전기전자기초설계실험 텀프로젝트

스마트 OTP 도어락

지도교수 : 조용범 교수님, 유위 교수님

목요일 7조

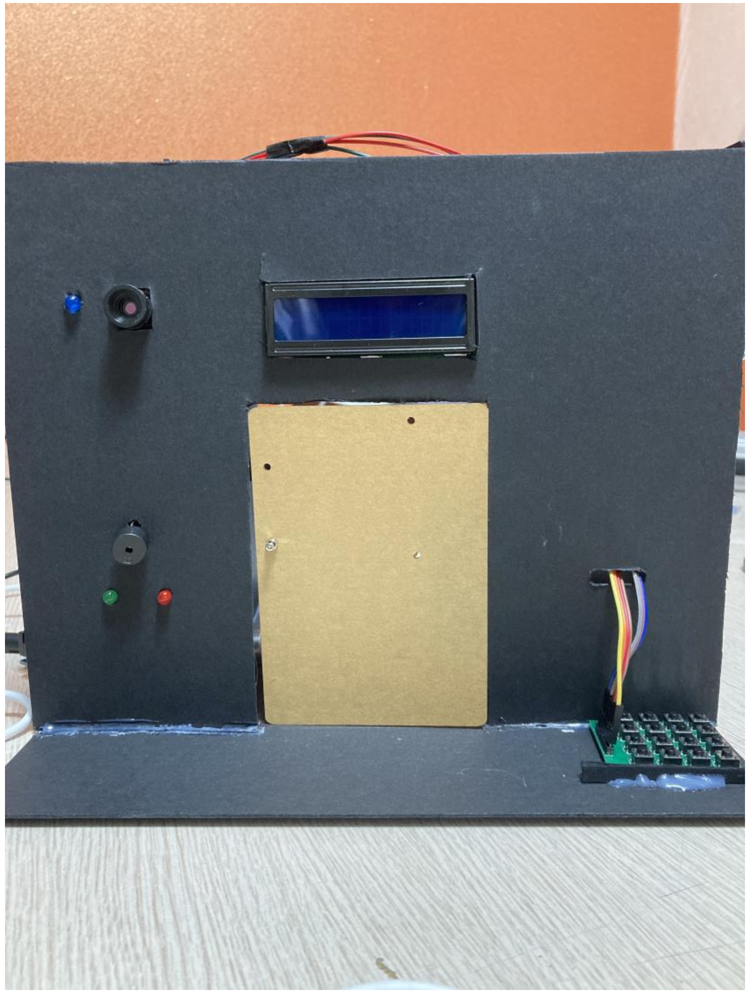
201711210 박원진

201710851 박중선

201810845 박종혁

KU KONKUK UNIVERSITY

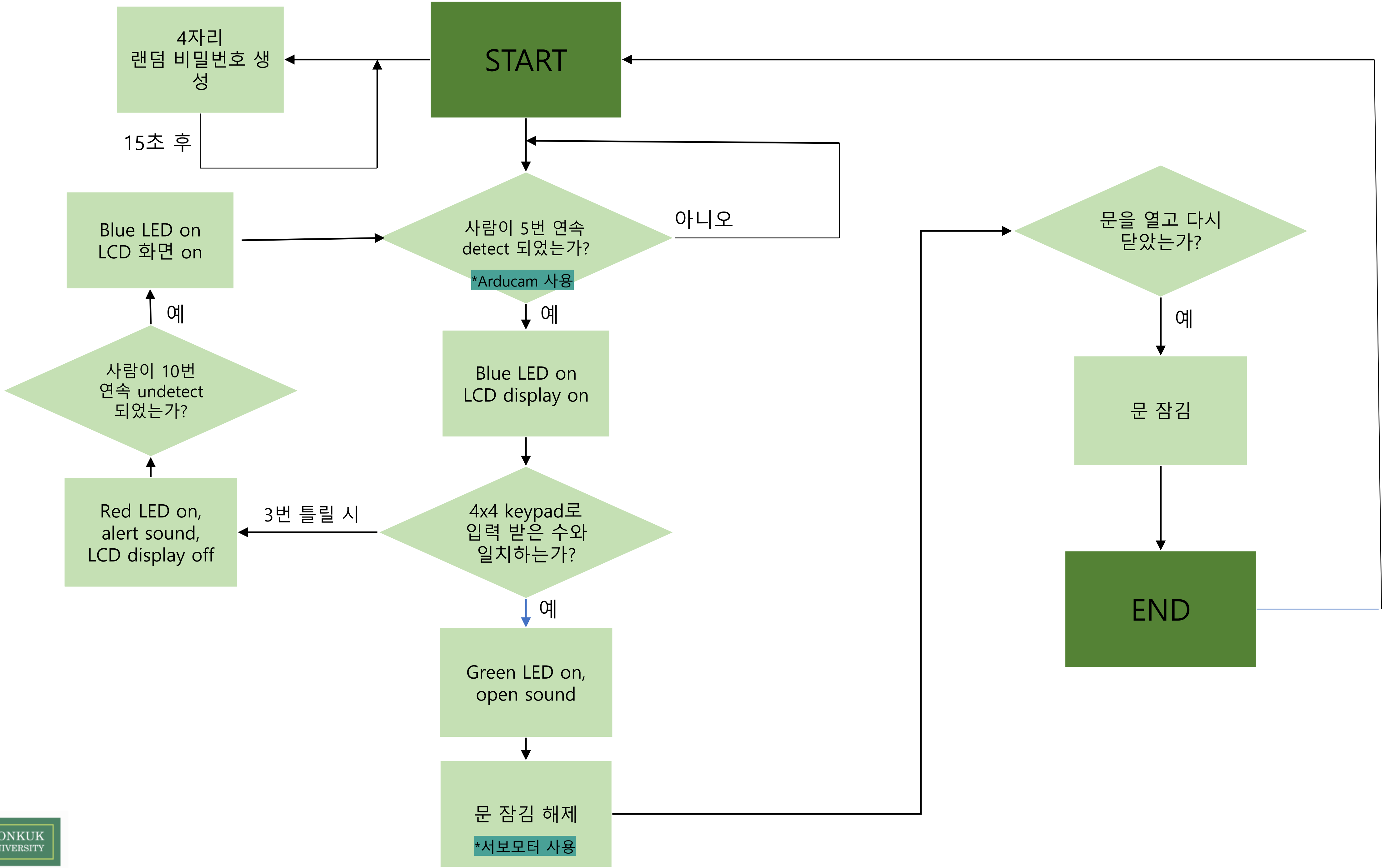
개요



스마트 + OTP

- 사람의 존재 여부를 파악해 스스로 동작을 제어

- 보다 보안성이 강한 OTP 기술을 현관문 도어락에 적용



START

4자리
랜덤 비밀번호 생
성

15초 후

사람이 5번 연속
detect 되었는가?

*Arducam 사용

아니오

예

Blue LED on
LCD display on

4x4 keypad로
입력 받은 수와
일치하는가?

3번 틀릴 시

Red LED on,
alert sound,
LCD display off

사람이 10번
연속 undetect
되었는가?

예

Blue LED on
LCD 화면 on

예

Green LED on,
open sound

문 잠김 해제

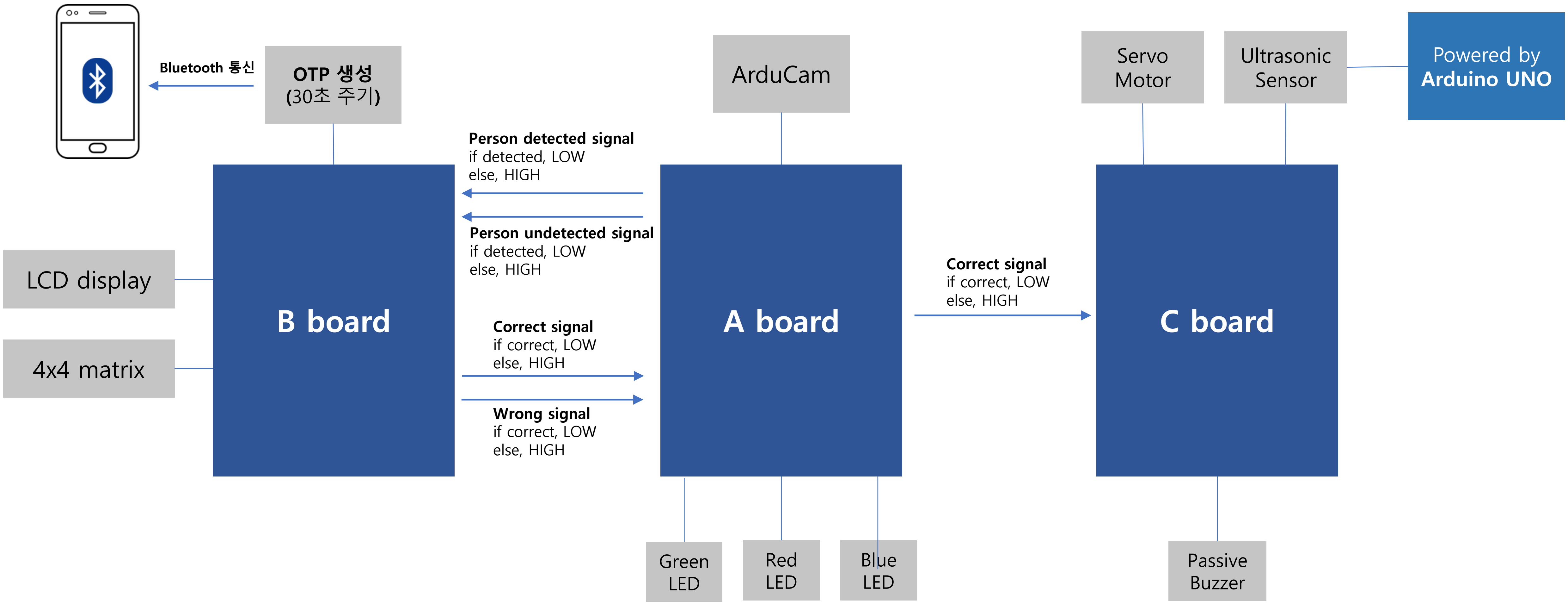
*서보모터 사용

문을 열고 다시
닫았는가?

예

문 잠김

END



Bluetooth 통신

OTP 생성
(30초 주기)

ArduCam

Servo Motor

Ultrasonic Sensor

Powered by
Arduino UNO

LCD display

4x4 matrix

B board

Person detected signal
if detected, LOW
else, HIGH

Person undetected signal
if detected, LOW
else, HIGH

Correct signal
if correct, LOW
else, HIGH

Wrong signal
if correct, LOW
else, HIGH

A board

Correct signal
if correct, LOW
else, HIGH

C board

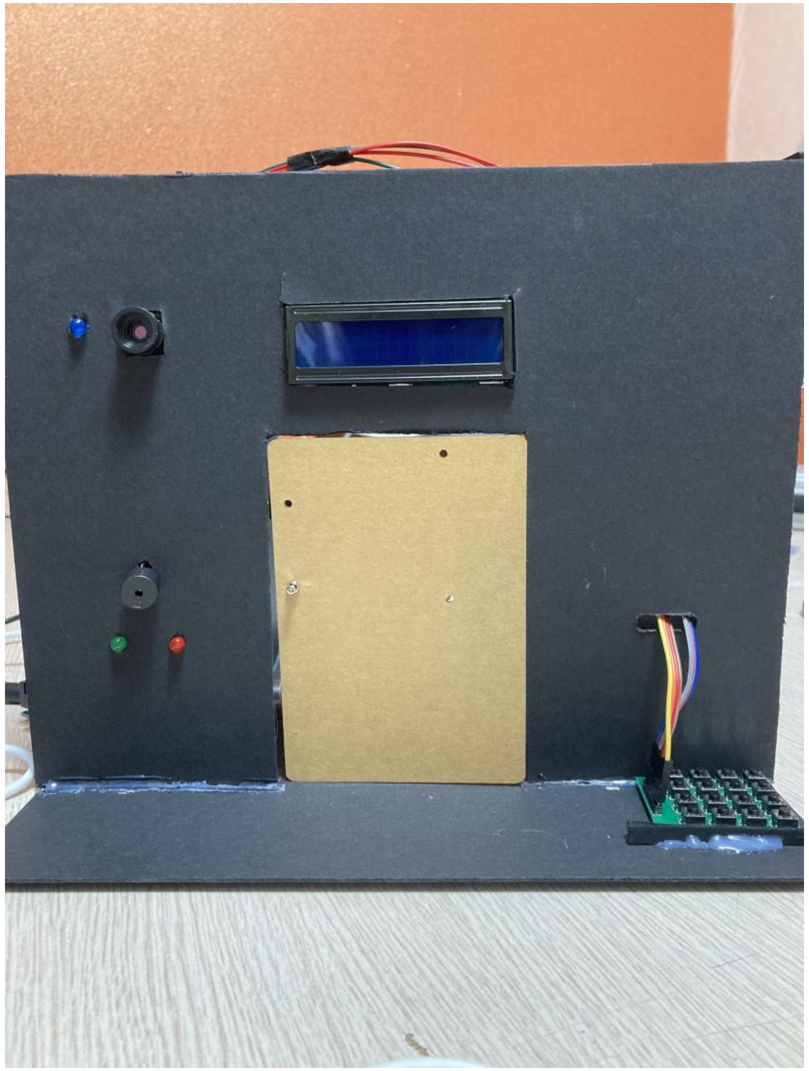
Green LED

Red LED

Blue LED

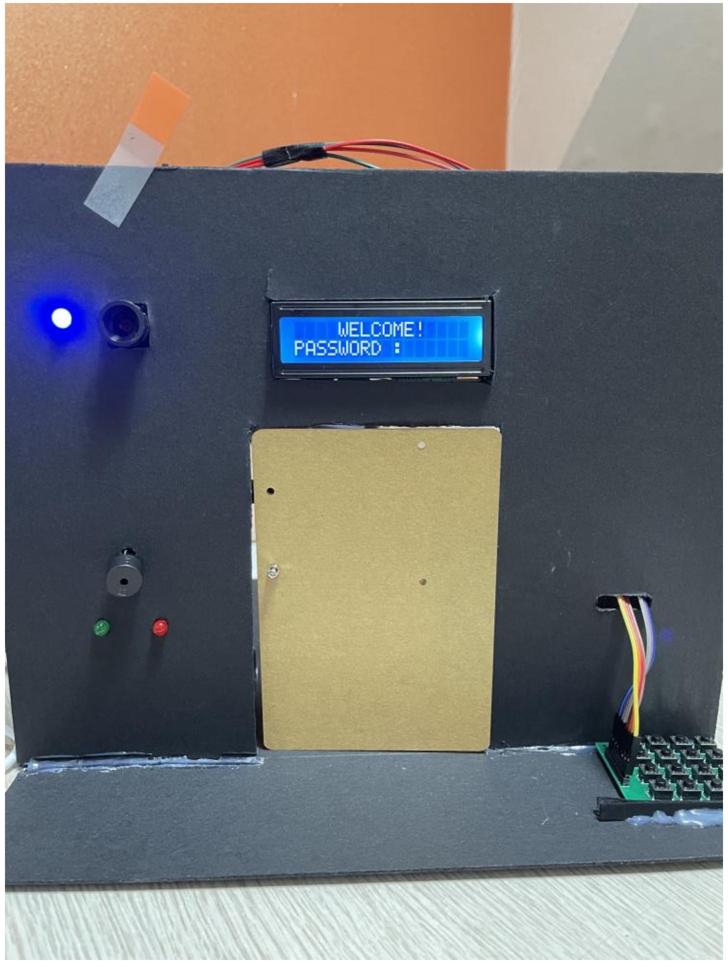
Passive Buzzer

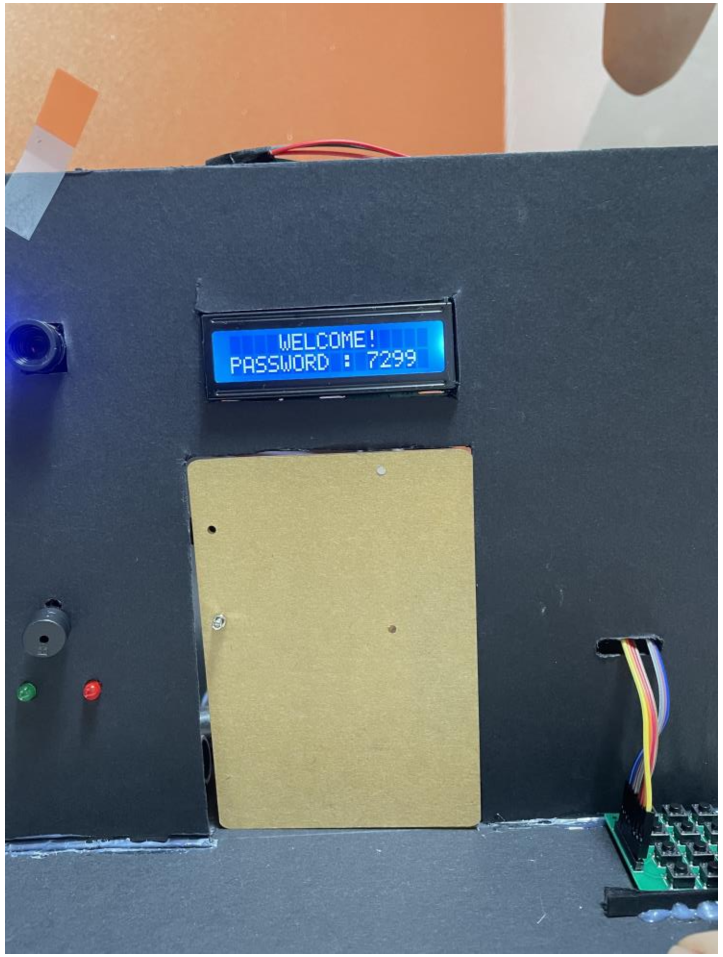
구현 동작 설명



(0) 초기상태

- 모든 display 및 소자 off
- 10초 이상 사람이 인식되지 않았을 경우



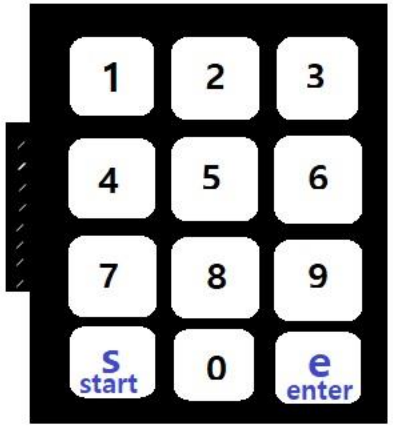


(1) 사람 인식

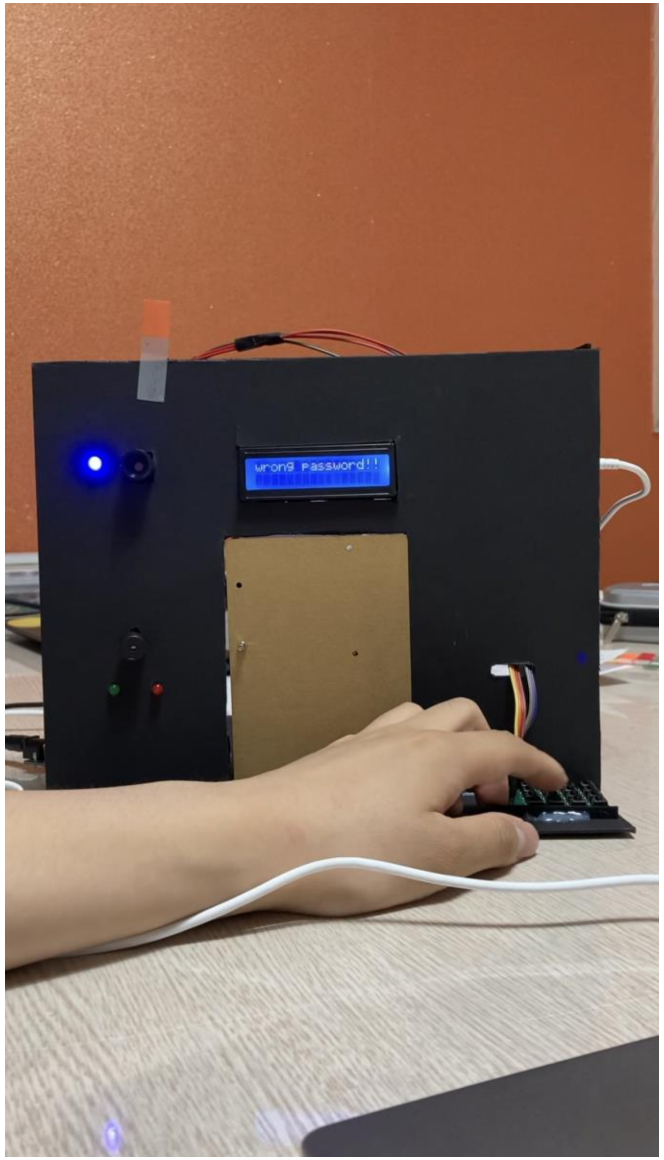
- 약 5초 동안 사람이 연속으로 감지되었을 때
- Blue LED on
- LCD display on

