

창의설계프로젝트 결과보고서

(졸업실험실습 보고서)

LTE Cat.M1 통신 기반 홈 IoT 시스템

2022년 12월

금오공과대학교 전자통신전공
김정인, 조은비, 김근호, 오창윤, 안정은

지도교수 : 전일수

LTE Cat.M1 통신 기반 홈 IoT 시스템

김정인(1)[†]·조은비(2)^{*}·김근호(2)^{*}·오창윤(2)^{*}·안정은(3)^{*}
전일수^{**}

Implementation of LTE Cat.M1 based IoT Home Automation system

Jeong-in Kim, Eunbi Cho, Geun-ho Kim, Chang-yoon Oh, Jeong-eun An
Ilsoo Jeon

Abstract

The rise of popularity of the Internet of Things (IoT) enabled growth of home automation system. Home automation system means monitoring and controlling of home appliances remotely using the concept of IoT. This paper proposes a door lock system based on face recognition technology. The proposed system will automatically give access to the door only if the system recognizes the face of the person trying to access the door. Furthermore, we presents a low-cost IoT prototype for automated LED lighting system based on sensors and LPWAN.

요약

홈 오토메이션(Home automation)은 집 외부에서 내부의 각종 기기를 조절하는 기능 등 집안의 필요기능을 자동으로 제어하는 기술이다. 본 연구에서는 홈 IoT 시스템 구축을 위해 현관문 도어락 자동화와 LED 디밍 제어 시스템의 프로토타입을 제안한다. 제안된 도어락 자동화 시스템에서는 얼굴인식 기능과 안드로이드 앱을 통한 사용자 인터페이스를 제공하였다. 또한, LED 조명 제어 시스템에서는 저전력, 저비용 무선통신 기술로 각광받고 있는 LTE Cat.M1 무선통신 기술을 접목하여 에너지 효율을 극대화할 수 있는 시스템을 제안하였다.

핵심주제어 : Face Recognition, Android app programming, Internet of Things, LTE Cat.M1, LPWAN

1. 작품과제 필요성

본 연구에서는 IoT 홈 자동화 시스템에서 카메라와 센서에 의해 수집한 데이터를 분석하여 홈의 외부, 내부 상태를 파악하고 판단된 결과에 따라 필수 가전제품을 제어하여 사용자가 편리함을 느낄 수 있는 맞춤형 홈 자동화 알고리즘을 제안한다. 최근에는 IoT 기술의 발전이 고도화됨에 따라 사용자의 요구사항을 직접 반영하고 저비용의 제품을 사용하려는 욕구가 더욱 상승하였으며 오픈소스를 통한 IoT 홈을 구축하는 연구가 증가하였다. 이에 따라 본 논문에서 제안하는 최종 시스템은 게이트웨이에 연결된 각종 센서의 이미지 데이터, 조도 데이터 값을 수집하고 수집된 데이터 값을 분석하여 외부 도어락 제어나 LED 점등을 자동

으로 해주는 IoT 홈 자동화 시스템을 목표로 한다.

2. 작품과제 해결 방안 및 과정

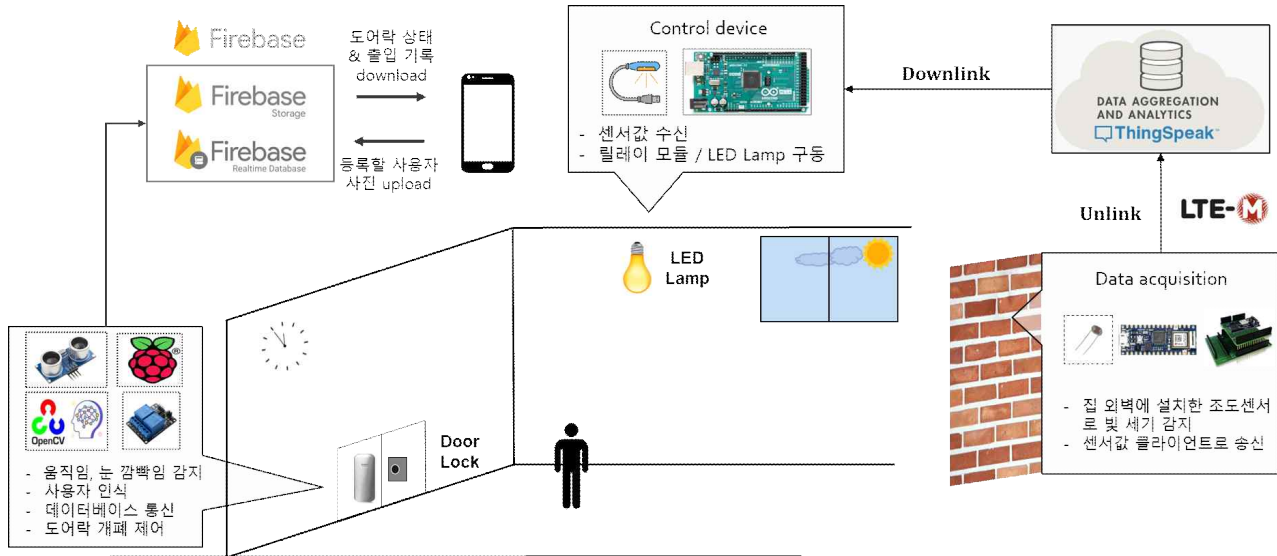


그림 1 전체적인 시스템 구조: 도어락 자동 제어 시스템, LED 점등 자동 제어 시스템

본 프로젝트에서는 4차 산업혁명의 핵심기술 중 하나인 IoT 기술을 통해 단일 보드 컴퓨터를 이용한 얼굴 인식 기반 도어락과 저전력 장거리 네트워크를 기반으로 통신하는 자동 전등 제어 시스템을 구축하여 더욱 높은 실용성과 보안성을 갖춘 홈 자동화 서비스를 개발한다. 본 작품은 크게 1) 얼굴인식부와 2) LED 디밍 제어로 나누어지며 서로 다른 Micro Controller Unit (MCU)와 다양한 센서들을 활용하여 홈 자동화 시스템을 구축한다.

