delay

// Function for delaying and printing the number of seconds until countdown ends
void delay_seconds(int seconds)
{
        int i = 0;
        for (i = seconds; i >= 0; i--){         // This for loop decrements until the integer i reaches 0
            sprintf(countdown, "Delay: %d", i); // sprintf puts the text "Delay: %d" into the countdown buffer. %d is the identifier for what data type is meant to take its place.
            GLCD_DrawString(168,9*24, countdown); // Printing the countdown on the screen
            HAL_Delay(1000);                    // Delays the program for 1 second, 1000 because it is miliseconds (1000 miliseconds = 1 second)
        }
}
// Function for checking the interrupt, the function checks for the interrupt and if there is one the callback function will execute
void interrupt_check(void)
{
        // This function will check for the shock sensor interrupt, the gpio pin is gpio_pin_6 as this is the interrupt handler vector I configured in shock.c
        HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_6);
}
// Callback function for the interrupt, this function writes the pin for the buzzer, uses a for loop to delay the resetting of the buzzer pin
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){

                    HAL_GPIO_WritePin(GPIOH, GPIO_PIN_6, GPIO_PIN_SET); // Setting the buzzer pin

                    for (i = 0; i < 10000000; i++){                     // for loop to delay the reset of the buzzer, causing it to continue sounding
                    }

                    HAL_GPIO_WritePin(GPIOH, GPIO_PIN_6, GPIO_PIN_RESET);   // resetting the buzzer
}

// Function to delay the system, based on the amount of attempts have accumulated in GLCDTOUCH.c
void timed_delay(int wrong_attempts){

    // If two wrong attempts
    if (wrong_attempts == 2)
    {

        GLCD_DrawString(72, 100, "Incorrect, try again.");  // User feedback

        delay_seconds(5);  // Calls the delay_seconds function, wait for 5 seconds

    }
    else if (wrong_attempts == 3)
    {
        // Wait for 10 seconds
                GLCD_DrawString(72, 100, "Incorrect, try again.");

                delay_seconds(10);

    }
    else if (wrong_attempts >= 4)
    {
        // Wait for 15 seconds
                GLCD_DrawString(72, 100, "Incorrect, try again.");


                delay_seconds(15);


    }
}

```

2.6 interrupt.h

```
/*
     ################################
     Principle Author:

     Tommy Peach, 100358179

     File name:

     interrupt.h

     Description:

     interrupt.h defines functions that will be used in other source files.
      ################################
 */
void timed_delay(int wrong_attempts);
```

2.7 keypad.c

```
#include "keypad.h"
#include "stm32f7xx_hal.h"
#include "stm32f7xx_hal_gpio.h"

void initializeMembranePins (void){
    GPIO_InitTypeDef gpio;

    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOI_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();

    gpio.Mode = GPIO_MODE_OUTPUT_PP;
    gpio.Pull = GPIO_PULLDOWN;
    gpio.Speed = GPIO_SPEED_HIGH;

    // D8
    gpio.Pin = GPIO_PIN_2;
    HAL_GPIO_Init(GPIOI, &gpio);

    // D9
    gpio.Pin = GPIO_PIN_15;
    HAL_GPIO_Init(GPIOA, &gpio);

    // D10
    gpio.Pin = GPIO_PIN_8;
    HAL_GPIO_Init(GPIOA, &gpio);

    // D11
    gpio.Pin = GPIO_PIN_15;
    HAL_GPIO_Init(GPIOB, &gpio);

    // D12
    gpio.Pin = GPIO_PIN_14;
    HAL_GPIO_Init(GPIOB, &gpio);

    // D13
    gpio.Pin = GPIO_PIN_1;
    HAL_GPIO_Init(GPIOI, &gpio);

    // D14
    gpio.Pin = GPIO_PIN_9;
    HAL_GPIO_Init(GPIOB, &gpio);

