2016년 2학기
컴퓨터하드웨어 실험 보고서
7조 최종보고서

한성희 이호욱 한경수

Table Of Contents

프로젝트 개요

각종 센서를 이용한 무인 택배 보관함 개발
부재 시, 택배 보관을 해결하기 위한 스마트폰 연동의 무인 택배 보관함

프로젝트 시나리오

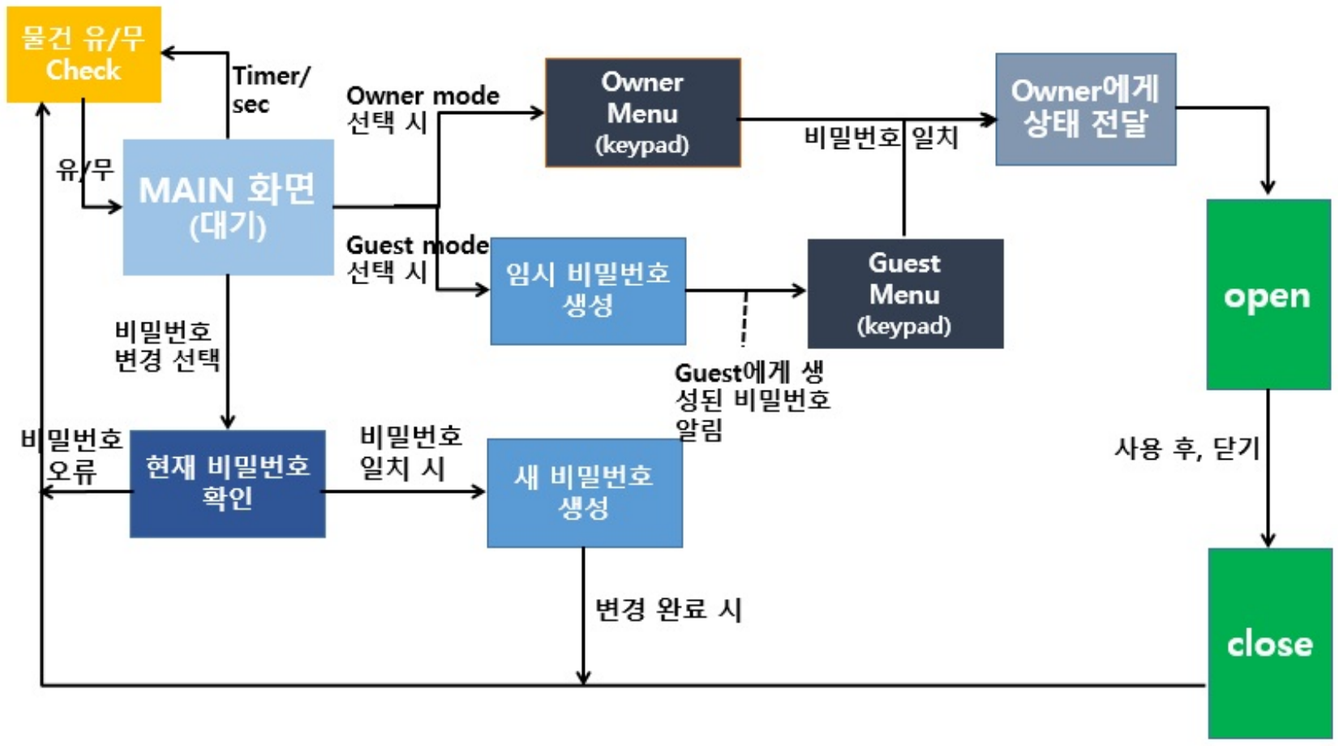
1. 사용자는 택배 함 사용을 위해 원하는 비밀번호를 설정 해준다.
2. 사용자가 비밀번호 변경을 하기 위해서는 메인 화면의 "CHANGE PASSWORD" 버튼을 이용하여 변경한다.
3. 택배함 이용
4. 사용자 및 택배기사가 택배함을 열게 되면 사용자의 스마트폰으로 "open door" 라는 문열림 알림이 온다.
5. 사용자가 물건을 빼거나 택배기사가 물건을 넣게되면 적외선 센서가 물건을 인식하여 사용자의 스마트폰에 택배가 도착했다는 알림이 오고 LCD 화면의 택배함 상태가 you -> moo 또는 moo -> yoo 로 변경된다.

- 사용자의 경우
  1) "OWN" 버튼을 이용하여 사용자 사용 모드로 들어간다.
  2) 설정해둔 비밀번호를 입력하면 택배함 문이 열린다.
  3) 이용이 끝나면 "close door" 버튼을 눌려 택배함 문을 닫는다.