2.1.1. 얼굴인식 도어락

제안하는 얼굴인식 도어락 시스템은 Raspberry Pi에 초음파센서를 연결하여 움직임을 감지하면 카메라를 실행시켜 얼굴인식 기능을 활성화한다. 얼굴인식은 오픈소스 라이브러리인 dlib을 사용해 감지된 얼굴을 벡터로 인코딩하여 Euclidean 거리를 계산함으로써 얼굴을 인식한다. 이때 눈 깜빡임 여부를 점검하여 실제 사람인지를 판단한다. 또한, Firebase 데이터베이스와 앱을 연동하여 앱에서 편리하게 사용자를 등록 및 확인하고 방문자 목록을 확인할 수 있다. 앱을 통해 촬영한 사진은 Firebase storage로 업로드하여 사용자 등록이 가능하다. Firebase에서는 출입을 시도한 사람의 방문 시간 및 사진을 realtime database에 저장하여 관리할 수 있다. 앱에서는 얼굴인식뿐만 아니라 지문인식 기능을 통해 보안성 향상을 꾀하였으며, 원격으로 도어락을 제어하고 사용자 목록 확인이 가능하게 하였다.

2.2. LED 디밍 제어

자동 전등 제어부는 조도센서를 통해 수집한 데이터를 ThingSpeak 이라는 웹서버로 업링크(Uplink)하여 관리자가 모니터링하고, 제어부 단말기에서 서버에 업로드된 센서값을 수신하여 Threshold 값을 바

탕으로 기기를 제어하는 시스템이다. 마이크로컨트롤러 유닛(MicroController Unit; MCU)으로 활용된 Arduino Nano 33 IoT는 외부 환경의 밝기를 측정할 조도센서 및 LTE Cat.m1 통신모듈과 연결된다. 제어부인 LED는 서버로부터 받아온 센서값의 정보를 바탕으로 LED 램프의 작동 및 밝기를 제어한다. 조도 센서는 집 외벽에 설치되어 실시간으로 센서 데이터를 스트리밍하여 밝기를 측정하였다. 수집한 센서 데이터는 LTE Cat.M1 통신모듈을 사용하여 HTTP 통신을 통해 클라이언트로 송신된다. 조도 센서값이 특정 Threshold 값 이하로 내려가면(어두워지면) 릴레이 모듈을 구동시켜 자동으로 LED 램프가 켜진다.

3. 개념설계 및 상세설계

3.1. 도어락 자동 제어 시스템

3.1.1. 하드웨어 구성

본 논문에서 제안하는 얼굴인식 도어락 시스템의 하드웨어는 Raspberry Pi 4, Logitech Web cam, 초음파 센서, 도어락, 릴레이 모듈, 안드로이드 휴대폰으로 구성된다.