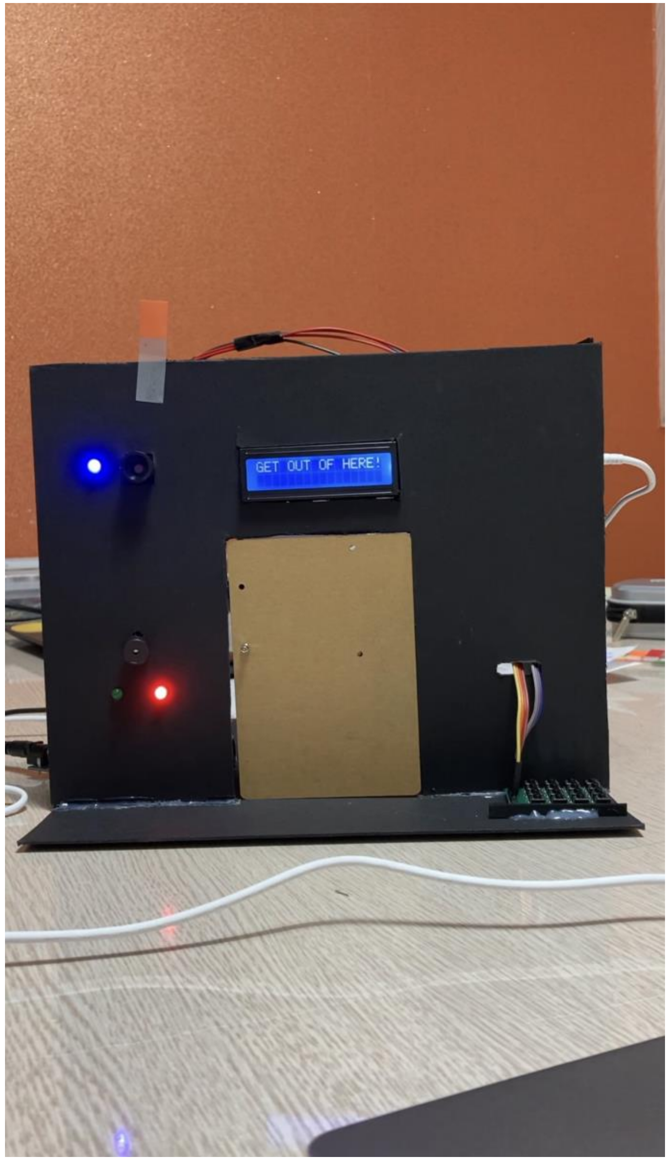
(1) – 1 사람 인식이 됐을 때

- LCD display가 ON일때만, 비밀번호 입력 가능



*** Keypad**

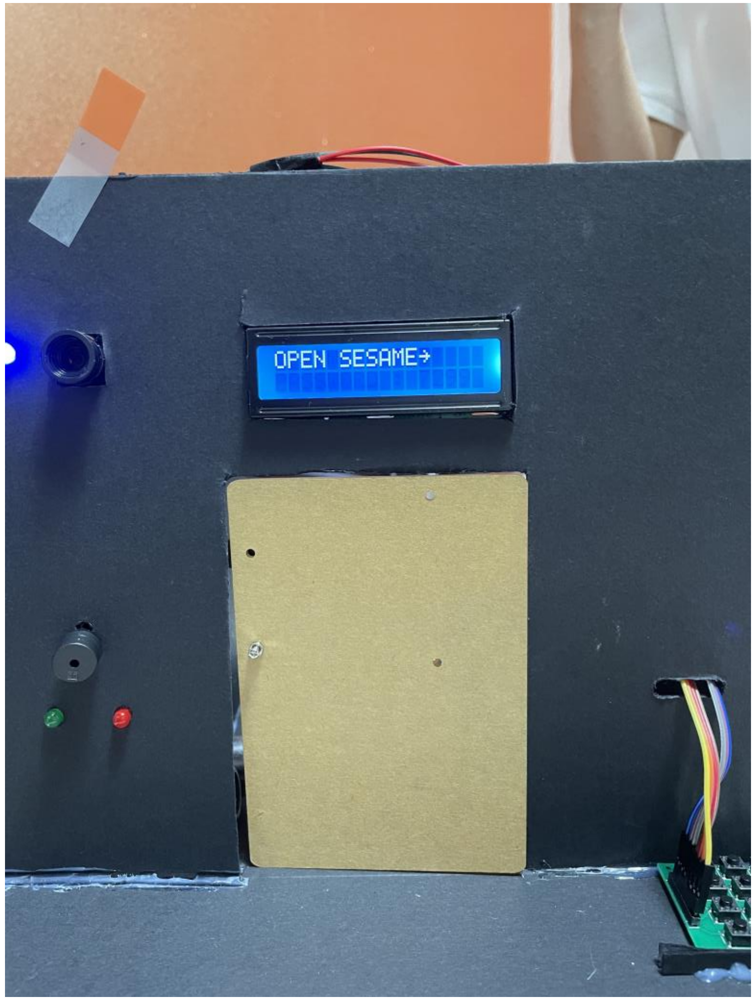


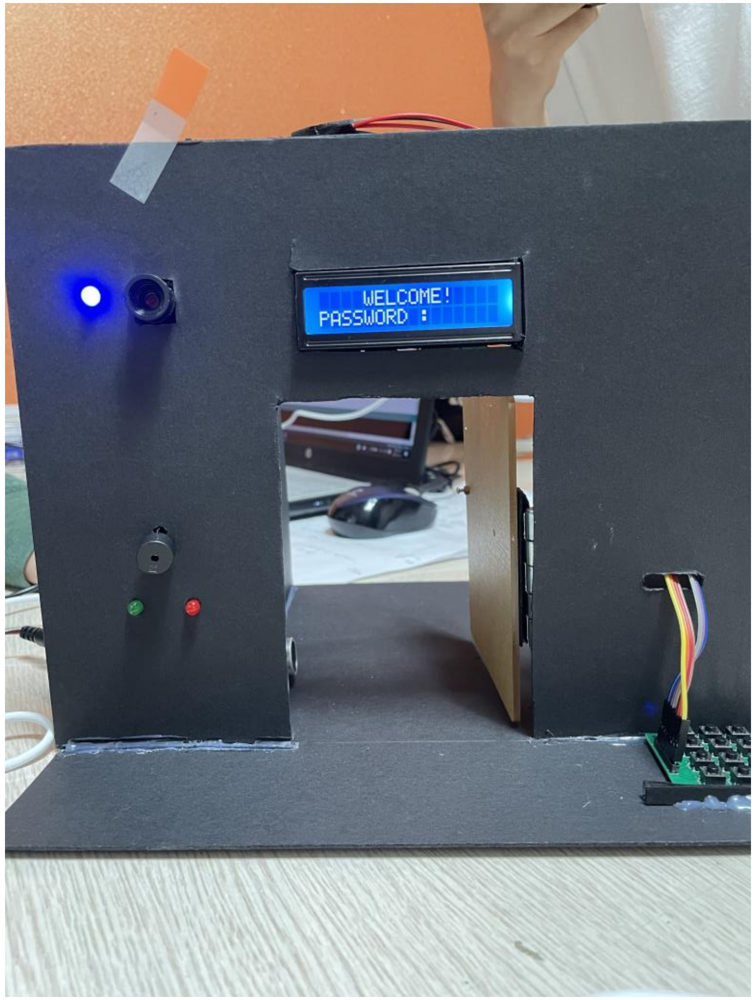


(2) 비밀번호가 틀렸을 때

- "Wrong password" 문구 출력
- 3번 틀렸을 경우, "Get out of here!" 문구와 함께, Red LED on, LCD display off

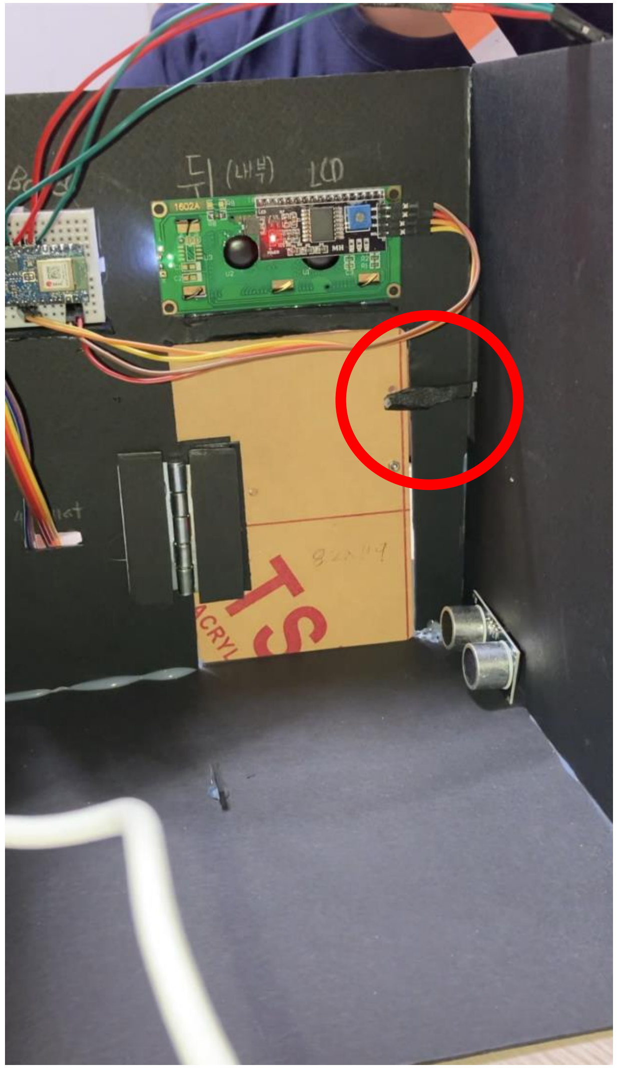
- 초기상태(Blue LED off인 상태)로 돌아가기 위해서는, 10번 연속 사람 인식이 되지 않아야 한다.



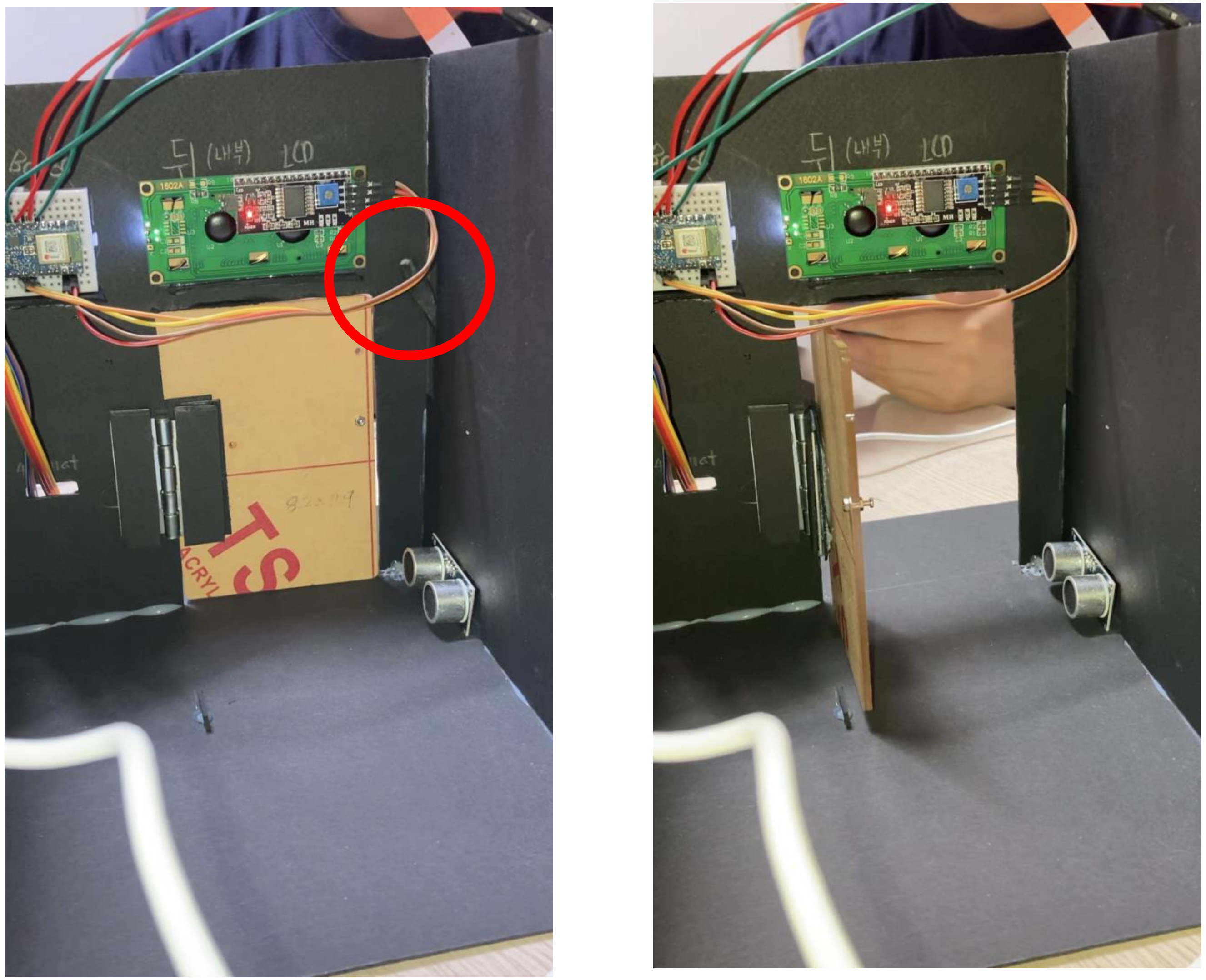


(3) 비밀번호가 맞았을 때

- "Open Sesame" 문구 출력
- 서보 모터 동작, 잠금 해제
- Green LED on
- 수동 부저 작동, "엘리제를 위하여"

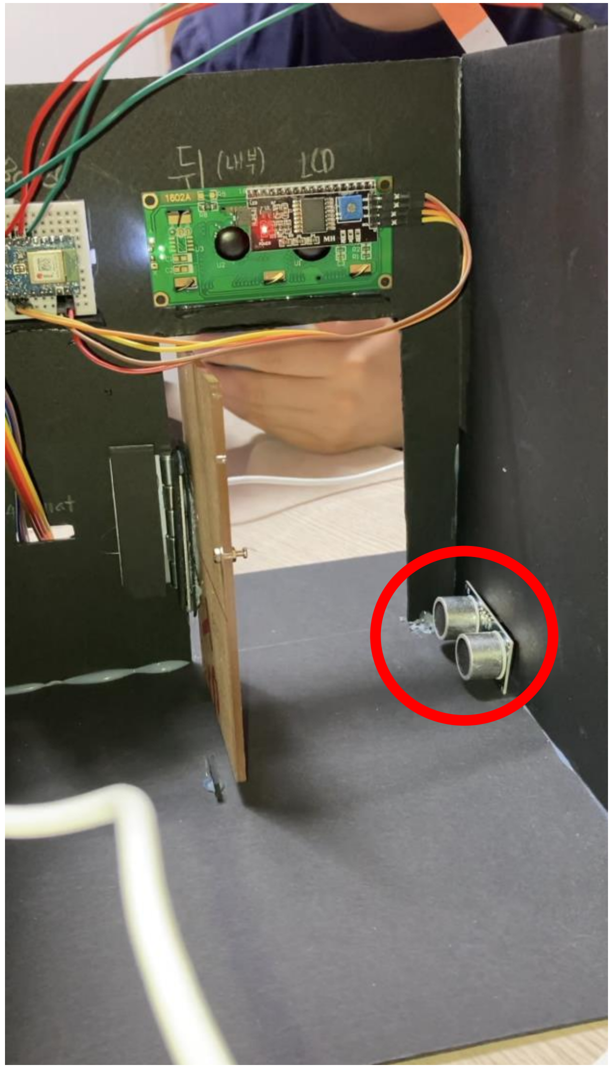


(3) -1 초기 상태 (내부)



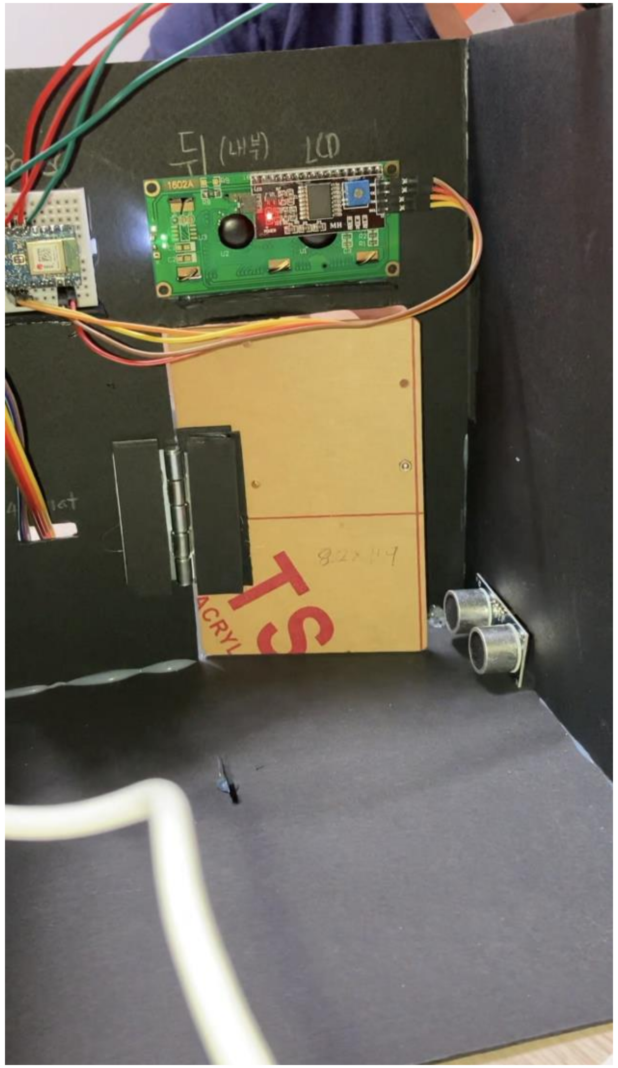
(3) - 2 비밀번호가 맞았을 때 (내부)

- 서보 모터의 회전으로 문의 잠금이 해제된다.



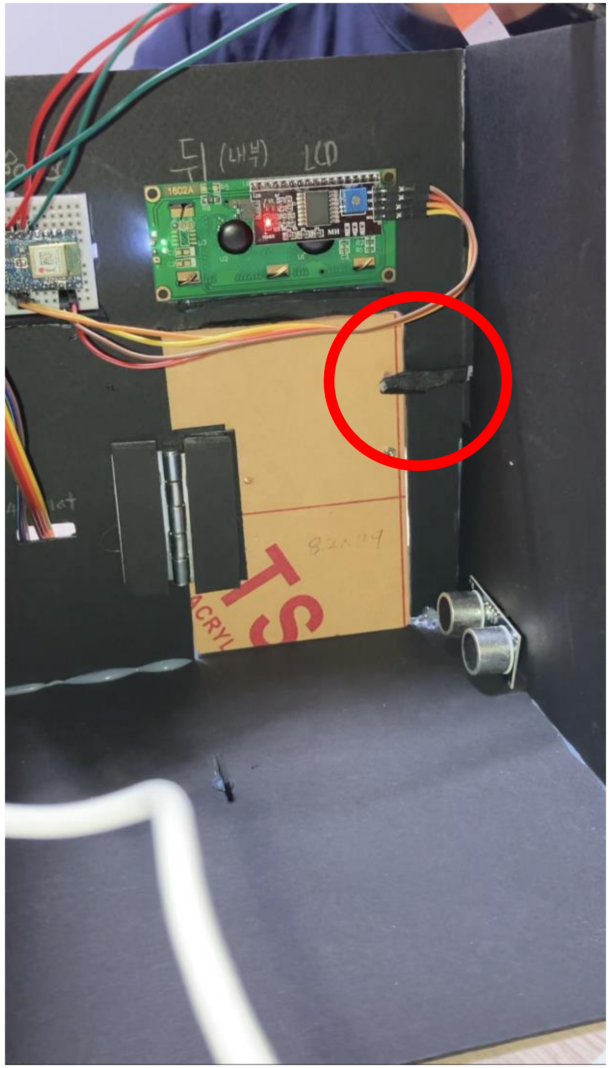
(4) 초음파 센서 작동

- 문이 열리고 2초 후, 초음파 센서 작동



(4) - 1 문 닫기

- 초음파 센서 작동 후, 문을 닫으면 센서가 거리가 길어질 때를 측정한다.



