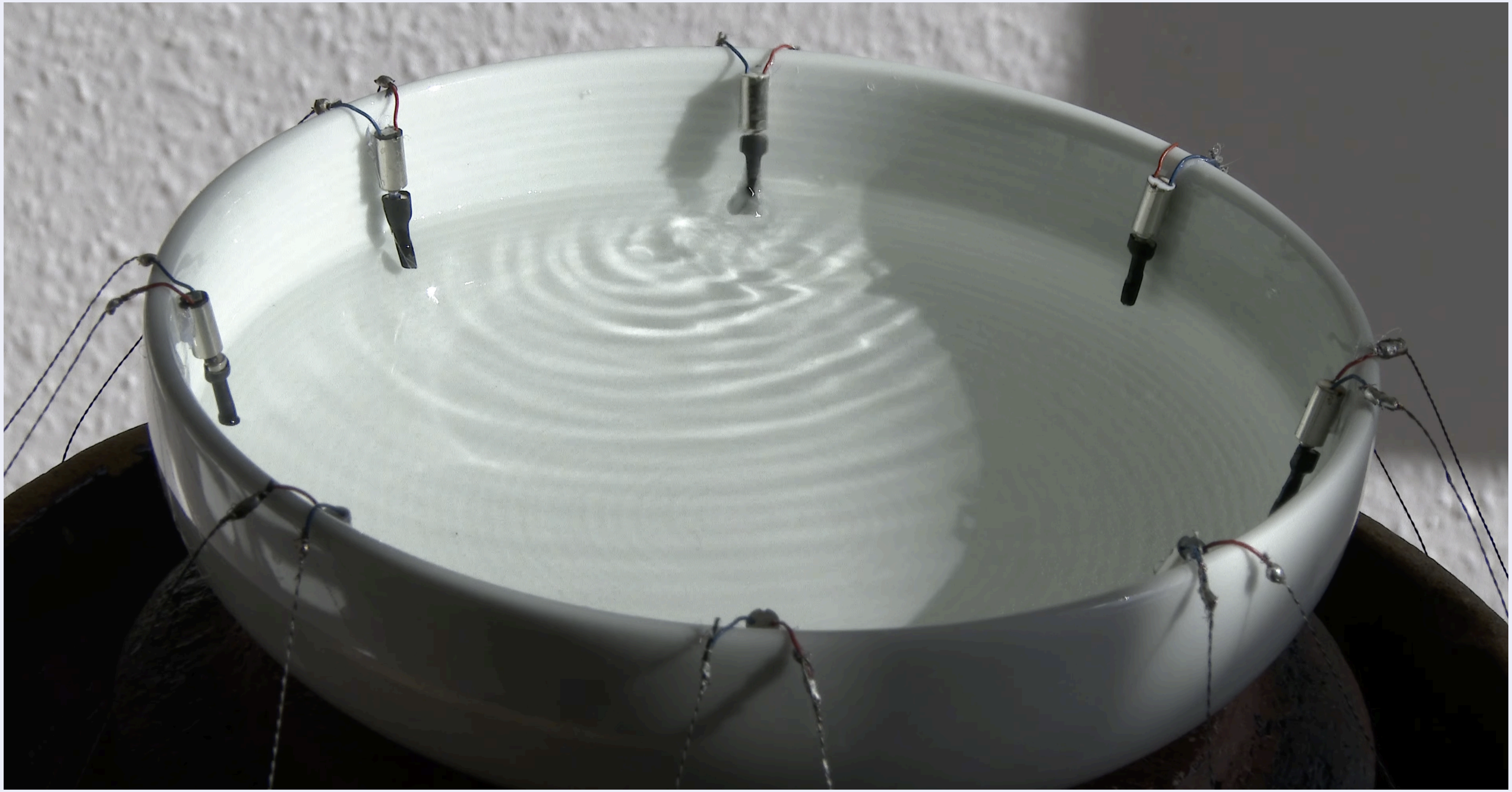
Final presentation

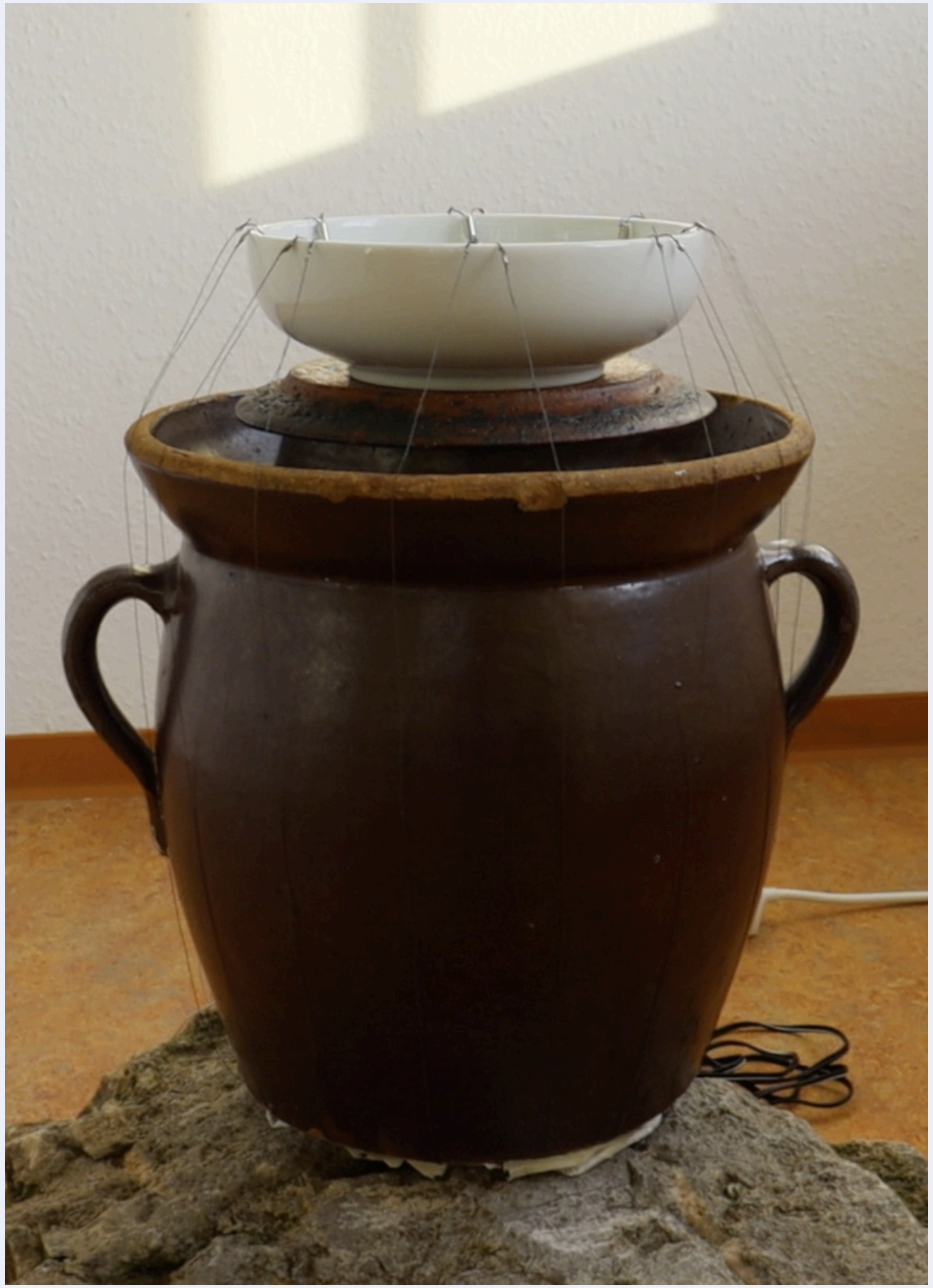
Bauhaus Universty Weimar SS 24
Media Enviroment
guest.-Prof. QUADRATURE
Project Modul : ‘Data as a artistic Material’

