4357. Embedded Firmware Essentials

Final Project Report

6/1/2015

Jae Yang Park (jaeyangp@gmail.com)

Objective

This project is to implement the Fingerprint recognition system using a fingerprint sensor module, and mbed platform and mbed application board hardware.

What's fingerprint recognition?

"It refers to automated method of verifying a match between two human fingerprints. Finger prints are one of many forms of biometrics used to identify individuals and verify their identity."

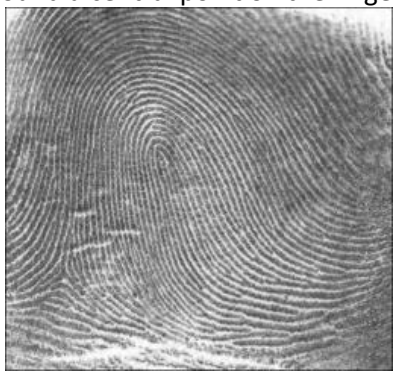
The fundamental of fingerprint

Patterns

The three basic patterns of fingerprint ridges are the arch, loop, and whorl.

- Arch: The ridges enter from one side of the finger, rise in the center forming an arc, and then exit the other side of the finger.
- Loop: The ridges enter from one side of a finger, form a curve, and then exit on that same side.
- Whorl: Ridges form circularly around a central point on the finger.

The arch pattern

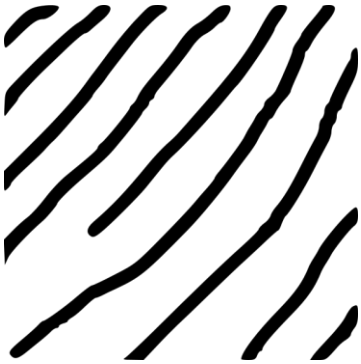


The loop pattern

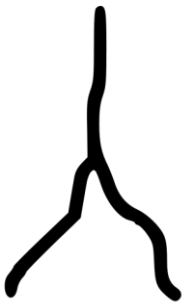
The whorl pattern

Minutia

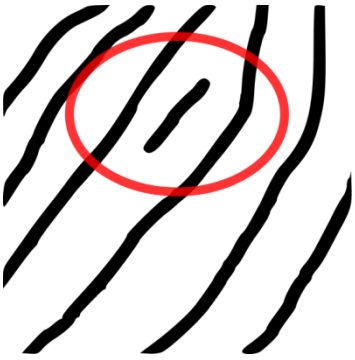
The major minutia features of fingerprint ridges are ridge ending, bifurcation, and short ridge (or dot). Minutiae and patterns are very important in the analysis of fingerprints since no two fingers have been shown to be identical.



Ridge ending

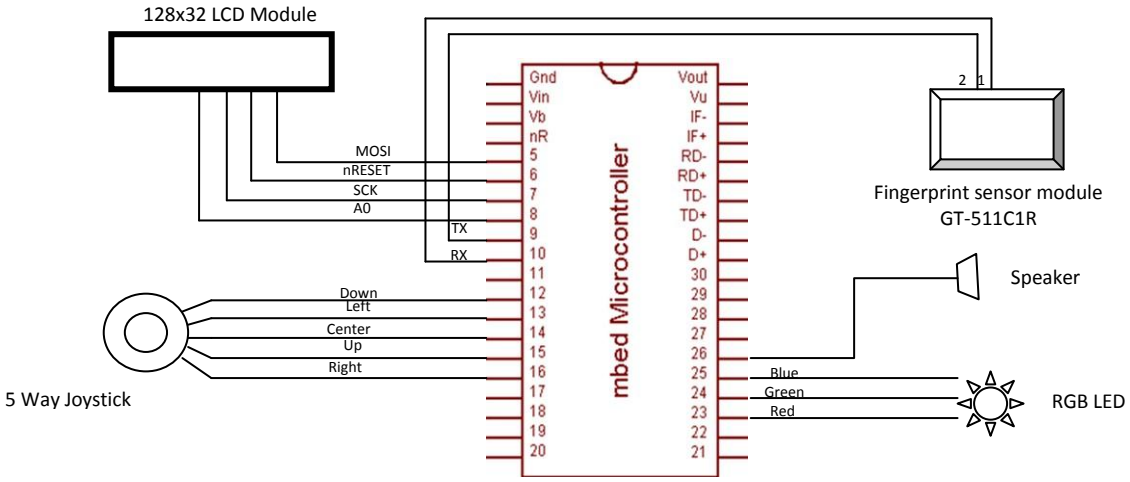


Bifurcation

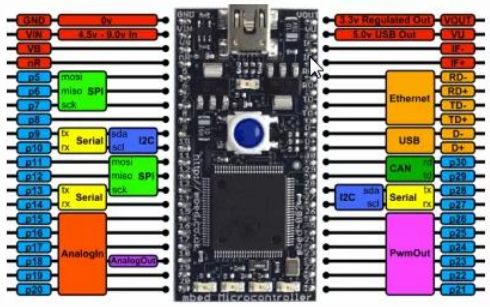


Short ridge (dot)