(4) – 2 문 잠김

- 측정하는 거리가 길어지면, 약 2초 뒤 서보 모터가 작동해 문이 잠긴다.

사용한 외부회로 및 기능

*Servo motor

*LCD display

*Ultrasonic sensor

*RGB LED

*4x4 keypad

*Passive buzzer

*ArduCam

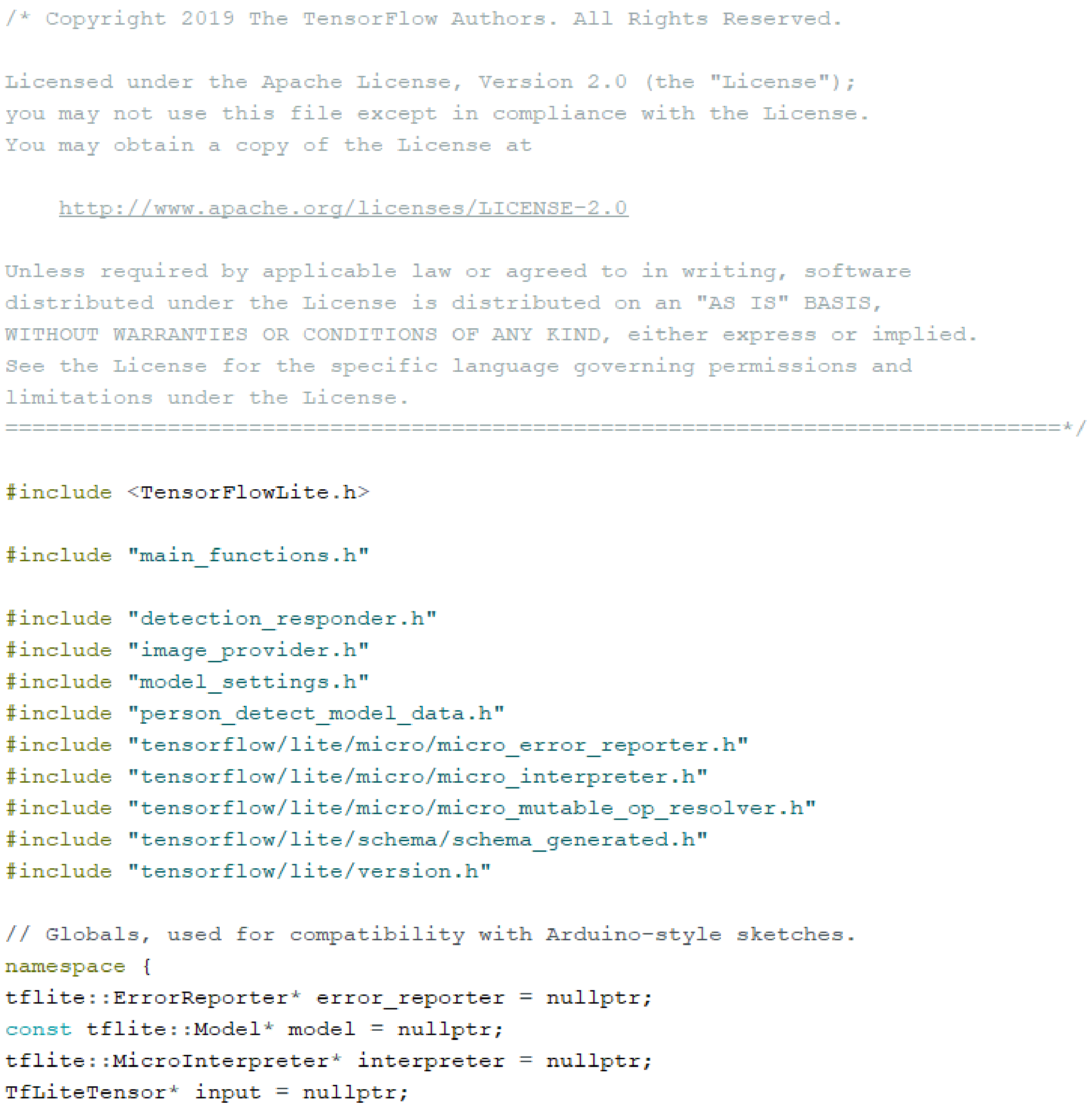
* Timer Interrupt

* Attach Interrupt

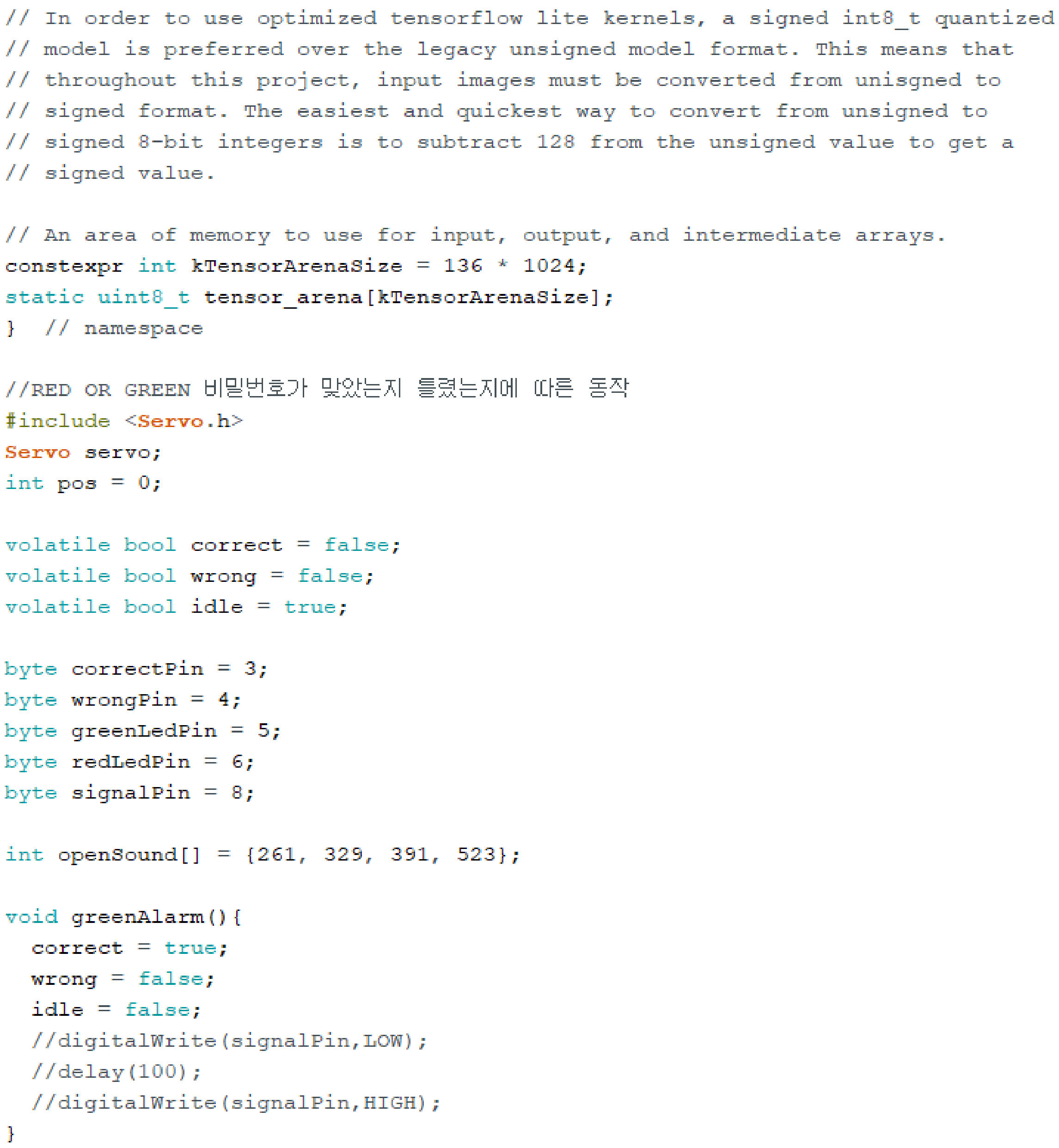
* Blutooth

<A board>
Arduino Code

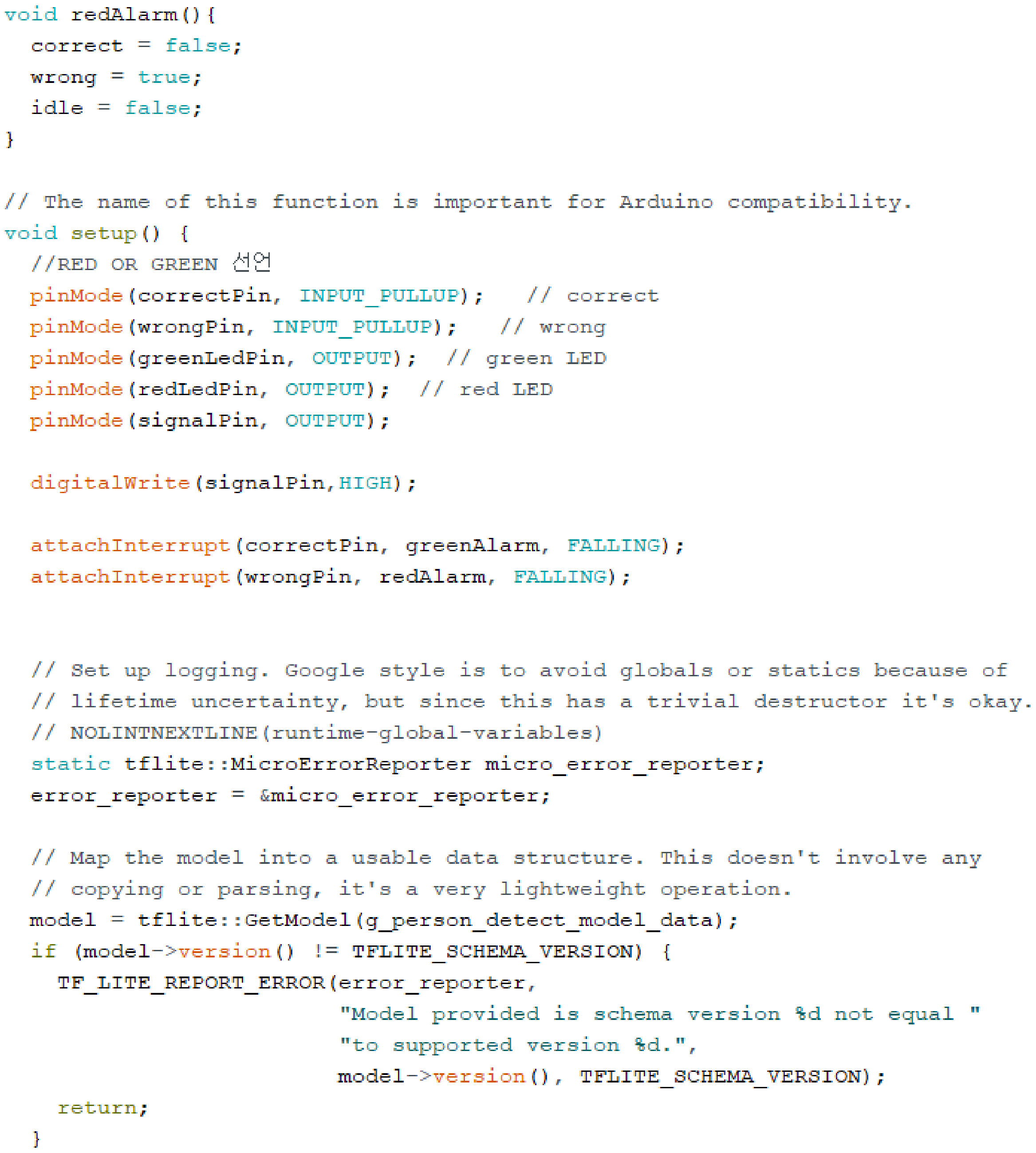
Person_detection_change



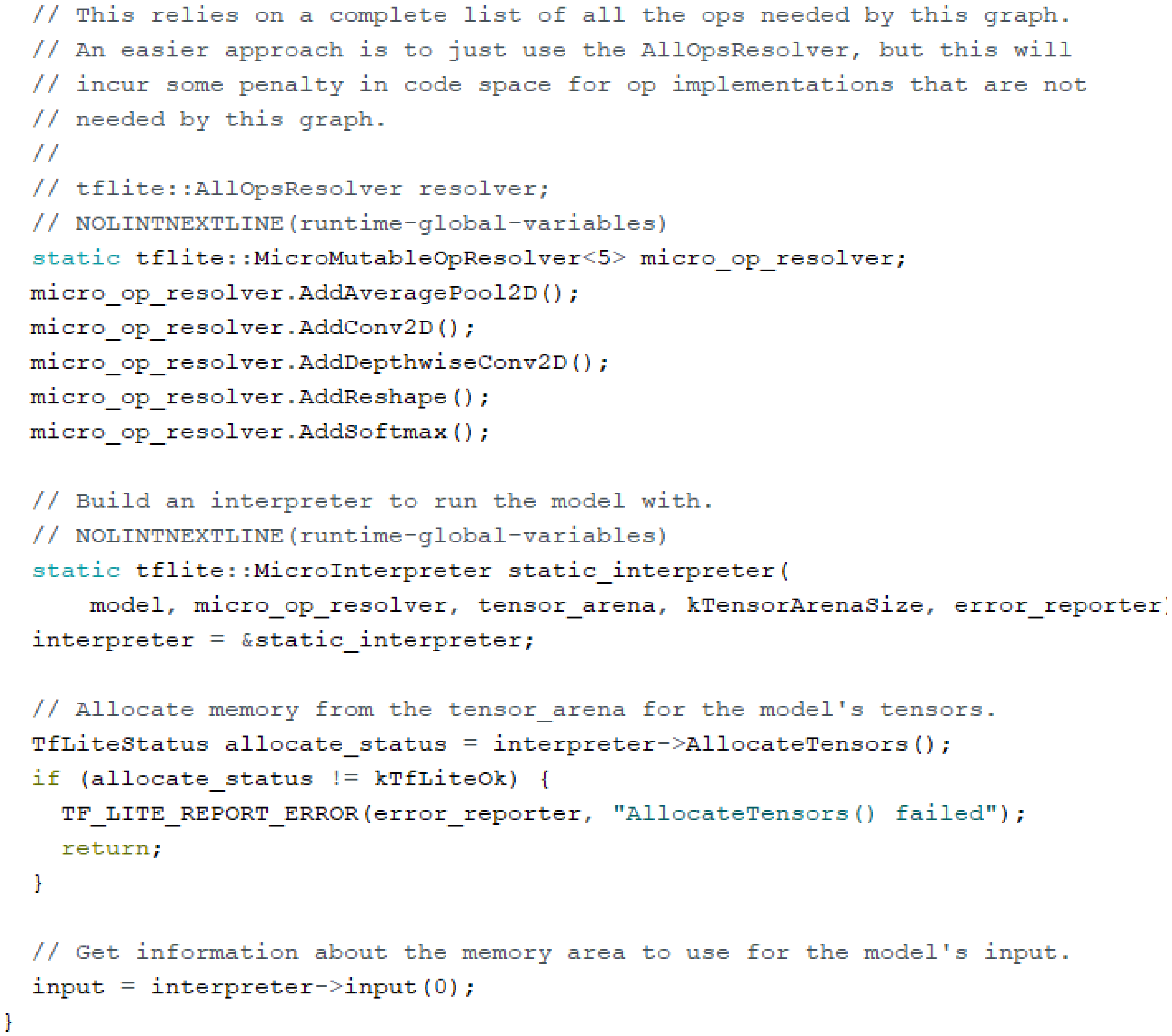
```
/* Copyright 2019 The TensorFlow Authors. All Rights Reserved.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
==============================================================================*/

#include <TensorFlowLite.h>

#include "main_functions.h"

#include "detection_responder.h"
#include "image_provider.h"
#include "model_settings.h"
#include "person_detect_model_data.h"
#include "tensorflow/lite/micro/micro_error_reporter.h"
#include "tensorflow/lite/micro/micro_interpreter.h"
#include "tensorflow/lite/micro/micro_mutable_op_resolver.h"
#include "tensorflow/lite/schema/schema_generated.h"
#include "tensorflow/lite/version.h"

// Globals, used for compatibility with Arduino-style sketches.
namespace {
tflite::ErrorReporter* error_reporter = nullptr;
const tflite::Model* model = nullptr;
tflite::MicroInterpreter* interpreter = nullptr;
TfLiteTensor* input = nullptr;
```



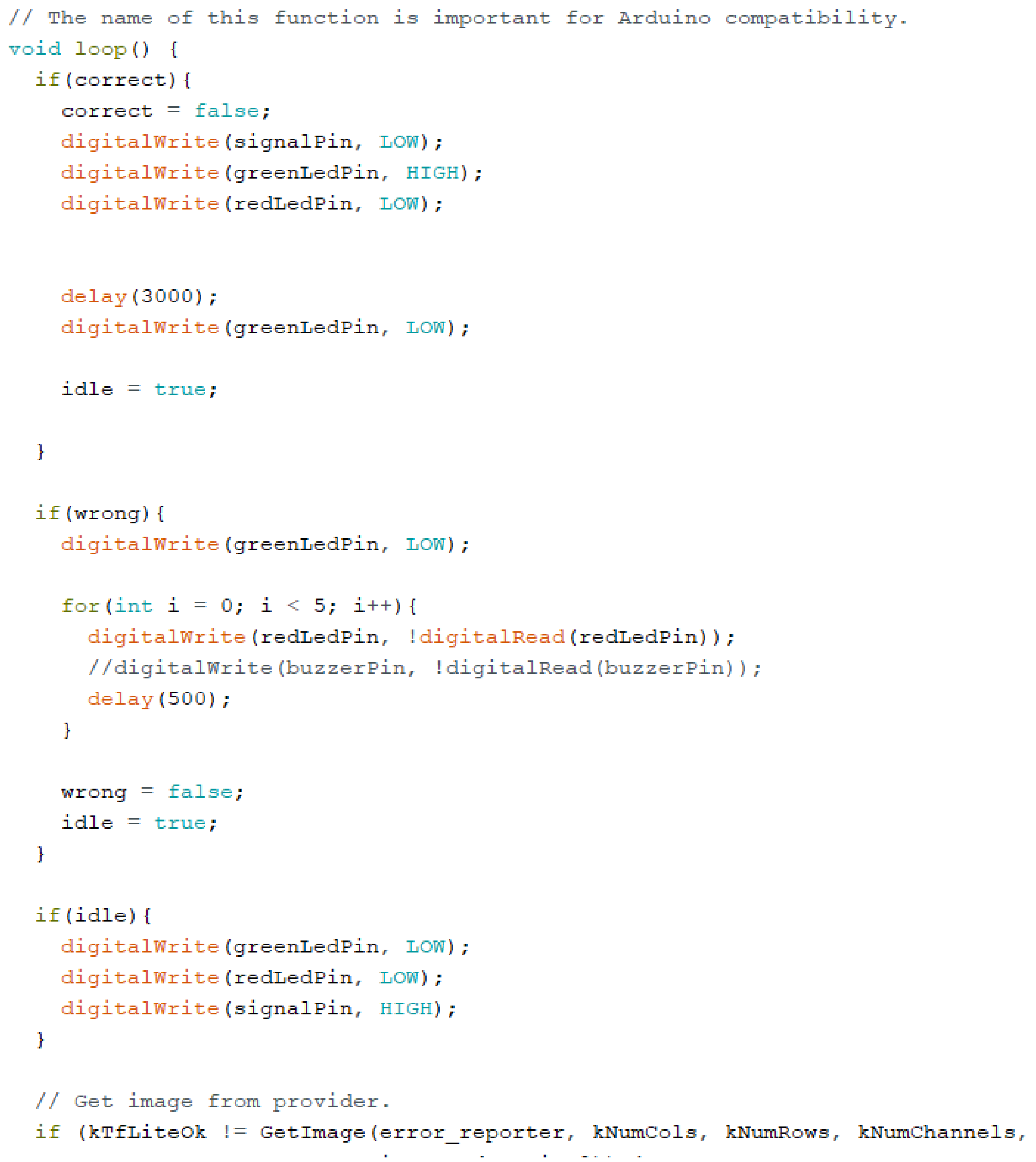
```
// In order to use optimized tensorflow lite kernels, a signed int8_t quantized
// model is preferred over the legacy unsigned model format. This means that
// throughout this project, input images must be converted from unisgned to
// signed format. The easiest and quickest way to convert from unsigned to
// signed 8-bit integers is to subtract 128 from the unsigned value to get a
// signed value.

// An area of memory to use for input, output, and intermediate arrays.
constexpr int kTensorArenaSize = 136 * 1024;
static uint8_t tensor_arena[kTensorArenaSize];
}  // namespace

//RED OR GREEN 비밀번호가 맞았는지 틀렸는지에 따른 동작
#include <Servo.h>
Servo servo;
int pos = 0;

volatile bool correct = false;
volatile bool wrong = false;
volatile bool idle = true;

byte correctPin = 3;
byte wrongPin = 4;
byte greenLedPin = 5;
byte redLedPin = 6;
byte signalPin = 8;

int openSound[] = {261, 329, 391, 523};

void greenAlarm(){
  correct = true;
  wrong = false;
  idle = false;
  //digitalWrite(signalPin,LOW);
  //delay(100);
  //digitalWrite(signalPin,HIGH);
}
```