8 = ∞

DAHYE SEO
moonwatcher1018@gmail.com

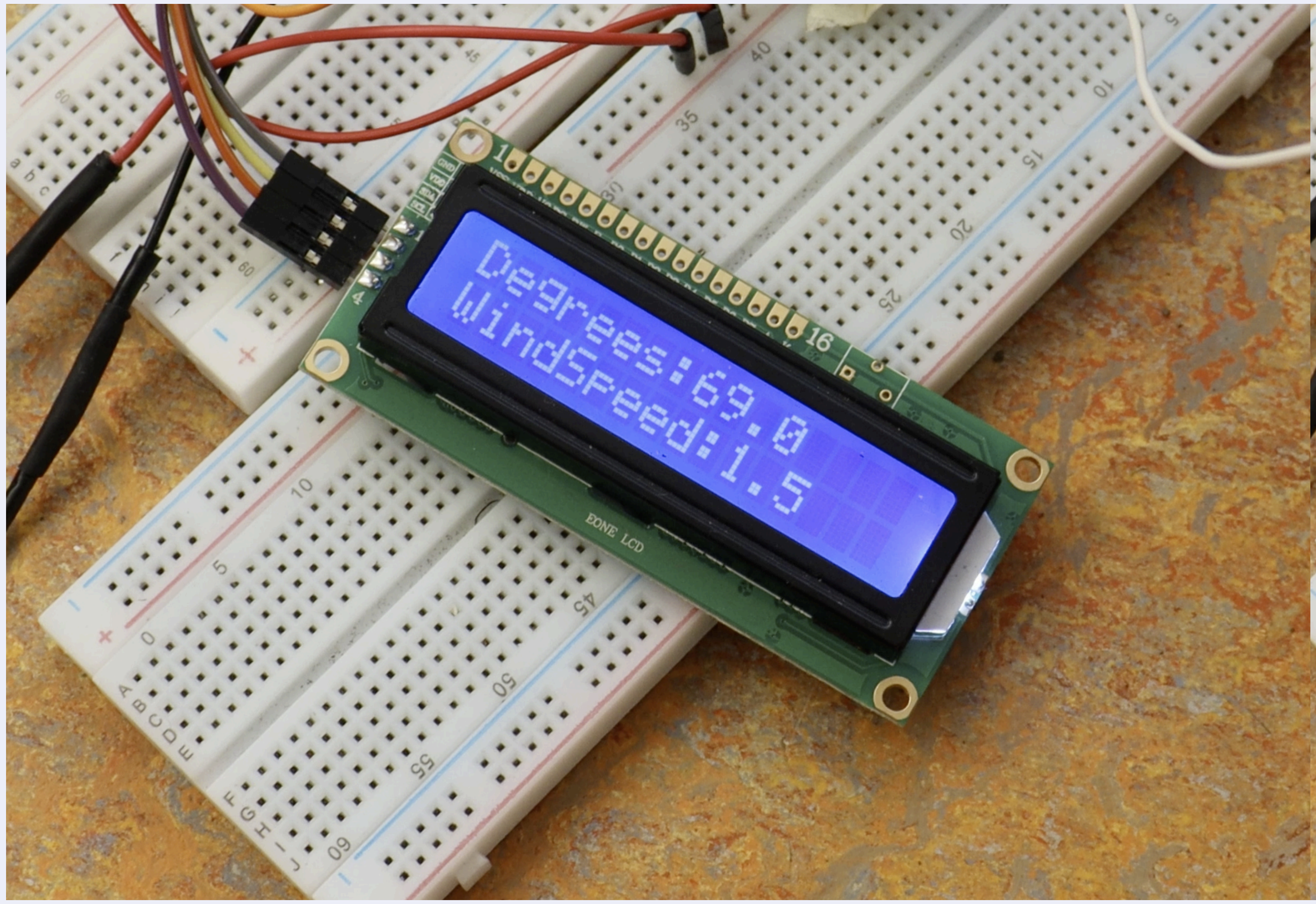


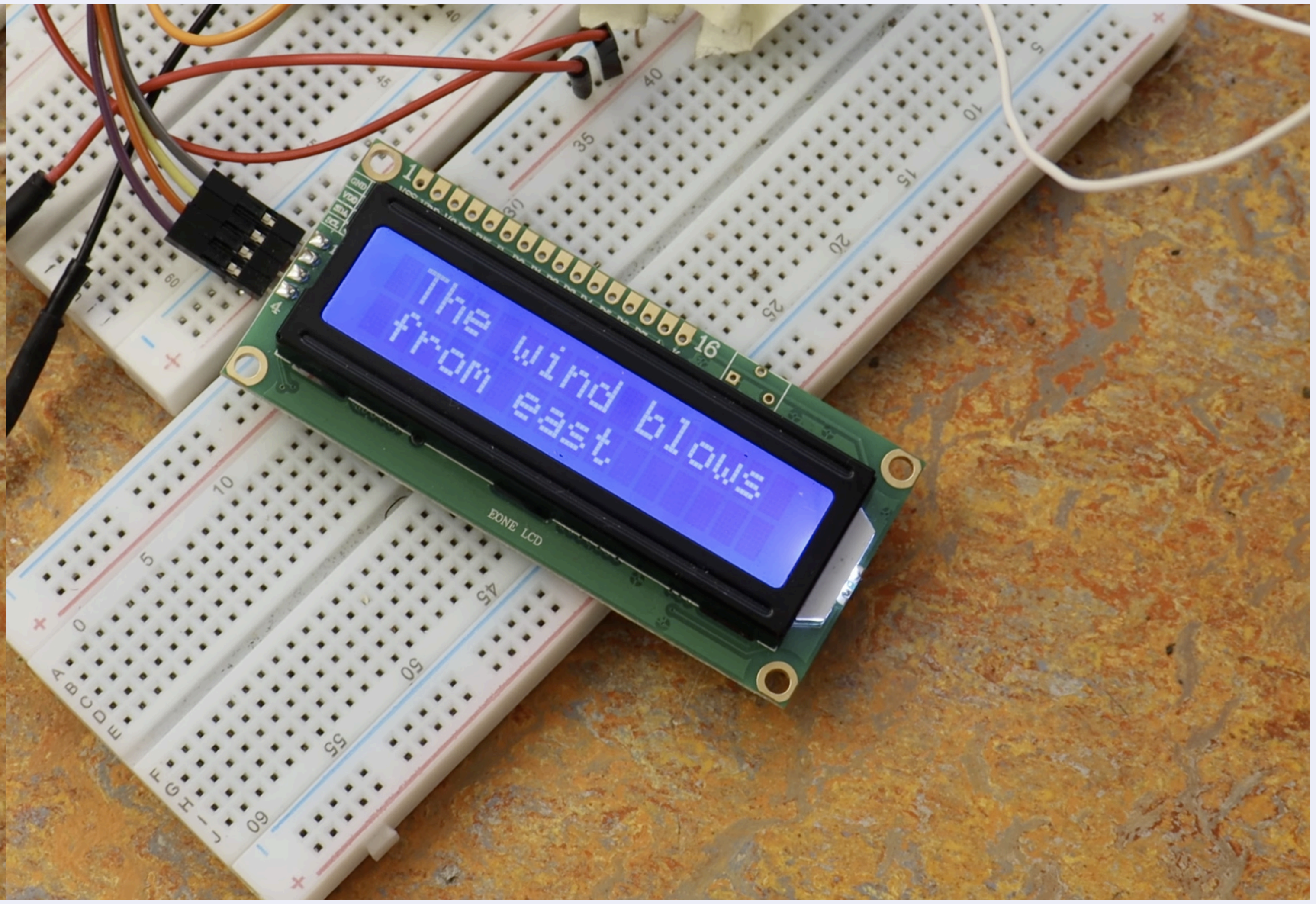
< 8 = ∞ > is an interactive installation that conveys real-time wind data from Anseong, South Korea, the artist's hometown, through vibrations in water. The work