컴퓨터 시스템 입문

기말고사 프로젝트

‘스마트 블라인드’

담당교수 : 김성필 교수님

제출일자 : 20.07.03

학 과 : 전자공학과

학 년 : 3학년

이 름 : 배준성

학 번 :

한국외국어대학교
HANKUK UNIVERSITY OF FOREIGN STUDIES

목 차

1. 개발 배경

예전부터 자동으로 동작하는 블라인드를 하나 가지고 싶었지만 상당히 가격이 나가는 편인 물건이었다. 그래서 이번 기말 프로젝트를 통해서 직접 만들어보면 어떨까? 하는 생각을 가지게 되었다.

전동우드블라인드 자동블라인드 전동블라인드

178,000원

가구/인테리어 > 커튼/블라인드 > 블라인드

소재 : 우드 | 작동방식 : 전동식 | 색상계열 : 브라운, 화이트, 블랙

리뷰 48 · 구매건수 32 · 등록일 2017.03. · 찜하기 83 · 신고하기 톡톡

브런트 블라인드 엔진 / 스마트 IoT 전동 블라인드 자동 커튼 / 구글홈 클로바 카카오미니 기가지니 연동

108,000원

가구/인테리어 > 커튼/블라인드 > 블라인드

리뷰 215 (네이버페이 86) · 등록일 2020.04. · 찜하기 101 · 신고하기

차세대 전동 블라인드모터/자동 블라인드모터[시스템]

117,480원

가구/인테리어 > 커튼/블라인드 > 블라인드

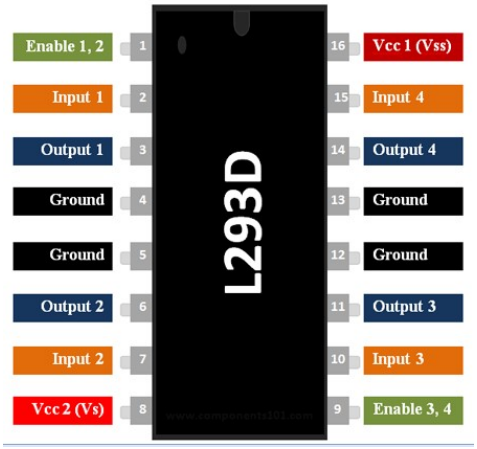
[KB국민카드 1% 청구할인] [KB국민/신한/현대 100만원 이상 SK pay 결제 시 최대 22개월 무이자] [KB국민/신한/현대/NH농협/삼성 20만원 이상 SKpay 결제 시 최대 12개월 무이자]

등록일 2019.03. · 찜하기 20 · 신고하기

또 방에 있는 창이 바람을 많이 받는 구조로 되어있어 비가 올 때 창문이 열려 있으면 비가 집안으로 엄청 들어와서 책상위의 물건이 비에 젖은 경우가 꽤 있었다. 그래서 블라인드가 창문에 설치되는 기기이니 일기예보와 연동되어 동작하는 기능을 추가하면 좋겠다는 생각이 들어 이러한 기능이 추가된 자동 블라인드를 만들게 되었다.

2. 배경 이론

우선 블라인드를 만들기 위해서는 모터를 한방향으로 돌려 블라인드를 감아 올리고 반대 방향으로 돌려 풀어 내려야한다. 따라서 하나의 모터를 양방향으로 회전하도록 제어해야 하는데 이 기능을 위해서는 L239D라는 IC가 필요하다.



이 IC의 datasheet와 pin map을 통해서 확인해보면 Enable1, 2를 이용해서 2, 7번핀(Input1,2)을 사용할지 말지 정할 수 있고 input1, 2에 모터 제어신호를 넣고 output1, 2핀으로 출력되는 신호를 이용하여 모터를 제어할 수 있다. IC의 오른쪽면도 이와 마찬가지로 동작한다. Vs핀에는 모터의 전원이, Vss는 IC의 전원이 들어간다. 이를 통해서 모터를 양방향으로 제어할 수 있다. IC 제어를 통해서 다음과 같이 제어할 수 있다.

Input 1	**Input 2**	**모터**
GND	GND	멈춤
5V	GND	방향A로 회전
GND	5V	방향B로 회전
5V	5V	멈춤

또한 Enable핀을 이용해서 모터가 동작할지 말지를 결정해 줄 수 있다.

이렇게 2개의 입력을 통해서 하나의 모터를 양방향으로 동작하도록 제어할 수 있다.

다음으로 일기예보와 연동되어서 동작하기 위해서는 인터넷에 연결되어 일기예보에 대한 정보를 받아올 수 있어야 한다. 이러한 동작을 위해서 기상청의 RSS를 이용하여 일기예보를 받아 오기로 하였다.

기상청 사이트에 접속하여 원하는 지역의 RSS주소를 얻어와 사용하였다.