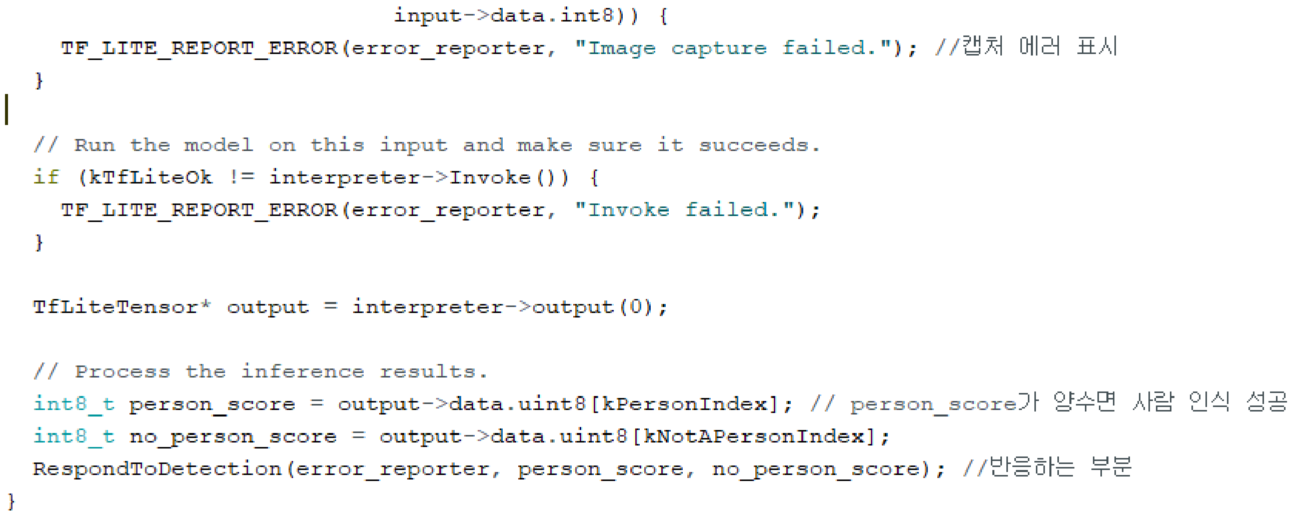
```
void redAlarm(){
  correct = false;
  wrong = true;
  idle = false;
}

// The name of this function is important for Arduino compatibility.
void setup() {
  //RED OR GREEN 선언
  pinMode(correctPin, INPUT_PULLUP);   // correct
  pinMode(wrongPin, INPUT_PULLUP);   // wrong
  pinMode(greenLedPin, OUTPUT);  // green LED
  pinMode(redLedPin, OUTPUT);  // red LED
  pinMode(signalPin, OUTPUT);

  digitalWrite(signalPin,HIGH);

  attachInterrupt(correctPin, greenAlarm, FALLING);
  attachInterrupt(wrongPin, redAlarm, FALLING);


  // Set up logging. Google style is to avoid globals or statics because of
  // lifetime uncertainty, but since this has a trivial destructor it's okay.
  // NOLINTNEXTLINE(runtime-global-variables)
  static tflite::MicroErrorReporter micro_error_reporter;
  error_reporter = &micro_error_reporter;

  // Map the model into a usable data structure. This doesn't involve any
  // copying or parsing, it's a very lightweight operation.
  model = tflite::GetModel(g_person_detect_model_data);
  if (model->version() != TFLITE_SCHEMA_VERSION) {
    TF_LITE_REPORT_ERROR(error_reporter,
                         "Model provided is schema version %d not equal "
                         "to supported version %d.",
                         model->version(), TFLITE_SCHEMA_VERSION);
    return;
  }
```



```
  // This relies on a complete list of all the ops needed by this graph.
  // An easier approach is to just use the AllOpsResolver, but this will
  // incur some penalty in code space for op implementations that are not
  // needed by this graph.
  //
  // tflite::AllOpsResolver resolver;
  // NOLINTNEXTLINE(runtime-global-variables)
  static tflite::MicroMutableOpResolver<5> micro_op_resolver;
  micro_op_resolver.AddAveragePool2D();
  micro_op_resolver.AddConv2D();
  micro_op_resolver.AddDepthwiseConv2D();
  micro_op_resolver.AddReshape();
  micro_op_resolver.AddSoftmax();

  // Build an interpreter to run the model with.
  // NOLINTNEXTLINE(runtime-global-variables)
  static tflite::MicroInterpreter static_interpreter(
      model, micro_op_resolver, tensor_arena, kTensorArenaSize, error_reporter)
  interpreter = &static_interpreter;

  // Allocate memory from the tensor_arena for the model's tensors.
  TfLiteStatus allocate_status = interpreter->AllocateTensors();
  if (allocate_status != kTfLiteOk) {
    TF_LITE_REPORT_ERROR(error_reporter, "AllocateTensors() failed");
    return;
  }

  // Get information about the memory area to use for the model's input.
  input = interpreter->input(0);
}
```

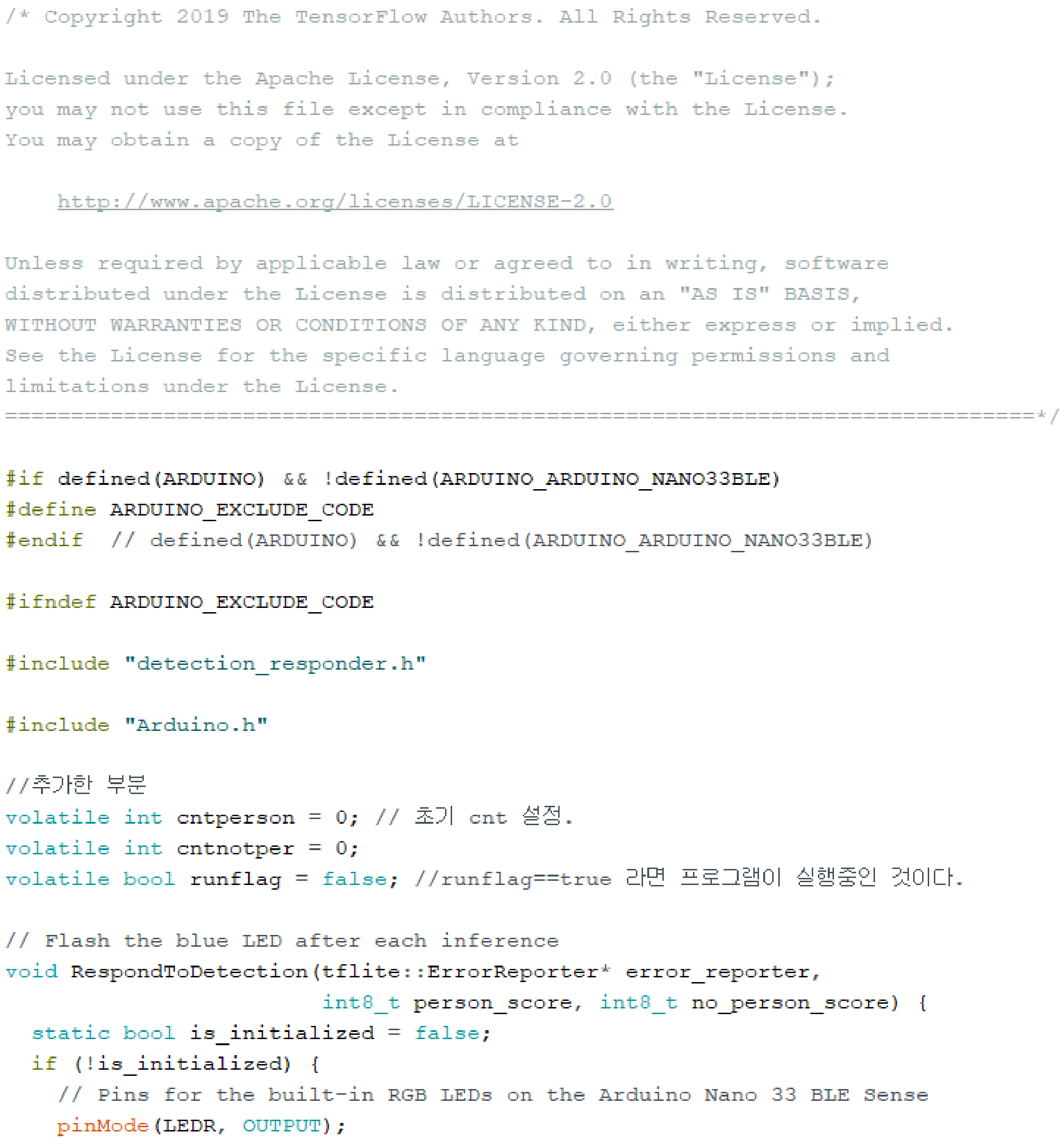


```
// The name of this function is important for Arduino compatibility.
void loop() {
  if(correct){
    correct = false;
    digitalWrite(signalPin, LOW);
    digitalWrite(greenLedPin, HIGH);
    digitalWrite(redLedPin, LOW);


    delay(3000);
    digitalWrite(greenLedPin, LOW);

    idle = true;

  }

  if(wrong){
    digitalWrite(greenLedPin, LOW);

    for(int i = 0; i < 5; i++){
      digitalWrite(redLedPin, !digitalRead(redLedPin));
      //digitalWrite(buzzerPin, !digitalRead(buzzerPin));
      delay(500);
    }

    wrong = false;
    idle = true;
  }

  if(idle){
    digitalWrite(greenLedPin, LOW);
    digitalWrite(redLedPin, LOW);
    digitalWrite(signalPin, HIGH);
  }

  // Get image from provider.
  if (kTfLiteOk != GetImage(error_reporter, kNumCols, kNumRows, kNumChannels,
```

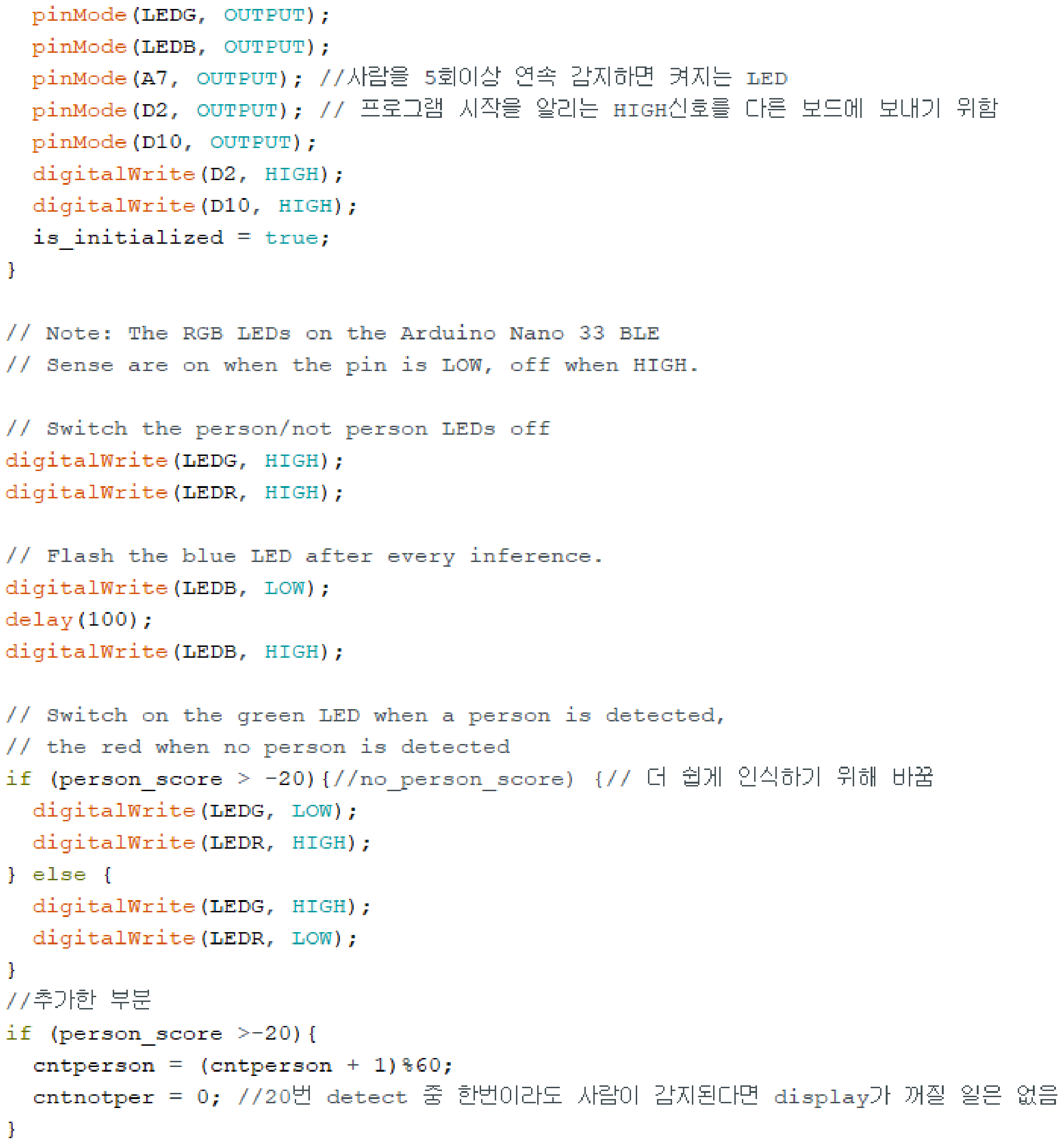


```
                           input->data.int8)) {
    TF_LITE_REPORT_ERROR(error_reporter, "Image capture failed."); //캡처 에러 표시
  }

  // Run the model on this input and make sure it succeeds.
  if (kTfLiteOk != interpreter->Invoke()) {
    TF_LITE_REPORT_ERROR(error_reporter, "Invoke failed.");
  }

  TfLiteTensor* output = interpreter->output(0);

  // Process the inference results.
  int8_t person_score = output->data.uint8[kPersonIndex]; // person_score가 양수면 사람 인식 성공
  int8_t no_person_score = output->data.uint8[kNotAPersonIndex];
  RespondToDetection(error_reporter, person_score, no_person_score); //반응하는 부분
}
```

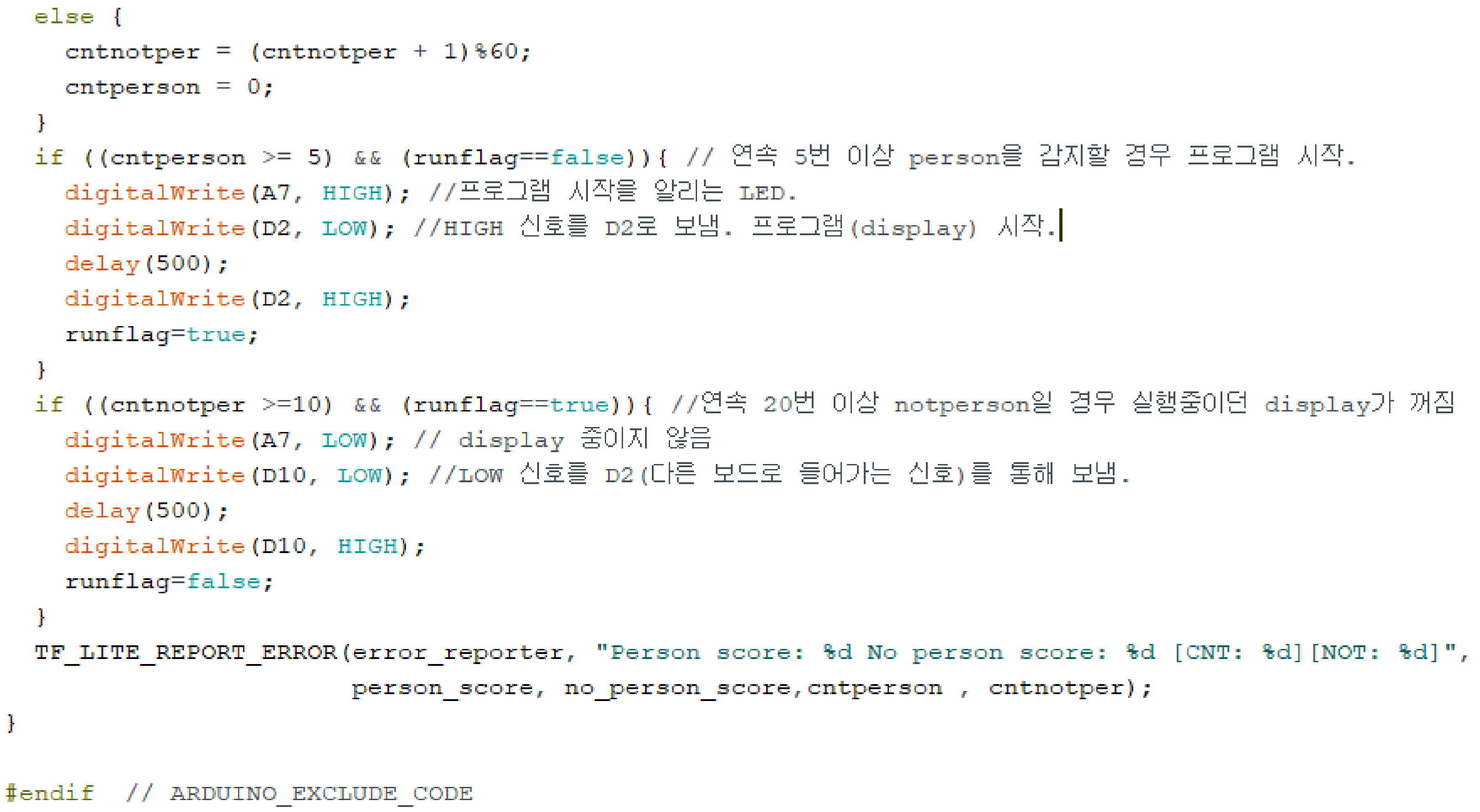
Arduino_detection_responder.cpp



```cpp
/* Copyright 2019 The TensorFlow Authors. All Rights Reserved.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
==============================================================================*/

#if defined(ARDUINO) && !defined(ARDUINO_ARDUINO_NANO33BLE)
#define ARDUINO_EXCLUDE_CODE
#endif  // defined(ARDUINO) && !defined(ARDUINO_ARDUINO_NANO33BLE)

#ifndef ARDUINO_EXCLUDE_CODE

#include "detection_responder.h"

#include "Arduino.h"

//추가한 부분
volatile int cntperson = 0; // 초기 cnt 설정.
volatile int cntnotper = 0;
volatile bool runflag = false; //runflag==true 라면 프로그램이 실행중인 것이다.

// Flash the blue LED after each inference
void RespondToDetection(tflite::ErrorReporter* error_reporter,
                        int8_t person_score, int8_t no_person_score) {
  static bool is_initialized = false;
  if (!is_initialized) {
    // Pins for the built-in RGB LEDs on the Arduino Nano 33 BLE Sense
    pinMode(LEDR, OUTPUT);
```