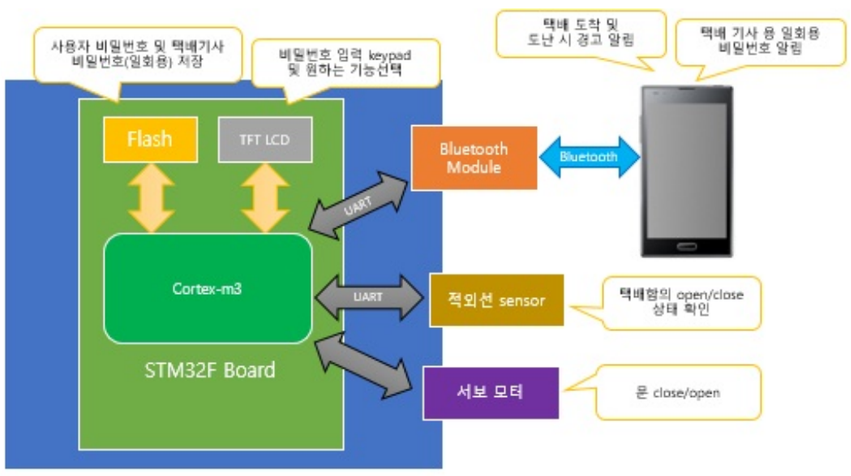
- 택배기사의 경우
  1) "GUEST" 버튼을 이용하여 택배기사 사용 모드로 들어간다.
  2) 택배기사의 스마트폰으로 일회용 비밀번호가 전송된다.
  3) 택배기사는 제공받은 비밀번호를 이용하여 택배함 문을 연다.
  4) 이용이 끝나면 "close door" 버튼을 눌려 택배함 문을 닫는다.

프로젝트 구조

흐름도 (Flow Chart)



블록 다이어그램 (Block diagram)

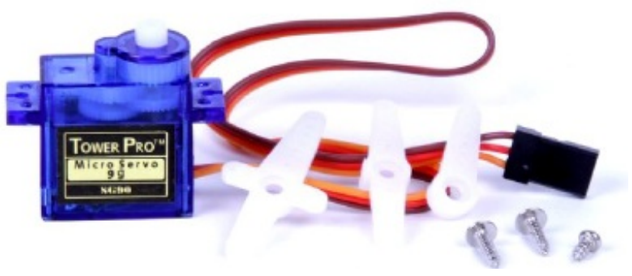


사용 부품

적외선 센서 (E18-D80NK)

택배함 내부의 물품 존재 여부를 감지해준다.
센서자체의 가변저항을 조절하여 감지 거리를 택배함의 내부에 맞게 조정한다.

서보모터 (SG90)



택배함의 문을 여닫는데 사용된다.
LCD 입력에 따라 상태변화에 맞게 문을 열고 닫아준다.

블루투스 모듈 (FB755AC)

기기의 상태를 스마트폰으로 전송하기 위해서 쓰인다. 택배 기사가 문을 열고 닫고 하는 정보나, 일회용 비밀번호에 대한 정보를 전송한다.

Application 설명 및 동작사진

위와 같이 택배함에 보드를 부착하여 LCD 를 조작할 수 있고 적외선 센서를 아래방향을 향하도록 위쪽에 부착하여 택배물을 감지할 수 있다.

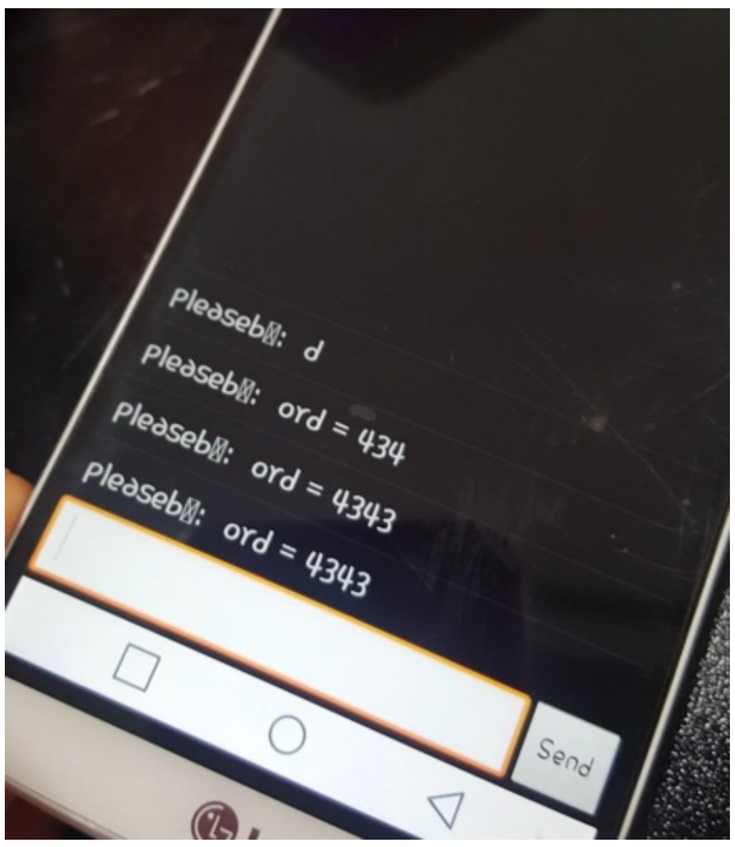
물건 감지 시 적외선 센서에 빨간 불이 드러온다.

시스템 메인 화면이다.