originates from the artist's simple desire to simultaneously feel the breeze that touches the parents' cheeks.



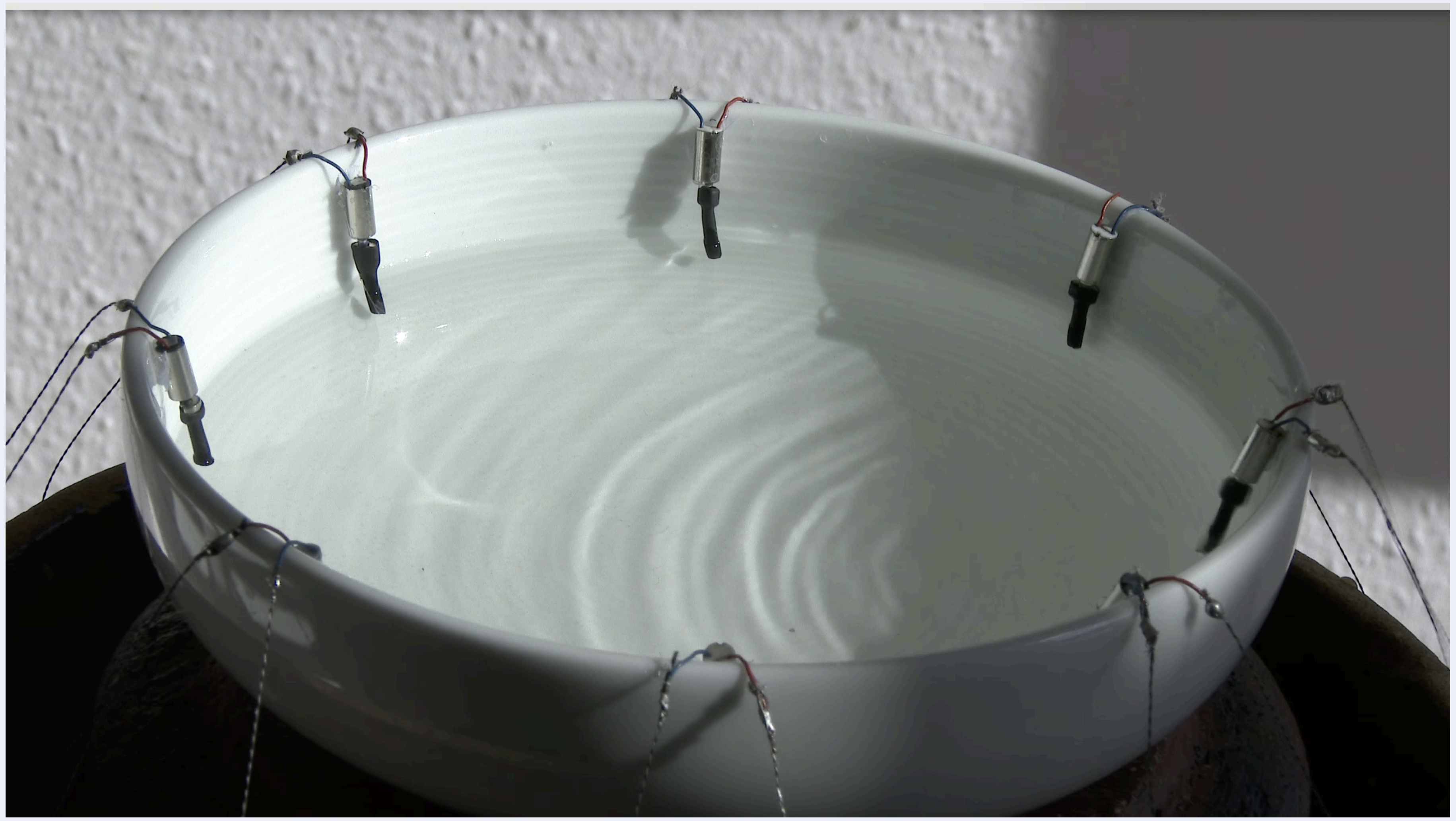
Aesthetically, it draws inspiration from the traditional Korean ritual of the 'Jeonghwaseu'* water ceremony, predominantly performed by women.