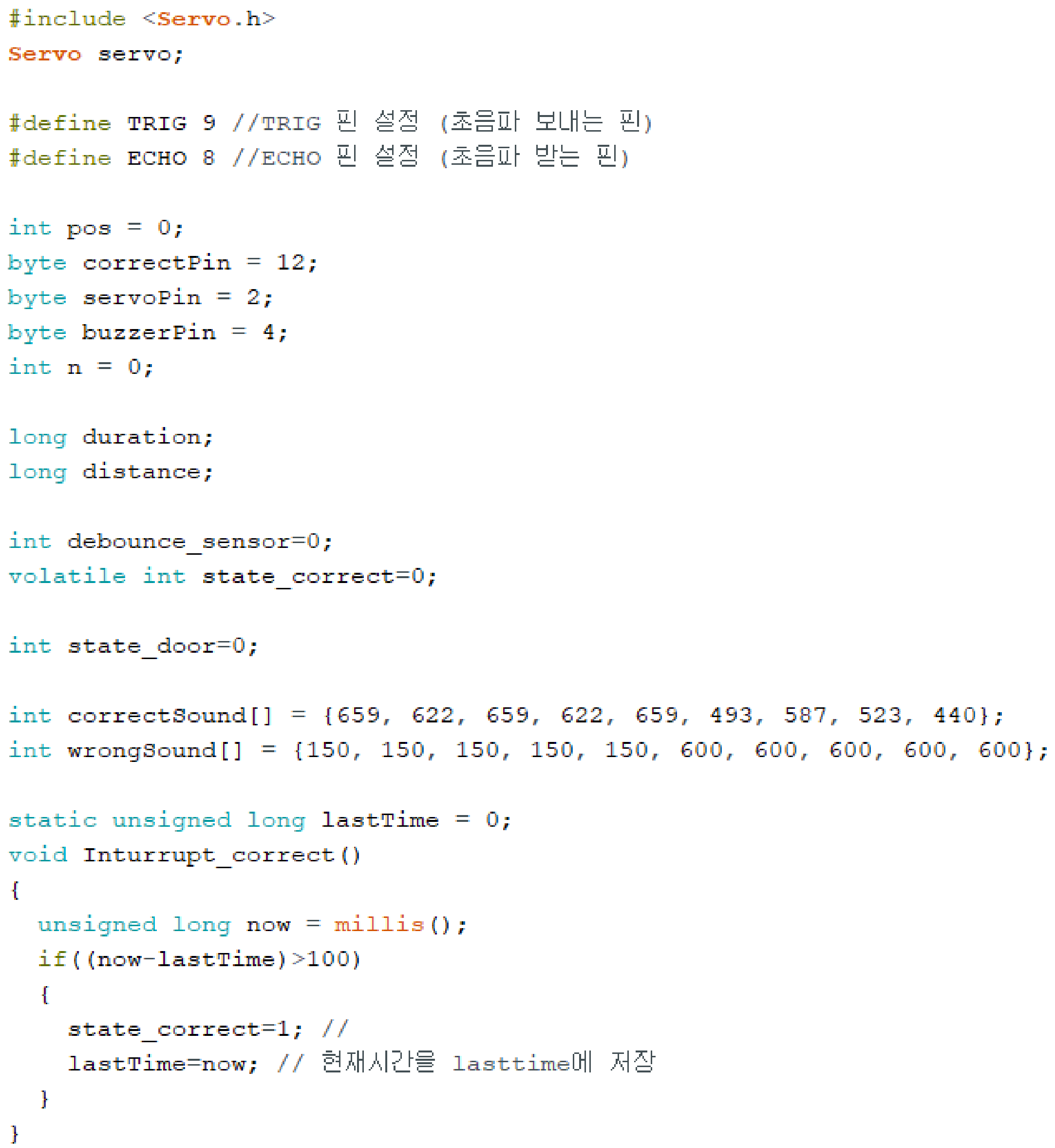


```cpp
    pinMode(LEDG, OUTPUT);
    pinMode(LEDB, OUTPUT);
    pinMode(A7, OUTPUT); //사람을 5회이상 연속 감지하면 켜지는 LED
    pinMode(D2, OUTPUT); // 프로그램 시작을 알리는 HIGH신호를 다른 보드에 보내기 위함
    pinMode(D10, OUTPUT);
    digitalWrite(D2, HIGH);
    digitalWrite(D10, HIGH);
    is_initialized = true;
  }

  // Note: The RGB LEDs on the Arduino Nano 33 BLE
  // Sense are on when the pin is LOW, off when HIGH.

  // Switch the person/not person LEDs off
  digitalWrite(LEDG, HIGH);
  digitalWrite(LEDR, HIGH);

  // Flash the blue LED after every inference.
  digitalWrite(LEDB, LOW);
  delay(100);
  digitalWrite(LEDB, HIGH);

  // Switch on the green LED when a person is detected,
  // the red when no person is detected
  if (person_score > -20){//no_person_score) {// 더 쉽게 인식하기 위해 바꿈
    digitalWrite(LEDG, LOW);
    digitalWrite(LEDR, HIGH);
  } else {
    digitalWrite(LEDG, HIGH);
    digitalWrite(LEDR, LOW);
  }
  //추가한 부분
  if (person_score >-20){
    cntperson = (cntperson + 1)%60;
    cntnotper = 0; //20번 detect 중 한번이라도 사람이 감지된다면 display가 꺼질 일은 없음
  }
```

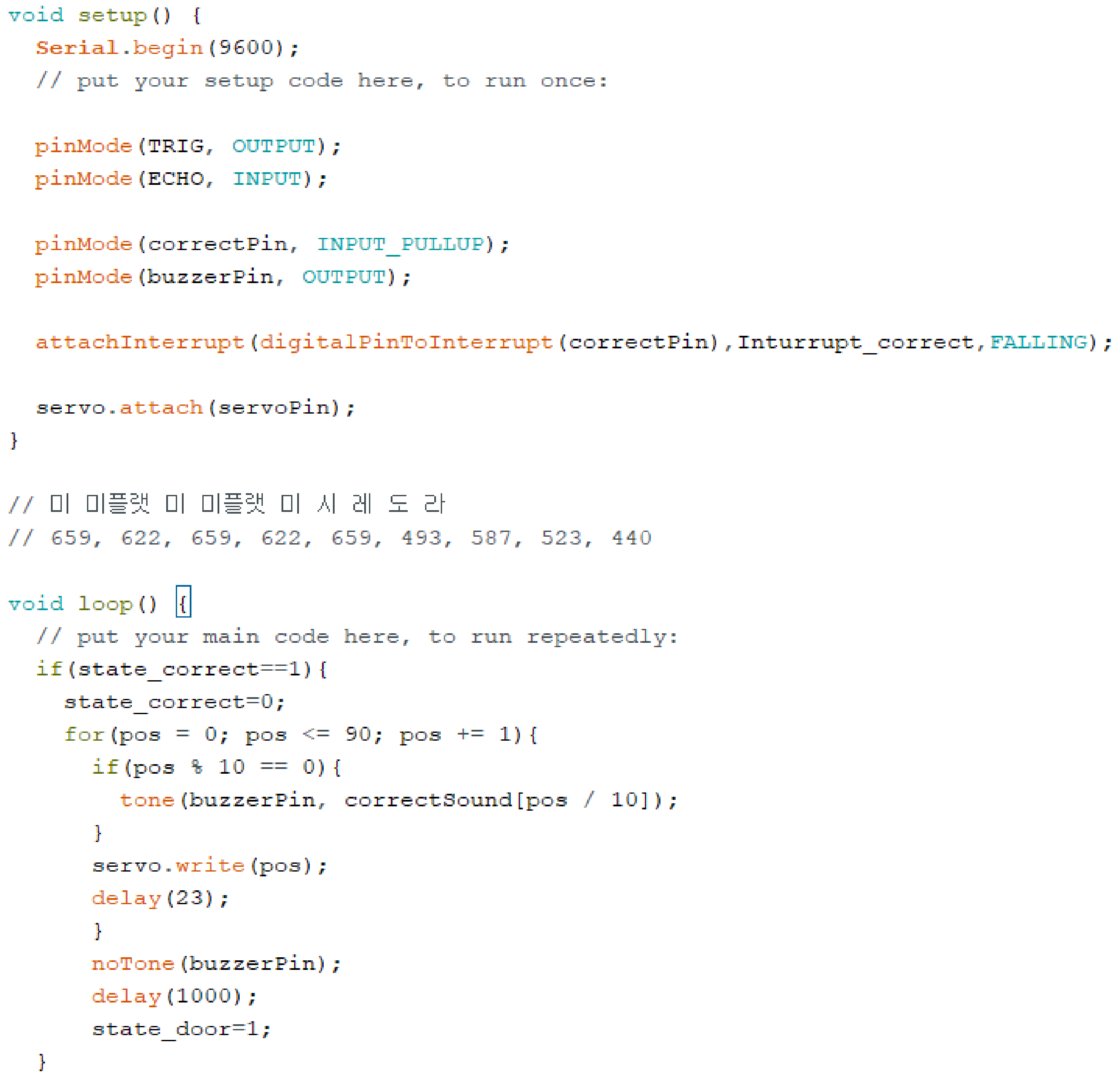


```
  else {
    cntnotper = (cntnotper + 1)%60;
    cntperson = 0;
  }
  if ((cntperson >= 5) && (runflag==false)){ // 연속 5번 이상 person을 감지할 경우 프로그램 시작.
    digitalWrite(A7, HIGH); //프로그램 시작을 알리는 LED.
    digitalWrite(D2, LOW); //HIGH 신호를 D2로 보냄. 프로그램(display) 시작.
    delay(500);
    digitalWrite(D2, HIGH);
    runflag=true;
  }
  if ((cntnotper >=10) && (runflag==true)){ //연속 20번 이상 notperson일 경우 실행중이던 display가 꺼짐
    digitalWrite(A7, LOW); // display 중이지 않음
    digitalWrite(D10, LOW); //LOW 신호를 D2(다른 보드로 들어가는 신호)를 통해 보냄.
    delay(500);
    digitalWrite(D10, HIGH);
    runflag=false;
  }
  TF_LITE_REPORT_ERROR(error_reporter, "Person score: %d No person score: %d [CNT: %d][NOT: %d]",
                       person_score, no_person_score,cntperson , cntnotper);
}

#endif  // ARDUINO_EXCLUDE_CODE
```

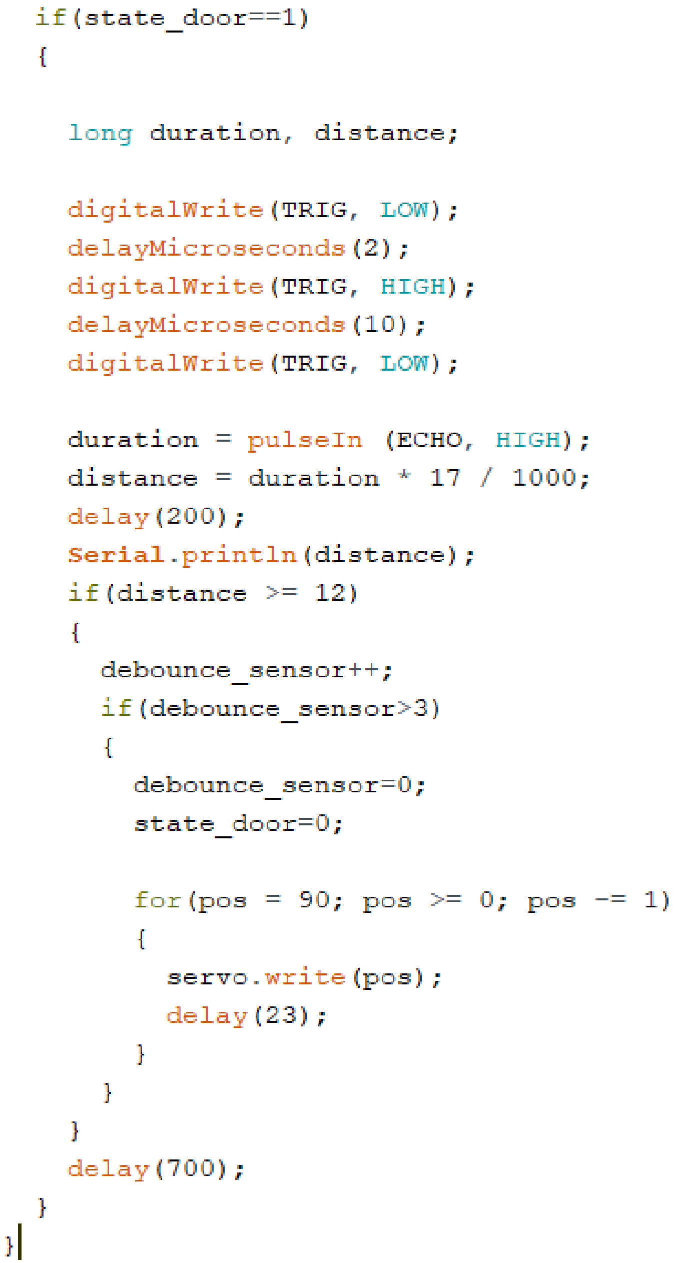
<C board>
Arduino Code



```
#include <Servo.h>
Servo servo;

#define TRIG 9 //TRIG 핀 설정 (초음파 보내는 핀)
#define ECHO 8 //ECHO 핀 설정 (초음파 받는 핀)

int pos = 0;
byte correctPin = 12;
byte servoPin = 2;
byte buzzerPin = 4;
int n = 0;

long duration;
long distance;

int debounce_sensor=0;
volatile int state_correct=0;

int state_door=0;

int correctSound[] = {659, 622, 659, 622, 659, 493, 587, 523, 440};
int wrongSound[] = {150, 150, 150, 150, 150, 600, 600, 600, 600, 600};

static unsigned long lastTime = 0;
void Inturrupt_correct()
{
  unsigned long now = millis();
  if((now-lastTime)>100)
  {
    state_correct=1; //
    lastTime=now; // 현재시간을 lasttime에 저장
  }
}
```



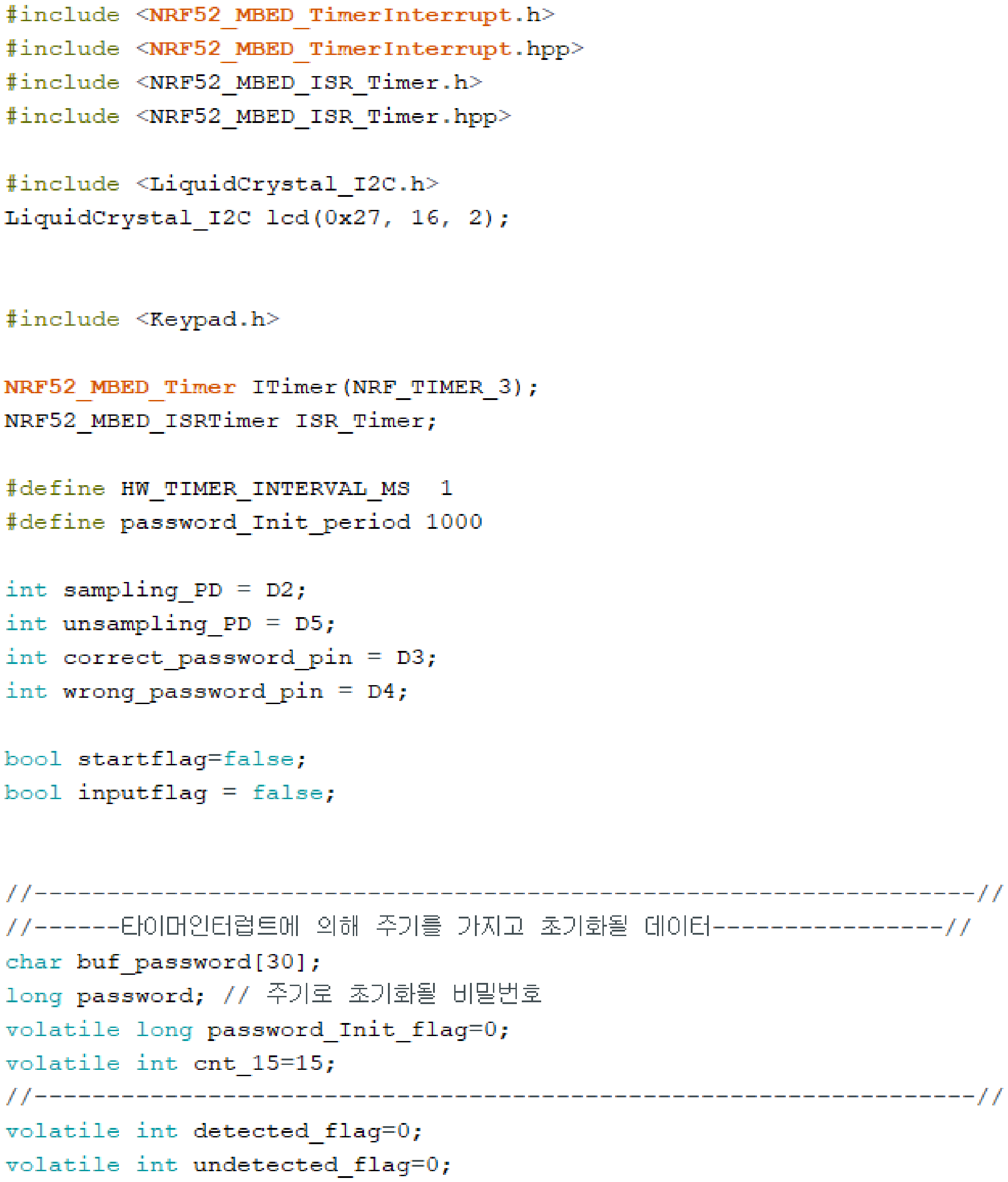
```
void setup() {
  Serial.begin(9600);
  // put your setup code here, to run once:

  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);

  pinMode(correctPin, INPUT_PULLUP);
  pinMode(buzzerPin, OUTPUT);

  attachInterrupt(digitalPinToInterrupt(correctPin),Inturrupt_correct,FALLING);

  servo.attach(servoPin);
}

// 미 미플랫 미 미플랫 미 시 레 도 라
// 659, 622, 659, 622, 659, 493, 587, 523, 440

void loop() {
  // put your main code here, to run repeatedly:
  if(state_correct==1){
    state_correct=0;
    for(pos = 0; pos <= 90; pos += 1){
      if(pos % 10 == 0){
        tone(buzzerPin, correctSound[pos / 10]);
      }
      servo.write(pos);
      delay(23);
      }
      noTone(buzzerPin);
      delay(1000);
      state_door=1;
  }
```



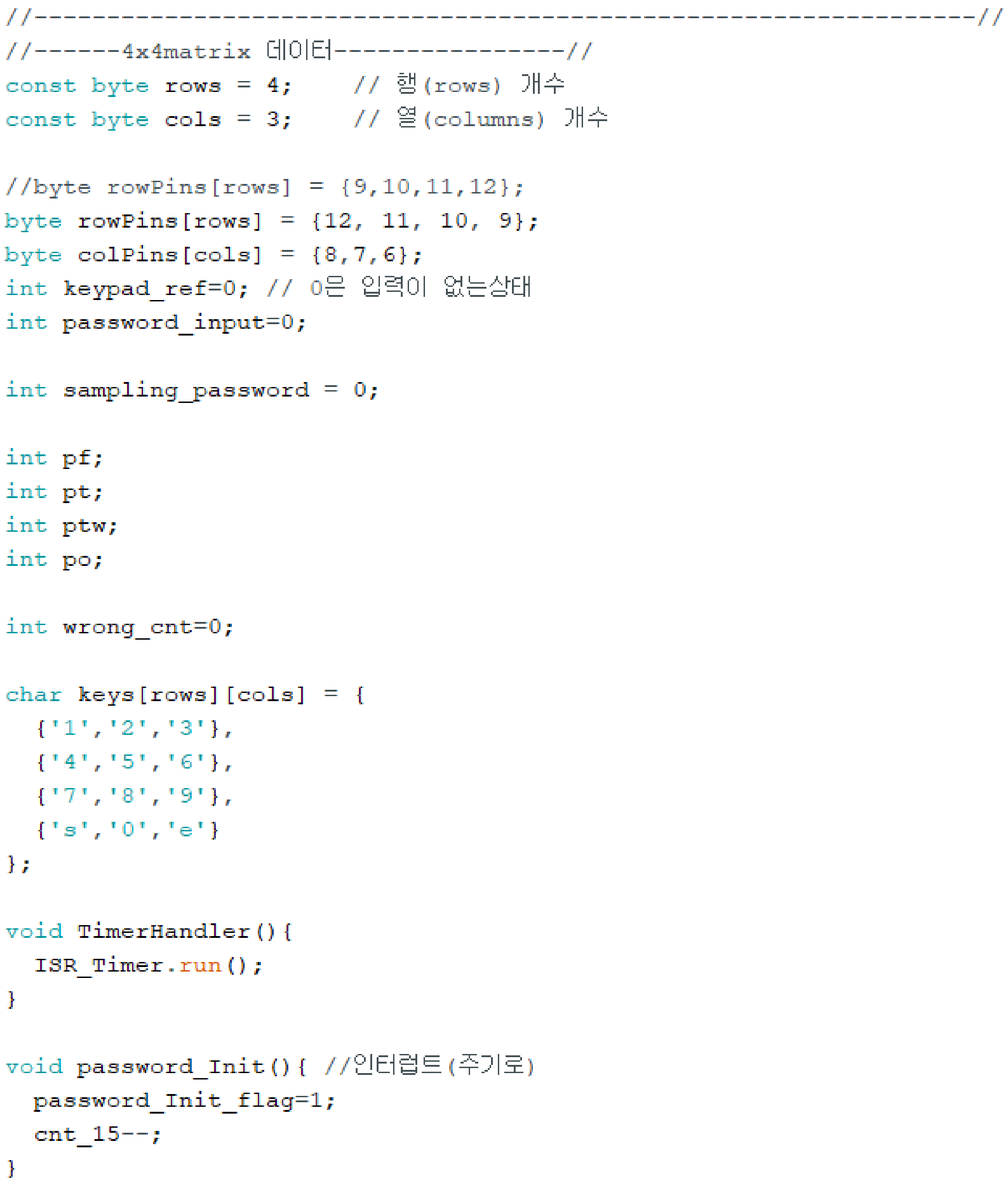
```
  if(state_door==1)
  {

    long duration, distance;

    digitalWrite(TRIG, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG, LOW);

    duration = pulseIn (ECHO, HIGH);
    distance = duration * 17 / 1000;
    delay(200);
    Serial.println(distance);
    if(distance >= 12)
    {
      debounce_sensor++;
      if(debounce_sensor>3)
      {
        debounce_sensor=0;
        state_door=0;

        for(pos = 90; pos >= 0; pos -= 1)
        {
          servo.write(pos);
          delay(23);
        }
      }
    }
    delay(700);
  }
}
```