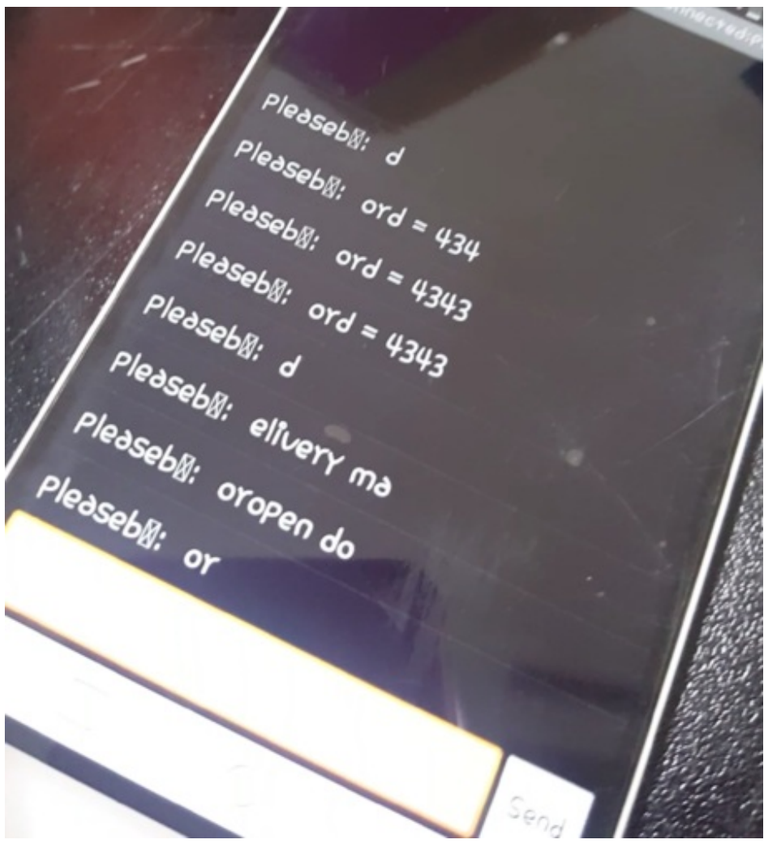
사용자가 "CHANGE PASSWORD" 를 이용하여 비밀번호 변경 시 LCD의 왼쪽 우측과 같이 비밀번호가 변경되게 된다. 변경 확인을 위해 hint 라고 해서 비밀번호를 띄어 놓았다.

keypad 화면에서 비밀번호가 틀리게 되면 메인 화면으로 돌아가게 되고

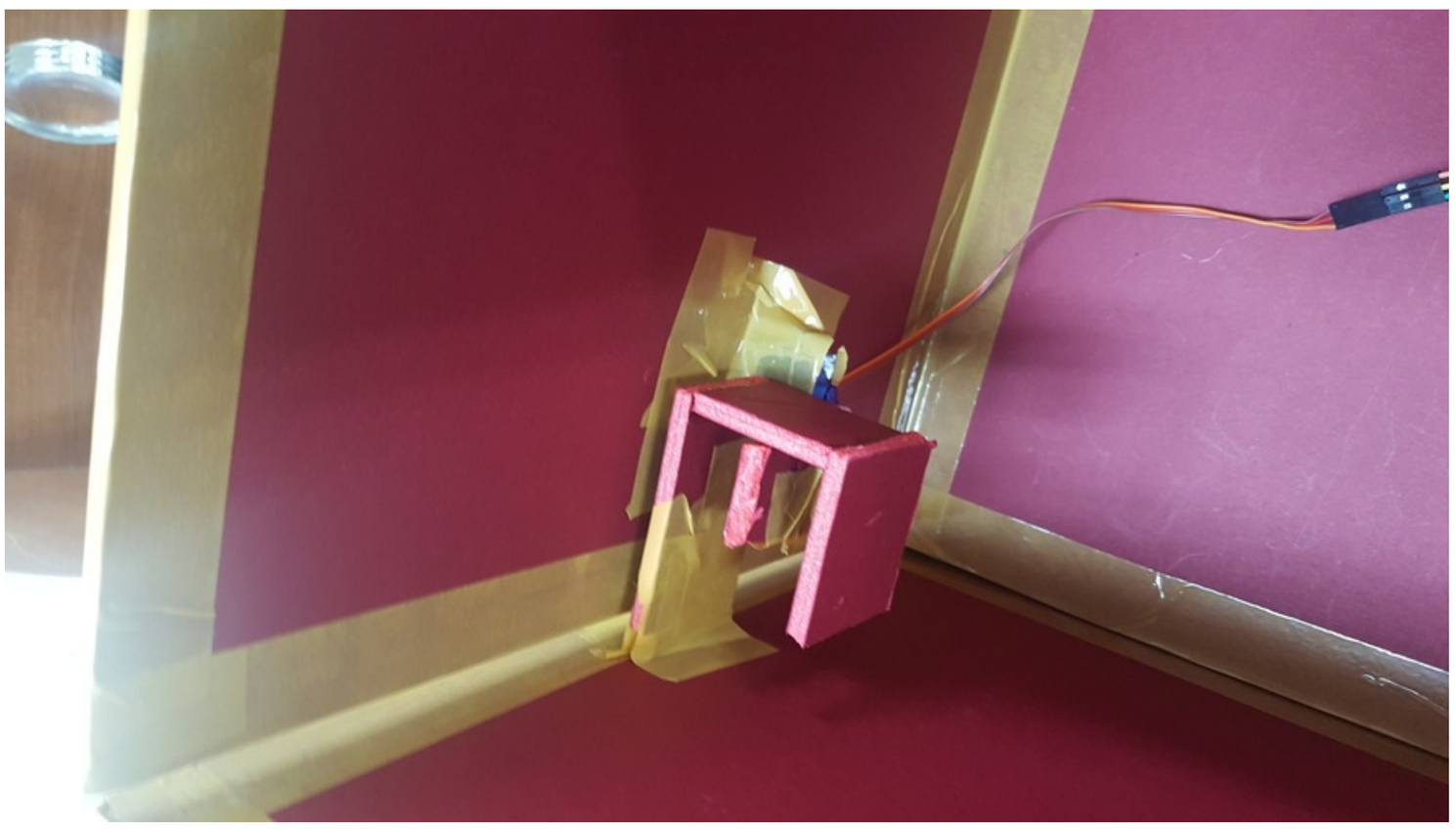
비밀번호 일치 시, 다음과 같은 창이 뜨고 택배함 이용이 마치면 화면 아래의 "close door" 버튼을 눌려 이용을 종료한다. 종료시 메인화면으로 돌아간다.