** The 'Jeonghwaseu' water ceremony is a traditional Korean ritual where purified water, often gathered at dawn, is used for cleansing and blessing purposes. It is a practice imbued with cultural and spiritual significance, typically carried out by women to bring peace and protection to their households.*


Degrees:69.0
WindSpeed:1.5


The wind blows
from east

8 = ∞



VIDEO

Short Description

8 = ∞ is an interactive real-time art installation that replicates the wind data from the artist's hometown of Anseong, South Korea, as vibrations in a bowl of water. Utilising an **API** provided by the Korea Meteorological Administration, it receives real-time (last minute) wind direction and speed data through an **ESP32** microcontroller. This wind data is mapped to **8 motors** positioned around the bowl in **8 directions** (North, Northeast, East, Southeast, South, Southwest, West, Northwest). The rotation speed of these motors varies according to the wind speed, creating a dynamic representation of the wind conditions.

Technical equipment Provided by artist

- Traditional Korean jar
- white bowl with 8 motors installed as conductive threads
- Brett board with ESP32 and cables designed
- 2 X 5v power sources : one for ESP32 and another to drive the 8 motors

General room requirements

As long as there is enough light, this piece doesn't require any special conditions, but it prefers to be near a window with natural light and a breeze from the site.

Set-up time : 1 day

More Details

Concept

'Jeong-hwa-su'

정화수

Water drawn from a well early in the morning

Purified Water in Korean Tradition

Humans cannot sustain life without water, so clear water used for drinking has become an object or medium of religious practice.
Women typically draw and offer Cheonghwaseu in a bowl, praying to the spirits for the health and harmony of their families.

Cheonghwaseu is clean water that women irregularly draw early in the morning with sincere devotion to make small wishes. It holds significance as an offering or sacrificial item. The water, collected at dawn with care and devotion, is considered sacred and symbolically valuable as a pure offering to spirits or gods. **Women usually offer Cheonghwaseu and make wishes in places such as by the well, in front of the jangdokdae (fermentation pots), or in the kitchen.** During the Joseon Dynasty, heavily influenced by Confucian thought, these were spaces primarily associated with women.

The objects of worship are nature deities like trees or rocks, or household gods such as the Seven Stars (Chilseong), the Kitchen God (Jo Wang), or the Earth God (Teoju), who are traditionally venerated by women. **Therefore, the space for prayer and the spirits invoked through Cheonghwaseu are closely related to women.**

Concept

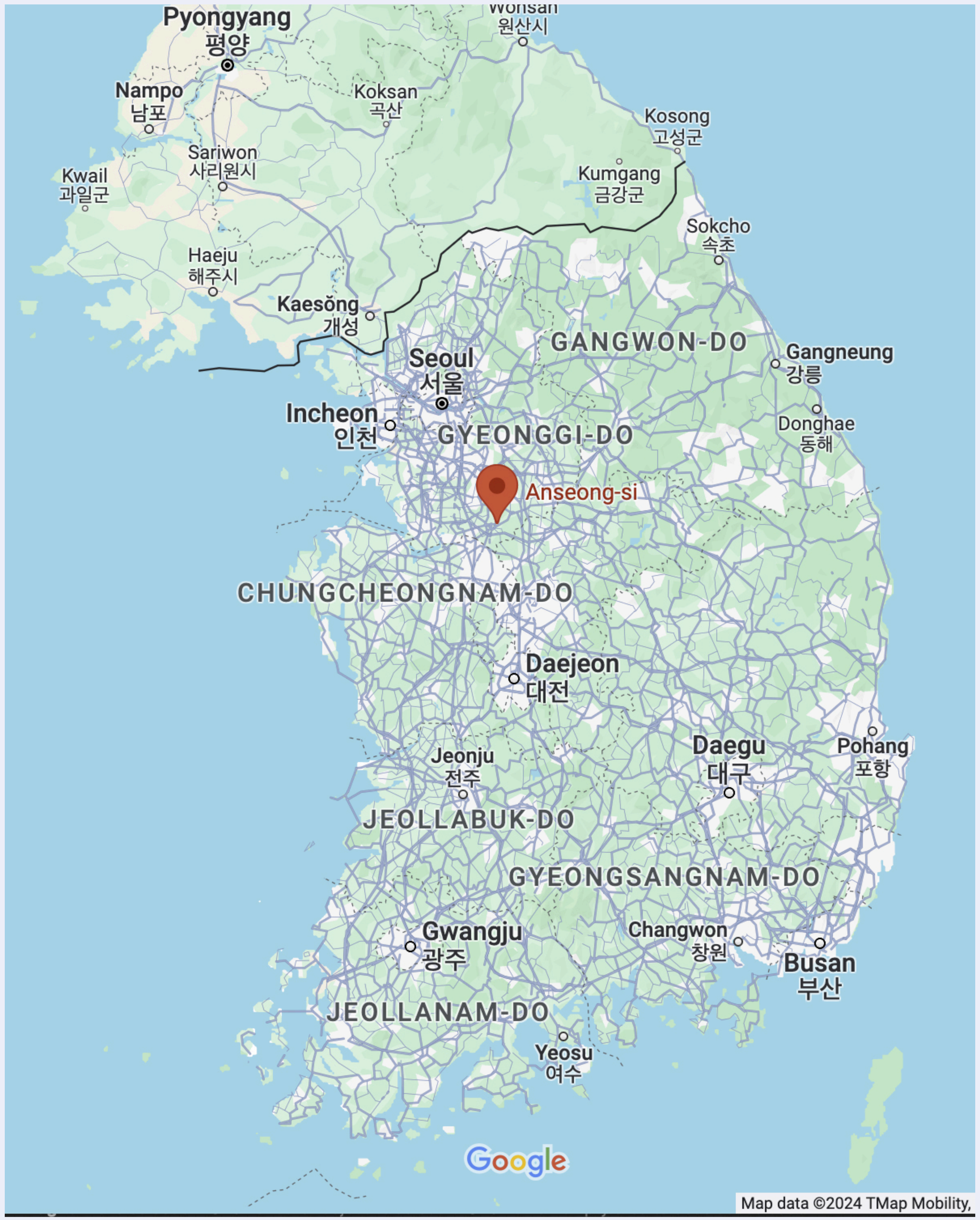
8 = ∞ aim to connect the two cities, **Anseong** and **Weimar**,