<B board>
Arduino Code
(no BLE)

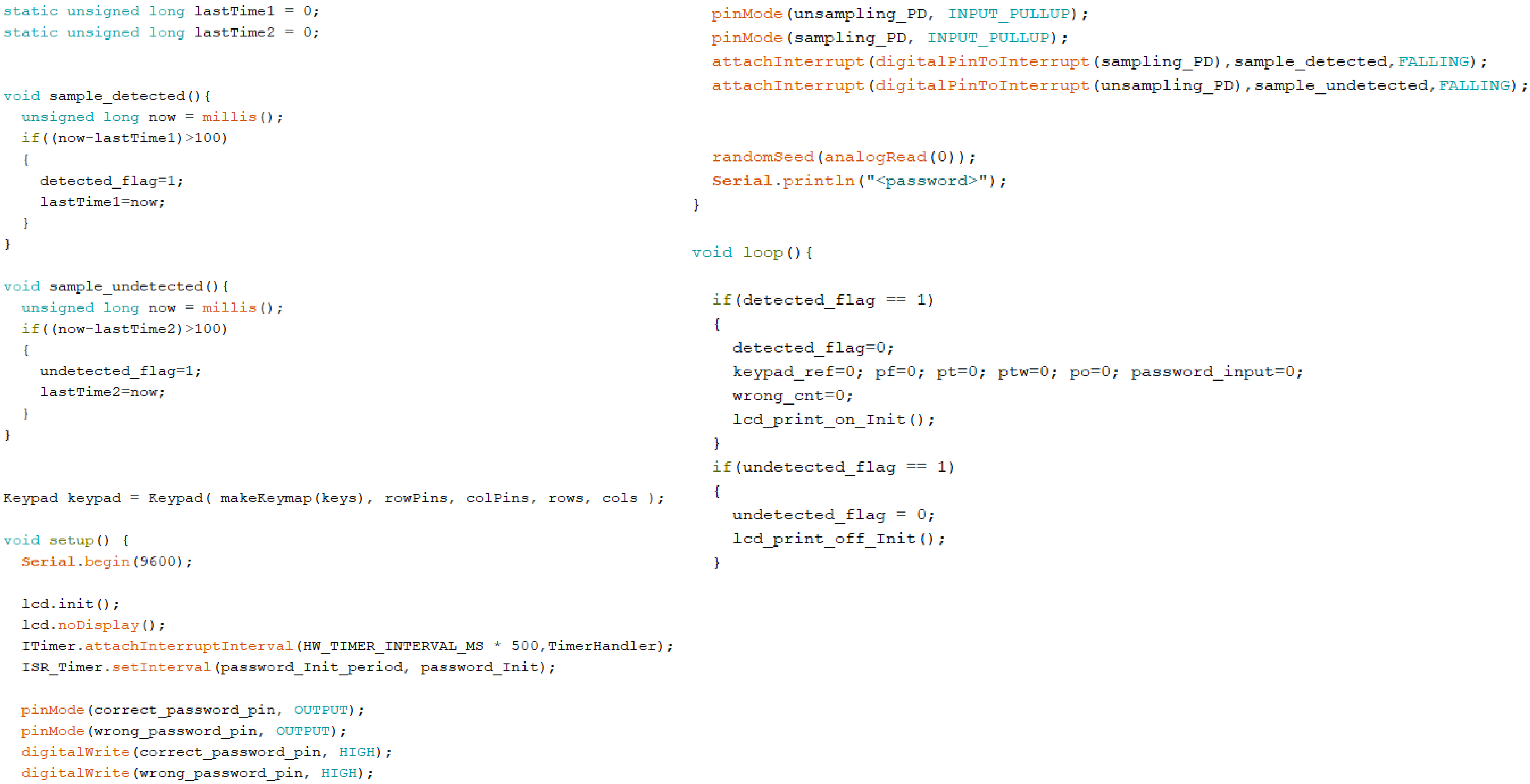
KU KONKUK UNIVERSITY



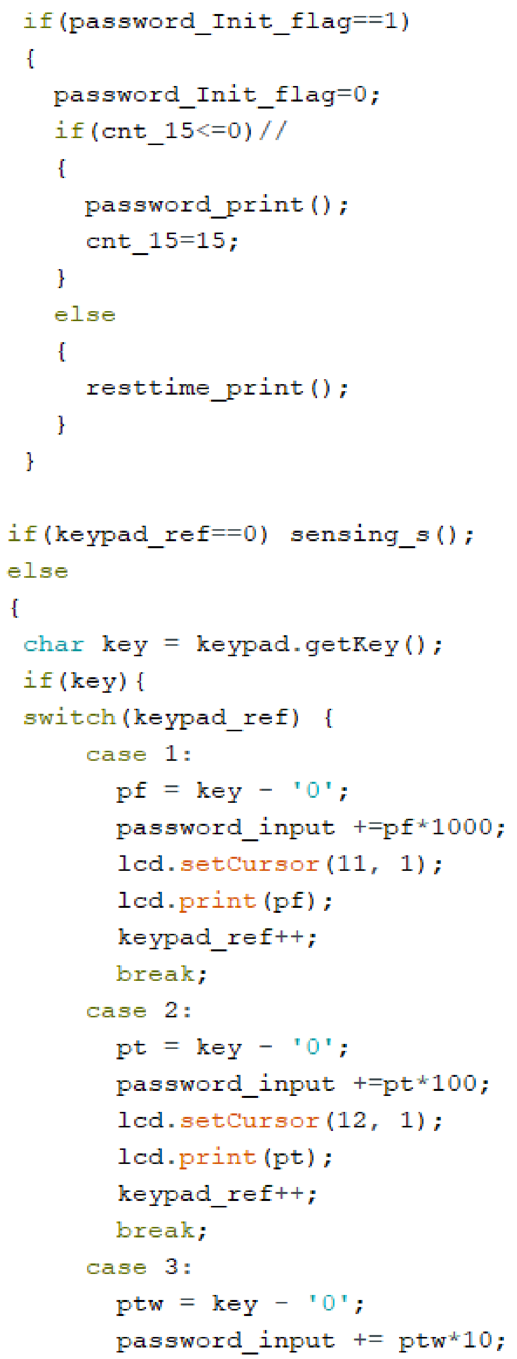
```
#include <NRF52_MBED_TimerInterrupt.h>
#include <NRF52_MBED_TimerInterrupt.hpp>
#include <NRF52_MBED_ISR_Timer.h>
#include <NRF52_MBED_ISR_Timer.hpp>

#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);


#include <Keypad.h>

NRF52_MBED_Timer ITimer(NRF_TIMER_3);
NRF52_MBED_ISRTimer ISR_Timer;

#define HW_TIMER_INTERVAL_MS  1
#define password_Init_period 1000

int sampling_PD = D2;
int unsampling_PD = D5;
int correct_password_pin = D3;
int wrong_password_pin = D4;

bool startflag=false;
bool inputflag = false;


//-----------------------------------------------------------------//
//------타이머인터럽트에 의해 주기를 가지고 초기화될 데이터----------------//
char buf_password[30];
long password; // 주기로 초기화될 비밀번호
volatile long password_Init_flag=0;
volatile int cnt_15=15;
//-----------------------------------------------------------------//
volatile int detected_flag=0;
volatile int undetected_flag=0;
```



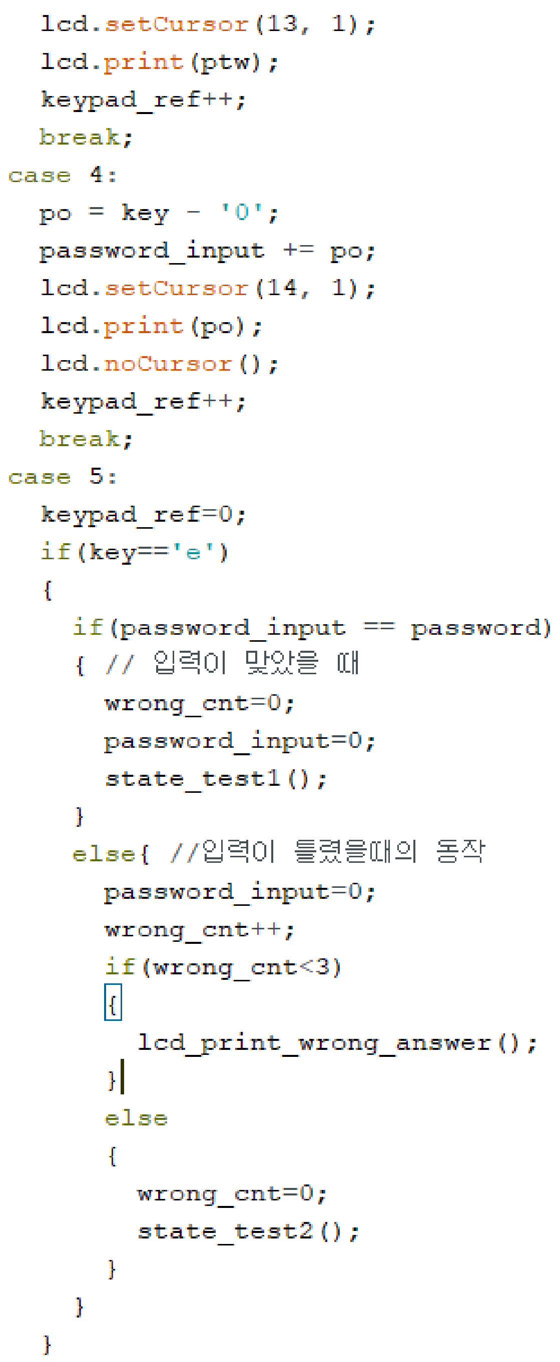
```
//-----------------------------------------------------------------//
//------4x4matrix 데이터----------------//
const byte rows = 4;    // 행(rows) 개수
const byte cols = 3;    // 열(columns) 개수

//byte rowPins[rows] = {9,10,11,12};
byte rowPins[rows] = {12, 11, 10, 9};
byte colPins[cols] = {8,7,6};
int keypad_ref=0; // 0은 입력이 없는상태
int password_input=0;

int sampling_password = 0;

int pf;
int pt;
int ptw;
int po;

int wrong_cnt=0;

char keys[rows][cols] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'s','0','e'}
};

void TimerHandler(){
  ISR_Timer.run();
}

void password_Init(){ //인터럽트(주기로)
  password_Init_flag=1;
  cnt_15--;
}
```



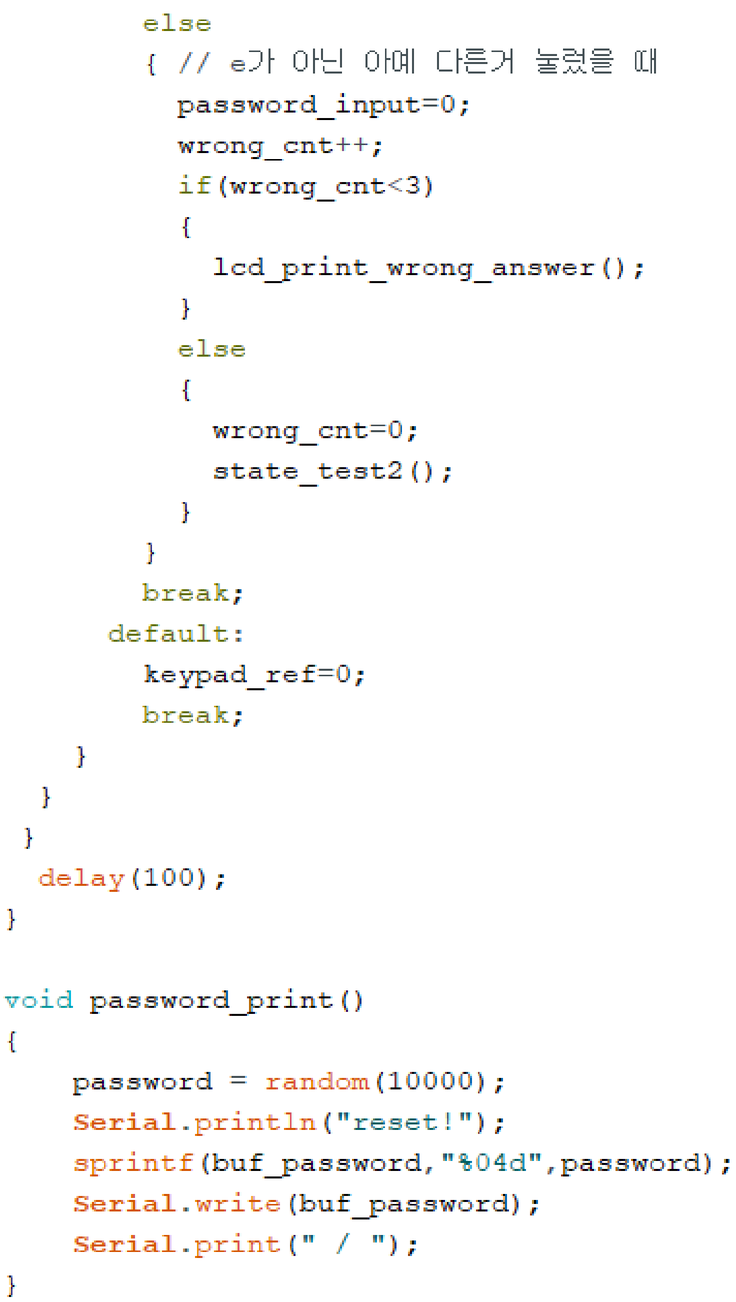
```
static unsigned long lastTime1 = 0;
static unsigned long lastTime2 = 0;


void sample_detected(){
  unsigned long now = millis();
  if((now-lastTime1)>100)
  {
    detected_flag=1;
    lastTime1=now;
  }
}

void sample_undetected(){
  unsigned long now = millis();
  if((now-lastTime2)>100)
  {
    undetected_flag=1;
    lastTime2=now;
  }
}


Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, rows, cols );

void setup() {
  Serial.begin(9600);

  lcd.init();
  lcd.noDisplay();
  ITimer.attachInterruptInterval(HW_TIMER_INTERVAL_MS * 500,TimerHandler);
  ISR_Timer.setInterval(password_Init_period, password_Init);

  pinMode(correct_password_pin, OUTPUT);
  pinMode(wrong_password_pin, OUTPUT);
  digitalWrite(correct_password_pin, HIGH);
  digitalWrite(wrong_password_pin, HIGH);
  pinMode(unsampling_PD, INPUT_PULLUP);
  pinMode(sampling_PD, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(sampling_PD),sample_detected,FALLING);
  attachInterrupt(digitalPinToInterrupt(unsampling_PD),sample_undetected,FALLING);


  randomSeed(analogRead(0));
  Serial.println("<password>");
}

void loop(){

  if(detected_flag == 1)
  {
    detected_flag=0;
    keypad_ref=0; pf=0; pt=0; ptw=0; po=0; password_input=0;
    wrong_cnt=0;
    lcd_print_on_Init();
  }
  if(undetected_flag == 1)
  {
    undetected_flag = 0;
    lcd_print_off_Init();
  }
```



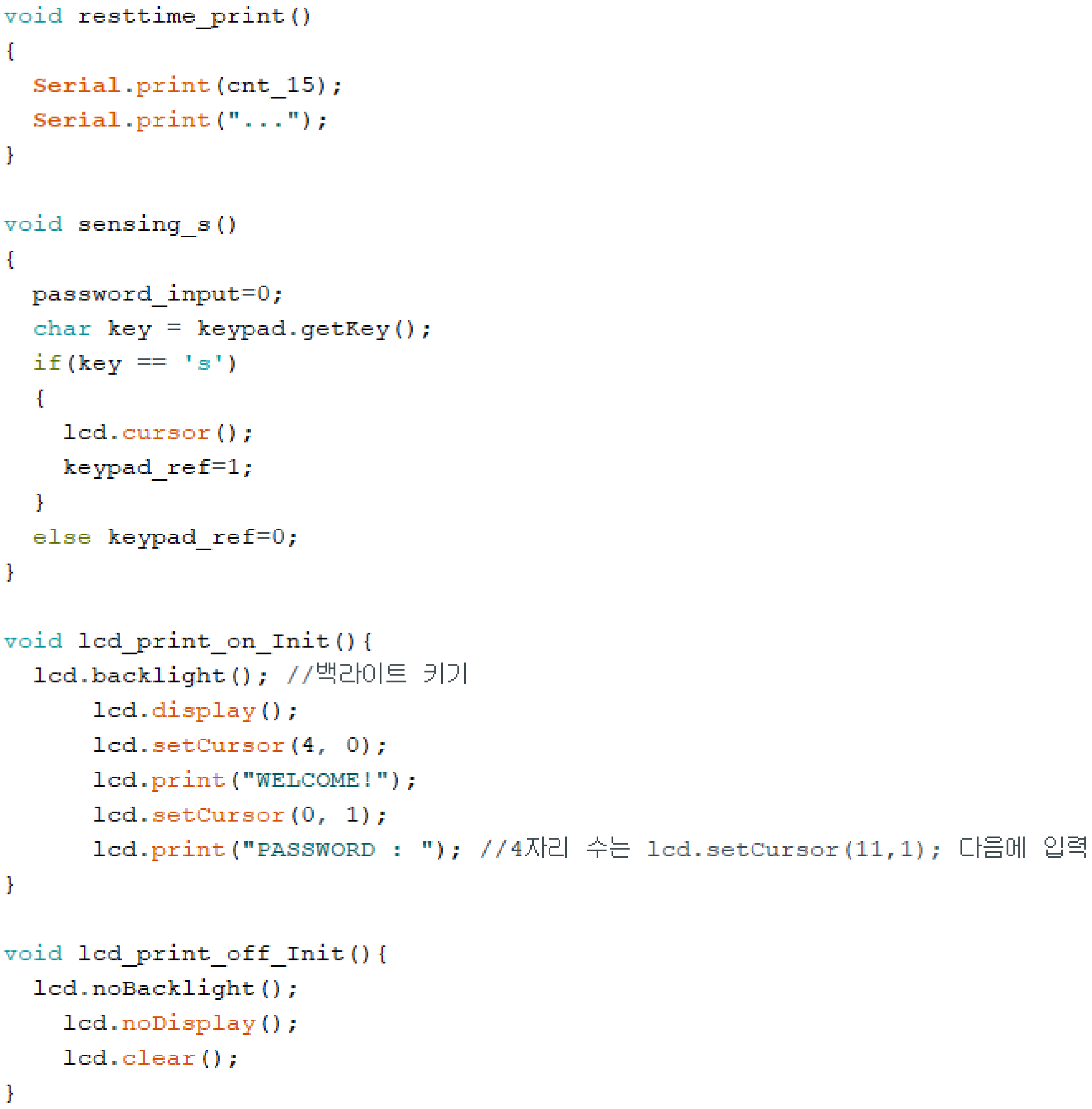
```
 if(password_Init_flag==1)
 {
   password_Init_flag=0;
   if(cnt_15<=0)//
   {
     password_print();
     cnt_15=15;
   }
   else
   {
     resttime_print();
   }
 }

if(keypad_ref==0) sensing_s();
else
{
 char key = keypad.getKey();
 if(key){
 switch(keypad_ref) {
     case 1:
       pf = key - '0';
       password_input +=pf*1000;
       lcd.setCursor(11, 1);
       lcd.print(pf);
       keypad_ref++;
       break;
     case 2:
       pt = key - '0';
       password_input +=pt*100;
       lcd.setCursor(12, 1);
       lcd.print(pt);
       keypad_ref++;
       break;
     case 3:
       ptw = key - '0';
       password_input += ptw*10;
```