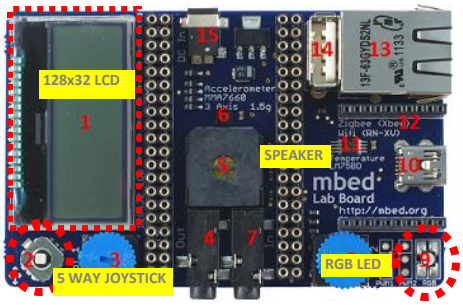
Hardware Diagram



mbed LPC1768



mbed application board



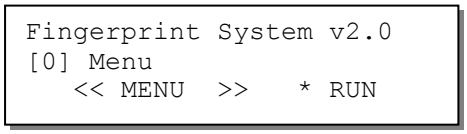
Fingerprint reader module GT-511C1R

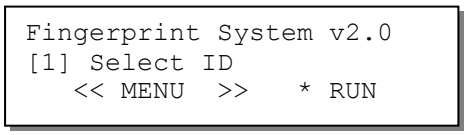
Specification of Fingerprint reader module GT-511C1R

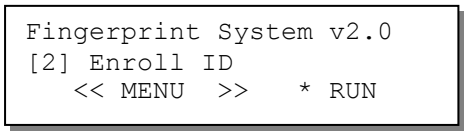
- High-accuracy and high-speed fingerprint identification technology
- Ultra-thin optical sensor
- 1:1 verification, 1:N identification

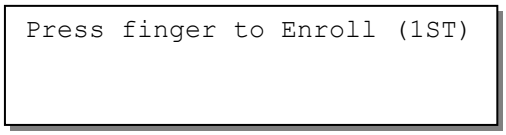
Item		Value
CPU		ARM Cortex M3 Core
Sensor		Optical sensor
Effective area of the sensor		12 x 12.5 (mm)
Image size		240 x 216 pixels
Resolution		450 dpi
The maximum number of fingerprints		20 fingerprints
Matching mode		1:1, 1:N
The size of template		504 bytes (template) + 2 bytes (checksum)
Communication interface		UART, default baud rate = 9600 bps after power on USB ver 1.1, Full speed
False Acceptance Rate (FAR)		< 0.001%
False Rejection Rate (FRR)		< 0.1%
Enrollment time		< 3 sec (3 fingerprints)
Identification time		< 1.5 sec (20 fingerprints)
Operating voltage		DC 3.3 ~ 6V
Operating current		< 130 mA
Operating environment	Temperature	-20˚C ~ +60˚C
	Humidity	20% ~ 80%
Storage environment	Temperature	-20˚C ~ +60˚C
	Humidity	10% ~ 80%

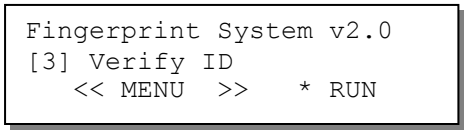
LCD Screen design

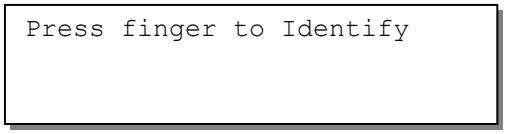


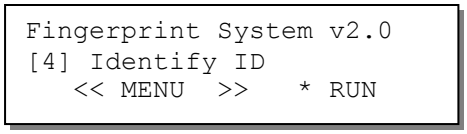


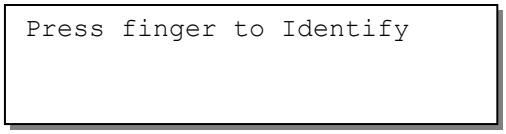


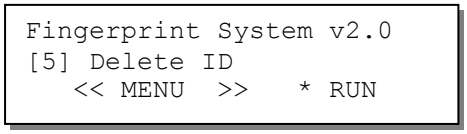












Source Code

finger.h

```
//
// UCSC ext.
// 2015 Embedded Firmware Essential
// Project
// Fingerprint Security Lock
// Jae-Yang Park
// jaeyangp@gmail.com
//

#ifndef FINGER_H
#define FINGER_H

Serial debug(USBTX,USBRX);
DigitalOut myled(LED1);

GT511C3 finger(p9,p10);
C12832 LCD(p5, p7, p6, p8, p11);

InterruptIn JOY_CENTER(p14);
InterruptIn JOY_UP(p15);
InterruptIn JOY_DOWN(p12);
InterruptIn JOY_LEFT(p13);
InterruptIn JOY_RIGHT(p16);

PwmOut SPEAKER(p26);
PwmOut LEDR(p23);
PwmOut LEDG(p24);
PwmOut LEDB(p25);

Timer debounce;

#endif
```