This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
▼<rss version="2.0">
  ▼<channel>
    <title>기상청 동네예보 웹서비스 - 서울특별시 동작구 신대방제2동 도표예보</title>
    <link>http://www.kma.go.kr/weather/main.jsp</link>
    <description>동네예보 웹서비스</description>
    <language>ko</language>
    <generator>동네예보</generator>
    <pubDate>2020년 07월 03일 (금)요일 17:00</pubDate>
   ▼<item>
      <author>기상청</author>
      <category>서울특별시 동작구 신대방제2동</category>
      <title>동네예보(도표) : 서울특별시 동작구 신대방제2동 [X=59, Y=125]</title>
      <link>http://www.kma.go.kr/weather/forecast/timeseries.jsp?searchType=INTEREST&dongCode=1159068000</link>
      <guid>http://www.kma.go.kr/weather/forecast/timeseries.jsp?searchType=INTEREST&dongCode=1159068000</guid>
     ▼<description>
       ▼<header>
          <tm>202007031700</tm>
          <ts>5</ts>
          <x>59</x>
          <y>125</y>
        </header>
       ▼<body>
         ▼<data seq="0">
            <hour>21</hour>
            <day>0</day>
            <temp>23.0</temp>
            <tmx>-999.0</tmx>
            <tmn>-999.0</tmn>
            <sky>3</sky>
            <pty>4</pty>
            <wfKor>소나기</wfKor>
            <wfEn>Shower</wfEn>
            <pop>60</pop>
            <r12>0.0</r12>
            <s12>0.0</s12>
```


기상청에서 받아온 RSS주소를 통해서 접속해보면 다음과 같은 방식으로 정보를 보내준다. 따라서 이곳에서 필요한 정보를 찾아내어 저장하는 방식으로 코딩할 필요가 있었다.

또한 우리나라 일기예보는 아래의 표와 같이 3시간마다 예보를 발표하고 각 시간의 4, 7,

10시간 뒤의 일기 예보를 발표한다. 따라서 예보와 발표 시간이 가장 가까운 때가 발표 1시간 후 비 예보가 있는 경우이다. 따라서 이러한 경우에는 비가 올 위험이 크기 때문에 스마트폰 알림을 통해서 알려주도록 할것이다.

2시 발표	5시 발표	8시 발표	11시 발표	14시 발표	17시 발표	20시 발표	23시 발표
6시 예보	9시 예보	12시 예보	15시 예보	18시 예보	21시 예보	0시 예보	3시 예보
9시 예보	12시 예보	15시 예보	18시 예보	21시 예보	0시 예보	3시 예보	6시 예보
12시 예보	15시 예보	18시 예보	21시 예보	0시 예보	3시 예보	6시 예보	9시 예보

다음으로는 blynk앱의 터미널 사용하여 받아온 정보를 출력해 주었다. 터미널의 사용법의 경우에는 아두이노 개발환경의 시리얼 모니터와 유사하였기 때문에 크게 어렵지 않았다. 또한 터미널을 이용하여 정보를 보드로부터 받기만 할 것이기 때문에 terminal.print문을 주로 이용하여 터미널 화면에 원하는 정보를 출력해 주었다.

일기예보 시스템의 경우 시간에 따라서 일기예보 정보를 분석할 필요가 있었기 때문에 현재 시간을 알아야 했는데 이것은 github를 통해서 필요한 라이브러리를 설치하여 blynk앱과 연동해 쉽게 시간을 불러와 사용하였다.

3. 소프트웨어 해설

nodeMCU의 코드는 다음과 같다.

```
#include <ESP8266WiFi.h> //와이파이와 http를 사용하기 위한 헤더
#include <ESP8266HTTPClient.h>

#define BLYNK_PRINT Serial //blynk와 그 위젯들을 사용하기 위한 헤더
#include <BlynkSimpleEsp8266.h>
#include <TimeLib.h>
#include <WidgetRTC.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
// blynk를 사용하기 위한 토큰
char auth[] = "[redacted]";
```

```
// Your WiFi credentials.
// Set password to "" for open networks.
// 네트워크 이름과 비밀번호
char ssid[] = "[redacted]";
char pass[] = "[redacted]";

// 기상청 RSS주소
const String endpoint =
"http://www.kma.go.kr/wid/queryDFSRSS.jsp?zone=2671025000";

// 날씨 저장 변수
char pty0;
//현재 시각을 LCD에 출력하기위한 변수
String currentTime;
String currentDate;

int counter;   //날씨알람 보낼 시간인지 확인

int timeStart ;

int pinV2;

char serialTest;
int testOn = 0;

float disToWin;

int numTones = 78; //부저 음악 출력_슈퍼마리오
int tones[] = {
  2637, 2637, 0, 2637,
  0, 2093, 2637, 0,
  3136, 0, 0,   0,
  1568, 0, 0, 0,

  2093, 0, 0, 1568,
  0, 0, 1319, 0,
  0, 1760, 0, 1976,
  0, 1865, 1760, 0,
```

```
  1568, 2637, 3136,
  3520, 0, 2794, 3136,
  0, 2637, 0, 2093,
  2349, 1976, 0, 0,

  2093, 0, 0, 1568,
  0, 0, 1319, 0,
  0, 1760, 0, 1976,
  0, 1865, 1760, 0,

  1568, 2637, 3136,
  3520, 0, 2794, 3136,
  0, 2637, 0, 2093,
  2349, 1976, 0, 0
};

String line = "";

WidgetTerminal terminal(V0);

BlynkTimer timer;

WidgetRTC rtc;




void setup() {
  Serial.begin(9600);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("Connected to the WiFi network");

  Blynk.begin(auth, ssid, pass);
```

```
  // terminal 화면을 clr
  terminal.clear();

  setSyncInterval(10 * 60); // 10 분마다 싱크 다시 맞춰줌

  // 1초마다 디스플레이에 표시_시간_초
  timer.setInterval(1000L, clockDisplay);

  counter = hour() - 1; //시작하자마자 일기예보를 한번 확인해보기 위해서 다음과 같이 입력

  pinMode (D0, OUTPUT);
  pinMode (D1, OUTPUT);
  pinMode (D2, OUTPUT);
  pinMode (D3, OUTPUT);
  pinMode (D4, INPUT);
  pinMode (D5, OUTPUT);

}

void loop() {
  Blynk.run();
  timer.run();

  rainChecker();

  Test();

}
// 부저에서 음악이 나오게 해줌
void buzzer() {
  for (int i = 0; i < numTones; i++) {
    tone(D5, tones[i]);
    delay(200);
  }
  noTone(D5);
}
// 설계한 프로그램을 테스트하기 위해서 만듦
void Test() {
```

```
  serialTest = Serial.read();
  if (serialTest == 't') {
    Serial.println("TEST Start");
    testOn = 1;
    rainChecker();
    testOn = 0;
    Serial.println("TEST Fin");

  }
}
//이름대로 비 예보가 있는지 확인
void rainChecker() {
  int Ho;
  if (testOn == 1) {
    Ho = 10;
    Serial.println("Test rainChecker");
  }
  else {
    Ho = hour();
  }

  if (counter != Ho) {
    get_weather();
    if (testOn == 1) {
      pty0 = 1;
    }
    Serial.println("*******   Time Checked   *******");
    for (int i = 10; i < 23; i += 3)
    {
      if (i == Ho) {
        rainNotice();
        Serial.println("*******   Rain Noticed   *******");
      }
    }
    counter = Ho;
    Serial.println("*******   Rain Checked   *******");
  }

}
```

```
// RSS로 날씨를 받아옴
void get_weather() {
  if ((WiFi.status() == WL_CONNECTED)) {
    Serial.println("Starting connection to server...");
    HTTPClient http;
    http.begin(endpoint); //기상청 RSS주소에 연결한다
    int httpCode = http.GET();
    if (httpCode > 0) { //요청 성공
      line = http.getString();
    }
    else {// 실패하면 에러가 발생했다고 출력
      Serial.println("Error on HTTP request");
    }
    parsing(); //이 함수를 이용해서 터미널에 일기예보를 출력
    http.end(); //Free the resources
  }
}