    // D15
    gpio.Pin = GPIO_PIN_8;
    HAL_GPIO_Init(GPIOB, &gpio);
}

int convertPinsToNum(int k, int r){
    if(k == 11 && r == 15){
            return 1;
    }
    if(k == 10 && r == 15){
            return 2;
    }
    if(k == 9 && r == 15){
            return 3;
    }
    if(k == 8 && r == 15){
            return 11; //A
    }
    if(k == 11 & r == 14){
        return 4;
    }
    if(k == 10 & r == 14){
        return 5;
    }
    if(k == 9 & r == 14){
        return 6;
    }
    if(k == 8 & r == 14){
        return 12; //B
    }
    if(k == 11 & r == 13){
        return 7;
    }
    if(k == 10 & r == 13){
        return 8;
    }
    if(k == 9 & r == 13){
        return 9;
    }
    if(k == 8 & r == 13){
        return 13; //C
    }
    if(k == 11 & r == 12){
        return 15; //*
    }
    if(k == 10 & r == 12){
        return 0;
    }
    if(k == 9 & r == 12){
        return 16; //#
    }
    if(k == 8 & r == 12){
        return 14; //D
    }

    return 0;
}

void turnOn(int pinNo){
    switch (pinNo){
        case 8:
            HAL_GPIO_WritePin(GPIOI, GPIO_PIN_2, GPIO_PIN_RESET);
            break;
        case 9:
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, GPIO_PIN_RESET);
            break;
        case 10:
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_RESET);
            break;
        case 11:
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, GPIO_PIN_RESET);
            break;
        case 12:
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14, GPIO_PIN_RESET);
            break;
        case 13:
            HAL_GPIO_WritePin(GPIOI, GPIO_PIN_1, GPIO_PIN_RESET);
            break;
        case 14:
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_9, GPIO_PIN_RESET);
            break;
        case 15:
            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, GPIO_PIN_RESET);
            break;
    }
}

void turnOff(int pinNo){
        switch (pinNo){
            case 8:
                HAL_GPIO_WritePin(GPIOI, GPIO_PIN_2, GPIO_PIN_SET);
                break;
            case 9:
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, GPIO_PIN_SET);
                break;
            case 10:
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
                break;
            case 11:
                HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, GPIO_PIN_SET);
                break;
            case 12:
                HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14, GPIO_PIN_SET);
                break;
            case 13:
                HAL_GPIO_WritePin(GPIOI, GPIO_PIN_1, GPIO_PIN_SET);
                break;
            case 14:
                HAL_GPIO_WritePin(GPIOB, GPIO_PIN_9, GPIO_PIN_SET);
                break;
            case 15:
                HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, GPIO_PIN_SET);
                break;
    }
}

GPIO_PinState readPin(int number){
    switch(number){
        case 8:
            return HAL_GPIO_ReadPin(GPIOI, GPIO_PIN_2);
        case 9:
            return HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_15);
        case 10:
            return HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_8);
        case 11:
            return HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_15);
        case 12:
            return HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_14);
        case 13:
            return HAL_GPIO_ReadPin(GPIOI, GPIO_PIN_1);
        case 14:
            return HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_9);
        case 15:
            return HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_8);
        default:
            return GPIO_PIN_RESET;
    }
}

void setColsIn(){
    //set columns to write
    GPIO_InitTypeDef gpio;
    gpio.Mode = GPIO_MODE_INPUT;
    gpio.Pull = GPIO_PULLDOWN; //
    gpio.Speed = GPIO_SPEED_HIGH; //
    // D8
    gpio.Pin = GPIO_PIN_2;
    HAL_GPIO_Init(GPIOI, &gpio);

    // D9
    gpio.Pin = GPIO_PIN_15;
    HAL_GPIO_Init(GPIOA, &gpio);

    // D10
    gpio.Pin = GPIO_PIN_8;
    HAL_GPIO_Init(GPIOA, &gpio);