택배기사 사용 모드시 택배기사의 스마트폰으로 일회용 비밀번호가 전송된 모습이다.



택배기사 뿐만 아니라 사용자가 문을 열게 되면 사용자의 스마트폰으로 문이 열렸다는 알림이 오게 된다.



택배함이 잠겼을 때의 모습으로 모터에 달린 네모 판이 고리에 걸려있어 문이 열리지 않는다.

택배함이 열렸을 때 모습으로 위와는 다르게 모터에 달린 네모 판이 모터의 작동으로 밑으로 내려가 있어 문을 열 수 있다.

구현 시 어려웠던 점

적외선 센서를 쓰는데 거리에 따라 어떻게 변하는지 몰라서 직접 저항을 달까 고민을 많이 했다. 잘 찾아보니 내부 저항이 있어서 손쉽게 그것이 제어가 되는것을 알았다.

모터를 다룰 때 pwm 이라는 개념을 몰라서 그냥 값만 전달하면 모터가 돌아갈 줄 알았다. 제조사 레퍼런스 찾아보는 법을 알게됐다. 이는 stm32 의 내장된 PWM sample 로 비교적 손쉽게 해결이 됐던 부분이다.

하지만 특히 블루투스모듈을 다루는 문제가 가장 큰 어려웠던 점인데, 먼저 블루투스가 현재 어떤 상태인지 알아보는게 급선무였다. status led

가 반짝반짝 거리거나 불이 안들어오거나 켜져있거나 이런 상태들이 있는데 이 정보를 토대로 문제를 해결해야 했다.

처음엔 AT+BTSCAN 명령어로 블루투스를 대기모드로 만들어놓으면 해결이 됐는데, 전원을 또 껐다 키면 안됐다. 삽질 끝에 Connection Mode 를 mode3 으로 고치면 된다는것을 알고 해결을 했다. 그러자 수신(RX) 은 되는데 TX) 전송은 안되는 문제가 생겼다.

알고보니 안드로이드 앱 구현체에 문제가 있었다. 글로 푸니까 큰 어려움이 아닌 것 같은데 여기에 2일 정도 시간을 썼다.

또한 문을 만드는 점에서 어떻게 잠금장치 및 모터와 문을 연동시킬것 인지가 가장 어려웠다. 이 문제는 컴퓨터과학의 문제를 떠난 문제 같아서 해결하기가 많이 어려웠다. 모터와 문이 헛돌고 접착을 하기도 어려워서 제어가 안됐다.

고민 끝에 잠금 장치를 제어하는 것으로 창의적으로 해결했다.

한계

- 보안 면에서 문 잠금 장치 외에는 아무런 조치가 없어서 문을 뜯어가면 어쩔 수 없다.
- 다양한 이미지 파일로 gui 를 구성하려 했지만 scatter file 을 늘리더라도 메모리쪽에서 fault 가 나서 진행이 어려웠다.
- 되도록이면 LCD 내용으로 표시하기보단 외부 출력장치로 해결해야 했는데 그러지 못한 점

부록 - 최종 코드

main.cpp

```
/* vi: set sw=4 ts=4 expandtab: */
#include <misc.h>
#include <stm32f10x.h>
#include <stm32f10x_exti.h>
#include <stm32f10x_gpio.h>
#include <stm32f10x_rcc.h>
#include <stm32f10x_usart.h>
#include <stm32f10x_adc.h>
#include <lcd.h>
#include <Touch.h>
#include <stdio.h>
#include <stdlib.h>
#include "config.h"
#include "interrupt.h"

int keypadMenu(int prevState, const char* msg);
/*
flash load C:\Users\USER\Desktopchoterm\Debugchoterm.axf
flash load C:\Users\USER\Desktopchoterm\Debug\flashclear.axf
*/
const uint16_t colorArray[] = {109, 65535,1, 25621,1, 19316,1, 17170,1, 13010,1, 10897,1, 4
591,10, 2478,1, 8817,1, 12977,1, 15090,1, 17202,1, 23509,214, 65535,1, 10864,1, 8752,1, 667
1,1, 4591,24, 2479,1, 4591,1, 6671,1, 8752,1, 8816,203, 65535,1, 21396,1, 10929,7, 2479,1,
10897,1, 17170,1, 19315,18, 65535,1, 19315,1, 17170,1, 10897,7, 2479,1, 10897,1, 21396,194,
65535,1, 17268,5, 2479,1, 4591,1, 6672,1, 8784,1, 23541,30, 65535,1, 23541,1, 8784,1, 6672,
1, 4591,5, 2479,1, 17268,187, 65535,1, 15122};

TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
uint16_t CCR1_Val[2] = {1500, 100};
uint16_t CCR2_Val[2] = {1500, 100};
uint16_t CCR3_Val[2] = {1500, 100};
uint16_t CCR4_Val[2] = {1500, 100};
uint16_t PrescalerValue = 0;

void send_phone(char buf[]) {
    char *s = buf;
    while (*s) {
        while (USART_GetFlagStatus(USART2, USART_FLAG_TXE) == RESET)
            ;
        USART_SendData(USART2, *s++);
    }
}

void openDoor(){
    TIM_OCInitTypeDef  TIM_OCInitStructure;
    TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
    TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
    TIM_OCInitStructure.TIM_Pulse = CCR1_Val[0];
    TIM_OC4Init(TIM3, &TIM_OCInitStructure);

    TIM_OC2Init(TIM3, &TIM_OCInitStructure);