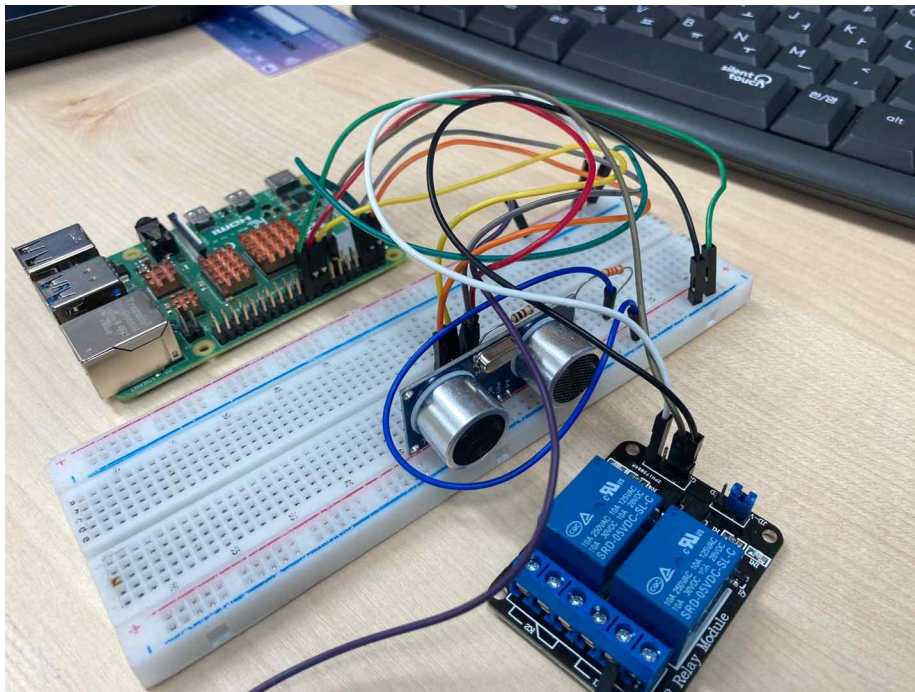


그림 2 Raspberry Pi, 초음파센서, 릴레이 모듈 연결 모습

Raspberry Pi는 초소형 컴퓨터로 얼굴인식 도어락의 핵심적인 역할을 담당한다. 앱에서 사용자가 등록을 요청하면 Firebase에서 해당 사용자의 사진을 가져와 얼굴을 align한 후 얼굴 데이터를 추출하여 Raspberry Pi 로컬 폴더에 저장하고 출입 시도가 있을 시 Logitech web cam을 통해 얻은 실시간 영상 속 얼굴과 로컬 폴더에 저장된 얼굴 데이터를 비교하여 도어락을 제어한다. 초음파 센서는 움직임 여부를 파악하는 역할로서 특정 물체나 사람이 도어락에 다가오면 Raspberry Pi 카메라를 실행시키고 일정 시간 동안 출입 시도가 없다면

Raspberry Pi 카메라를 종료시키고 다시 움직임을 감지한다. 도어락은 개폐 회로 부분을 납땜하여 릴레이 모듈에 연결하여 제어된다. 안드로이드 휴대폰은 앱을 이용하여 지문인식을 통한 도어락 제어, 사용자 등록 및 확인, 출입 기록에 대한 알림을 받고 목록을 확인할 수 있다.



그림 3 도어락 납땜

3.1.2. 소프트웨어 구성

본 시스템의 소프트웨어는 크게 얼굴인식과 사용자 인터페이스 부분으로 나눌 수 있다. 사용자 인터페이스는 안드로이드 앱을 통해 제공하였으며 앱 기능은 상세하게 지문인식, 회원가입 및 로그인, 사용자 등록, 출입 기록 알림 및 조회로 구성된다.

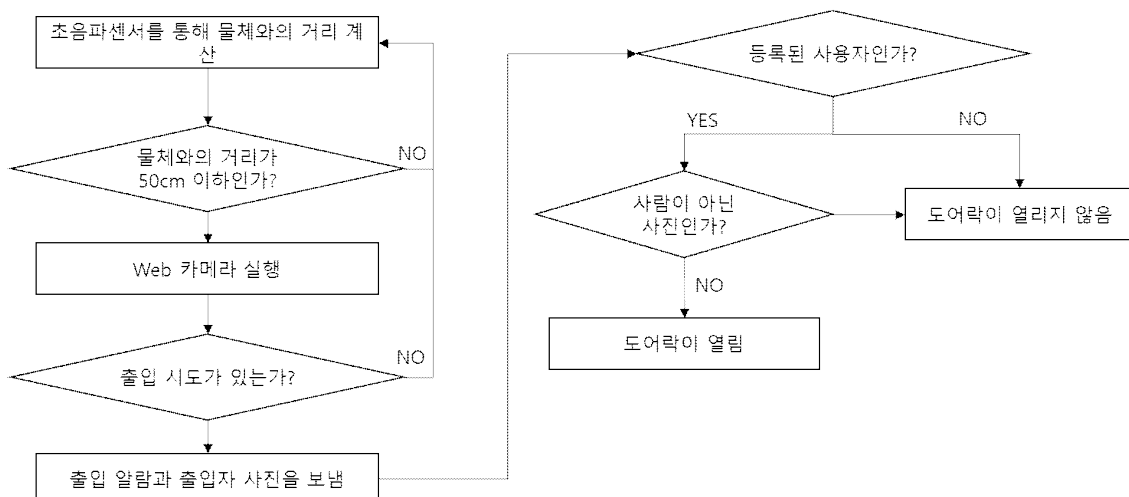


그림 4 얼굴인식 흐름도

그림 3은 Raspberry Pi에서 실행되는 얼굴인식 알고리즘의 흐름도이다. Raspberry Pi에 GPIO로 연결된 초음파센서가 실시간으로 물체와의 거리를 계산하여 출력하면 Threshold 값이 50cm 이하가 될 때 얼굴인식을 위한 Web 카메라가 실행된다. 카메라를 통해 얼굴이 감

지되면 출입 알람과 출입 시도자 사진을 각각 Firebase realtime database와 Firebase storage 로 전송한다. 또한, 앱을 통해 등록된 사용자들의 사진도 Firebase storage에 업로드되므로 Raspberry pi에서 이를 가져와 로컬 폴더에 저장한다. Raspberry Pi의 얼굴 저장 코드는 로컬 카메라를 통해 수집된 사진과 앱을 통해 등록된 사진들의 정렬된 결과를 인코딩하여 로컬 폴더에 저장시켜 놓는다.