    // D11
    gpio.Pin = GPIO_PIN_15;
    HAL_GPIO_Init(GPIOB, &gpio);
}

void setColsOut(){
    //set columns to write
    GPIO_InitTypeDef gpio;
    gpio.Mode = GPIO_MODE_OUTPUT_PP;
    gpio.Pull = GPIO_PULLDOWN; //
    gpio.Speed = GPIO_SPEED_HIGH; //
    // D8
    gpio.Pin = GPIO_PIN_2;
    HAL_GPIO_Init(GPIOI, &gpio);

    // D9
    gpio.Pin = GPIO_PIN_15;
    HAL_GPIO_Init(GPIOA, &gpio);

    // D10
    gpio.Pin = GPIO_PIN_8;
    HAL_GPIO_Init(GPIOA, &gpio);

    // D11
    gpio.Pin = GPIO_PIN_15;
    HAL_GPIO_Init(GPIOB, &gpio);
}

int getInput (void){
    unsigned int k, r;

    //set rows high one at a time
    for (r = 12; r <= 15; r++){

            //set columns to low
            setColsOut();
            for (k = 8; k<=11; k++){
                turnOn(k);
            }
            setColsIn();

            turnOff(r);

            // check the value of each column
            for (k = 8; k <= 11; k++){
                    if(readPin(k) == GPIO_PIN_SET){
                        turnOn(r);
                        return convertPinsToNum(k, r);
                    }
            }
            turnOn(r);
    }

    return -1;
}
```

2.8 keypad.h

```
void initializeMembranePins (void);
int getInput (void);
```

2.9 led_main.c

```
/*
 ***********************************************************************************
 Author: Chaya Punnasri, 100386454
 Date of creation: 30 April 2024
 Name: led_main.c
 Description: Initialise and set/reset selected ports and pins for the use of an LED
 ***********************************************************************************
 */

#include <stdio.h> //include the library for a standard input and output
#include "stm32f7xx_hal.h" //include the library for HAL
#include "stm32f7xx_hal_gpio.h" // include the HAL library for a GPIO use
#include "led_main.h" // include the header file called led_main.h

// LED Red and Green gpio initialisation
void initialise_led(void){

    // this is for pin and port of green
    GPIO_InitTypeDef gpio; // initialise the type definition
    __HAL_RCC_GPIOI_CLK_ENABLE(); // enable clock for port I
    gpio.Pin = GPIO_PIN_0; // using pin 0
    gpio.Mode = GPIO_MODE_OUTPUT_PP; // output mode
    gpio.Pull = GPIO_NOPULL; // no pull for the GPIO output
    gpio.Speed = GPIO_SPEED_HIGH; // set the speed to high
    HAL_GPIO_Init(GPIOI, &gpio); // initialise port I pin 0

    // this is for pin and port of red
    __HAL_RCC_GPIOG_CLK_ENABLE(); // enable the clock for port G
    gpio.Pin = GPIO_PIN_7; // using pin 7
    gpio.Mode = GPIO_MODE_OUTPUT_PP; //output mode
    gpio.Pull = GPIO_NOPULL; // no pull for the GPIO output
    gpio.Speed = GPIO_SPEED_HIGH; // set the speed to high
    HAL_GPIO_Init(GPIOG, &gpio); // initialise port G pin 7
}

// function to set or clear the selected data port bit of a red colour
void turnOnRed(void){

    // set port G pin 7
    HAL_GPIO_WritePin(GPIOG, GPIO_PIN_7, GPIO_PIN_SET);
    // clear/reset port I pin 0
    HAL_GPIO_WritePin(GPIOI, GPIO_PIN_0, GPIO_PIN_RESET);
}

// function to set or clear the selected data port bit of a green colour
void turnOnGreen(void){

    // set port I pin 0
    HAL_GPIO_WritePin(GPIOI, GPIO_PIN_0, GPIO_PIN_SET);
    // clear/reset port G pin 7
    HAL_GPIO_WritePin(GPIOG, GPIO_PIN_7, GPIO_PIN_RESET);
}


```

2.10 led_main.h

```
/*
 ***************************************************************************************************
 Author: Chaya Punnasri, 100386454
 Date of creation: 30 April 2024
 Name: led_main.h
 Description: It provides function prototypes to be used in other c file for a further implementation
 ***************************************************************************************************
 */