main.cpp

```
//
// UCSC ext.
// 2015 Embedded Firmware Essential
// Project
// Fingerprint Security Lock
// Jae-Yang Park
// jaeyangp@gmail.com
//
#include "mbed.h"
#include "GT511C3.hpp"
#include "C12832.h"
#include "finger.h"

int progress(int, char *);
void err_beep(void);
void good_beep(void);
void red_on(void);
void green_on(void);
void blue_on(void);
void rgb_off(void);
void init(void);
void disp_GT_info(void);
void disp_menu(void);
```

```
void chk_Enrolled(int);
void enroll_id(void);
void verify_id(void);
void identify_id(void);
void delete_id(void);
void select_id(void);

void joy_center_ISR(void);
void joy_right_ISR(void);
void joy_left_ISR(void);
void joy_up_ISR(void);
void joy_down_ISR(void);

int EnrollID = 0;
int current = 0;

#define MENU_SZ 6
#define MAX_ID 20

char *menu[] = {"[0] Menu              >>",
                "[1] Select ID      << >>",
                "[2] Enroll ID      << >>",
                "[3] Verify ID      << >>",
                "[4] Identify ID    << >>",
                "[5] Delete ID      <<   "};

void (*ptr_func[MENU_SZ])(void) = {disp_menu, select_id, enroll_id, identify_id,
identify_id, delete_id};

////////////////////

void err_beep(void)
{
    SPEAKER.period(1.0 / 2000.0);
    SPEAKER = 0.5;
    wait(0.1);
    SPEAKER = 0.0;
}

void good_beep(void)
{
    SPEAKER.period(1.0 / 2000.0);
    SPEAKER = 0.8;
    wait(0.1);
    SPEAKER = 0.0;
}

void red_on(void)
{
    LEDR.period(0.001);
    LEDR = 0.8;
    LEDG = 1.0;
    LEDB = 1.0;
}

void green_on(void)
{
    LEDG.period(0.001);
    LEDR = 1.0;
    LEDG = 0.8;
    LEDB = 1.0;
}
```

```
void blue_on(void)
{
    LEDB.period(0.001);
    LEDR = 1.0;
    LEDG = 1.0;
    LEDB = 0.8;
}

void rgb_off(void)
{
    LEDR = 1.0;
    LEDG = 1.0;
    LEDB = 1.0;
}

//////////////////

int progress(int status, char *msg)
{
    debug.printf("%s", msg);

    LCD.cls();
    LCD.locate(0, 1);
    LCD.printf("%s", msg);

    good_beep();

    if (status >= 1 && status <= 100) {
        blue_on();
    }

    return 0;
}

void disp_menu(void)
{
    LCD.cls();
    LCD.locate(0, 1);
    //          123456789012345678901234
    LCD.printf("Fingerprint System v2.0  \n");
    LCD.locate(0, 10);
    LCD.printf("%s\n", menu[current]);
    LCD.printf("   << MENU >>  * RUN     \n");
    finger.CmosLed(0);
    rgb_off();
}

void init(void)
{
    debug.format(8, Serial::None, 1);
    debug.baud(115200);
    debug.printf("Fingerprint reader module \"GT-511C3 / GT-511C31\" test program.\n");

    good_beep();
    rgb_off();

    debug.printf("Build: %s %s\n", __DATE__, __TIME__);
    debug.printf("Open\n");
}

void disp_GT_info(void)
```

```
{
    int i;

    debug.printf("FirmwareVersion = %lx\n", finger.FirmwareVersion);
    debug.printf("IsoAreaMaxSize = %ld\n", finger.IsoAreaMaxSize);
    debug.printf("DeviceSerialNumber = ");

    for (i = 0; i < sizeof(finger.DeviceSerialNumber); i++){
        debug.printf("%02X", finger.DeviceSerialNumber[i]);
    }

    debug.printf("\n");
}

void chk_Enrolled(int EnrollID)
{
    if (finger.CheckEnrolled(EnrollID) == 0) {
        debug.printf("EnrollID(%d) is already enrolled.\nDelete!\n", EnrollID);

        LCD.cls();
        LCD.printf("EnrollID(%d) is already enrolled.\nDelete!\n", EnrollID);

        if (finger.DeleteID(EnrollID) == 0){
            debug.printf("Delete OK!\n");
            LCD.printf("Delete OK!\n");
        }
    }
}

void select_id(void)