through a **purified water ritual** reinterpreted with the elements of **Wind, Water**, and **Data**

Wind Data

from

Anseong, South Korea



Concept

Herzquelle in Oberweimar, Germany

Concept

Wind Data
of
Anseong, South Korea.

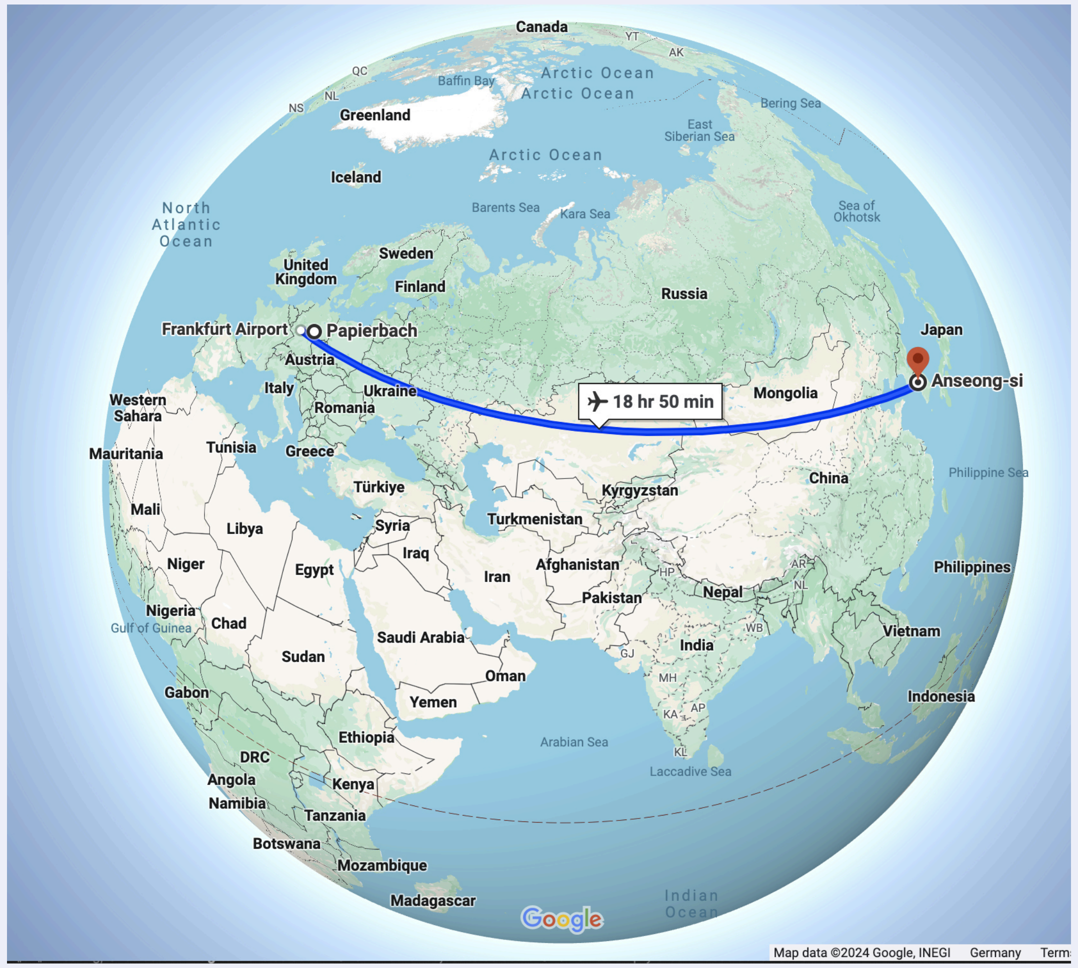
Vibration of Water
from
Herzquelle in Weimar

Concept

Wind Data

of

Anseong, South Korea.



Vibration of Water

from

Herzquelle in Weimar

How to get a real time wind data of Anseong ?