// providing a function prototype for the initialise_led() function
void initialise_led(void);

// providing a function prototype for the turnOnRed() function
void turnOnRed(void);

// providing a function prototype for thw turnOnGreen() function
void turnOnGreen(void);
```

2.11 photoresistor.c

```
/*
 ***************************************************************************************
 Author: Chaya Punnasri, 100386454
 Date of creation: 5 May 2024
 Name: photoresistor.c
 Description: Initialise the system clock configuration for the RCC and the PWR which
 will be used in an analogue signal generation, after that the signal is converted into
 the digital signal
 ***************************************************************************************
 */

#include <stdio.h>
#include "stm32f7xx_hal.h"
#include "GLCD_Config.h" // include the GLCD_config.h
#include "Board_GLCD.h" // include the Board_GLCD.h
#include "Board_Touch.h" // include Board_Touch.h
#include "photoresistor.h" // include the photoresistor.h
#define wait_delay HAL_Delay
#include <math.h> // include the use of a math library

```

```
// calling the function from GLCD_Touch.h
extern GLCD_FONT GLCD_Font_16x24;


/*
* System Clock Configuration
*/

void SystemClock_Config_Photoresistor(void) {
    RCC_OscInitTypeDef RCC_OscInitStruct;
    RCC_ClkInitTypeDef RCC_ClkInitStruct;
    /* Enable Power Control clock */
    __HAL_RCC_PWR_CLK_ENABLE();
    /* The voltage scaling allows optimizing the power
    consumption when the device is clocked below the
    maximum system frequency. */
    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
    /* Enable HSE Oscillator and activate PLL
    with HSE as source */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
    RCC_OscInitStruct.HSEState = RCC_HSE_ON;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
    RCC_OscInitStruct.PLL.PLLM = 25;
    RCC_OscInitStruct.PLL.PLLN = 336;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
    RCC_OscInitStruct.PLL.PLLQ = 7;
    HAL_RCC_OscConfig(&RCC_OscInitStruct);
    /* Select PLL as system clock source and configure
    the HCLK, PCLK1 and PCLK2 clocks dividers */
    RCC_ClkInitStruct.ClockType = 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_DIV4;
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
    HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5);
}

// initialise the use of analogue to digital convertor
ADC_HandleTypeDef hadc;

void MX_ADC_Init(void)
{
    ADC_ChannelConfTypeDef sConfig;

    /* Enable ADC CLOCK For every ADC */
    __HAL_RCC_ADC1_CLK_ENABLE();
    __HAL_RCC_ADC2_CLK_ENABLE();
    __HAL_RCC_ADC3_CLK_ENABLE();

    /* Configure the global features of the ADC (Clock, Resolution, Data Alignment and number
    of conversion) */
    hadc.Instance = ADC3; // choosing ADC3 as we are using the port A pin 0
    hadc.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV2;
    hadc.Init.Resolution = ADC_RESOLUTION_10B;
    hadc.Init.DataAlign = ADC_DATAALIGN_RIGHT;
    hadc.Init.NbrOfConversion = 1;
    hadc.Init.ScanConvMode = ENABLE;
    hadc.Init.ContinuousConvMode = ENABLE;
    hadc.Init.DiscontinuousConvMode = DISABLE;
    HAL_ADC_Init(&hadc);
    //configure channal
    sConfig.Rank = 1;
    sConfig.Channel = ADC_CHANNEL_0; // choosing to use a channel 0 because it is compatible for port
A pin 0
    sConfig.SamplingTime = ADC_SAMPLETIME_28CYCLES;
    HAL_ADC_ConfigChannel(&hadc, &sConfig);
    HAL_ADC_Start(&hadc);
    HAL_ADC_PollForConversion(&hadc, HAL_MAX_DELAY);
}