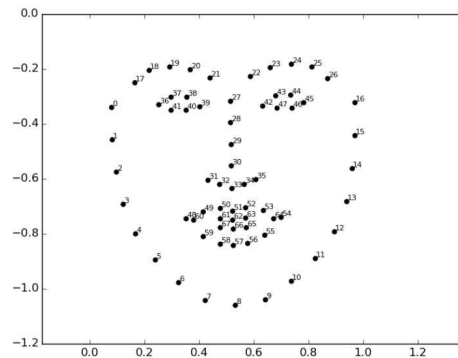


그림 5 Openface를 통해 랜드마크를 정규화한 모습

얼굴 정렬은 'dlib' 라이브러리를 통해 얼굴 랜드마크를 찾고, 그 랜드마크를 그림 4와 같이 'Openface' 라이브러리를 이용해 눈, 코, 입을 모두 동일한 위치에 정렬하는 기법을 사용하였다. 정렬된 얼굴은 'dlib'의 딥러닝 네트워크 기반으로 사전 학습된 얼굴인식 모델인 "dlib_face_recognition_resnet_model_v1.dat"을 활용하여 numpy 배열로 인코딩하고 인코딩된 얼굴 벡터 간의 Euclidean 거리를 비교하여 최소 거리를 찾아 얼굴을 인식한다. 전송된 사진이 사전에 등록된 사용자라고 판단하면 마지막으로 dlib을 통해 얻은 얼굴 랜드마크에서 eye point를 추출하여 눈 깜빡임을 점검하고 사진이 아님을 판단한 후 도어락의 잠금을 해제한다.

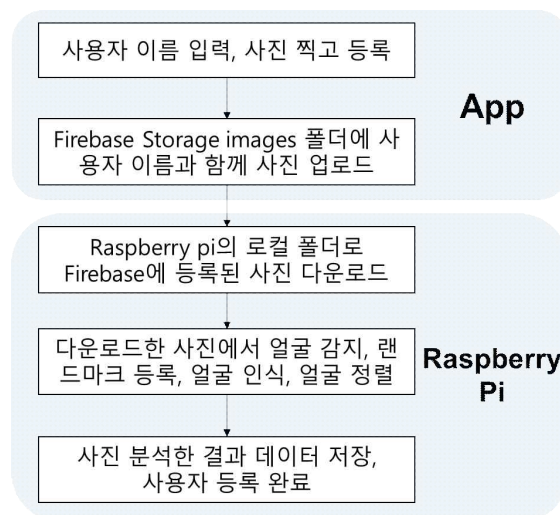


그림 6 사용자 등록 흐름도

그림 6는 사용자가 안드로이드 앱을 통해 등록할 사진을 찍고 Firebase storage로 업로드하고 얼굴인식 엔진이 들어있는 Raspberry Pi의 로컬 폴더로 다운로드하는 사용자 등록과정의 전체적인 흐름을 나타낸다. Firebase 업로드에는 'firebase_admin' 라이브러리의 `firebase.storage.bucket.blob의 upload_from_filename()` 함수를 사용하였다.

