Information Techonology University

Attendance System

Microcontrollers & Interfacing

Rehan Hafiz

Submitted by:

Muhammad Miqdad Ahmad (BSCE22001)

Muhammad Moiz Ahmad (BSCE22029)

Theory

This project focuses on making an attendance system using RFID cards and readers. RFID stands for radio frequency identification. It focuses on storing some data over an RFID card using radio waves and then reading the data from that card.

An RFID card is made from a microchip with a coiled antenna that can communicate with a nearby reader.

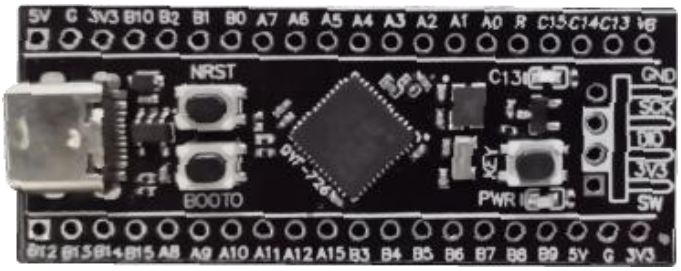
List of components

List of the required components is given under.

Components	Cost	Datasheets	Operation
STM32F407G	15000 Rs	User manual	Controller
EM-18 RFID reader/ MFRC522 RFID card reader	1900 Rs	Data sheet	Card readers
16 x 2 LCD display module	350 Rs	Data sheet	Liquid crystal
Bread Boards	300 Rs	No sheet	Bread boards
RFID cards/ RFID tags	90 Rs	No sheet	RFID cards
Wires	200 Rs	No sheet	Jumper wires
Potentiometer	25 RS	NO sheet	Brightness control

Components detail STM32F411 black pill board

This is the board that we will be using for this project. This is the black pill board



and was designed by the WeAct studio. The black pill board uses an STM32F411 processor. It features the STM32F411CEU6 chip with 512KB of ROM, 128KB of SRAM

and runs at 100MHz. There is a spot on the bottom for SOIC flash memory, which allows soldering an SPI flash memory for more space for data logging or file storage.

Use:

Will be used to run the entire project as the central processing unit.

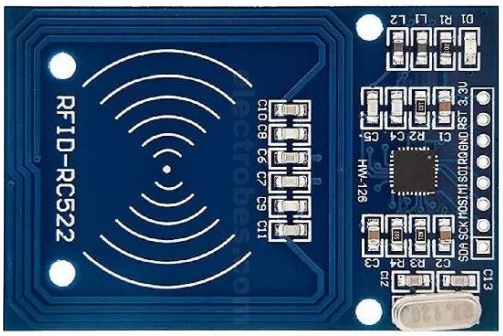
EM-18 reader module

EM18 is a RFID reader which is used to read RFID cards of 125 kHz frequency. After reading tags, it transmits unique ID serially to the PC or microcontroller using UART communication or Wiegand format on respective pins. EM18 RFID reader reads the data from RFID tags which contains stored ID which is of 12 bytes.

- **Use**:
  - This module will be used to read the RFID cards.

- **Operating Principle:**
  - It uses radio waves to identify a specific RFID tag.