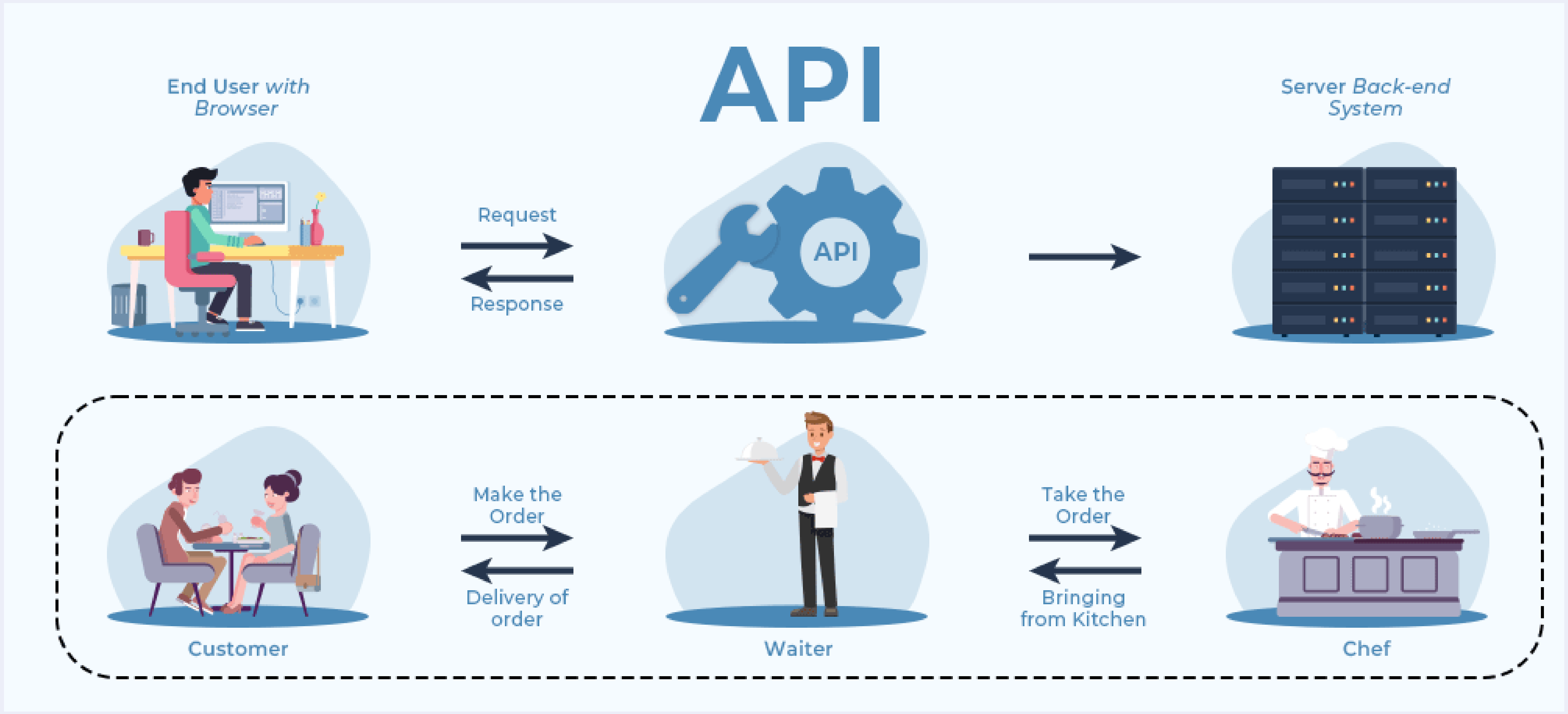
API DATA

from

National Weather Service in South Korea

what is an API – (Application Programming Interface)?

An API (Application Programming Interface) is a set of rules and tools that allows different software applications to communicate with each other.



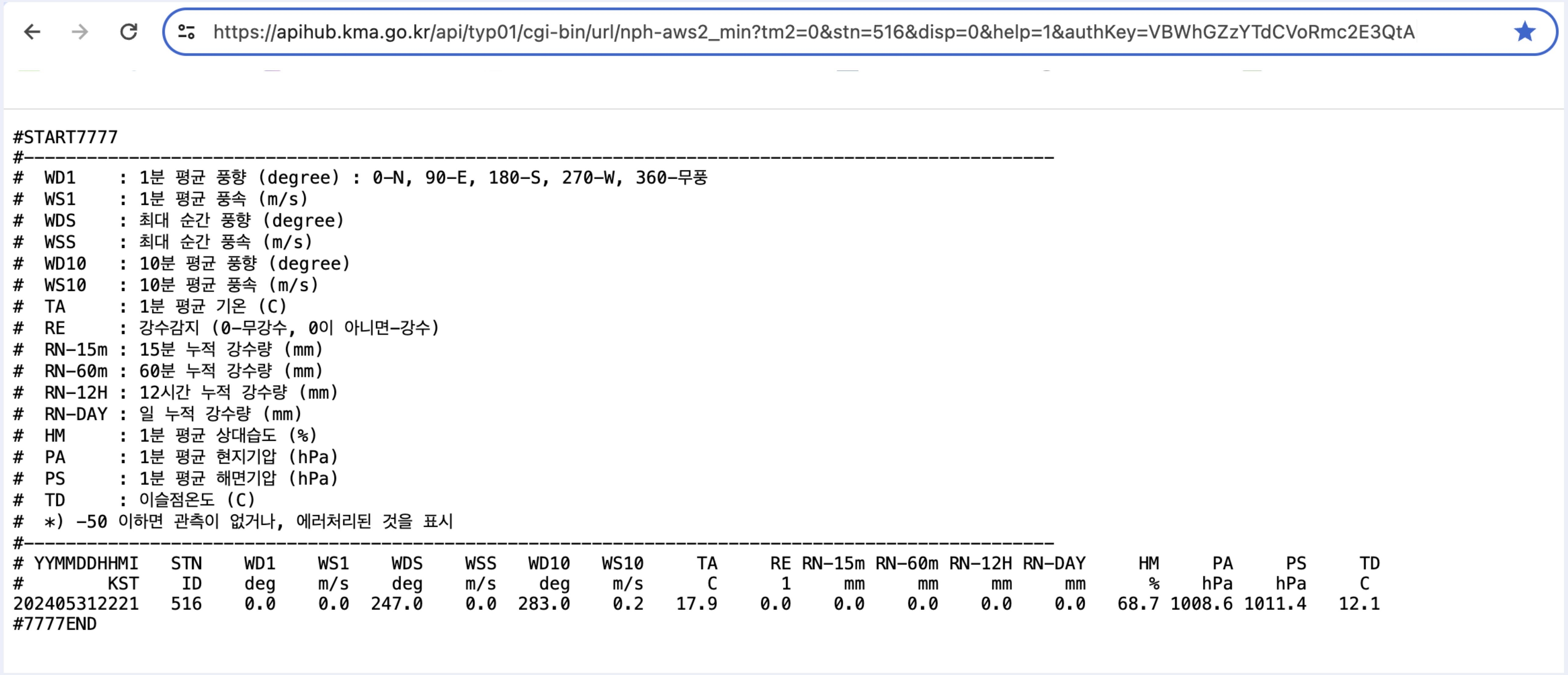
source: https://www.geeksforgeeks.org/what-is-an-api/

AWS

(Automatic Weather System)

Disaster prevention weather observation refers to ground observation conducted to prevent natural disasters caused by meteorological phenomena such as earthquakes, typhoons, floods, and droughts. In order to eliminate observation gaps and identify local weather phenomena, **automatic weather observation equipment (AWS, Automatic Weather System)** is installed at approximately 510 locations across the country to automatically observe.