// initialise a GPIO for analog
GPIO_InitTypeDef GPIO_InitStruct;
void MX_GPIO_Init(void){
    __HAL_RCC_GPIOA_CLK_ENABLE(); // enable clock for port A
    GPIO_InitStruct.Mode = GPIO_MODE_ANALOG; // configure to analog input mode
    GPIO_InitStruct.Pin = GPIO_PIN_0; // pin 0
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
}



// control LCD brightness
void ctrl_brightness (float adc_read){
    float V_pwr=5.0; //configure the V_pwr
    float V_sense= (adc_read * V_pwr)/1024; //calculate the V_sense which 1024 is the reading range
    float brightness=V_sense/V_pwr; // calculate the ratio of brightness

    if(brightness > 0.2)    {
        // toggle the dark mode
            GLCD_SetBackgroundColor(GLCD_COLOR_BLACK);
            GLCD_SetForegroundColor(GLCD_COLOR_WHITE);
            GLCD_ClearScreen();
        }
        else {
            // toggle the light mode
            GLCD_SetBackgroundColor(GLCD_COLOR_WHITE);
            GLCD_SetForegroundColor(GLCD_COLOR_BLACK);
            GLCD_ClearScreen();

        }
}

// main loop for a photoresistor
int photoresistor_main(void){

    uint16_t adc_read; // declare a variable for the ADC reading
    adc_read = HAL_ADC_GetValue(&hadc);
    SystemClock_Config_Photoresistor(); //Config Clocks
    MX_GPIO_Init();
    MX_ADC_Init();
        // read ADC value
        ctrl_brightness(adc_read);
        wait_delay(1);
}
```

2.12 photoresistor.h

```
/*
 ****************************************************************************************************
 Author: Chaya Punnasri, 100386454
 Date of creation: 5 May 2024
 Name: photoresistor.h
 Description: It provides function prototypes to be used in other c file for a further implementation
 ****************************************************************************************************
 */

int photoresistor_main(void);
void MX_GPIO_Init(void);
void MX_ADC_Init(void);
void SystemClock_Config_Photoresistor(void);
```

2.13 Safe.c

```
/*
     ################################
     Principle Author:

     Tommy Peach, 100358179

     File name:

     safe.c

     Description:

     safe.c is responsible for initialising all of the components that are not already initialised and
also checks for touch screen presses

     ################################
 */
#include "GLCDTOUCH.h"
#include "buzzer.h"
#include "keypad.h"
#include "photoresistor.h"
 #include "Board_Touch.h"
 #include "shock.h"
 #include "screens.h"

  int main(void){

                initializeMembranePins();
                HAL_Init(); //Init Hardware Abstraction Layer

                SystemClock_Config();
                Touch_Initialize();
                GLCD_Initialize();
                TIM2_Init();
                PA15_Init();
                initialise_input();
                initialise_led();
                MX_GPIO_Init();
                MX_ADC_Init();
                SystemClock_Config_Photoresistor();
                __HAL_RCC_GPIOC_CLK_ENABLE();
                __USART6_CLK_ENABLE();
                init_shock();
                init_buzzer();
                changeScreen(0); // go to set pin screen

            for(;;){
                Touch_GetState(&tsc_state);
                    if(tsc_state.pressed){
                        tsc_state.pressed = 0;
                        CheckCoords(tsc_state.x, tsc_state.y);
                        continue;
                    }
                }
}
```

2.14 screens.c

```
/*
     ################################
     Principle Author:

     Tommy Peach, 100358179

     File name:

     screens.c

     Description:

     Screens.c handles the screens, specifically drawing the elements on each screen, as well as handling the transition between screens through CheckCoords.
     From the stm website, I was able to determine that the resolution of the screen was 480x272 pixels, so this is the number the calculations were based on for centering the text.

     Calculations for centering text goes as follows:

     (16 pixels (size of the font width) * (number of letters)) / 2) - 240.
     This calculation takes the midpoint of the sentence/text and minuses half of the screen size to find the point where the text should start.