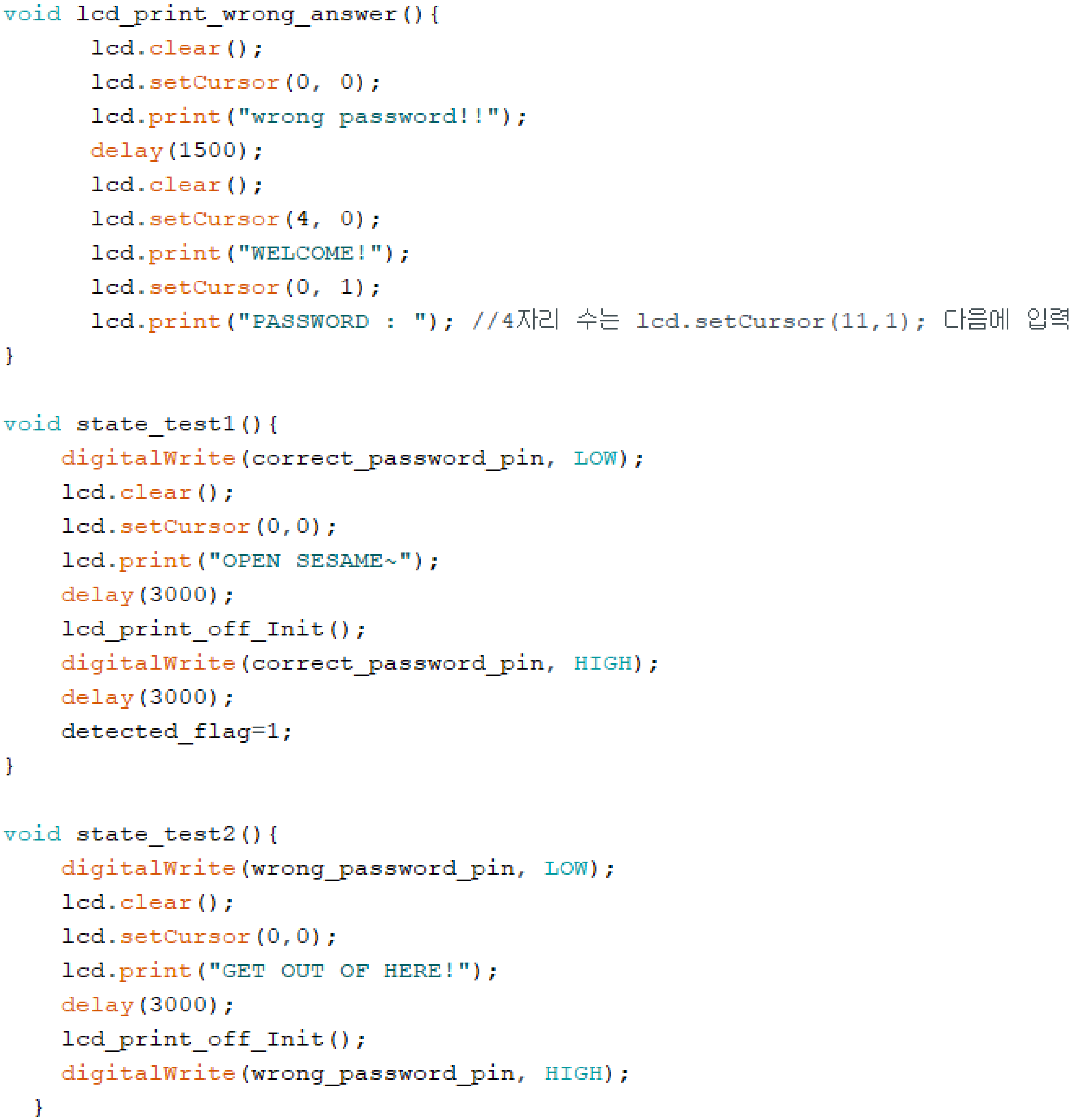
```
       lcd.setCursor(13, 1);
       lcd.print(ptw);
       keypad_ref++;
       break;
     case 4:
       po = key - '0';
       password_input += po;
       lcd.setCursor(14, 1);
       lcd.print(po);
       lcd.noCursor();
       keypad_ref++;
       break;
     case 5:
       keypad_ref=0;
       if(key=='e')
       {
         if(password_input == password)
         { // 입력이 맞았을 때
           wrong_cnt=0;
           password_input=0;
           state_test1();
         }
         else{ //입력이 틀렸을때의 동작
           password_input=0;
           wrong_cnt++;
           if(wrong_cnt<3)
           {
             lcd_print_wrong_answer();
           }
           else
           {
             wrong_cnt=0;
             state_test2();
           }
         }
       }
```



```
        else
        { // e가 아닌 아예 다른거 눌렀을 때
          password_input=0;
          wrong_cnt++;
          if(wrong_cnt<3)
          {
            lcd_print_wrong_answer();
          }
          else
          {
            wrong_cnt=0;
            state_test2();
          }
        }
        break;
      default:
        keypad_ref=0;
        break;
    }
  }
 }
  delay(100);
}

void password_print()
{
    password = random(10000);
    Serial.println("reset!");
    sprintf(buf_password,"%04d",password);
    Serial.write(buf_password);
    Serial.print(" / ");
}
```



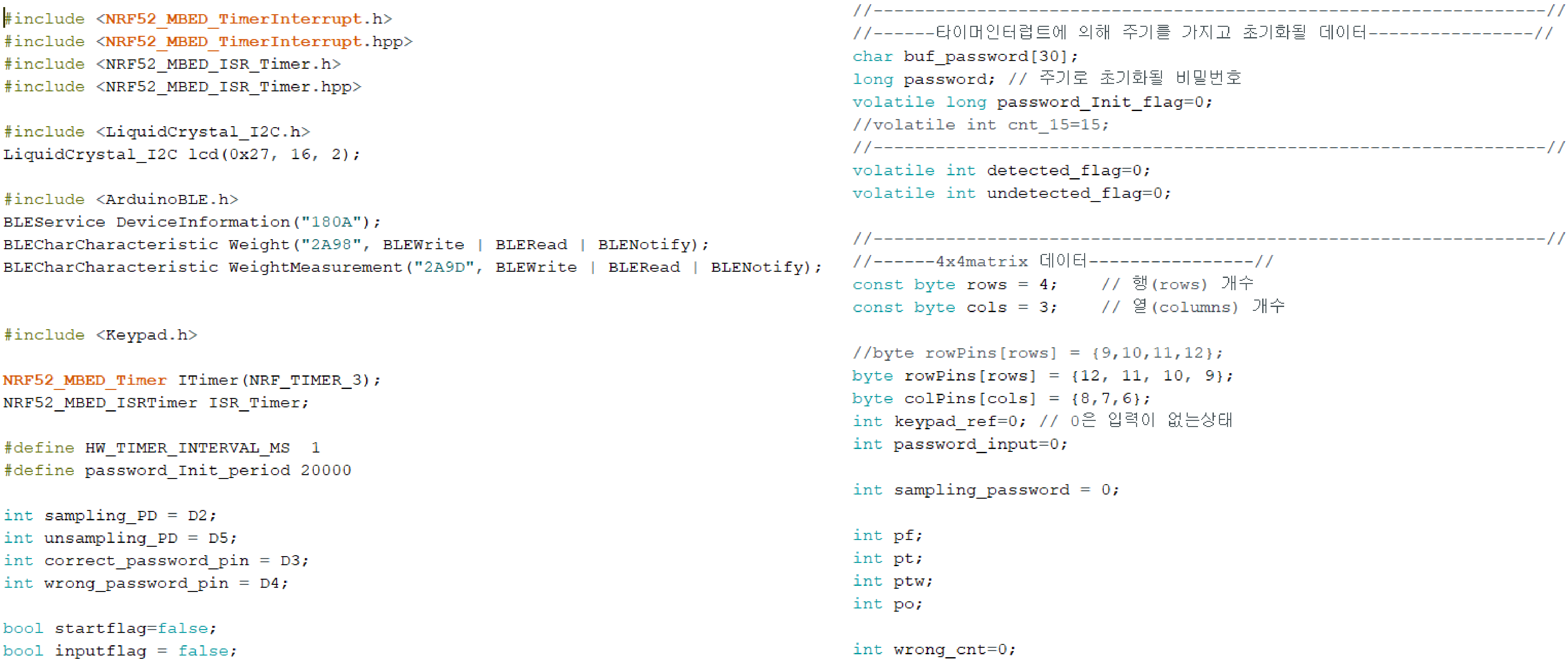
```
void resttime_print()
{
  Serial.print(cnt_15);
  Serial.print("...");
}

void sensing_s()
{
  password_input=0;
  char key = keypad.getKey();
  if(key == 's')
  {
    lcd.cursor();
    keypad_ref=1;
  }
  else keypad_ref=0;
}

void lcd_print_on_Init(){
  lcd.backlight(); //백라이트 키기
      lcd.display();
      lcd.setCursor(4, 0);
      lcd.print("WELCOME!");
      lcd.setCursor(0, 1);
      lcd.print("PASSWORD : "); //4자리 수는 lcd.setCursor(11,1); 다음에 입력
}

void lcd_print_off_Init(){
  lcd.noBacklight();
    lcd.noDisplay();
    lcd.clear();
}
```



```
void lcd_print_wrong_answer(){
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("wrong password!!");
      delay(1500);
      lcd.clear();
      lcd.setCursor(4, 0);
      lcd.print("WELCOME!");
      lcd.setCursor(0, 1);
      lcd.print("PASSWORD : "); //4자리 수는 lcd.setCursor(11,1); 다음에 입력
}

void state_test1(){
    digitalWrite(correct_password_pin, LOW);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("OPEN SESAME~");
    delay(3000);
    lcd_print_off_Init();
    digitalWrite(correct_password_pin, HIGH);
    delay(3000);
    detected_flag=1;
}

void state_test2(){
    digitalWrite(wrong_password_pin, LOW);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("GET OUT OF HERE!");
    delay(3000);
    lcd_print_off_Init();
    digitalWrite(wrong_password_pin, HIGH);
  }
```

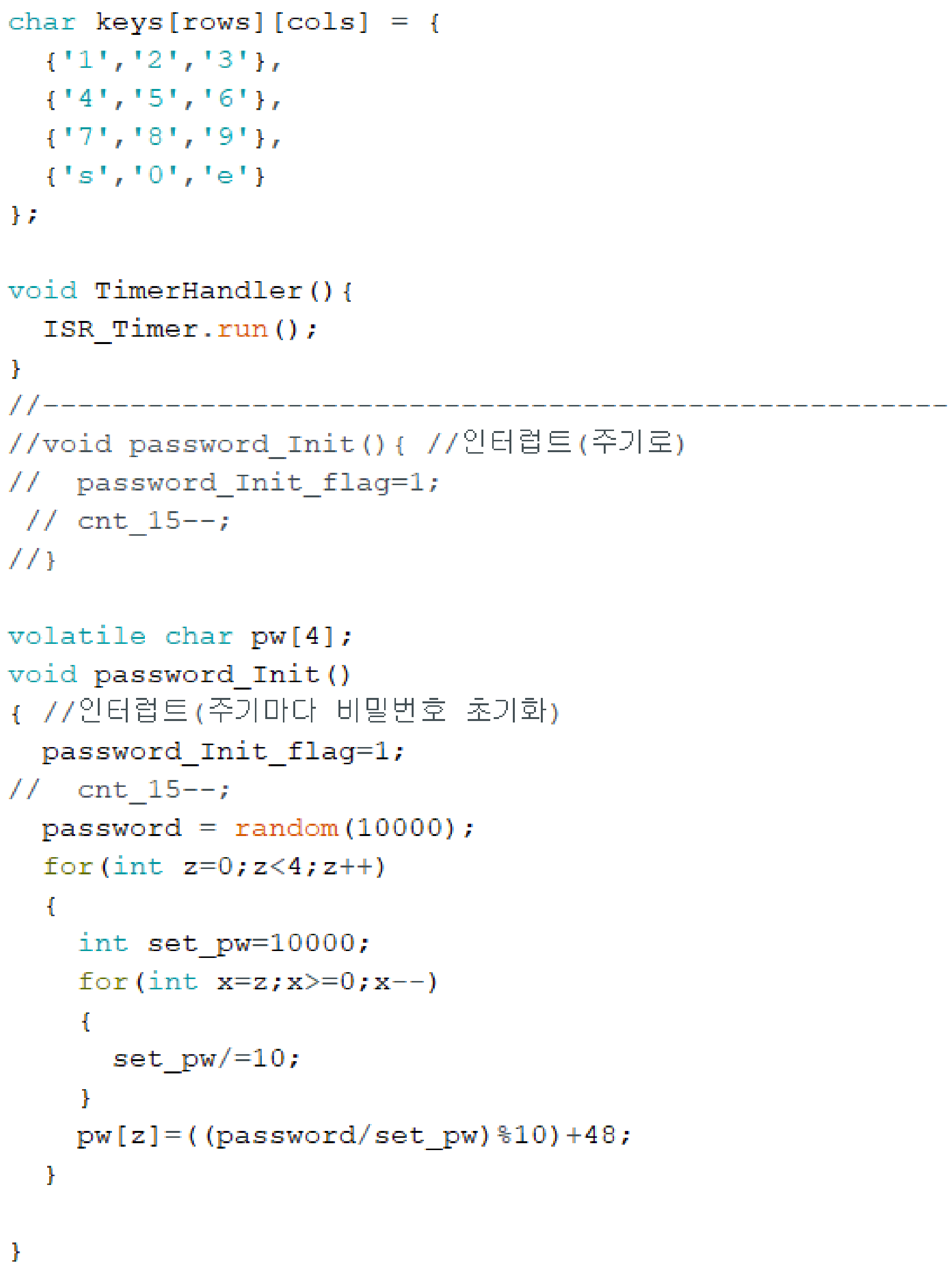
<B board>
Arduino Code
(with BLE)

KU KONKUK UNIVERSITY

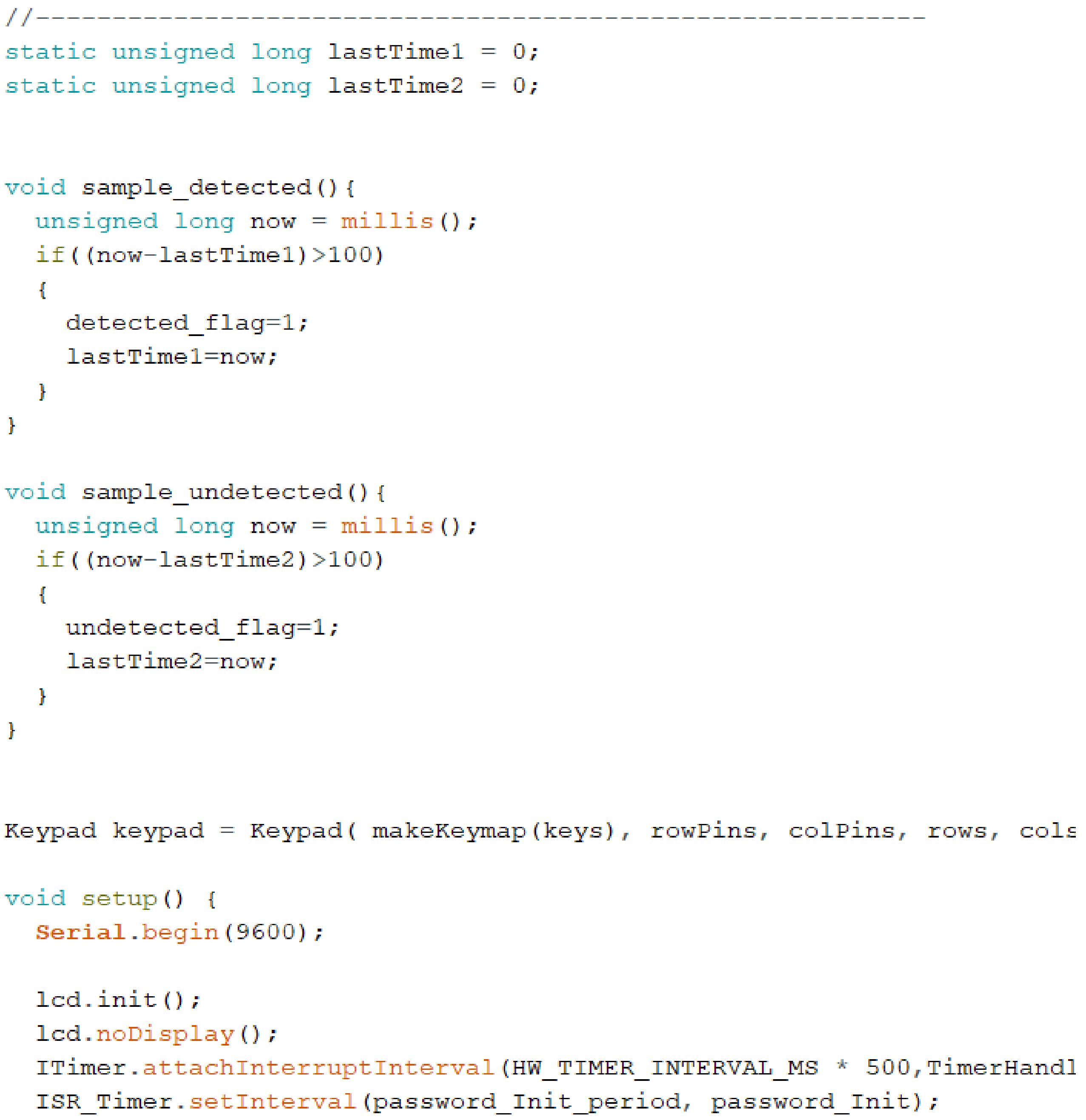


```
#include <NRF52_MBED_TimerInterrupt.h>
#include <NRF52_MBED_TimerInterrupt.hpp>
#include <NRF52_MBED_ISR_Timer.h>
#include <NRF52_MBED_ISR_Timer.hpp>

#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

#include <ArduinoBLE.h>
BLEService DeviceInformation("180A");
BLECharCharacteristic Weight("2A98", BLEWrite | BLERead | BLENotify);
BLECharCharacteristic WeightMeasurement("2A9D", BLEWrite | BLERead | BLENotify);


#include <Keypad.h>

NRF52_MBED_Timer ITimer(NRF_TIMER_3);
NRF52_MBED_ISRTimer ISR_Timer;

#define HW_TIMER_INTERVAL_MS  1
#define password_Init_period 20000

int sampling_PD = D2;
int unsampling_PD = D5;
int correct_password_pin = D3;
int wrong_password_pin = D4;

bool startflag=false;
bool inputflag = false;
```

```
//--------------------------------------------------------------------//
//------타이머인터럽트에 의해 주기를 가지고 초기화될 데이터---------------//
char buf_password[30];
long password; // 주기로 초기화될 비밀번호
volatile long password_Init_flag=0;
//volatile int cnt_15=15;
//--------------------------------------------------------------------//
volatile int detected_flag=0;
volatile int undetected_flag=0;

//--------------------------------------------------------------------//
//------4x4matrix 데이터---------------//
const byte rows = 4;    // 행(rows) 개수
const byte cols = 3;    // 열(columns) 개수

//byte rowPins[rows] = {9,10,11,12};
byte rowPins[rows] = {12, 11, 10, 9};
byte colPins[cols] = {8,7,6};
int keypad_ref=0; // 0은 입력이 없는상태
int password_input=0;

int sampling_password = 0;

int pf;
int pt;
int ptw;
int po;

int wrong_cnt=0;
```