{
    LCD.locate(0, 10);
    LCD.printf("ID = %d                \n", EnrollID);
    LCD.printf("   << MENU >>  * CHANGE\n");
}

void enroll_id(void)
{
    finger.Open();

    chk_Enrolled(EnrollID);
    finger.Enroll(EnrollID, progress);
    finger.CmosLed(1);

    LCD.locate(0, 10);
    LCD.printf("                        ");
    LCD.locate(0, 10);
    LCD.printf("%s\n", menu[current]);
    LCD.printf("   << MENU >>  * RUN     \n");
}

void verify_id(void)
{
    LCD.locate(0, 10);
    LCD.printf("%s\n", menu[current]);
    //          123456789012345678901234
    LCD.printf("   << MENU >>  * RUN     \n");
}

void identify_id(void)
{
    int id;
```

```
    finger.CmosLed(1);

    debug.printf("Press finger for Identify\n");
    LCD.locate(0, 20);
    LCD.printf("Press finger for Identify\n");

    finger.WaitPress(1);

    if (finger.Capture(1) != 0) {
        debug.printf("Press finger for Identify\n");
        LCD.cls();
        LCD.locate(0, 20);
        //          12345678901234567890123456789
        LCD.printf("Press finger for Identify\n");
    }

    id = finger.Identify();

    debug.printf("ID = %d\n", id);
    debug.printf("Remove finger\n");

    if (id == -1) {
        err_beep();
        red_on();

        LCD.cls();
        LCD.locate(0, 1);
        LCD.printf("ID = %d, Not matched!\n", id);
    }
    else {
        good_beep();
        green_on();

        LCD.cls();
        LCD.locate(0, 1);
        LCD.printf("ID = %d, Matched!\n", id);
    }

    LCD.locate(0, 10);
    LCD.printf("%s\n", menu[current]);
    LCD.printf("   << MENU >>  * RUN     \n");

    finger.WaitPress(0);
    finger.CmosLed(0);
}

void delete_id(void)
{
    if (finger.DeleteID(EnrollID) == 0) {
        LCD.locate(0, 1);
        //          12345678901234567890123456789
        LCD.printf("%d deleted               \n", EnrollID);
    }
    else {
        LCD.locate(0, 1);
        LCD.printf("%d is not found          \n", EnrollID);
    }

    wait(0.5);

    LCD.locate(0, 10);
```

```
    LCD.printf("%s\n", menu[current]);
    LCD.printf("   << MENU >>  * RUN     \n");
}

void joy_center_ISR1(void)
{
    ptr_func[current]();
}

void joy_right_ISR(void)
{
    current++;
    if (current > MENU_SZ - 1)
                current = 0;

    LCD.locate(0, 10);
    LCD.printf("                         ");
    LCD.locate(0, 10);
    LCD.printf("%s\n", menu[current]);
    LCD.printf("   << MENU >>  * RUN     \n");

    debounce.reset();
}

void joy_left_ISR(void)
{
    current--;
    if (current < 0)
                current = MENU_SZ - 1;

    LCD.locate(0, 10);
    LCD.printf("                         ");
    LCD.locate(0, 10);
    LCD.printf("%s\n", menu[current]);
    LCD.printf("   << MENU >>  * RUN     \n");

    debounce.reset();
}

void joy_up_ISR(void)
{
    EnrollID++;
    if (EnrollID > (MAX_ID - 1))
         EnrollID = 0;

    debounce.reset();
}

void joy_down_ISR(void)
{
    EnrollID--;
    if (EnrollID < 0)
         EnrollID = MAX_ID - 1;

    debounce.reset();
}

void set_ISR(void)
{
    debounce.start();
    JOY_CENTER.rise(&joy_center_ISR1);
    JOY_RIGHT.rise(&joy_right_ISR);
```

```
    JOY_LEFT.rise(&joy_left_ISR);
    JOY_UP.rise(&joy_up_ISR);
    JOY_DOWN.rise(&joy_down_ISR);
}

int main()
{
    int sts = 0;

    init();
    disp_menu();
    set_ISR();

    sts = finger.Open();
    debug.printf("sts = %d\n", sts);

    if (sts == 0) disp_GT_info();
}
```