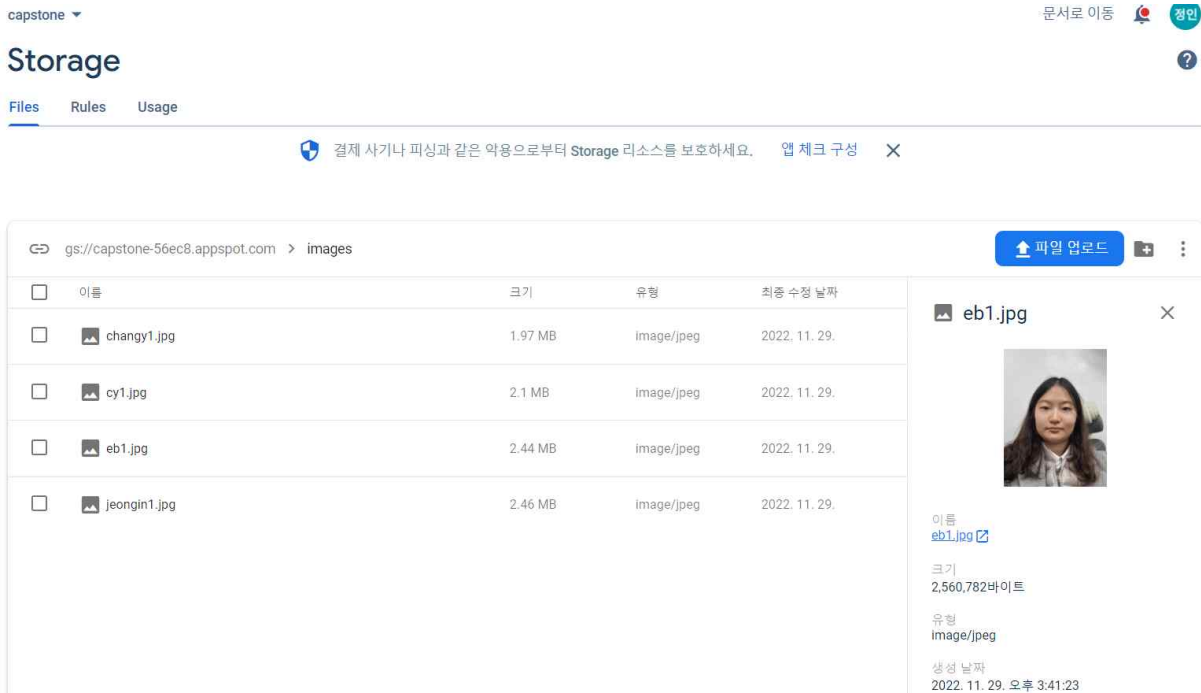


그림 7 Firebase storage - Registered users images

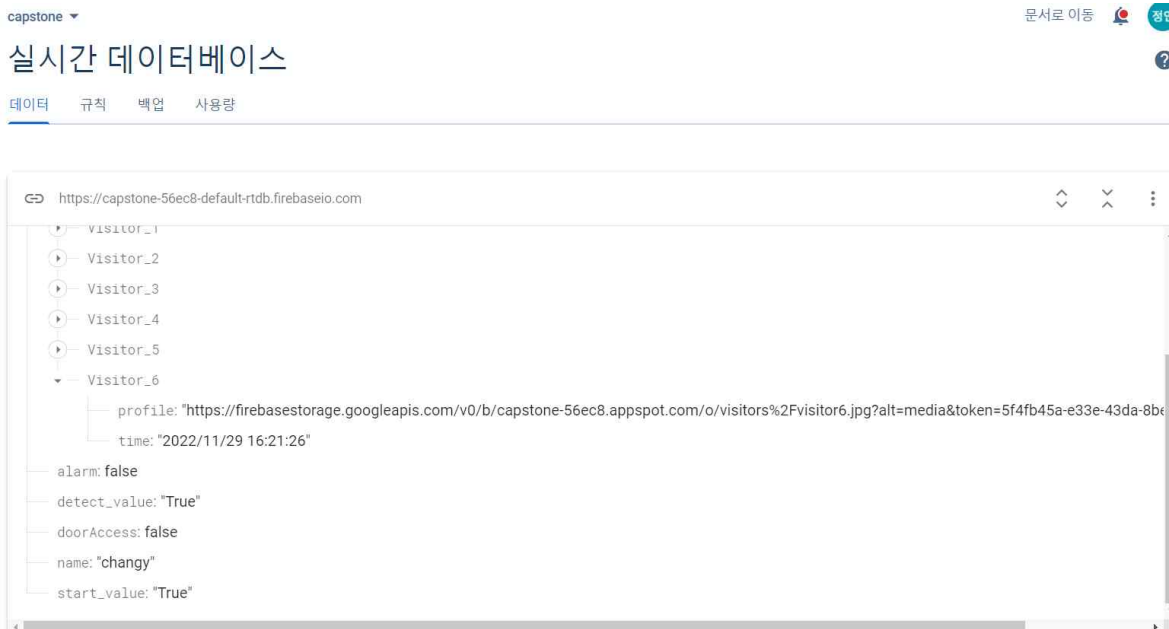


그림 8 Firebase Realtime database

Storage

Files Rules Usage

결제 사기나 피싱과 같은 악용으로부터 Storage 리소스를 보호하세요. [앱 체크 구성](#) ✕







gs://capstone-56ec8.appspot.com > visitors				
<input type="checkbox"/>	이름	크기	유형	최종 수정 날짜
<input type="checkbox"/>	 visitor1.jpg	73.14 KB	image/jpeg	2022. 11. 29.
<input type="checkbox"/>	 visitor2.jpg	69.25 KB	image/jpeg	2022. 11. 29.
<input type="checkbox"/>	 visitor3.jpg	87.3 KB	image/jpeg	2022. 11. 29.
<input type="checkbox"/>	 visitor4.jpg	81.56 KB	image/jpeg	2022. 11. 29.
<input type="checkbox"/>	 visitor5.jpg	81.16 KB	image/jpeg	2022. 11. 29.
<input type="checkbox"/>	 visitor6.jpg	75.78 KB	image/jpeg	2022. 11. 29.

그림 9 Firebase storage - Raspberry Pi web cam에서 수집한 이미지

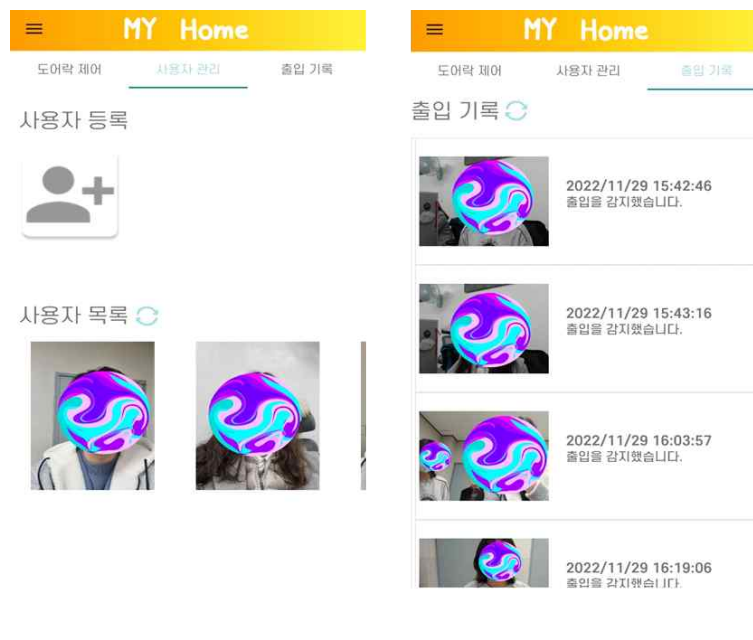


그림 10 앱 구동 사진 (좌: 사용자 등록, 우: 출입 감지 기록 조회)

3.2. LED 점등 제어 시스템

3.2.1. LED 점등

본 논문에서 제안하는 LED 자동 점등 제어 시스템의 하드웨어는 Arduino Nano 33 IoT, 조도센서, LTE Cat.M1 모듈, USB-type LED로 구성된다. 센서값을 모니터링하는 데이터 수집부는 LTE Cat.M1 모듈, 조도 센서 그리고 Arduino Nano 33 IoT로 구성된다. Arduino Nano 33 IoT는 MCU로 조도 센서와 연결되어 집 외벽에 설치되고 빛 세기를 스트리밍한다. 수집된 데이터는 LTE Cat.M1 모듈의 오픈소스 라이브러리를 통해 HTTP 프로토콜로 웹 서버 ThingSpeak으로 전송된다. 구동부인 LED는 또 다른 MCU와 연결되어 ThingSpeak에 업로드된 센서 데이터값을 수신한다. 조도 센서값이 특정 Threshold 값을 초과하면 LED를 서서히 점등하는 디밍 제어를 시작한다.