// LCD에 시간을 표시한다.
void clockDisplay()
{

  currentTime = String(hour()) + ":" + minute() + ":" + second();
  currentDate = String(year()) + " " + month() + " " + day();

  // 가상핀을 이용해 앱으로 시간을 보냄
  Blynk.virtualWrite(V3, currentTime);
  // 가상핀을 이용해 앱으로 날짜를 보냄
  Blynk.virtualWrite(V4, currentDate);
}

// 블라인드를 올리거나 내리는 것을 제어
void blindMove() {
  timeStart = millis();

  if (pinV2 == 1) {
    digitalWrite (D0, HIGH);
    digitalWrite (D1, LOW);
```

```
    Serial.println("A");
    digitalWrite (D2, HIGH);
  }

  else {
    digitalWrite (D0, LOW);
    digitalWrite (D1, HIGH);

    Serial.println("B");
    digitalWrite (D2, HIGH);
  }

  int timeStoE = 0;
  while (timeStoE < 5000) { //블라인드가 동작하는 시간을 정해주고 1초마다 정해진
시간동안 동작하였는지 확인
    timeStoE = millis() - timeStart;
    Serial.println("C");
    Serial.println(timeStoE);
    delay(1000);
  }
  digitalWrite (D2, LOW); //동작 후 정지

}



BLYNK_CONNECTED() {
  rtc.begin();
}

BLYNK_WRITE (V1) { //날씨에 따른 안내 가상핀 V1은 날짜정보 불러오기 버튼에 연결

  int pinValue = param.asInt ();
  if (pinValue == 1) {
    get_weather();
    terminal.flush();

    if (pty0 != '0') { //일기 예보에 따라 문구를 출력
      terminal.print(F("₩n날씨가 흐리니 창문을 닫아주세요."));
    }
```

```
      else {
        terminal.print(F("₩n날씨가 맑으니 환기하세요."));
      }

      int winCh = winCheck(); //창문이 열려 있는지 닫혀 있는지 확인가능
      if (winCh == 0) {
        terminal.print(F("₩n창문이 닫혀 있습니다."));
      }
      else {
        terminal.print(F("₩n창문이 열려 있습니다."));
      }

      terminal.flush();
    }
}

BLYNK_WRITE (V2) { //가상핀 2번은 블라인드 제어 버튼들에 연결
   pinV2 = param.asInt (); // V2에서 들어오는 값을 변수에 할당
   blindMove();
}

BLYNK_WRITE (V5) { //가상핀 5번은 창문거리 셋팅 버튼에 연결
   int pinV5 = param.asInt ();
   if (pinV5 == 1) {
      disToWin = dist();
   }
}

// 초음파 센서를 통해서 거리를 측정
float dist() {
   digitalWrite(D3, LOW);
   digitalWrite(D4, LOW);
   delayMicroseconds(2);
   digitalWrite(D3, HIGH);
   delayMicroseconds(10);
   digitalWrite(D3, LOW);

   // echoPin 이 HIGH를 유지한 시간을 저장한다.
   unsigned long duration = pulseIn(D4, HIGH);
```

```
  // HIGH 였을 때 시간(초음파가 보냈다가 다시 들어온 시간)을 가지고 거리를 계산한다.
  float distance = ((float)(340 * duration) / 10000) / 2;

  Serial.print("거리 : ");
  Serial.print(distance);
  Serial.println("cm");

  return distance; //측정한 거리값을 출력
}

// 창문이 열려있는지 확인
int winCheck() {
  int d = dist();
  int distance_error = disToWin + 5;
  if (d > distance_error) {
    return 1;
  }
  else {
    return 0;
  }
}

// 만약 1시간 시점에 비 예보가 있으면 스마트폰 앱 알림을 통해서 알려준다.
void rainNotice() {
  if (pty0 != '0') {
    Blynk.notify("현재시각  " + String (hour()) + "시 " + minute() + "분 " + second() + "초 ₩n" + "1시간후 비예보가 있으니 창문을 닫아주세요!");
    int winCh = winCheck(); //1시간 시점에 비 예보가 있는데 창문이 열려있으면 부저를 통해서 음악을 출력
    if (winCh == 1) {
      buzzer();
    }
  }
}

//RSS로부터 데이터를 받아오고 터미널과 시리얼 모니터를 통해 출력
void parsing() {
  terminal.clear();
```

```
  String announce_time;
  int pubDate_start = line.indexOf(F("<pubDate>")); // "<pubDate>"문자가 시작되는 인덱스 값('<'의 인덱스)을 반환한다.
  int pubDate_end = line.indexOf(F("</pubDate>"));
  announce_time = line.substring(pubDate_start + 9, pubDate_end); // +9: "<pubDate>"스트링의 크기 9바이트, 9칸 이동
  Serial.print(F("announce_time: "));
  Serial.println(announce_time);

  terminal.print("기상예보 발표 시각 : ");
  terminal.println(announce_time);
  terminal.println("--------------------");

  String hour;
  int hour_start = line.indexOf(F("<hour>"));
  int hour_end = line.indexOf(F("</hour>"));
  hour = line.substring(hour_start + 6, hour_end);
  Serial.print(F("hour: "));
  Serial.println(hour);

  terminal.print(hour);
  terminal.println("시 기상예보 ");

  String temp;
  int temp_start = line.indexOf(F("<temp>"));
  int temp_end = line.indexOf(F("</temp>"));
  temp = line.substring(temp_start + 6, temp_end);
  Serial.print(F("temp: "));
  Serial.println(temp);

  terminal.print("기온 : ");
  terminal.print(temp);
  terminal.println("도");

  String wfKor;
  int wfKor_start = line.indexOf(F("<wfKor>"));
  int wfKor_end = line.indexOf(F("</wfKor>"));
  wfKor = line.substring(wfKor_start + 7, wfKor_end);
  Serial.print(F("weather: "));
```

```
Serial.println(wfKor);

terminal.print(F("날씨 : "));
terminal.println(wfKor);
terminal.println("--------------------");
terminal.flush();