```

```
}


void closeDoor(){
    TIM_OCInitTypeDef  TIM_OCInitStructure;
    TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
    TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
    TIM_OCInitStructure.TIM_Pulse = CCR1_Val[1];
    TIM_OC4Init(TIM3, &TIM_OCInitStructure);
    TIM_OC2Init(TIM3, &TIM_OCInitStructure);
}
void setBoards(){
    SystemInit();

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);      // interrupt
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE,ENABLE);      // RCC GPIO E
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC,ENABLE);      // RCC GPIO C
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD,ENABLE);      // RCC GPIO D
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);      // ADC1
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);        // DMA1

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1 | RCC_APB2Periph_AFIO |
            RCC_APB2Periph_GPIOA, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);

    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
    ////////////////////////////////////////////////////////////////////////////////
//////////////


    ADC_InitTypeDef ADC_InitStructure;
    ADC_DeInit(ADC1);
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
    ADC_InitStructure.ADC_ScanConvMode = ENABLE;
    ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
    ADC_InitStructure.ADC_NbrOfChannel = 2;
    ADC_RegularChannelConfig(ADC1, ADC_Channel_11, 1,
            ADC_SampleTime_55Cycles5);
    ADC_RegularChannelConfig(ADC1, ADC_Channel_12, 2,
            ADC_SampleTime_55Cycles5);
    ADC_Init(ADC1, &ADC_InitStructure);

    ADC_DMACmd(ADC1, ENABLE);
    ADC_Cmd(ADC1, ENABLE);

    ADC_ResetCalibration(ADC1);
    while (ADC_GetResetCalibrationStatus(ADC1))
        ;
    ADC_StartCalibration(ADC1);
    while (ADC_GetCalibrationStatus(ADC1))
        ;
    ADC_SoftwareStartConvCmd(ADC1, ENABLE);


    DMA_InitTypeDef DMA_InitStructure;
    DMA_DeInit(DMA1_Channel1);
    DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t) &ADC1->DR;
    DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t) ADC_Value;
    DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
    DMA_InitStructure.DMA_BufferSize = 2;
    DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
    DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
    DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Word;
    DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Word;
```

```
    DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
    DMA_InitStructure.DMA_Priority = DMA_Priority_High;
    DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
    DMA_Init(DMA1_Channel1, &DMA_InitStructure);
    DMA_Cmd(DMA1_Channel1, ENABLE);


    GPIO_InitTypeDef LED;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);     // RCC GPIO D
    LED.GPIO_Mode = GPIO_Mode_Out_PP;
    LED.GPIO_Pin = GPIO_Pin_2;
    LED.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOD, &LED);


    // 타이머 설정.
    NVIC_InitTypeDef NVIC_InitStructure; // for interreupt
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure; // timerbase...
    /* TIM2 Clock Enable */
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
    /* Enable TIM2 Global Interrupt */
    NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
    /* TIM2 Initialize */
    TIM_TimeBaseStructure.TIM_Period = 600;
    TIM_TimeBaseStructure.TIM_Prescaler = 60000;
    //계산방법 : 1/72mhz * 1200 * 60000
    TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
    TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
    /* TIM2 Enale */
    TIM_ARRPreloadConfig(TIM2, ENABLE);
    TIM_Cmd(TIM2, ENABLE);
    TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE); // interrupt enable

    ////////////////////////////////////////
    // 이하 모다 소스

    /* TIM3 clock enable */
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);

    /* GPIOA and GPIOB clock enable */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB |
            RCC_APB2Periph_GPIOC | RCC_APB2Periph_AFIO, ENABLE);

    /*GPIOB Configuration: TIM3 channel1, 2, 3 and 4 */
    GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_6 | GPIO_Pin_7 | GPIO_Pin_8 | GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
    GPIO_PinRemapConfig(GPIO_FullRemap_TIM3, ENABLE);

    TIM_OC4PreloadConfig(TIM3, TIM_OCPreload_Enable);
    TIM_OC2PreloadConfig(TIM3, TIM_OCPreload_Enable);
    TIM_ARRPreloadConfig(TIM3, ENABLE);
    TIM_Cmd(TIM3, ENABLE);
    PrescalerValue = (uint16_t) (SystemCoreClock / 1000000) - 1;
    TIM_TimeBaseStructure.TIM_Period = 20000-1;
    TIM_TimeBaseStructure.TIM_Prescaler = PrescalerValue;
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;

    TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);


    // bluetooth setting

    // pc 와의  usart 통신
    USART_InitTypeDef usart1_init_struct;
    GPIO_InitTypeDef gpioa_init_struct;