API request : AWS data of Anseong

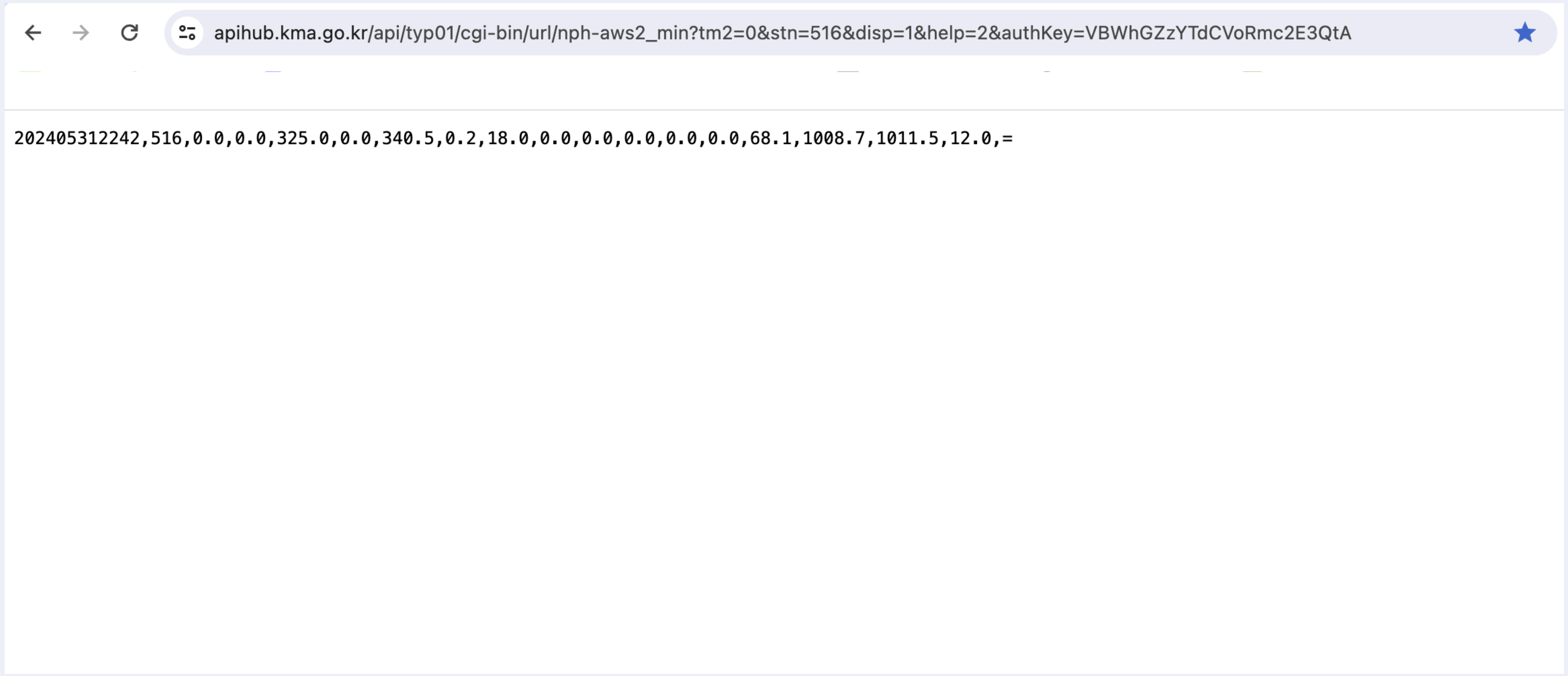


https://apihub.kma.go.kr/api/typ01/cgi-bin/url/nph-aws2_min?tm2=0&stn=516&disp=0&help=1&authKey=VBWhGZzYTdCVoRmc2E3QtA

```
#START7777
#--------------------------------------------------------------------------------------------------
#  WD1    : 1분 평균 풍향 (degree) : 0-N, 90-E, 180-S, 270-W, 360-무풍
#  WS1    : 1분 평균 풍속 (m/s)
#  WDS    : 최대 순간 풍향 (degree)
#  WSS    : 최대 순간 풍속 (m/s)
#  WD10   : 10분 평균 풍향 (degree)
#  WS10   : 10분 평균 풍속 (m/s)
#  TA     : 1분 평균 기온 (C)
#  RE     : 강수감지 (0-무강수, 0이 아니면-강수)
#  RN-15m : 15분 누적 강수량 (mm)
#  RN-60m : 60분 누적 강수량 (mm)
#  RN-12H : 12시간 누적 강수량 (mm)
#  RN-DAY : 일 누적 강수량 (mm)
#  HM     : 1분 평균 상대습도 (%)
#  PA     : 1분 평균 현지기압 (hPa)
#  PS     : 1분 평균 해면기압 (hPa)
#  TD     : 이슬점온도 (C)
#  *) -50 이하면 관측이 없거나, 에러처리된 것을 표시
#--------------------------------------------------------------------------------------------------
# YYMMDDHHMI   STN    WD1    WS1    WDS    WSS   WD10   WS10      TA     RE RN-15m RN-60m RN-12H RN-DAY     HM     PA     PS     TD
#        KST    ID    deg    m/s    deg    m/s    deg    m/s       C      1     mm     mm     mm     mm      %    hPa    hPa      C
202405312221   516    0.0    0.0  247.0    0.0  283.0    0.2    17.9    0.0    0.0    0.0    0.0    0.0   68.7 1008.6 1011.4   12.1
#7777END
```