String pty;
int pty_start = line.indexOf(F("<pty>"));
int pty_end = line.indexOf(F("</pty>"));
pty = line.substring(pty_start + 5, pty_end);
Serial.print(F("pty: "));
Serial.println(pty);
pty0 = pty.charAt(0);
Serial.print(F("pty0: "));
Serial.println(pty0);

int del_index = line.indexOf(F("</data>"));
line.remove(0, del_index + 7); // 시작 인덱스 부터 "</data>" 스트링 포함 삭제
hour_start = line.indexOf(F("<hour>"));
hour_end = line.indexOf(F("</hour>"));
hour = line.substring(hour_start + 6, hour_end);
Serial.print(F("hour: "));
Serial.println(hour);

terminal.print(hour);
terminal.println("시 기상예보 ");

temp_start = line.indexOf(F("<temp>"));
temp_end = line.indexOf(F("</temp>"));
temp = line.substring(temp_start + 6, temp_end);
Serial.print(F("temp: "));
Serial.println(temp);

terminal.print("기온 : ");
terminal.print(temp);
terminal.println("도");

wfKor_start = line.indexOf(F("<wfKor>"));
```

```
wfKor_end = line.indexOf(F("</wfKor>"));
wfKor = line.substring(wfKor_start + 7, wfKor_end);
Serial.print(F("weather: "));
Serial.println(wfKor);

terminal.print(F("날씨 : "));
terminal.println(wfKor);
terminal.println("--------------------");
terminal.flush();

del_index = line.indexOf(F("</data>"));
line.remove(0, del_index + 7);  // 시작 인덱스 부터 "</data>" 스트링 포함 삭제
hour_start = line.indexOf(F("<hour>"));
hour_end = line.indexOf(F("</hour>"));
hour = line.substring(hour_start + 6, hour_end);
Serial.print(F("hour: "));
Serial.println(hour);

terminal.print(hour);
terminal.println("시 기상예보 ");

temp_start = line.indexOf(F("<temp>"));
temp_end = line.indexOf(F("</temp>"));
temp = line.substring(temp_start + 6, temp_end);
Serial.print(F("temp: "));
Serial.println(temp);

terminal.print("기온 : ");
terminal.print(temp);
terminal.println("도");

wfKor_start = line.indexOf(F("<wfKor>"));
wfKor_end = line.indexOf(F("</wfKor>"));
wfKor = line.substring(wfKor_start + 7, wfKor_end);
Serial.print(F("weather: "));
Serial.println(wfKor);

line = ""; // 스트링 변수 line 데이터 추출 완료
```

```
  terminal.print(F("날씨 : "));
  terminal.println(wfKor);

  terminal.flush();
}
```

코드는 위와 같으며 주요 부분에 대해서 설명하면 다음과 같다.

```
void setup() {
  Serial.begin(9600);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("Connected to the WiFi network");
```

WiFi.begin(ssid, pass) 함수를 이용하여 공유기에 연결한다. 이어지는 while문을 통해서 공유기에 연결중인 것을 확인할 수 있다. 공유기에 연결되면 while문을 탈출하여 wifi에 연결되었다고 출력해준다.

```
void buzzer() {
  for (int i = 0; i < numTones; i++) {
    tone(D5, tones[i]);
    delay(200);
  }
  noTone(D5);
}
```

부저는 미리 배열의 형식으로 원하는 노래를 악보로 만들어 입력해두면 for문을 이용해서 출력 가능하다.

```
void rainChecker() {
  int Ho;
  if (testOn == 1) {
    Ho = 10;
    Serial.println("Test rainChecker");
  }
  else {
    Ho = hour();
  }

  if (counter != Ho) {
    get_weather();
    if (testOn == 1) {
      pty0 = 1;
    }
    Serial.println("*******  Time Checked  *******");
    for (int i = 10; i < 23; i += 3)
    {
      if (i == Ho) {
        rainNotice();
        Serial.println("*******  Rain Noticed  *******");
      }
    }
    counter = Ho;
    Serial.println("*******  Rain Checked  *******");
  }

}
```

rainChecker의 경우 테스트를 위한 코드를 포함하고 있으며 counter과 현재 시간을 비교하여 다를 경우 동작에 들어간다. 즉 정상적으로 동작하면 1시간에 1번 동작한다. 이 함수의 동작은 일기예보시간과 예보 발표시간의 간격이 1시간인 시각이 되면 rainNotice를 동작 시킨다.

```
void rainNotice() {
  if (pty0 != '0') {
    Blynk.notify("현재시각  " + String (hour()) + "시 " + minute() + "분 " + second() + "초 \n" + "1시간이내 비예보가 있으니 창문을 닫아주세요!");
    int winCh = winCheck();
    if (winCh == 1) {
      buzzer();
    }
  }
}
```

rainNotice는 1시간 시점에 비예보가 있으면 스마트폰 알림을 보내 비 예보가 있음을 알려준다. 이 알림은 현재시각을 포함한다. 또한 알림을 보낼 때 창문이 열려 있으면 부저

에서 노래를 재생하여 창문이 열려있음을 알려준다.

```
void get_weather() {
  if ((WiFi.status() == WL_CONNECTED)) {
    Serial.println("Starting connection to server...");
    HTTPClient http;
    http.begin(endpoint);
    int httpCode = http.GET();
    if (httpCode > 0) {
      line = http.getString();
    }
    else {
      Serial.println("Error on HTTP request");
    }
    parsing();
    http.end();
  }
}
```

get_weather 함수의 경우 WiFi를 이용하여 기상청 RSS에서 데이터를 받아온다. HTTPClient의 클래스 오브젝트는 http이며 endpoint는 기상청의 RSS주소이다. 따라서 해당하는 주소로 접속하여 GET를 실행한다. 요청이 성공하면 응답을 line에 저장한다. 실패하면 에러가 발생했다고 출력한다. 그 다음으로 parsing 함수를 불러온다.
parsing 함수는 내부에 비슷한 구조가 반복되므로 일부만 꺼내서 확인해보도록 한다.