MFRC522 RFID



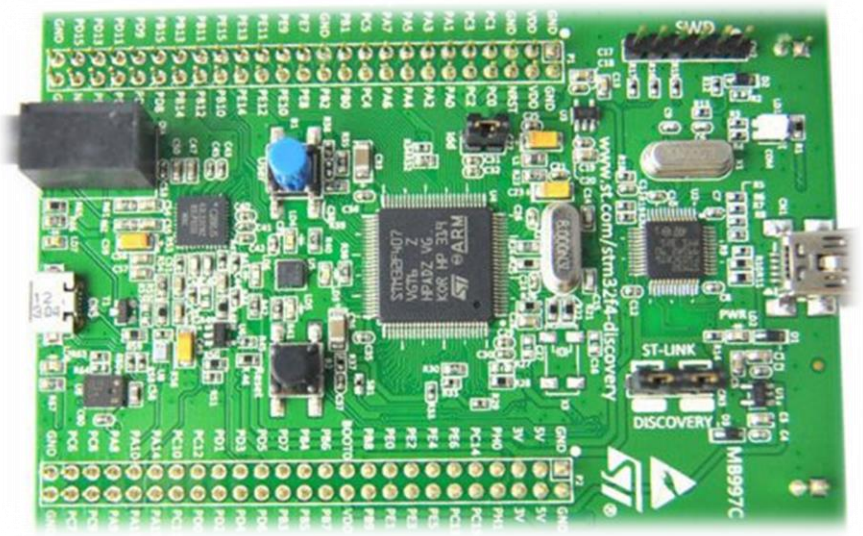
This is a highly integrated reader/writer module for contactless communication at 13.56 MHz. It has Buffered output drivers to connect an antenna with minimum number of external components. The typical distance for communication with this module is 50 mm. Supports high speed transfers of about 848 kbits/s. Comfortable 16 byte send and receive FIFO buffer. Flexible interrupts, free programmable io pins internal self

- **Use**:
  - This module will be used to read the RFID cards.

- **Operating Principle:**
  - It uses radio waves to identify a specific RFID tag.

STM32F407 discovery board

The STM32F407 Discovery board is a development kit designed to provide a simple and efficient way for users to get started with the STM32F4 series of microcontrollers from STMicroelectronics. It is particularly aimed at developers and engineers working on applications that require high-performance processing and advanced features. Here's an introduction to the STM32F407 Discovery board



- *STM32F407VG*: This is the core microcontroller on the board. It features:
- ARM Cortex-M4 32-bit RISC core operating at up to 168 MHz
- 1 MB of Flash memory
- 192 KB of SRAM
- Floating-point unit (FPU)
- Digital signal processing (DSP) instructions

RFID tags/ cards

RFID stands for radio frequency identification. These are made up of RFID tags are made up of a microchip with a coiled antenna that can communicate with a nearby reader wirelessly. Different kind of RFID tags with a different kind of shapes and sizes are available in the market. Few of them use different frequency for communication purpose. We will use 125Khz Passive RFID cards which holds the unique ID data.

Passive RFID tags draw power from the magnetic field that was created by the reader module like EM-18 and use it to power the microchip's circuits. The chip then sends information to the reader.

Active RFID tags requires separate power supply and contain up to 1MB of read/write memory.

- **Use**:
  - These are the RFID cards that will be used to store the data of a person.

- **Operating Principle:**
  - It communicates wirelessly wait a nearby reader to transfer data.