```

```
    gpioa_init_struct.GPIO_Pin = GPIO_Pin_9;
    gpioa_init_struct.GPIO_Speed = GPIO_Speed_50MHz;
    gpioa_init_struct.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOA, &gpioa_init_struct);
    gpioa_init_struct.GPIO_Pin = GPIO_Pin_10;
    gpioa_init_struct.GPIO_Speed = GPIO_Speed_50MHz;
    gpioa_init_struct.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &gpioa_init_struct);

    usart1_init_struct.USART_BaudRate = 57600;
    usart1_init_struct.USART_WordLength = USART_WordLength_8b;
    usart1_init_struct.USART_StopBits = USART_StopBits_1;
    usart1_init_struct.USART_Parity = USART_Parity_No;
    usart1_init_struct.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
    usart1_init_struct.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART_Init(USART1, &usart1_init_struct);
    USART_Cmd(USART1, ENABLE);
    USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);

    NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x01;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x01;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);

    USART_InitTypeDef usart2_init_struct;

    // 보드 - 맛폰간의 블루투스 통신
    // tx, rx 설정
    gpioa_init_struct.GPIO_Pin = GPIO_Pin_2;
    gpioa_init_struct.GPIO_Speed = GPIO_Speed_50MHz;
    gpioa_init_struct.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOA, &gpioa_init_struct);

    gpioa_init_struct.GPIO_Pin = GPIO_Pin_3;
    gpioa_init_struct.GPIO_Speed = GPIO_Speed_50MHz;
    gpioa_init_struct.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &gpioa_init_struct);

    usart2_init_struct.USART_BaudRate = 9600;
    usart2_init_struct.USART_WordLength = USART_WordLength_8b;
    usart2_init_struct.USART_StopBits = USART_StopBits_1;
    usart2_init_struct.USART_Parity = USART_Parity_No;
    usart2_init_struct.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
    usart2_init_struct.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART_Init(USART2, &usart2_init_struct);
    USART_Cmd(USART2, ENABLE);
    USART_ITConfig(USART2, USART_IT_RXNE, ENABLE);

    NVIC_InitStructure.NVIC_IRQChannel = USART2_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x01;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x01;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);

}



/*
   flash load C:\Users\USER\Desktop\team7choterm\Debugchoterm.axf
   flash load C:\Users\USER\Desktop\team7choterm\Debug\flashclear.axf
   */
void delay(int i){
    int j;
    for(j=0; j<=i * 100000; j++);
}




/*
   flash load C:\Users\USER\Desktop\team7choterm\Debugchoterm.axf
   flash load C:\Users\USER\Desktop\team7choterm\Debug\flashclear.axf
```

```
   */

int inBox(int x, int y, int sx, int sy, int ex, int ey){
    return sx <= x && x <= ex &&
        sy <= y && y <= ey;
}

int userMenu(int prevState){
    LCD_Clear(WHITE);
    screenState = OWN;

    LCD_DrawRectangle(5, 35, 65, 100);
    LCD_DrawRectangle(70, 35, 150, 100);
    LCD_DrawRectangle(155, 35, 235, 100);

    LCD_ShowString(5, 225, "close door", RED, WHITE);
    LCD_DrawRectangle(5, 225, 235, 280);
    // 물건이 있는지 표시!?
    // 닫기 버튼 누르면 문닫기.

    uint16_t pos_x,pos_y;
    uint16_t pix_x,pix_y;
    Touch_GetXY(&pos_x, &pos_y, 1); // 터치
    Convert_Pos(pos_x, pos_y, &pix_x, &pix_y);
    if (inBox(pix_x, pix_y, 5, 225, 235, 280)){
        // 문닫기.
        // 모터 돌림.
        closeDoor();
    }

    screenState = prevState;
    return 0;
}

int guestMenu(int prevState){
    LCD_Clear(WHITE);
    screenState = GUEST;


    send_phone("delivery man open door");


    LCD_DrawRectangle(5, 35, 65, 100);
    LCD_DrawRectangle(70, 35, 150, 100);
    LCD_DrawRectangle(155, 35, 235, 100);

    LCD_ShowString(5, 225, "close door", RED, WHITE);
    LCD_DrawRectangle(5, 225, 235, 280);
    // 물건이 있는지 표시!?
    // 닫기 버튼 누르면 문닫기.

    uint16_t pos_x,pos_y;
    uint16_t pix_x,pix_y;
    Touch_GetXY(&pos_x, &pos_y, 1); // 터치
    Convert_Pos(pos_x, pos_y, &pix_x, &pix_y);
    if (inBox(pix_x, pix_y, 5, 225, 235, 280)){
        // 문닫기.
        // 모터 돌림.
        closeDoor();
    }

    screenState = prevState;
    return 0;
}
void showLogo(){
    uint32_t index=0;
    LCD_SetCursor(0x00,0x0000);
    LCD_WriteRAM_Prepare();
    for(int i=0; i<sizeof(colorArray) / sizeof(colorArray[0]); i+=2){
        for(int j=0; j<colorArray[i]; j++){
            LCD_WR_DATA(colorArray[i+1]);
        }
    }
    /*for(index=0;index<76800;index++)*/
    /*{*/
```

```
354             /*LCD_WR_DATA(Color);*/
355         /*}*/
356     }
357     int mainMenu() {
358         char Magnetic[30];
359         uint16_t pos_x,pos_y;
360         uint16_t pix_x,pix_y;
361         int ret;
362         while(1){
363             /*showLogo();*/
364             LCD_Clear(WHITE);
365             delay(3);
366             showLogo();
367             char hint[30];
368             sprintf(hint, "hint : %d", ownerPasswd);
369             sprintf(Magnetic, "%d", MagneticValue);