```
void parsing() {
  terminal.clear();
  String announce_time;
  int pubDate_start = line.indexOf(F("<pubDate>")); // "<pubDate>"문자가 시작되는 인덱스 값('<'의 인덱스)을 반환한다.
  int pubDate_end = line.indexOf(F("</pubDate>"));
  announce_time = line.substring(pubDate_start + 9, pubDate_end); // +9: "<pubDate>"스트링의 크기 9바이트, 9칸 이동
  Serial.print(F("announce_time: "));
  Serial.println(announce_time);

  terminal.print("기상예보 발표 시각 : ");
  terminal.println(announce_time);
  terminal.println("--------------------");
```

새로운 값을 출력하기 위해서 터미널 화면을 clear한다. 다음으로 string에 값을 저장하기 위해서 코드를 실행하는데 이것은 RSS데이터를 참고하면 이해가 쉽다.

This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
▼<rss version="2.0">
  ▼<channel>
    <title>기상청 동네예보 웹서비스 - 서울특별시 동작구 신대방제2동 도표예보</title>
    <link>http://www.kma.go.kr/weather/main.jsp</link>
    <description>동네예보 웹서비스</description>
    <language>ko</language>
    <generator>동네예보</generator>
    <pubDate>2020년 07월 03일 (금)요일 17:00</pubDate>
   ▼<item>
      <author>기상청</author>
      <category>서울특별시 동작구 신대방제2동</category>
      <title>동네예보(도표) : 서울특별시 동작구 신대방제2동 [X=59,Y=125]</title>
      <link>http://www.kma.go.kr/weather/forecast/timeseries.jsp?searchType=INTEREST&dongCode=1159068000</link>
      <guid>http://www.kma.go.kr/weather/forecast/timeseries.jsp?searchType=INTEREST&dongCode=1159068000</guid>
     ▼<description>
       ▼<header>
          <tm>202007031700</tm>
          <ts>5</ts>
          <x>59</x>
          <y>125</y>
        </header>
       ▼<body>
         ▼<data seq="0">
            <hour>21</hour>
            <day>0</day>
            <temp>23.0</temp>
            <tmx>-999.0</tmx>
            <tmn>-999.0</tmn>
            <sky>3</sky>
            <pty>4</pty>
            <wfKor>소나기</wfKor>
            <wfEn>Shower</wfEn>
            <pop>60</pop>
            <r12>0.0</r12>
            <s12>0.0</s12>
```


- RSS에서 받아오는 데이터의 형태가 위와 같으므로 pubData에 해당하는 값을 받아오기 위해서는 <pubData>로 시작하는 문장을 찾아내서 <pubData>사이에 있는 값을 저장하면 된다. 따라서 문장을 찾아내는데 <pubData>를 이용하여 인덱스를 정하는데 이 경우 '<'의 인덱스로 각각의 인덱스가 지정된다. 이를 통해 값을 저장해야 하므로 앞의 인덱스의 경우 <pubData>의 제일 첫 부분부터 9바이트를 버리고 사이에 있는 값을 저장한다. 저장한 값을 확인하기 위해서 시리얼 모니터로 출력하고 또한 앱의 터미널로 출력해준다. 이 함수 내부에서는 이러한 형태의 코드가 계속 반복된다.

```
String hour;
int hour_start = line.indexOf(F("<hour>"));
int hour_end = line.indexOf(F("</hour>"));
hour = line.substring(hour_start + 6, hour_end);
Serial.print(F("hour: "));
Serial.println(hour);

terminal.print(hour);
terminal.println("시 기상예보 ");
```

위의 코드가 이전 코드 바로 다음에 오는 부분으로 유사한 구조를 가지고 있음을 확인할 수 있다.

```
void clockDisplay()
{
  // You can call hour(), minute(), ... at any time
  // Please see Time library examples for details

  currentTime = String(hour()) + ":" + minute() + ":" + second();
  currentDate = String(year()) + " " + month() + " " + day();

  // Send time to the App
  Blynk.virtualWrite(V3, currentTime);
  // Send date to the App
  Blynk.virtualWrite(V4, currentDate);
}
```

clockDisplay 함수의 경우 LCD로 출력되는 시간과 날짜를 만들어 가상핀을 이용하여 출력한다. Git-hub의 라이브러리를 이용하였다.(https://github.com/PaulStoffregen/Time)

```
void blindMove() {
  timeStart = millis();

  if (pinV2 == 1) {
    digitalWrite (D0, HIGH);
    digitalWrite (D1, LOW);

    Serial.println("A");
    digitalWrite (D2, HIGH);
  }

  else {
    digitalWrite (D0, LOW);
    digitalWrite (D1, HIGH);

    Serial.println("B");
    digitalWrite (D2, HIGH);
  }
  int timeStoE = 0;
  while (timeStoE < 5000) {
    timeStoE = millis() - timeStart;
    Serial.println("C");
    Serial.println(timeStoE);
    delay(1000);
  }
  digitalWrite (D2, LOW);

}
```

blindMove 함수는 모터의 동작을 제어한다. 또한 모터가 동작할 시간을 지정하여 지정한 시간 동안만 모터가 동작하도록 확인하여 중단시켜주는 역할도 겸한다.

```
BLYNK_WRITE (V1) {//날씨에 따른 안내
  int pinValue = param.asInt ();
  if (pinValue == 1) {
    get_weather();
    terminal.flush();

    if (pty0 != '0') {
      terminal.print(F("\n날씨가 흐리니 창문을 닫아주세요."));
    }
    else {
      terminal.print(F("\n날씨가 맑으니 환기하세요."));
    }

    int winCh = winCheck();
    if (winCh == 0) {
      terminal.print(F("\n창문이 닫혀 있습니다."));
    }
    else {
      terminal.print(F("\n창문이 열려 있습니다."));
    }
    terminal.flush();
  }
```

가상핀 1번의 값이 변하면 동작하며 RSS에서 읽어 저장해둔 값을 이용하여 각각의 문구를 출력한다. Pty0에 저장된 값은 날씨가 맑으면 0, 비나 눈이 오면 이외의 값이 저장된다. 이를 통해서 판별한다. 또한 창문의 개폐여부를 winCheck을 통해 읽어 들여 문구를 출력한다.