Menu selection

Due to limitation of switch, menu selection is implemented with switch interrupt. When joystick is pressed with direction or center, each interrupt service routine is executed. The main action of ISR (up, down, left, and right) is changing value for menu index, and the center button ISR launch the functions for menu respectively.

```
Joystick: Left 🡐 Right 🡒 Up 🡑 Down 🡓 Center ⊙
          Menu change  🡐 🡒 and ⊙
          Value change 🡑  🡓 and ⊙
```

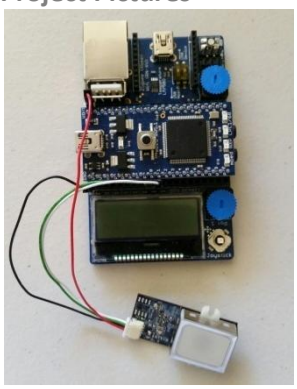
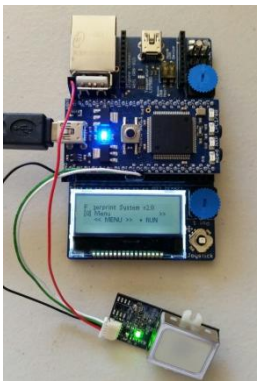
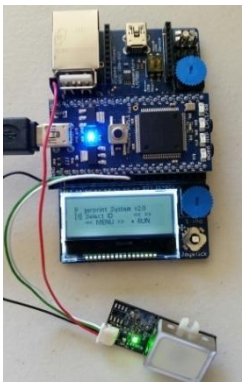
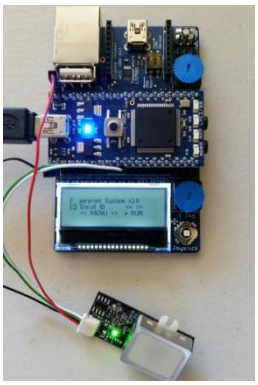
Further work

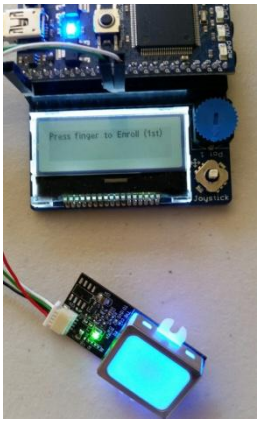
In this project, few functions were implemented, and some improvement is needed for already implemented functions, such as, Verify and Identify function.
At this time, these functions, “Get Image”, “Get Raw Image”, “Get / Set Template”, “Get / Set Database”, are not implemented. With these function, handling image file function (copy, transfer) from fingerprint module to mbed or external SD memory, or PC file system is one of want-to-be functions.
Moreover, it can be applicable to security system like entrance / computer locking system. When fingerprint is matched with previously enrolled fingerprint in the database, then unlock the system.

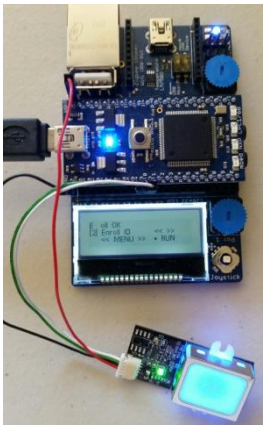
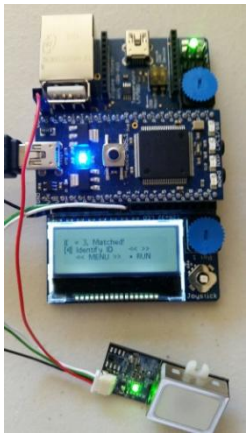
Conclusion

Through this project and course, Embedded Firmware Essentials, I can learn the trend of small embedded firmware and hardware system and its potential capabilities of IoT. CMSIS-DAP (The Cortex Microcontroller Software Interface Standard - Cortex Debug Access Port), OCD (On-Chip Debug), pyOCD, mbed platform debugging, these are things what I learned from the class newly.

Project Pictures



References

1. ARM mbed Developer Site, https://developer.mbed.org/
2. mbed Application Board, https://developer.mbed.org/components/mbed-Application-Board/
3. Fingerprint recognition, http://en.wikipedia.org/wiki/Fingerprint_recognition
4. GT-511C1R datasheet, http://www.adh-tech.com.tw/files/GT-511C1R_datasheet_V1%205_20140312[1].pdf
5. Toshihisa T, GT511C3 library, https://developer.mbed.org/users/tosihisa/code/GT511C3/
6. Toshihisa T, GT511C3test wiki, https://developer.mbed.org/users/tosihisa/code/GT511C3test/wiki/Homepage