표 2 LED 점등 코드

```
void control_usbled(int t) {
    if( t > 900){
        analogWrite(LEDLIGHT_PIN, 255);
    }
    else if( t < 900 && t > 700){
        analogWrite(LEDLIGHT_PIN, 191);
    }

    else if( t < 700 && t > 500){
        analogWrite(LEDLIGHT_PIN, 127);
    }

    else if( t < 500 && t > 300){
        analogWrite(LEDLIGHT_PIN, 64);
    }
    else{
        analogWrite(LEDLIGHT_PIN, 0);
    }
}
```

조도 센서의 값이 저장된 'light_val'에 따라 LED가 아두이노의 PWM 출력을 통해 단계적으로 밝기가 조절된다. PWM(Pulse Width Modulation)은 사각파의 펄스폭(duty cycle) 변조를 의미하며 펄스의 폭을 통해 평균값을 변화시켜 아날로그 전압(0~5V)처럼 사용할 수 있으며 PWM 파형이 LED 출력 회로에 연결되면 LED의 ON, OFF 시간을 조절해 밝기 조절이 가능해진다.

3.2.2. LTE Cat.M1 통신

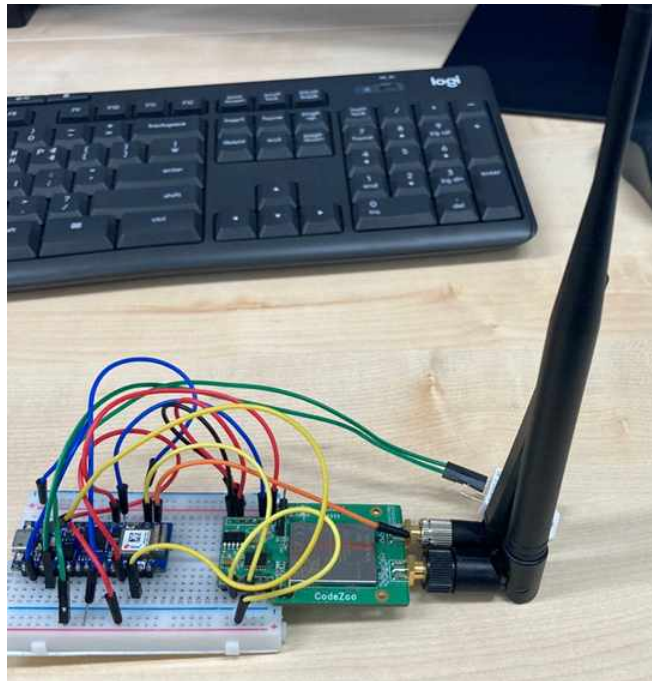


그림 11 LTE Cat.M1과 Arduino Nano 33 IoT 연결모습

본 LED 점등 제어 시스템의 센서값 송신부는 그림 12과 같이 이루어진다. LTE Cat.M1 내장형 모듈인 BG96이 탑재된 Codezoo LTE Cat.M1을 활용하였다. BG96은 최대 300/375Kbps(D/U) 속도의 무선테이터, SMS 송수신을 지원한다. 또한 GNSS(GPS & Glonass) 수신 기능을 내장하고 있다. 오픈소스 라이브러리로 공개된 “BG96.h” 헤더파일을 활용하여 아두이노 IDE 개발 환경에서 Thingspeak 클라우드로 조도 센서값을 전송하였다.