     ################################
 */
#include "GLCD_Config.h"
#include "Board_GLCD.h"
#include "GLCDTOUCH.h"
int ScreenState = 0; // Initialising ScreenState, 0 represents the first screen which is the set pin screen

/* CheckCoords checks the coordinates that a user has pressed on the screen, and passes these values into if statements.
 * The conditions inside of the if statements make sure that the correct coordinates are pressed for the back buttons on each screen.
 * It also checks for the "Unlock" and "Reset Pin" buttons on the home screen.
 * Each if statement includes the ScreenState for which screens these conditions should apply for. This avoids changing the screen for unintended screens.
 */

void CheckCoords(int x, int y){
    // If statement to transition back to the home screen
    if ((x > 0 && x < 70) && (y > 0 && y < 25) && (ScreenState == 2 || ScreenState == 3
        || ScreenState == 4 || ScreenState==5 )){
        changeScreen(1);
    }
    // If statement for transition to the unlock screen
    if ((x < 288 && x > 192) && (y < 140 && y > 116) && ScreenState == 1){
        changeScreen(2);
    }
    // If statement for transition to the reset pin screen
    if ((x < 312 && x > 168) && (y < 92 && y > 68) && ScreenState == 1){
        changeScreen(3);
    }
    // Transition for the set pin confirmation screen, if the user wants to change the pin on the first time setting
    if ((x > 0 && x < 70) && (y > 0 && y < 25) && (ScreenState == 6)){
        changeScreen(0);
    }
}


void unlockScreen(){
            int x = 37;
            int i = 0;
            GLCD_DrawString(192, 0*24, "UNLOCK");
            GLCD_DrawString(104 , 3*24, "Input Combination");
            GLCD_DrawString(0*24, 0*24, "Back");
            GLCD_DrawRectangle(0*24, 0*24, 65, 25); // Drawing back button box
            // Drawing input boxes
            for (i = 0; i <= 7; i++) {
                x += 42;
                GLCD_DrawString(x, 136, "_");
            }
    }

void resetScreen(){
    int x = 37;
    int i = 0;
    GLCD_DrawString(168, 0*24, "Reset Pin");
    GLCD_DrawString(40 , 3*24, "Input current combination");
    GLCD_DrawString(0*24, 0*24, "Back");
    GLCD_DrawRectangle(0*24, 0*24, 65, 25); // Drawing back button box

    // Drawing input boxes
    for (i = 0; i <= 7; i++){
        x += 42;
        GLCD_DrawString(x, 136, "_");
    }
}

void hscreen(){
    GLCD_DrawString(192, 116, "Unlock");
    GLCD_DrawString(168, 68, "Reset Pin");
}


void lockedScreen(){
        GLCD_DrawString(72, 4*24, "Please close door to");
        GLCD_DrawString(40, 5*24, "automatically lock system");
}

void confirmScreen(){
    int x = 37;
    int i = 0;
    GLCD_DrawString(152, 0*24, "Confirm Pin");
    GLCD_DrawString(88, 3*24, "Confirm combination");
    GLCD_DrawString(0*24, 0*24, "Back");
    GLCD_DrawRectangle(0*24, 0*24, 65, 25);
    // Drawing input boxes
    for (i = 0; i <= 7; i++){
        x += 42;
        GLCD_DrawString(x, 136, "_");
    }
}
void resetSetScreen(){
    int x = 37;
    int i = 0;
    GLCD_DrawString(168, 0*24, "Reset Pin");
    GLCD_DrawString(72, 3*24, "Input new combination");
    GLCD_DrawString(0*24, 0*24, "Back");
    GLCD_DrawRectangle(0*24, 0*24, 65, 25);