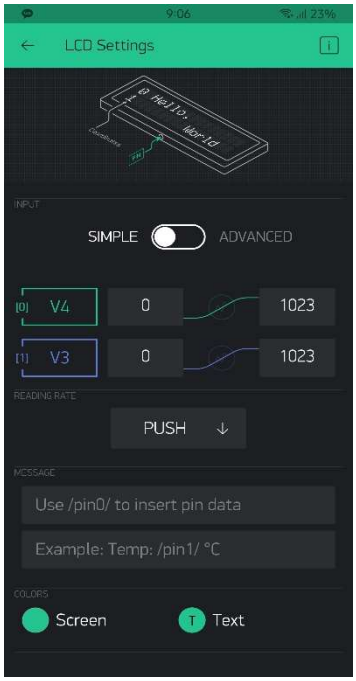
```
int winCheck() {
  int d = dist();
  int distance_error = disToWin + 5;
  if (d > distance_error) {
    return 1;
  }
  else {
    return 0;
  }
}
```

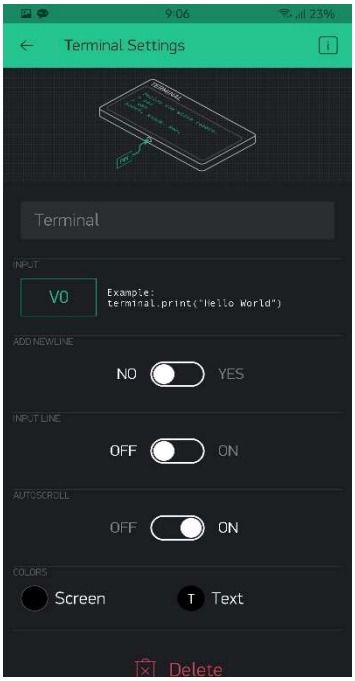
winCheck 함수의 경우 디폴트값으로 지정된 창문과의 거리와 5cm 이상 더 멀다고 확인되면 문이 열렸다고 판단한다. 이것은 초음파 센서 오차를 보정하기 위함이다.

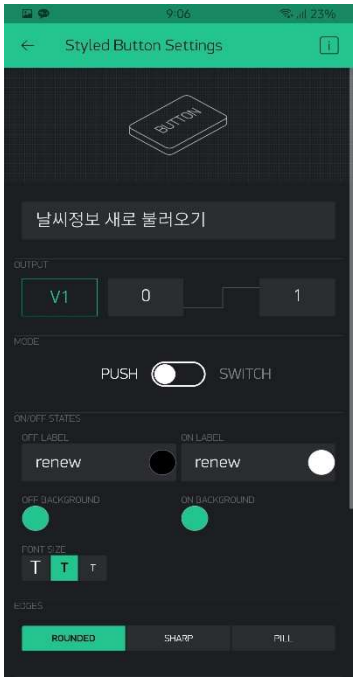
```
float dist() {
  digitalWrite(D3, LOW);
  digitalWrite(D4, LOW);
  delayMicroseconds(2);
  digitalWrite(D3, HIGH);
  delayMicroseconds(10);
  digitalWrite(D3, LOW);


  // echoPin 이 HIGH를 유지한 시간을 저장 한다.
  unsigned long duration = pulseIn(D4, HIGH);
  // HIGH 였을 때 시간(초음파가 보냈다가 다시 들어온 시간)을 가지고 거리를 계산 한다.
  float distance = ((float)(340 * duration) / 10000) / 2;

  Serial.print("거리 : ");
  Serial.print(distance);
  Serial.println("cm");

  return distance;
}
```

dist 함수는 초음파 센서를 이용하여 거리를 측정하는 함수이다.

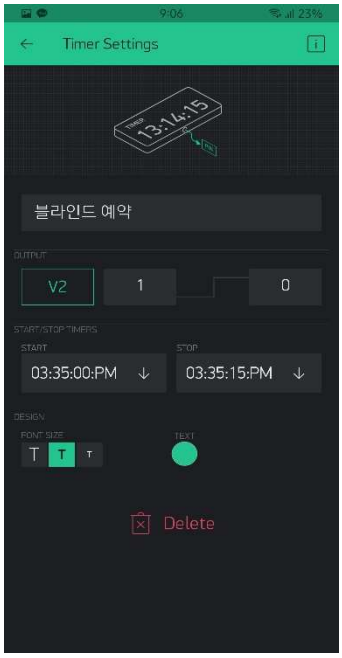
다음으로 Blynk앱의 설정과 각 버튼의 설정은 다음과 같다.