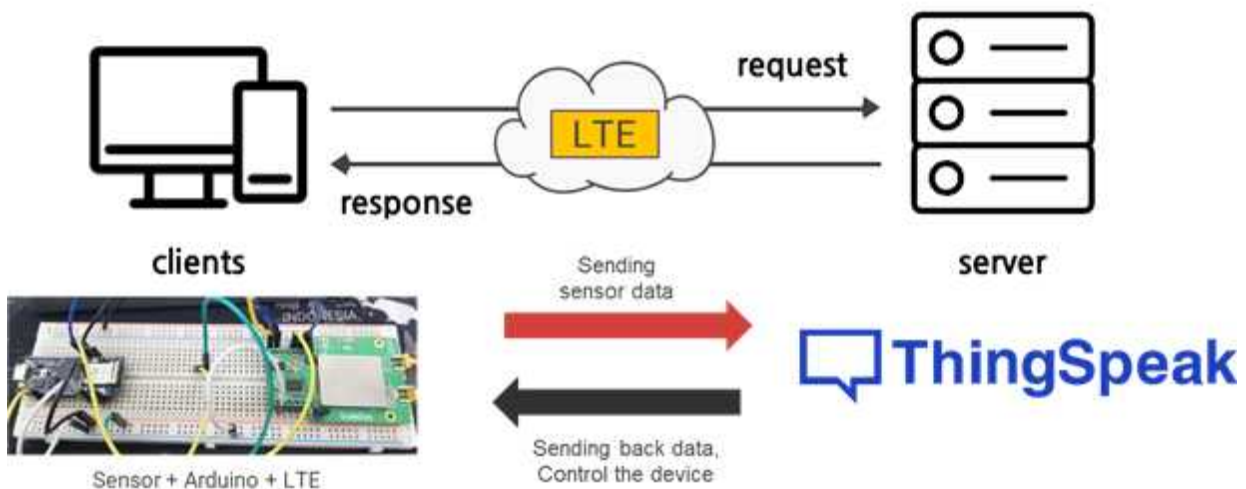


그림 12 LTE Cat.M1 통신모듈을 통한 센서값 클라우드 송신 메커니즘

표 3 LTE Cat.M1통신을 활용한 클라우드 송신 코드

```
#include "BG96.h"
#include <Arduino.h>
#include <Time.h>
#include <TimeAlarms.h>
// AM2302(DHT22) Temperature & Humidity Sensor
#include "DHT.h"
#define DebugSerial Serial
#define M1Serial Serial1
#define PWR_PIN 2
#define STAT_PIN 3
#define DHT_PIN A0

#define ALARM_CYCLE 3600 /* Seconds, 1hour */

String WApiKey = "3UE1NLZS1DBMY0LZ"; // Thing Speak Write API Key 16Character
float temp = 0.0; // Stores temperature value
float humi = 0.0; // Stores humidity value

String fieldTemp = "field1"; // Air temperature
String fieldHumi = "field2"; // Air humidity

DHT dht;
int _sendTag = 0;

BG96 BG96(M1Serial, DebugSerial, PWR_PIN, STAT_PIN);

void setup() {

    dht.setup(DHT_PIN);

    M1Serial.begin(115200);
    DebugSerial.begin(115200);

    /* Power On Sequence */
    if (BG96.isPwrON()) {
        DebugSerial.println("BG96 Power ON Status");
        if (BG96.pwrOFF()) {
            DebugSerial.println("BG96 Power Off Error");
        } else {
            DebugSerial.println("BG96 Power Off Success");
        }
    }
}
```

```

    DebugSerial.println("Module Power ON Sequence Start");
    if (BG96.pwrON()) {
        DebugSerial.println("BG96 Power ON Error");
    } else
        DebugSerial.println("BG96 Power ON Success");
}
} else {
    DebugSerial.println("BG96 Power OFF Status");
    if (BG96.pwrON()) {
        DebugSerial.println("BG96 Power ON Error");
    } else
        DebugSerial.println("BG96 Power ON Success");
}

/* BG96 Module Initialization */
if (BG96.init()) {
    DebugSerial.println("BG96 Module Error!!!");
}

/* BG96 Module Power Saving Mode Disable */
if (BG96.disablePSM()) {
    DebugSerial.println("BG96 PSM Disable Error!!!");
}

/* Network Regsistraiton Check */
while (BG96.canConnect() != 0) {
    DebugSerial.println("Network not Ready !!!");
    delay(2000);
}

char szCCLK[32];
int _year, _month, _day, _hour, _min, _sec;

/* Get Time information from the Telco base station */
if (BG96.getCCLK(szCCLK, sizeof(szCCLK)) == 0) {
    DebugSerial.println(szCCLK);
    sscanf(szCCLK, "%d/%d/%d,%d:%d:%d+*%d", &_year, &_month, &_day, &_hour,
        &_min, &_sec);

    /* Time Initialization */
    setTime(_hour, _min, _sec, _month, _day, _year);
}

```

```

/* Alarm Enable */
Alarm.timerRepeat(ALARM_CYCLE, Repeats);

DebugSerial.println("BG96 Module Ready!!!");
}

void Repeats() {
    DebugSerial.println("alarmed timer!");
    _sendTag = 1;
}

void loop() {

    if (_sendTag) {

        while (1) {
            delay(dht.getMinimumSamplingPeriod());

            /* Get DHT22 Sensor */
            if (dht.getStatusString() == "OK") {
                temp = dht.getTemperature();
                humi = dht.getHumidity();
                if (String(temp) != "nan" && String(humi) != "nan")
                    break;
            }
            else {
                DebugSerial.println("case nan ...");
                delay(1000);
            }
        }
    }

    char _IP[] =
        "api.thingspeak.com"; // ThingSpeak Report Server api.thingspeak.com
    unsigned long _PORT = 80;

    char recvBuffer[1024];
    int recvSize;
    String data = "GET /update";
    data += "?api_key=" + WApiKey + "&" + fieldTemp + "=" + String(temp) + "&" +
        fieldHumi + "=" + String(humi);
    data += " HTTP/1.1\r\n";

```

```

data += "Host: api.thingspeak.com\r\n";
data += "Connection: close\r\n\r\n";

if (BG96.actPDP() == 0) {
    DebugSerial.println("BG96 PDP Activation !!!");
}

if (BG96.socketCreate(1, _IP, _PORT) == 0)
    DebugSerial.println("TCP Socket Create!!!");

/* Socket Send */
if (BG96.socketSend(data.c_str()) == 0) {
    DebugSerial.print("[TCP Send] >> ");
    DebugSerial.println(data);
} else
    DebugSerial.println("Send Fail!!!");

if (BG96.socketRecv(recvBuffer, sizeof(recvBuffer), &recvSize, 10000) ==
    0) {
    DebugSerial.print(recvBuffer);
    DebugSerial.println(" <<----- Read Data");
    DebugSerial.println(recvSize, DEC);
    DebugSerial.println("Socket Read!!!");
}

if (BG96.socketClose() == 0) {
    DebugSerial.println("Socket Close!!!");
}

delay(5000);

if (BG96.deActPDP() == 0) {
    DebugSerial.println("BG96 PDP DeActivation!!!");
}

_sendTag = 0;
}

/* Time Test */
DebugSerial.print(hour());
DebugSerial.print(":");