    // Drawing input boxes
    for (i = 0; i <= 7; i++){
        x += 42;
        GLCD_DrawString(x, 136, "_");
    }
}
void setPinScreen(){
    int x = 37;
    int i = 0;
    GLCD_DrawString(184, 0*24, "Set pin");
    GLCD_DrawString(104, 3*24, "Input combination");
    // Drawing input boxes
    for (i = 0; i <= 7; i++){
        x += 42;
        GLCD_DrawString(x, 136, "_");
    }
    ScreenState = 0;
}
void setPinConfirmScreen(){
    int x = 37;
    int i = 0;
    GLCD_DrawString(184, 0*24, "Set pin");
    GLCD_DrawString(104, 3*24, "Confirm combination");
    GLCD_DrawString(0*24, 0*24, "Back");
    GLCD_DrawRectangle(0*24, 0*24, 65, 25);
    // Drawing input boxes
    for (i = 0; i <= 7; i++){
        x += 42;
        GLCD_DrawString(x, 136, "_");
    }
}
```

2.15 screens.h

```

/*
     ################################
     Principle Author:

     Tommy Peach, 100358179

     File name:

     screens.h

     Description:

     screens.h defines functions to be used in other source files.
   ################################
 */
    #include "GLCD_Config.h"
    #include "Board_GLCD.h"
        #include "String.h"
        extern int ScreenState;

        void unlockScreen();
        void resetScreen();
        void homeScreen();
        void hscreen();
        void lockedScreen();
        void confirmScreen();
        void resetSetScreen();
        void setPinScreen();
        void setPinConfirmScreen();
        int CheckCoords(int x, int y);
```

2.16 servo.c

```
/*
 ***************************************************************************************
 Author: Chaya Punnasri, 100386454
 Date of creation: 25 April 2024
 Name: servo.c
 Description: The servo will be working by initialise the GPIO pin PB4 which supports
 the PWM and that pin is using TIM3 channel 1. The servo turns different angles
 ***************************************************************************************
 */

#include "stm32f7xx_hal.h" // HAL library is being used
#include "servo.h"

TIM_HandleTypeDef htim2;

void PA15_Init(){
    GPIO_InitTypeDef gpio;

    __HAL_RCC_GPIOB_CLK_ENABLE(); // enable the clock for a base B

    gpio.Pin = GPIO_PIN_4; // pin 4
    gpio.Mode = GPIO_MODE_AF_PP;
    gpio.Pull = GPIO_NOPULL;
    gpio.Speed = GPIO_SPEED_HIGH;
    gpio.Alternate = GPIO_AF2_TIM3;
    HAL_GPIO_Init(GPIOB, &gpio);
}


void TIM2_Init(void){
    TIM_ClockConfigTypeDef sClockSourceConfig;
    TIM_OC_InitTypeDef sConfigOC;

    //Timer configuration
    htim2.Instance = TIM3;
    htim2.Init.Prescaler = 32000; // set prescaler to 32000 and the signal 500Hz on the TIM3
    htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
    htim2.Init.Period = 9; // set period to 9 then it becomes 10, so the frequency will be 50 Hz cycle
    htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;

    HAL_TIM_Base_Init(&htim2);

    //Set the timer in PWM mode
    sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
    HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig);
    HAL_TIM_PWM_Init(&htim2);

    //Channel configuration
    sConfigOC.OCMode = TIM_OCMODE_PWM1;
    sConfigOC.Pulse = 0;
    sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
    sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
    HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_1);

    PA15_Init();
}

// opened angle which is 90 degrees
void opened_angle(void){

    //Reset of all peripherals, initializes the Flash interface and the Systick
    HAL_Init();

    //Enable peripheral clock for TIM3
    __HAL_RCC_TIM3_CLK_ENABLE();

    //Initialize TIM3, CH1 and PB4
    TIM2_Init();

    //Start PWM on TIM3_CH1
    HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);

    // change the CCR to 1 so the output will be high
    htim2.Instance->CCR1 = 1;

    // perform a delay for 1 second
    HAL_Delay(1000);
}

// closed angle which is 0 degree
int closed_angle(void){

    //Reset of all peripherals, initializes the Flash interface and the Systick
    HAL_Init();

    //Enable peripheral clock for TIM3
    __HAL_RCC_TIM3_CLK_ENABLE();

    //Initialize TIM3, CH1 and PB4
    TIM2_Init();

    //Start PWM on TIM3_CH1
    HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);