370             LCD_ShowString(100, 10, Magnetic, BLACK, WHITE);
371             LCD_ShowString(0, 10, hint, RED, WHITE);
372             LCD_ShowString(50, 150, "OWN", BLACK, WHITE);
373             LCD_ShowString(120 + 50, 150, "GUEST", BLACK, WHITE);
374             LCD_ShowString(50, 230, "CHANGE PASSWORD", RED, WHITE);
375 
376             LCD_DrawRectangle(5, 35, 65, 100);
377             LCD_DrawRectangle(70, 35, 150, 100);
378             LCD_DrawRectangle(155, 35, 235, 100);
379 
380             LCD_DrawRectangle(5, 120, 115, 220);
381 
382             LCD_DrawRectangle(120, 120, 235, 220);
383 
384             LCD_DrawRectangle(5, 225, 235, 280);
385             LCD_ShowString(10, 300, "TEST", RED, WHITE);
386 
387             ////////////////////////
388 
389             // ADC_Value[0] >= 0xF00 이면 물건 감지.
390 
391             /*LCD_ShowNum(100, 100, ADC_Value[0], 10, BLACK, WHITE);*/
392             /*LCD_ShowNum(100, 150, ADC_Value[1], 4, BLACK, WHITE);*/
393             ////////////////////////
394 
395 
396             ////////////////////////
397             Touch_GetXY(&pos_x, &pos_y, 1); // 터치
398             Convert_Pos(pos_x, pos_y, &pix_x, &pix_y);
399             if (inBox(pix_x, pix_y, 5, 120, 115, 220)){   // 주인
400                 int retNum = keypadMenu(MAIN, "input password");
401                 if(retNum == ownerPasswd){
402                     // 문 열고,
403                     openDoor();
404                     userMenu(MAIN);
405                 }
406                 else{
407                     continue;
408                 }
409             }
410             if (inBox(pix_x, pix_y, 120, 120, 235, 220)) {   // 택배
411                 char buf[50];
412                 sprintf(buf, "delivery man's password = %d", guestPasswd);
413                 send_phone(buf);
414                 int retNum = keypadMenu(MAIN, "input pw to insert item");
415                 if(retNum == guestPasswd){
416                     // 서보모터를 돌린다.
417                     openDoor();
418                     // && 택배기사 메뉴 띄움.
419                     guestPasswd = rand() % 10000;
420                     guestMenu(MAIN);
421                 }
422                 else{
423                     continue;
424                 }
425             }
426 
427             if (inBox(pix_x, pix_y, 5, 225, 235, 280)) {   // 사용자 비밀번호 변경.
428                 int retNum = keypadMenu(MAIN, "input original password");
```

```
            if(retNum == ownerPasswd){
                int changePass = keypadMenu(MAIN, "input new password");
                ownerPasswd = changePass;
            }
            // 비밀번호 변경 루틴.
        }
    }
    return 0;
}

int keypadMenu(int prevState, const char* msg){

    screenState = KEYPAD;
    uint16_t pos_x,pos_y;
    uint16_t pix_x,pix_y;
    int x, y;
    int h, w;
    int r, c;
    r = 4;
    c = 3;
    h = 235;
    w = 235;
    int marginX = 0;
    int dy = h / r;
    int dx = w / c;
    int nInput = 4;

    LCD_Clear(WHITE);

    // 가로 줄 그리기
    for(y=0; y<=r; y++){
        int startPointX = 0;
        int startPointY = dy * y;
        int endPointY = dy * y;
        int endPointX = w;

        LCD_DrawLine(startPointX, startPointY, endPointX, endPointY);
    }

    // 세로줄 그리기
    for(x = 0; x<=c; x++){
        int startPointX = dx * x;
        int startPointY = 0;
        int endPointX = dx * x;
        int endPointY = h;

        LCD_DrawLine(startPointX, startPointY, endPointX, endPointY);
    }

    // 숫자 넣기.
    int printNum = 1;
    int pointsX[12];
    int pointsY[12];
    for(y = 0; y<r; y++){
        for(x = 0; x<c; x++, printNum++){
            int leftTopX = dx * x;
            int leftTopY = dy * y;
            LCD_ShowNum(leftTopX + dx / 2, leftTopY + dy / 2, printNum - 1, 2, BLACK, WHITE
);
            pointsX[printNum-1] = leftTopX + dx / 2;
            pointsY[printNum-1] = leftTopY + dy / 2;
        }
    }

    LCD_ShowString(10, 300, (u8*)msg, RED, WHITE);
    int retValue = 0;
    for(int currentInput=0; currentInput < 4; ){
        Touch_GetXY(&pos_x, &pos_y, 1);
        for(volatile int i=0; i<100000; i++);
        Convert_Pos(pos_x, pos_y, &pix_x, &pix_y);

        for (int i = 0; i < 12; i++) {
            if (inBox(pix_x, pix_y, pointsX[i] - dx / 2, pointsY[i] - dy / 2,
                      pointsX[i] + dx / 2, pointsY[i] + dy / 2)) {
                int selectNum = i + 1;
```

```
                retValue *= 10;
                retValue += selectNum;

                currentInput++;
                break;
            }
        }
    }
    retValue -= 1111;

    screenState = prevState;
    LCD_Clear(WHITE);
    return retValue;
}

int main() {
    setBoards();
    LCD_Init();
    Touch_Configuration();
    Touch_Adjust();
    GPIOD->CRL = (GPIO_CRL_MODE2_0 | GPIO_CRL_MODE3_0 | GPIO_CRL_MODE4_0 | GPIO_CRL_MODE7_0
);
    delay(3);
    /*LCD_Clear(WHITE);*/
    mainMenu();
}
```