---

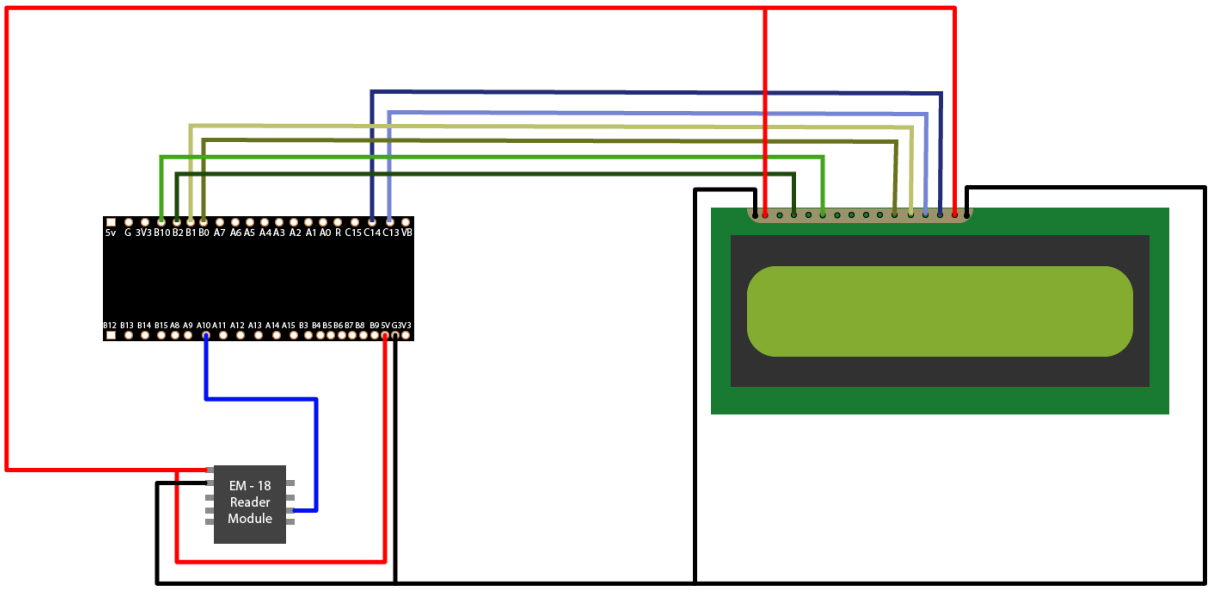
16 x 2 LCD display module

An LCD screen is an electronic display module that uses liquid crystal to produce a visible image. The 16×2 LCD display is a very basic module commonly used in circuits. The 16×2 translates a display of 16 characters per line in 2 such lines. In this LCD, each character is displayed in a 5×7 pixel matrix.

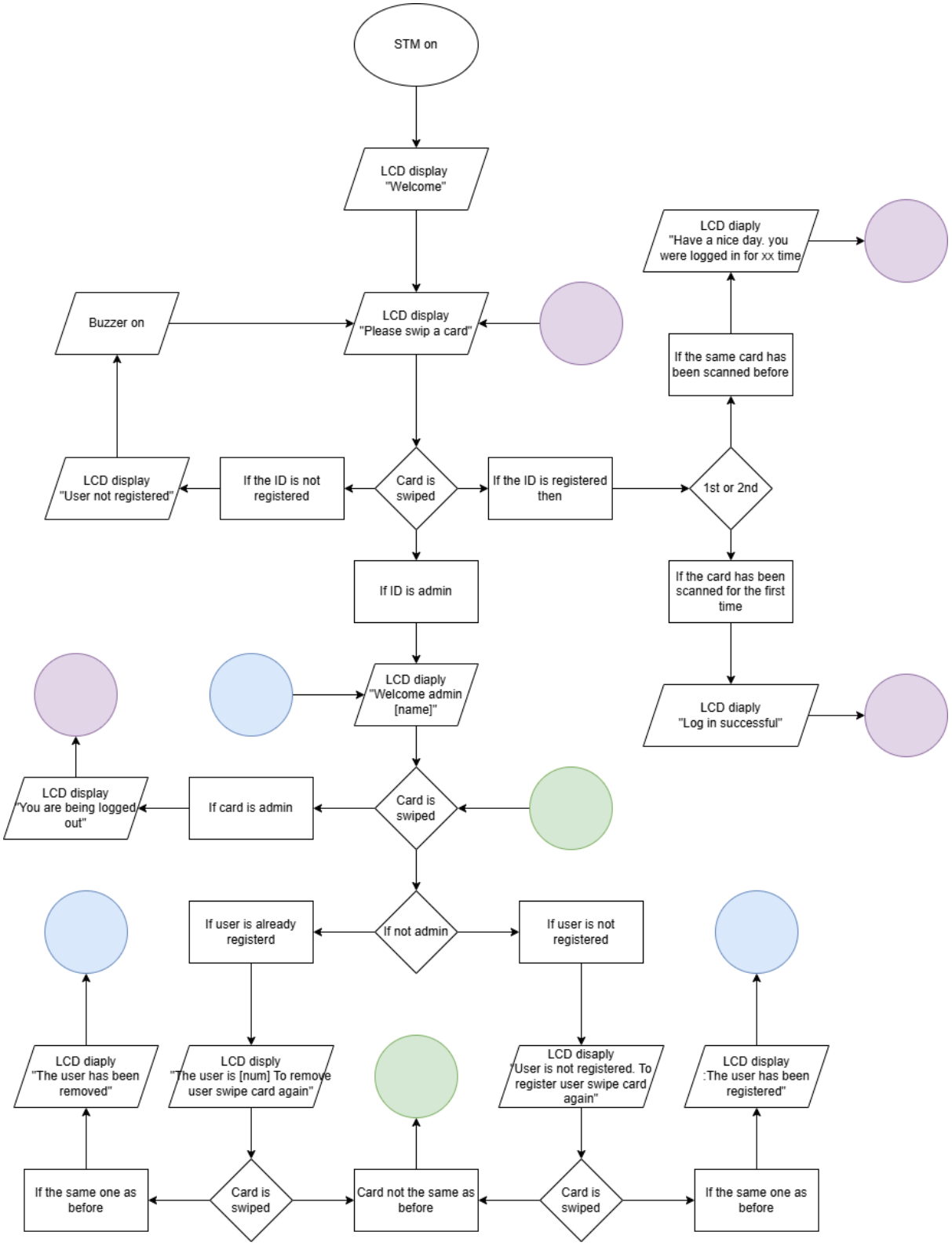
A 16X2 LCD has two registers, namely, command and data. The register select is used to switch from one register to other. RS=0 for the command register, whereas RS=1 for the data register.