    // change the CCR to 3 so the output will be low
    htim2.Instance->CCR1 = 3;

    //perform a delay for a second
    HAL_Delay(1000);
}
```

2.17 servo.h

```
/*
 ****************************************************************************************************
 Author: Chaya Punnasri, 100386454
 Date of creation: 25 April 2024
 Name: servo.h
 Description: It provides function prototypes to be used in other c file for a further implementation
 ****************************************************************************************************
 */

void PA15_Init();
void TIM2_Init(void);
void opened_angle(void);
void closed_angle(void);

// link to the function in GLCD_touch.c
void SystemClock_Config(void);
```

2.18 shock.c

```
/*
     ################################
     Principle Author:

     Tommy Peach, 100358179

     File name:

     shock.c

     Description:

     shock.c is responsible for initialising the shock sensor. It also sets the priority for the
interrupt for the shock sensor and enables the handler for the shock sensor.

     ################################
 */
#include <stdio.h>
#include "stm32f7xx_hal.h"
#include "stm32f7xx_hal_gpio.h"
#include "shock.h"
void init_shock(void){

    GPIO_InitTypeDef gpio;
    __HAL_RCC_GPIOG_CLK_ENABLE();
    gpio.Pin = GPIO_PIN_6;      // Setting relevant pin 6 for GPIOG

    /* Interrupt mode, detects interrupt if the voltage goes low to high.
     * This seems to make sense as after looking up how the shock sensor works, it seems once the
inside coil touches the outer casing it will return high.
     * This mode was found in stm32f7xx_hal_gpio.h.
     */
    gpio.Mode = GPIO_MODE_IT_RISING;

    gpio.Pull = GPIO_NOPULL;
    gpio.Speed = GPIO_SPEED_HIGH;
    HAL_GPIO_Init(GPIOG, &gpio);

    /* Setting the interrupt priority, 0 being the highest priority as stated in lecture notes.
     * According to the reference manual
(https://learn.uea.ac.uk/bbcswebdav/pid-4513906-dt-content-rid-29049770_1/xid-29049770_1, page 296)
     * and the vector table in startup_stm32f746xx.s,
     * EXTI9_5_IRQn seems to be the interrupt service handler for pins 5 - 9, as shock is pin 6.
     */
     HAL_NVIC_SetPriority(EXTI9_5_IRQn, 0, 0);
     // Enabling the interrupt service
     HAL_NVIC_EnableIRQ(EXTI9_5_IRQn);

}
```

2.19 shock.h

```
/*
     ################################
     Principle Author:

     Tommy Peach, 100358179

     File name:

     shock.h

     Description:

     shock.h declares functions to be used in other source files.
   ################################
 */

void init_shock(void);
```

2.20 tracking.c

```c
/*
 ***************************************************************************************
 Author: Chaya Punnasri, 100386454
 Date of creation: 8 May 2024
 Name: tracking.c
 Description: The tracking sensor is being initialised and being used as an input GPIO
 ***************************************************************************************
 */

#include <stdio.h>
#include "stm32f7xx_hal.h"
#include "stm32f7xx_hal_gpio.h"

void initialise_input(void){
    // tracking sensor port initialisation
    GPIO_InitTypeDef gpio;
    __HAL_RCC_GPIOI_CLK_ENABLE(); // enable clock for base I
    gpio.Pin = GPIO_PIN_3; // pin 3
    gpio.Mode = GPIO_MODE_INPUT; // set mode to input as it needs to detect for the object moving
nearby
    gpio.Pull = GPIO_PULLUP;
    gpio.Speed = GPIO_SPEED_HIGH;
    HAL_GPIO_Init(GPIOI, &gpio);
}
```

2.21 tracking.h

```c
/*
 ***************************************************************************************
 Author: Chaya Punnasri, 100386454
 Date of creation: 8 May 2024
 Name: tracking.h
 Description: It provides function prototypes to be used in other c file for a further implementation
 ***************************************************************************************
 */

void initialise_input(void);
```

Index