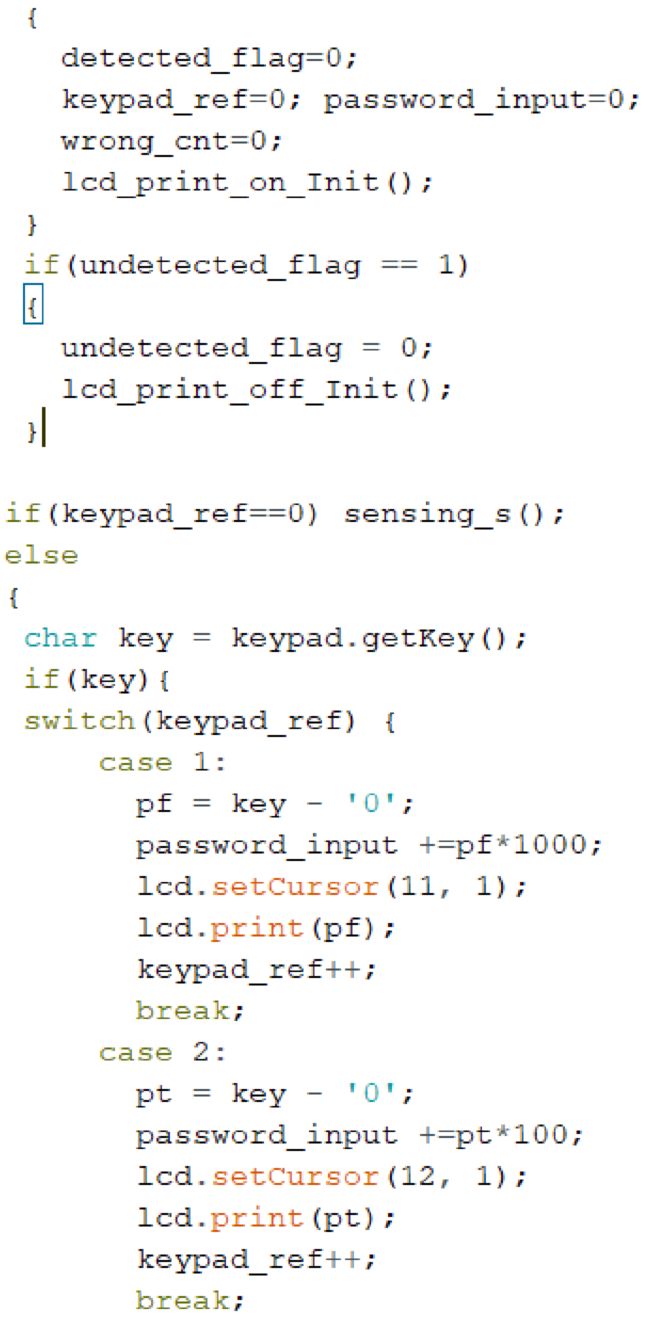


```
char keys[rows][cols] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'s','0','e'}
};

void TimerHandler(){
  ISR_Timer.run();
}
//----------------------------------------------------
//void password_Init(){ //인터럽트(주기로)
//  password_Init_flag=1;
 // cnt_15--;
//}

volatile char pw[4];
void password_Init()
{ //인터럽트(주기마다 비밀번호 초기화)
  password_Init_flag=1;
//  cnt_15--;
  password = random(10000);
  for(int z=0;z<4;z++)
  {
    int set_pw=10000;
    for(int x=z;x>=0;x--)
    {
      set_pw/=10;
    }
    pw[z]=((password/set_pw)%10)+48;
  }

}
```

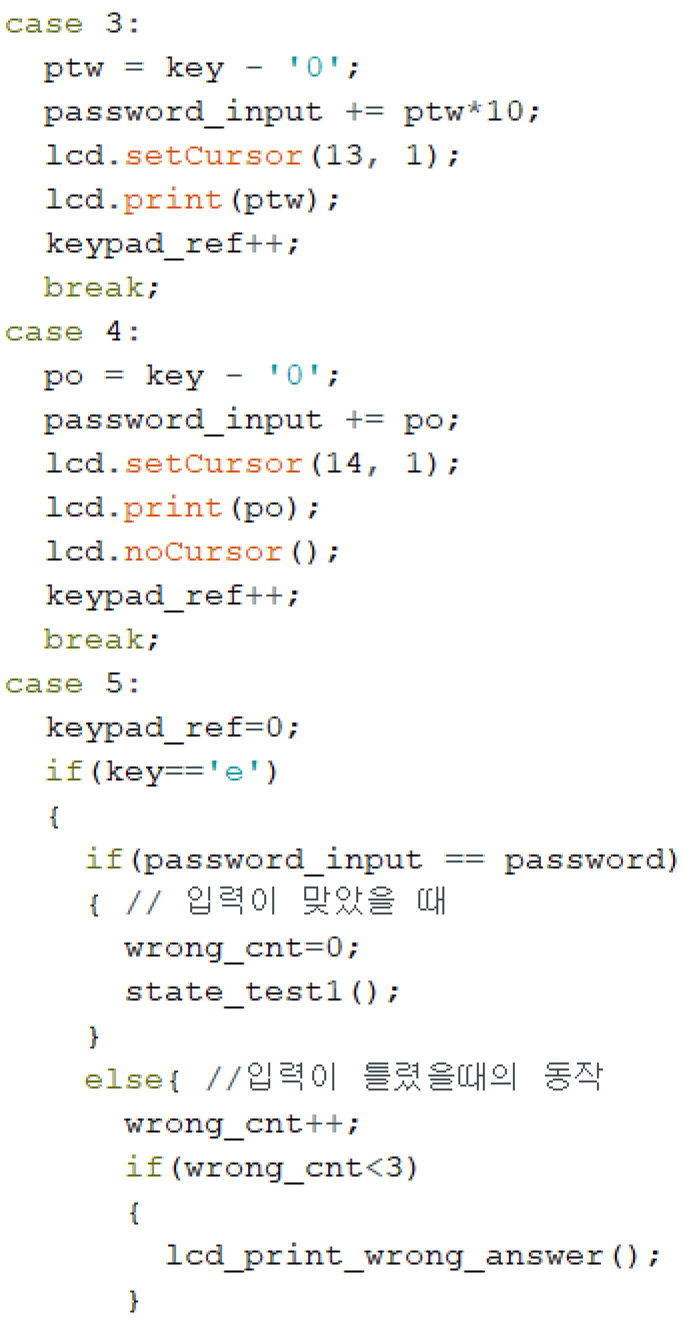


```
//------------------------------------------------------------
static unsigned long lastTime1 = 0;
static unsigned long lastTime2 = 0;


void sample_detected(){
  unsigned long now = millis();
  if((now-lastTime1)>100)
  {
    detected_flag=1;
    lastTime1=now;
  }
}

void sample_undetected(){
  unsigned long now = millis();
  if((now-lastTime2)>100)
  {
    undetected_flag=1;
    lastTime2=now;
  }
}


Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, rows, cols

void setup() {
  Serial.begin(9600);

  lcd.init();
  lcd.noDisplay();
  ITimer.attachInterruptInterval(HW_TIMER_INTERVAL_MS * 500,TimerHandl
  ISR_Timer.setInterval(password_Init_period, password_Init);
  pinMode(correct_password_pin, OUTPUT);
  pinMode(wrong_password_pin, OUTPUT);
  digitalWrite(correct_password_pin, HIGH);
  digitalWrite(wrong_password_pin, HIGH);


  pinMode(unsampling_PD, INPUT_PULLUP);
  pinMode(sampling_PD, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(sampling_PD),sample_detected,FALLING);
  attachInterrupt(digitalPinToInterrupt(unsampling_PD),sample_undetected,FALLING);


  randomSeed(analogRead(0));
//  Serial.println("<password>");

  if (!BLE.begin()) {
    Serial.println("starting BLE failed!");
    while (1);
  }
  BLE.setLocalName("BLE chat machine");

  BLE.setAdvertisedService(DeviceInformation);
  DeviceInformation.addCharacteristic(Weight);
  DeviceInformation.addCharacteristic(WeightMeasurement);
  BLE.addService(DeviceInformation);

  Weight.writeValue(NULL);
  WeightMeasurement.writeValue(NULL);

  BLE.advertise();
  Serial.println("BLE LED Peripheral");
}
int v=0;
void loop(){
```

```
BLEDevice central = BLE.central();

if (central)
{
  Serial.print("Connected to central: ");
  Serial.println(central.address());

  while (central.connected())
  {
    if(password_Init_flag==1)
      {
        if(v==0)
        {
          Serial.print("password : ");
          Serial.println(password);
        }
         char val1=pw[v++];
         Weight.writeValue(val1);
         delay(2000);
      }
    if(v==4) {password_Init_flag=0; v=0;}

    char val2 = WeightMeasurement.value();
    if (val2 != NULL)
    {
      Serial.println(val2);
      WeightMeasurement.writeValue(NULL);
    }

    if(detected_flag == 1)
```



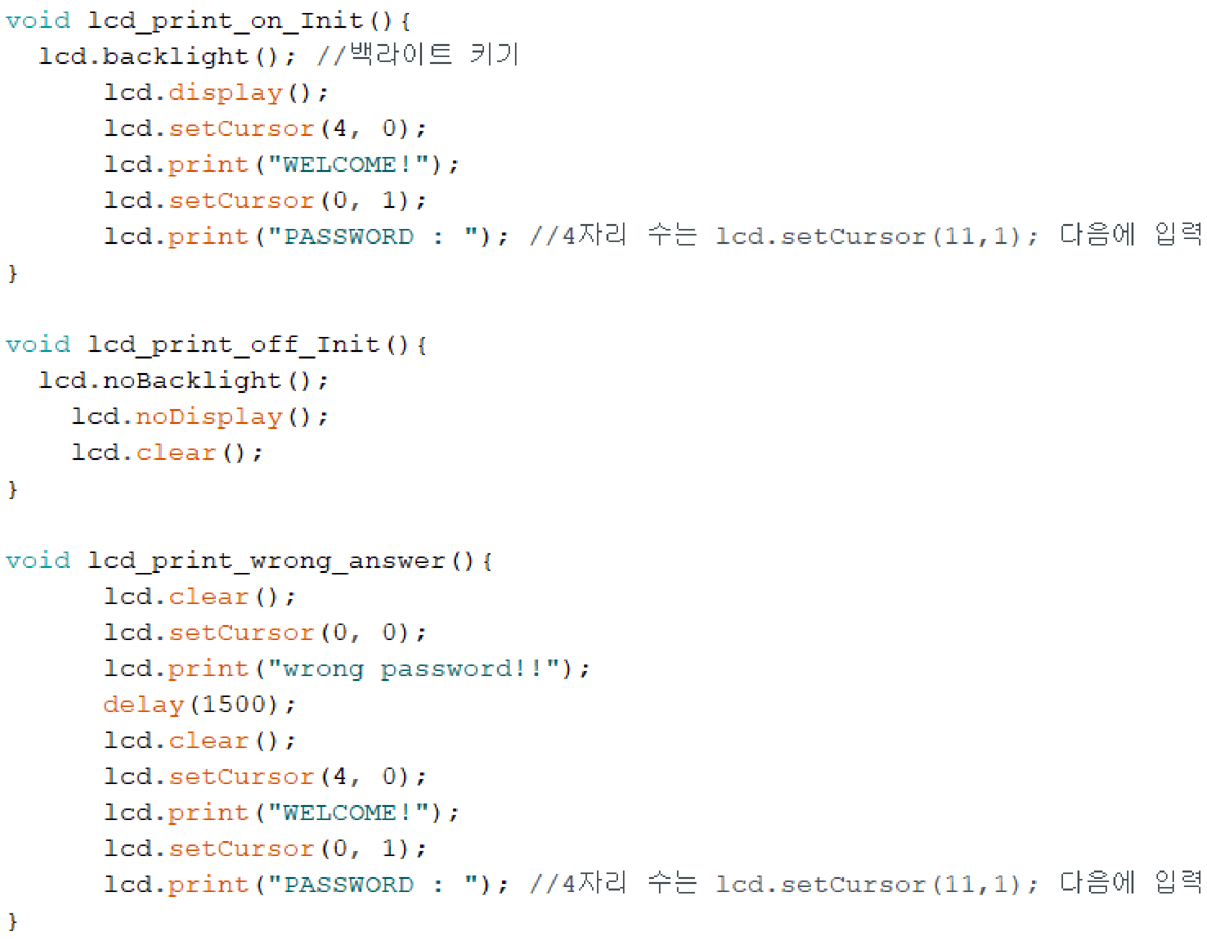
```
 {
   detected_flag=0;
   keypad_ref=0; password_input=0;
   wrong_cnt=0;
   lcd_print_on_Init();
 }
 if(undetected_flag == 1)
 {
   undetected_flag = 0;
   lcd_print_off_Init();
 }

if(keypad_ref==0) sensing_s();
else
{
 char key = keypad.getKey();
 if(key){
 switch(keypad_ref) {
     case 1:
       pf = key - '0';
       password_input +=pf*1000;
       lcd.setCursor(11, 1);
       lcd.print(pf);
       keypad_ref++;
       break;
     case 2:
       pt = key - '0';
       password_input +=pt*100;
       lcd.setCursor(12, 1);
       lcd.print(pt);
       keypad_ref++;
       break;
```



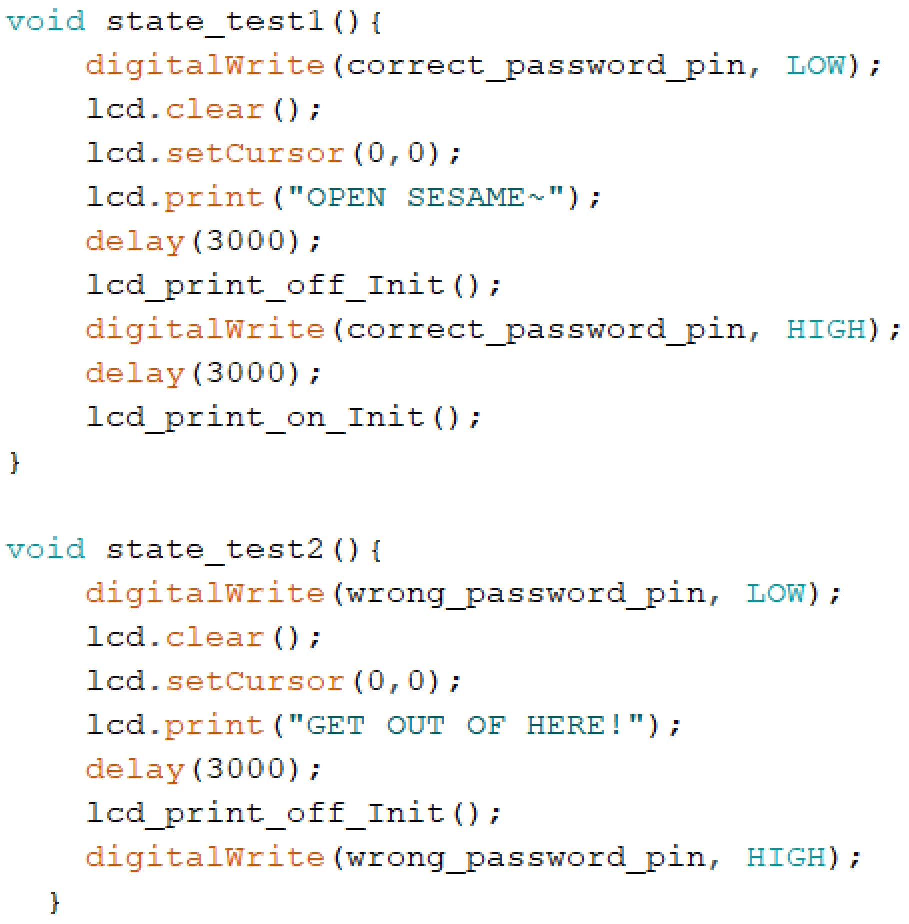
```
case 3:
  ptw = key - '0';
  password_input += ptw*10;
  lcd.setCursor(13, 1);
  lcd.print(ptw);
  keypad_ref++;
  break;
case 4:
  po = key - '0';
  password_input += po;
  lcd.setCursor(14, 1);
  lcd.print(po);
  lcd.noCursor();
  keypad_ref++;
  break;
case 5:
  keypad_ref=0;
  if(key=='e')
  {
    if(password_input == password)
    { // 입력이 맞았을 때
      wrong_cnt=0;
      state_test1();
    }
    else{ //입력이 틀렸을때의 동작
      wrong_cnt++;
      if(wrong_cnt<3)
      {
        lcd_print_wrong_answer();
      }
```

```
           else
           {
             wrong_cnt=0;
             state_test2();
           }
         }
         password_input=0;
       }
       else
       { // e가 아닌 아예 다른거 눌렀을 때
         password_input=0;
         wrong_cnt++;
         if(wrong_cnt<3)
         {
           lcd_print_wrong_answer();
         }
         else
         {
           wrong_cnt=0;
           state_test2();
         }
       }
       break;
     default:
       keypad_ref=0;
       break;
   }
 }
}
 delay(100);


   }
```

```
    Serial.print(F("Disconnected from central: "));
      Serial.println(central.address());
  }
}

void password_print()
{
    password = random(10000);
    Serial.println("reset!");
    sprintf(buf_password,"%04d",password);
    Serial.write(buf_password);
    Serial.print(" / ");
}

//void resttime_print()
//{
 // Serial.print(cnt_15);
 // Serial.print("...");
//}

void sensing_s()
{
  password_input=0;
  char key = keypad.getKey();
  if(key == 's')
  {
    lcd.cursor();
    keypad_ref=1;
  }
  else keypad_ref=0;
}
```

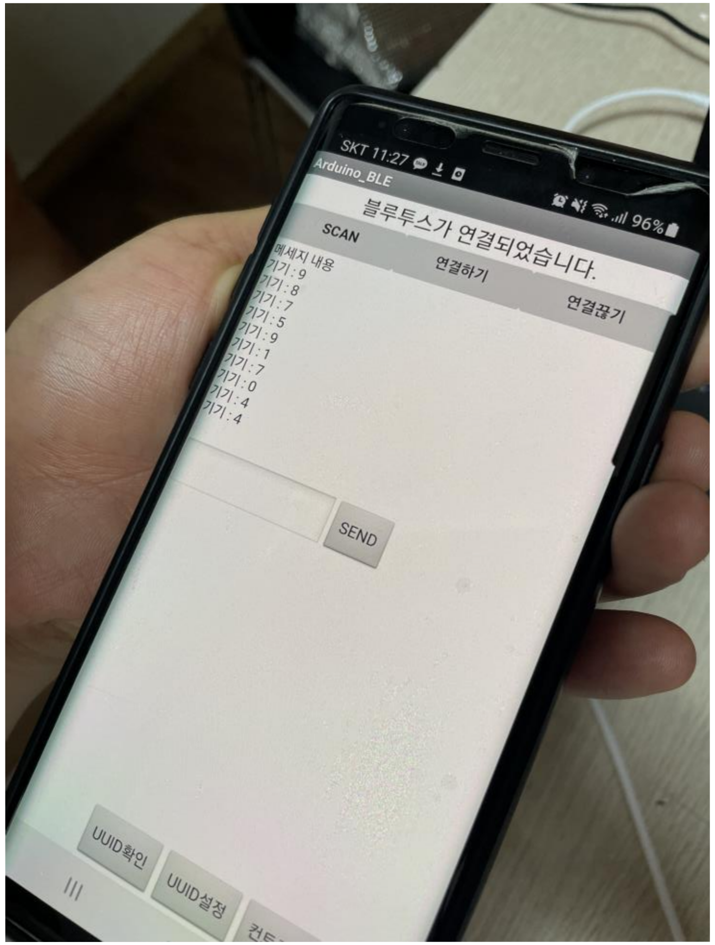


```
void lcd_print_on_Init(){
  lcd.backlight(); //백라이트 키기
      lcd.display();
      lcd.setCursor(4, 0);
      lcd.print("WELCOME!");
      lcd.setCursor(0, 1);
      lcd.print("PASSWORD : "); //4자리 수는 lcd.setCursor(11,1); 다음에 입력
}

void lcd_print_off_Init(){
  lcd.noBacklight();
    lcd.noDisplay();
    lcd.clear();
}

void lcd_print_wrong_answer(){
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("wrong password!!");
      delay(1500);
      lcd.clear();
      lcd.setCursor(4, 0);
      lcd.print("WELCOME!");
      lcd.setCursor(0, 1);
      lcd.print("PASSWORD : "); //4자리 수는 lcd.setCursor(11,1); 다음에 입력
}
```



```
void state_test1(){
    digitalWrite(correct_password_pin, LOW);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("OPEN SESAME~");
    delay(3000);
    lcd_print_off_Init();
    digitalWrite(correct_password_pin, HIGH);
    delay(3000);
    lcd_print_on_Init();
}

void state_test2(){
    digitalWrite(wrong_password_pin, LOW);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("GET OUT OF HERE!");
    delay(3000);
    lcd_print_off_Init();
    digitalWrite(wrong_password_pin, HIGH);
  }
```

BLE (additional function)



블루투스가 연결되었습니다.

```
메세지 내용
기기 : 9
기기 : 8
기기 : 7
기기 : 5
기기 : 9
기기 : 1
기기 : 7
기기 : 0
기기 : 4
기기 : 4
```

- Bluetooth 통신을 통해, 아두이노에서 스마트폰으로 비밀번호를 전송
- Arduino BLE tool app 활용