- **Use**:
  - The LCD module will be used to display the data of the user.
- **Operating Principle:**
  - It is an electronic display that uses liquid crystals to produce a visible image. It consists of a grid of 16 columns and 2 rows of characters. Each character position can display a standard ASCII character or custom-defined symbol.

Block Diagram



Flow Diagram



Pin configuration

PINS	Usage
5V	LCD (A, VDD), EM 18 (Vcc)
PD13	(LCD) D6
PD8	(LCD) RS
PD9	(LCD) RW
PD10	(LCD) E
PD11	(LCD) D4
PD12	(LCD) D5
PD14	(LCD) D7
GND	LC (K) RC 522 (GND)
PA4	SDA(rc-522)
PA5	SCK(rc-522)
PA6	MISO(rc-522)
PA7	MOSI(rc-522)
PB0	RST(rc-522)
PB1	Buzzer

Code:

Header

```
#pragma once
#include <stdio.h>
#include <stdint.h>
#include <inttypes.h>
#include <string.h>
#include "stm32f407xx.h"
#include "stm32f4xx_hal.h"
#include "LiquidCrystal.h"

void admin();
void lcd_init();
void spi_config();
void SPI1_IRQnHandler();
void SysTick_Handler(void);
void print_floats(float val);
void print_ints(uint32_t val);
```