config.c

```
#include "config.h"

 const int MAIN = 0;
 const int OWN = 1;
 const int GUEST = 2;
 const int CHANGE = 3;
 const int KEYPAD = 4;
int ownerPasswd = 4343;
int guestPasswd = 4343;
int screenState = MAIN;
uint32_t ADC_Value[4];
uint32_t MagneticValue;
```

config.h

```
#ifndef __CONFIG__H_
#define __CONFIG__H_
#include <misc.h>
extern int ownerPasswd;
extern int guestPasswd;

extern const int MAIN ;
extern const int OWN ;
extern const int GUEST ;
extern const int CHANGE ;
extern const int KEYPAD ;
extern int screenState;
extern uint32_t ADC_Value[4];
extern uint32_t MagneticValue;


#endif

```

interrupt.c

```
#include <misc.h>
#include <stm32f10x.h>
#include <stm32f10x_exti.h>
#include <stm32f10x_gpio.h>
```

```
#include <stm32f10x_rcc.h>
#include <stm32f10x_usart.h>
#include <stm32f10x_adc.h>
#include <lcd.h>

#include "config.h"

void TIM2_IRQHandler(void) {

  if(screenState == MAIN){
    uint32_t adc_value = ADC_Value[0];

    if(adc_value < 350){
      LCD_ShowString(100, 50, "you", BLACK, WHITE);
      LCD_ShowString(200, 50, "moo", WHITE, WHITE);
      // 뱀젙 醫뚰묤����異쒕젰.
    }
    else{
        LCD_ShowString(100, 50, "you", WHITE, WHITE);
      LCD_ShowString(200, 50, "moo", BLACK, WHITE);
      // 뱀젙 醫뚰묤��뗫異쒕젰.
    }
  }

  TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
    //Clears the TIMx's interrupt pending bits.
}

void USART1_IRQHandler(void) {
    unsigned char d;
    while (USART_GetFlagStatus(USART1, USART_FLAG_TXE) == RESET)
        ;

    d = (unsigned char) USART_ReceiveData(USART1);
    USART_SendData(USART2, d);
    USART_ClearITPendingBit(USART1, USART_IT_RXNE);
}

void USART2_IRQHandler(void) {
    unsigned char d;
    while (USART_GetFlagStatus(USART2, USART_FLAG_TXE) == RESET)
        ;

    d = (unsigned char) USART_ReceiveData(USART2);
    USART_SendData(USART1, d);
    USART_ClearITPendingBit(USART2, USART_IT_RXNE);
}
```

interrupt.h

```
#ifndef __INT__H_
#define __INT__H_

void TIM2_IRQHandler(void);

void USART1_IRQHandler(void);
void USART2_IRQHandler(void);
#endif

```

scatter file

```
LR_IROM1 0x08000000 0x000F0000  {    ; load region size_region
  ER_IROM1 0x08000000 0x000F0000  {  ; load address = execution address
   *.o(RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
  }
  RW_IRAM1 0x20000000 0x0008000  {  ; RW data
   .ANY (+RW +ZI)
  }
}
```

file list

```
▼ Libraries/
  ▶ CMSIS/
  ▼ STM32F10x_StdPeriph_Driver/
    ▼ inc/
        font.h
        lcd.h
        misc.h
        stm32f10x_adc.h
        stm32f10x_bkp.h
        stm32f10x_can.h
        stm32f10x_cec.h
        stm32f10x_crc.h
        stm32f10x_dac.h
        stm32f10x_dbgmcu.h
        stm32f10x_dma.h
        stm32f10x_exti.h
        stm32f10x_flash.h
        stm32f10x_fsmc.h
        stm32f10x_gpio.h
        stm32f10x_i2c.h
        stm32f10x_iwdg.h
        stm32f10x_pwr.h
        stm32f10x_rcc.h
        stm32f10x_rtc.h
        stm32f10x_sdio.h
        stm32f10x_spi.h
        stm32f10x_tim.h
        stm32f10x_usart.h
        stm32f10x_wwdg.h
        Touch.h
    ▼ src/
        lcd.c
        misc.c
        stm32f10x_adc.c
        stm32f10x_bkp.c
        stm32f10x_can.c
        stm32f10x_cec.c
        stm32f10x_crc.c
        stm32f10x_dac.c
        stm32f10x_dbgmcu.c
        stm32f10x_dma.c
        stm32f10x_exti.c
        stm32f10x_flash.c
        stm32f10x_fsmc.c
        stm32f10x_gpio.c
        stm32f10x_i2c.c
        stm32f10x_iwdg.c
        stm32f10x_pwr.c
        stm32f10x_rcc.c
        stm32f10x_rtc.c
        stm32f10x_sdio.c
        stm32f10x_spi.c
        stm32f10x_tim.c
        stm32f10x_usart.c
        stm32f10x_wwdg.c
        Touch.c
    Release_Notes.html
```