```

```
DebugSerial.print(minute());  
DebugSerial.print(":");  
DebugSerial.print(second());  
DebugSerial.println("");  
  
Alarm.delay(1000);  
}
```

4. 결론 및 기대효과

본 작품은 모바일을 통해 사용자 주도하에 사용 행동을 자동화하거나 원격으로 디바이스를 조작 및 관리하는 스마트 홈 서비스 시스템이다. 기존의 고가인 스마트 홈 시스템에 비해 저렴하지만, 신뢰성과 보안성을 고루 갖추었고, 앱을 통한 사용자 인터페이스도 제공하여 사용자에게 편의를 제공한다. 또한 전력 소모가 적은 통신 방식을 활용하였다는 점에서 IoT 시스템을 위한 소용량 센서 데이터를 효율적으로 송수신하기에 적합하다. 이는 다양한 IoT 애플리케이션의 선행 연구 사례로 활용될 수 있다.

5. 후기

김정인: 이번 프로젝트를 기획하고 진행하면서 Database, Cloud, Communication, App programming, Deep learning 기반의 다양한 소프트웨어 개발을 하였고 프로그래밍역량을 기를 수 있었습니다. Raspberry Pi, Arduino 등의 다양한 마이크로컨트롤러도 활용하여 하드웨어에 대한 경험도 쌓을 수 있었습니다. 또한 팀원들이 적극적으로 참여하여 프로젝트를 잘 마무리할 수 있었던 것 같습니다.

조은비: 3학년 때 IoT (Internet of Things)과목과 모바일 프로그래밍 수업을 들으며 생활에 도움이 되는 프로젝트를 하고 싶다고 생각하게 되었습니다. 평소에 관심있던 홈 IoT 프로젝트에 참여하게 되어 더 열정적으로 참여할 수 있었습니다. 애플리케이션에 통신을 접목해 보는 것은 처음이라 어려움이 있었지만, 팀원들과 힘을 합쳐 해결하며 실무적인 부분을 많이 배울 수 있었습니다.

김근호: 이번 창의설계프로젝트를 통해 어색하기만 했던 아두이노, 라즈베리 파이와 같은 마이크로프로세서를 활용하여 하드웨어를 동작하는 것에 대해 자신감이 생겼으며 이는 오일수 교수님의 '마이크로프로세서'를 수강했던 경험이 아주 큰 도움이 되었습니다. 또한, 한 가지를 해결하면 다른 문제가 생기는 고난의 상황에서 개인주의 성향이 강하고 혼자 하는 것이 익숙했던 저에게 집단지성, 팀플레이로 이러한 고난을 해결하는 과정은 저에게 큰 의미로 다가왔습니다.

오창윤: 아두이노라는 프로그램을 처음으로 다루어보았고 인터넷에 LED 예제 자료들이 많이 있어 따라 해볼 수 있었습니다, 1학년 때 수강한 전일수 교수님의 c언어 수업을 통해 조금이라도 이해할 수 있어 도움이 되었습니다.

처음 해보는 게 많아서 미숙했지만 팀원들이 도움을 많이 주어 무사히 끝낼 수 있어서 감사하고 그 과정에서 많이 배웠습니다.

팀원간 역할 분담

성명	역할	참여도(%)
김정인	프로젝트 기획 및 총괄, 앱 제작, 데이터베이스 구축, 얼굴인식 코드 분석, 통신 코드 분석, LED 제어 코드 분석	100%
조은비	앱 제작, 얼굴인식 코드 분석, 데이터베이스 구축, 도어락 하드웨어 프로토타입 제작	100%
김근호	도어락 하드웨어 프로토타입 제작, LED 제어 코드 분석	100%
오창윤	도어락 하드웨어 프로토타입 제작, 통신 코드 분석, LED 제어 코드 분석	100%
안정은	LED 제어 코드 분석	50%

비용 분석

항목	세부항목	소요비용
재료비	LTE-CatM1 내장형 모듈, Raspberry Pi, Arduino Nano 33 IoT, Arduino Mega, 아두이노 릴레이 2채널 절연회로 B57, 10 LED 5V USB 램프 형광등색 LED 전구, C415(r) USB_A type Adapter, USB 타입 220V LED 조명, 해강 디지털도어락 SRB100-실버 ,Global IoT SIM - Telenor Connexion 30MB	284,162
시제품가공비	아이리스 모던 도어형 공간박스, 흡착식 도어핸들	22,270
기타 경비	회의비	80,800

참고문헌

- (1) https://github.com/dudtjs1021ej/2021ESWContest_free_1032
- (2) <https://openface-api.readthedocs.io>
- (3) Danielle Jaye S Agron, Jeong-in Kim, and Wansu Lim. "IoT-enabled Smart Streetlamp Controller with LPWA LTE Cat M1 Module." 한국통신학회 학술대회논문집 2022.6 (2022): 691-692.
- (3) https://github.com/codezoo-ltd/CodeZoo_CATM1_Arduino

별첨

도어락 자동제어, LED 자동제어 프로토타입