Functions

```
#include "functions.h"
uint32_t ID_1 = 0x0010322033;
uint32_t ID_2 = 0x0010322033;
uint32_t ID_3 = 0x0010322033;
uint32_t ID_Admin = 0x0010277850;
```

```
void SysTick_Handler(void)
{
    HAL_IncTick();
}

void lcd_init()
{
    HAL_Init();
    LiquidCrystal(GPIOD, GPIO_PIN_8, GPIO_PIN_9, GPIO_PIN_10, GPIO_PIN_11, GPIO_PIN_12, GPIO_PIN_13,
GPIO_PIN_14);
    // RCC->AHB1ENR |= (1 << 3);
}

void print_floats(float val)
{
    int temp = val;
    int count_num = 0;
    while (temp != 0)
    {
        temp = temp / 10;
        count_num++;
    }
    {
        char numbers[count_num - 1];
        temp = val;
        for (int i = count_num - 1; i >= 0; i--)
        {
            numbers[i] = (temp % 10) + 48;
            temp = temp / 10;
        }
        for (int i = 0; i < count_num; i++)
        {
            write(numbers[i]);
        }
    }
    write('.');
    {
        temp = val * 100;
        char decimals[2];
        for (int i = 0; i < 2; i++)
        {
            decimals[i] = (temp % 10) + 48;
            temp = temp / 10;
        }
        for (int i = 0; i < 2; i++)
        {
            write(decimals[i]);
        }
    }
}

void print_ints(uint32_t val)
{
    char b[64];
    sprintf(b, "%lu", val);
    print(b);
}

void spi_config()
{
    // Enabling the clocks
    RCC->AHB1ENR |= (1 << 1) | (1 << 0);                                        // Enable port
B
    GPIOB->MODER |= (1 << 0) | (1 << 2);                                        // general
purpose output mode
    GPIOA->MODER |= (1 << 9 /*4*/) | (1 << 11 /*5*/) | (1 << 13 /*6*/) | (1 << 15 /*7*/);   // set the
pins in alternate function mode
    GPIOA->AFR[0] |= (5 << 16 /*4*/) | (5 << 20 /*5*/) | (5 << 24 /*6*/) | (5 << 28 /*7*/); // set the
alternate function mode
```

```
    RCC->APB2ENR |= (1 << 12); // Enabling the SPI1 that is on the ports A4 ,5, 6, 7

    // Now to configure the SPI1
    // SPI1->CR1 &= ~(1 << 15); // set unidirectional mode
    // SPI1->CR1 |= (1 << 10);  // set to recive only
    SPI1->CR1 |= (1 << 15);  // set to bidirectional mode
    SPI1->CR1 &= ~(1 << 14); // output disable, recieve only mode
    SPI1->CR1 |= (1 << 2);   // set it as master
    SPI1->CR2 |= (1 << 6);   // set the recieve interrupt
    SPI1->CR1 |= (1 << 4);   // shayad prescaler to 8
    NVIC_EnableIRQ(SPI1_IRQn);
    NVIC_SetPriority(SPI1_IRQn, 0);

    SPI1->CR1 |= (1 << 6); // spi enable
}

void SPI1_IRQnHandler()
{
    if (ID_1 == SPI1->DR || ID_2 == SPI1->DR || ID_3 == SPI1->DR)
    {
        clear();
        setCursor(0, 0);
        print("Login successful");
        setCursor(0, 1);
        print("Welcome");
        HAL_Delay(5000);
        clear();
    }
    else if (SPI1->DR == ID_Admin)
    {
        admin();
    }
    else
    {
        clear();
        setCursor(0, 0);
        print("Invalid card");
        GPIOB->ODR |= (1 << 1);
        HAL_Delay(5000);
        clear();
        GPIOB->ODR &= ~(1 << 1);
    }
}

void admin()
{
    clear();
    setCursor(0, 0);
    print("Welcome Admin");
}
```

Main

```
#include "functions.h"

int main()
{
    lcd_init();
    spi_config();
    clear();
    setCursor(0,0);
    print("Welcome");
    HAL_Delay(5000);
    clear();
    setCursor(0,0);
    print("Please wait");
    setCursor(0,1);
    print("Loading");
    for (int i = 0; i < 5; i++)
    {
```

```
        print(".");
        HAL_Delay(3000);
    }
    clear();
    while(1)
    {
        setCursor(0,0);
        print("Tap a card...");
    }
    return 0;
}
```

References:

- https://thongrobot.wordpress.com/2017/07/22/stm32f407-rfid-project-full-code/
- https://stm32f4-discovery.net/2014/07/library-23-read-rfid-tag-mfrc522-stm32f4xx-devices/