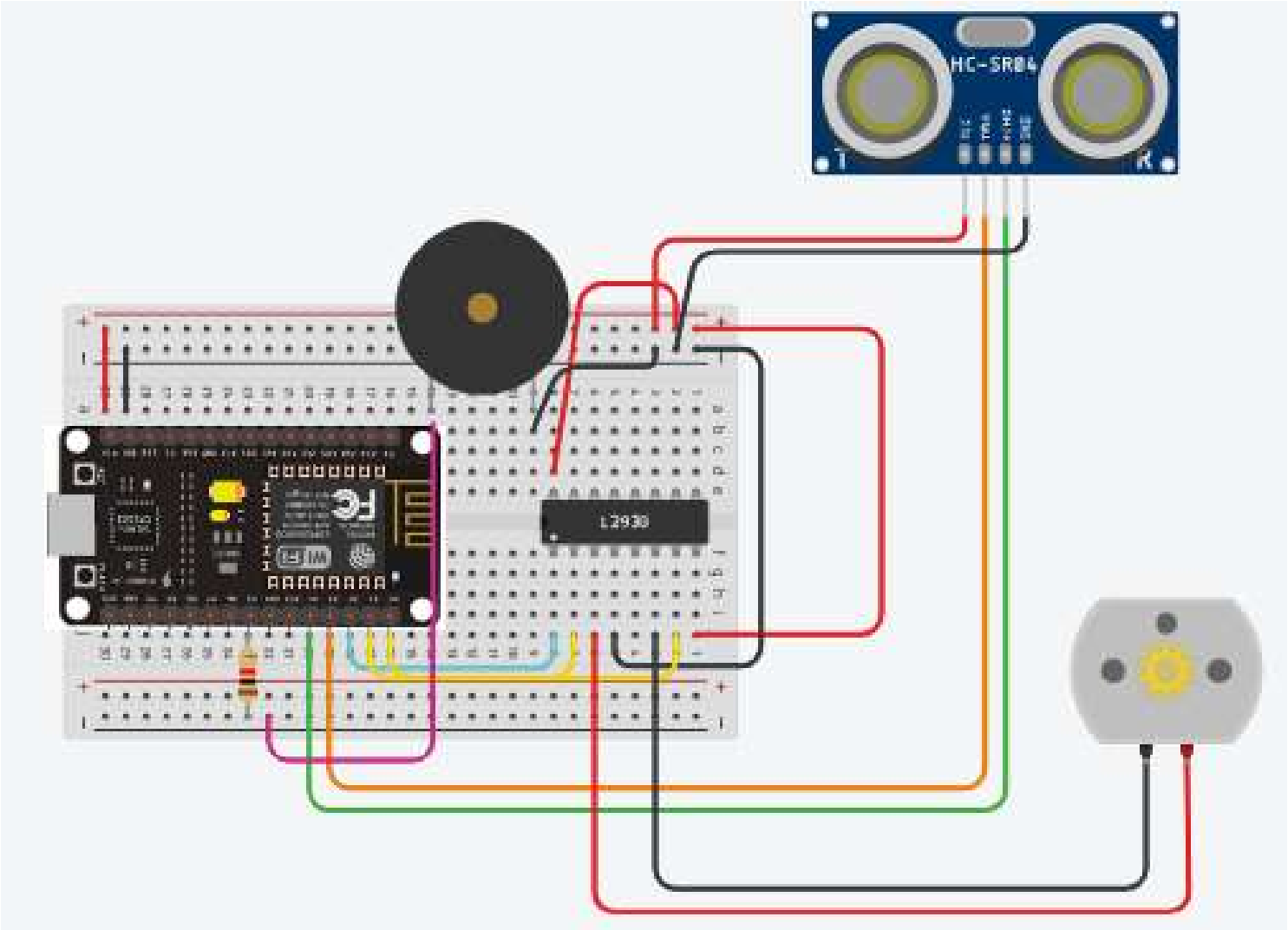
왼쪽부터 blynk앱의 메인화면이며 각각의 버튼에 연결된 가상핀의 번호를 확인할 수 있다. 다음으로 LCD화면에 연결된 가상핀을 확인할 수 있으며 이는 날짜와 시간을 출력한다. 그 옆에는 터미널 화면이 있으며 터미널 입력은 사용하지 않는다. 출력만 해주는 용도로 사용하며 일기예보와 여러 정보를 출력해준다. 마지막으로 날씨정보 새로 불러오기 버튼은 가상핀 1번과 연결되어 get_weather를 실행시키고 이외의 다른 정보를 출력하도록 한다.



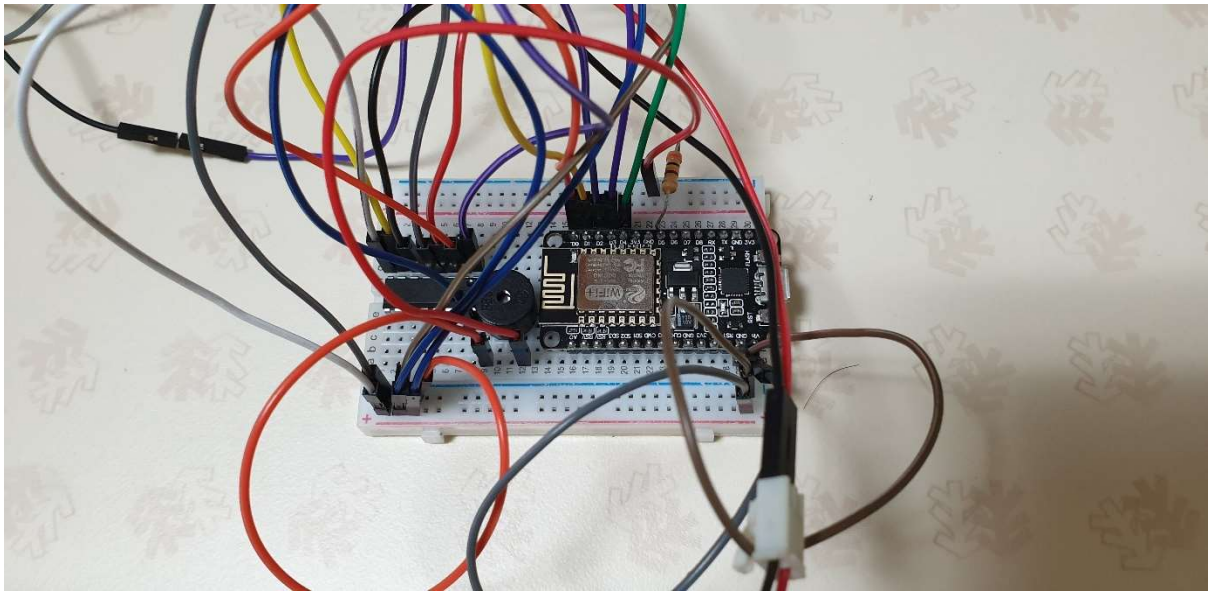
왼쪽부터 블라인드 예약버튼으로 이것은 가상핀 2번에 연결되며 블라인드를 자동으로 제어하는 역할이다. START의 시간이 되면 블라인드가 올라가도록 STOP의 시간이 되면 내려

가도록 핀의 신호를 조정해 주었다. 다음으로 블라인드 수동제어 버튼으로 내리기 올리기로 스위칭 되도록 해주었고 해당하는 동작을 하도록 가상핀의 신호를 지정해준다. 마지막으로 창문 거리측정 버튼으로 버튼이 PUSH되면 함수가 작동해 거리를 측정하도록 한다.

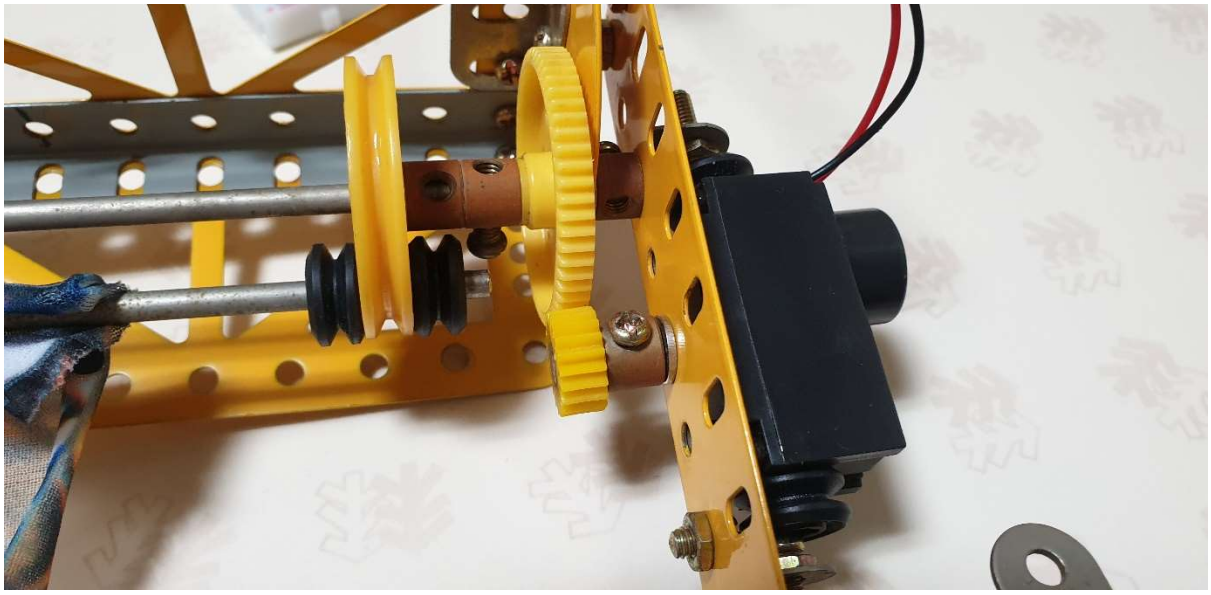
4. 하드웨어 해설



D1핀과 D2핀을 이용하여 모터를 양방향으로 제어한다. 이것은 앞서 설명한 IC칩을 이용하여 제어하는 방법이다. D3핀과 D4핀은 초음파 센서를 제어하는데 사용한다. D3핀을 Trig핀에 D4핀을 Echo 핀에 사용한다. 다음으로 D5핀은 피에조 부저를 제어하는데 사용한다. 부저에 들어가는 신호는 저항을 연결하여 입력하도록 한다.

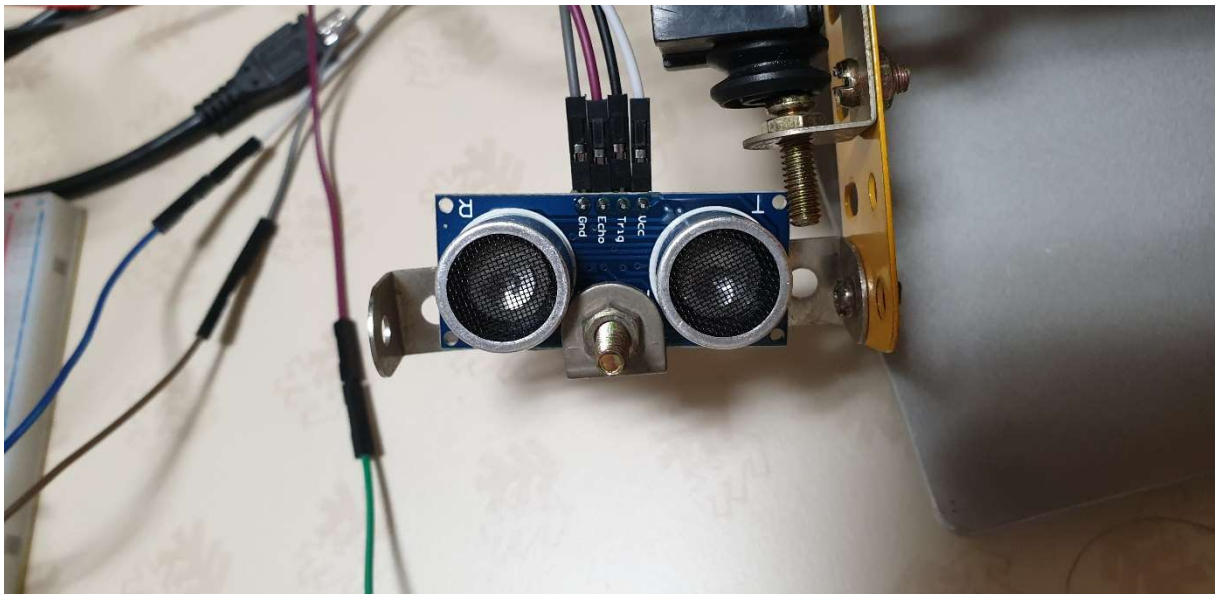


실제 회로의 모습



서보모터의 출력이 약했기 때문에 기어를 이용하여 작은 힘으로 큰 출력을 낼 수 있도록 연결하였다. 추가적인 부품을 부착하여 모터 전압을 높여주면 더 무겁고 큰 블라인드도 사용 가능하다.

2개의 봉을 이용하여 커튼을 감을 수 있도록 설계



초음파 센서는 창문 쪽을 바라보도록 설치되어 창문과 블라인드 사이의 거리를 측정하여 창문이 열려 있고 닫혀 있음을 확인할 수 있다. 초음파 센서의 경우 너무 가까운 거리를 측정하게 되면 오류가 발생하기 때문에 창문이 닫혀 있을 때 창문과의 거리가 충분히 떨어져 있도록 장착해야 함에 유의하여 설치한다.

5. 맺음말

이번 자동 블라인드의 경우 충분히 가벼운 블라인드를 사용하여 동작하였지만 무거운 블라인드를 사용하려면 모터의 출력이 더 커야 할 것입니다. 따라서 추가적인 모터 드라이버를 이용하여 모터전력을 추가적으로 공급해주고 더 힘이 강한 모터를 사용했으면 더 실용적인 블라인드를 만들 수 있었을 것 같습니다.

또 현재 블라인드는 블라인드가 내려가는 도중에 동작을 멈출 수 없습니다. 실제 블라인드를 사용할 때에는 반만 내리고 사용하거나 내리는 도중 다시 올릴 수도 있기 때문에 이러한 동작을 보완할 수 있으면 더 좋은 결과물이 나올 수 있었을 것 같습니다.

하지만 이번 스마트 블라인드의 경우 IoT와 일기예보와의 연동에 더 중점을 두고 설계하였기 때문에 그 부분을 더 관심있게 평가해 주시면 감사하겠습니다.

이상으로 기말고사 프로젝트 '스마트 블라인드'에 대한 보고서를 마치도록 합니다. 읽어주셔서 감사합니다.