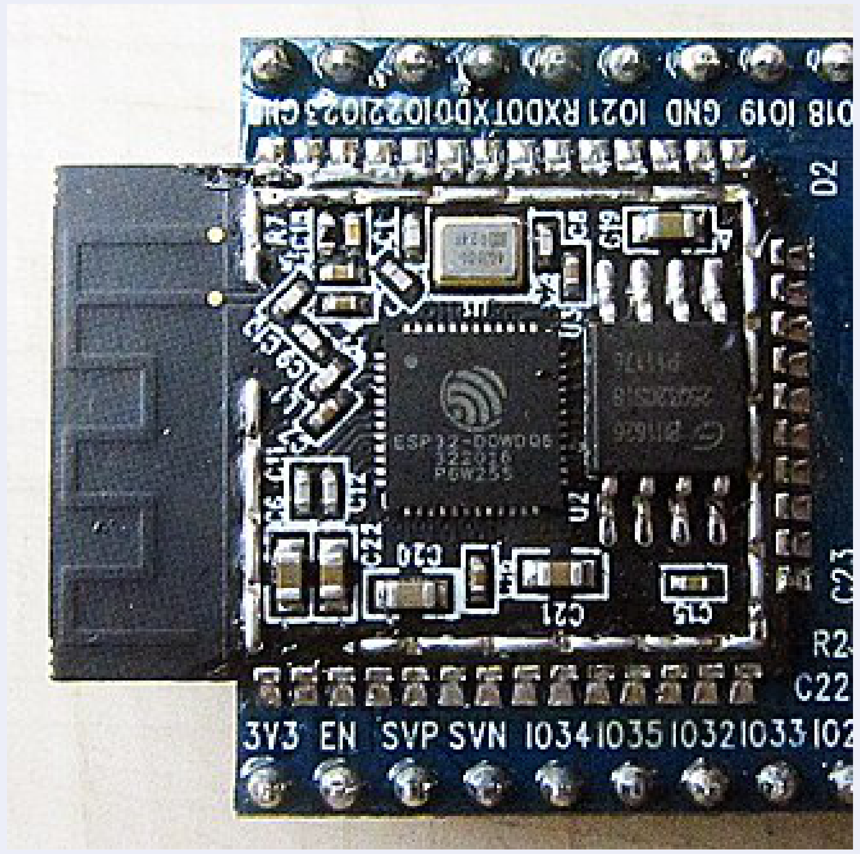
https://apihub.kma.go.kr/api/typ01/cgi-bin/url/nph-aws2_min?tm2=0&stn=516&disp=0&help=1&authKey=VBWhGZzYTdCVoRmc2E3QtA

Scientific/Technical



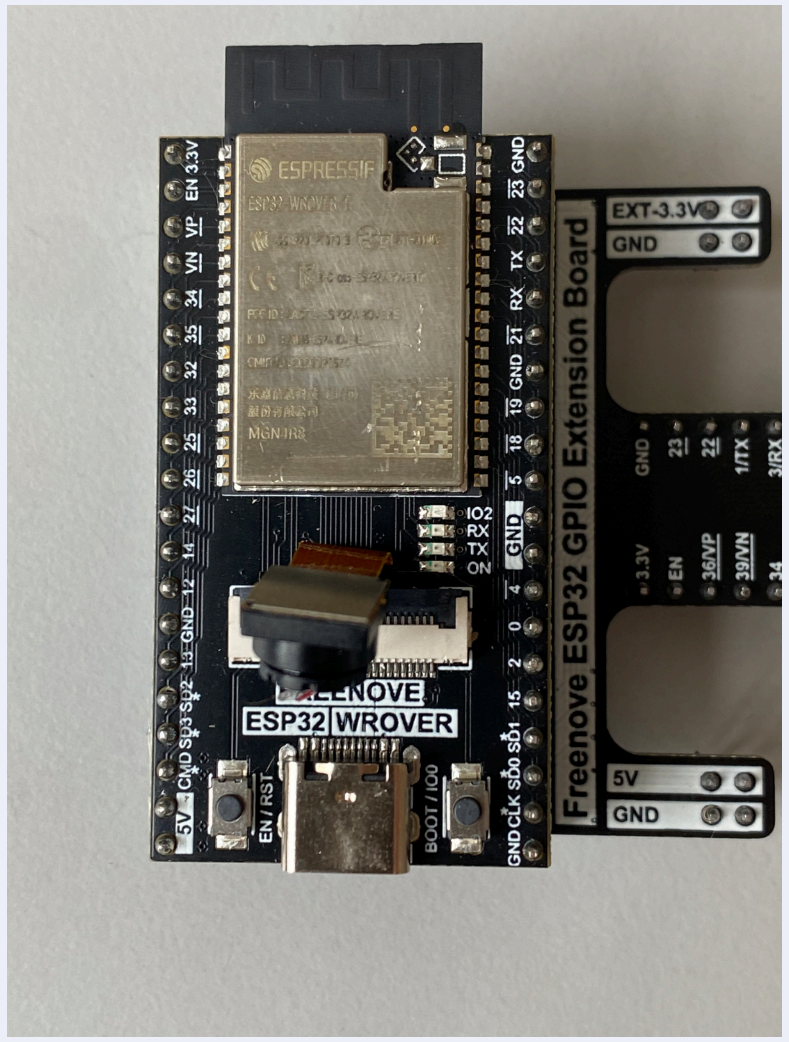
```
202405312242,516,0.0,0.0,325.0,0.0,340.5,0.2,18.0,0.0,0.0,0.0,0.0,0.0,68.1,1008.7,1011.5,12.0,=
```

https://apihub.kma.go.kr/api/typ01/cgi-bin/url/nph-aws2_min?tm2=0&stn=516&disp=1&help=2&authKey=VBWhGZzYTdCVoRmc2E3QtA


CHI IO23 IO22 TXD0 RXD0 IO21 GND IO19 IO18
3V3 EN SVP SVN IO34 IO35 IO32 IO33
ESP32-D0WDQ6

ESP 32

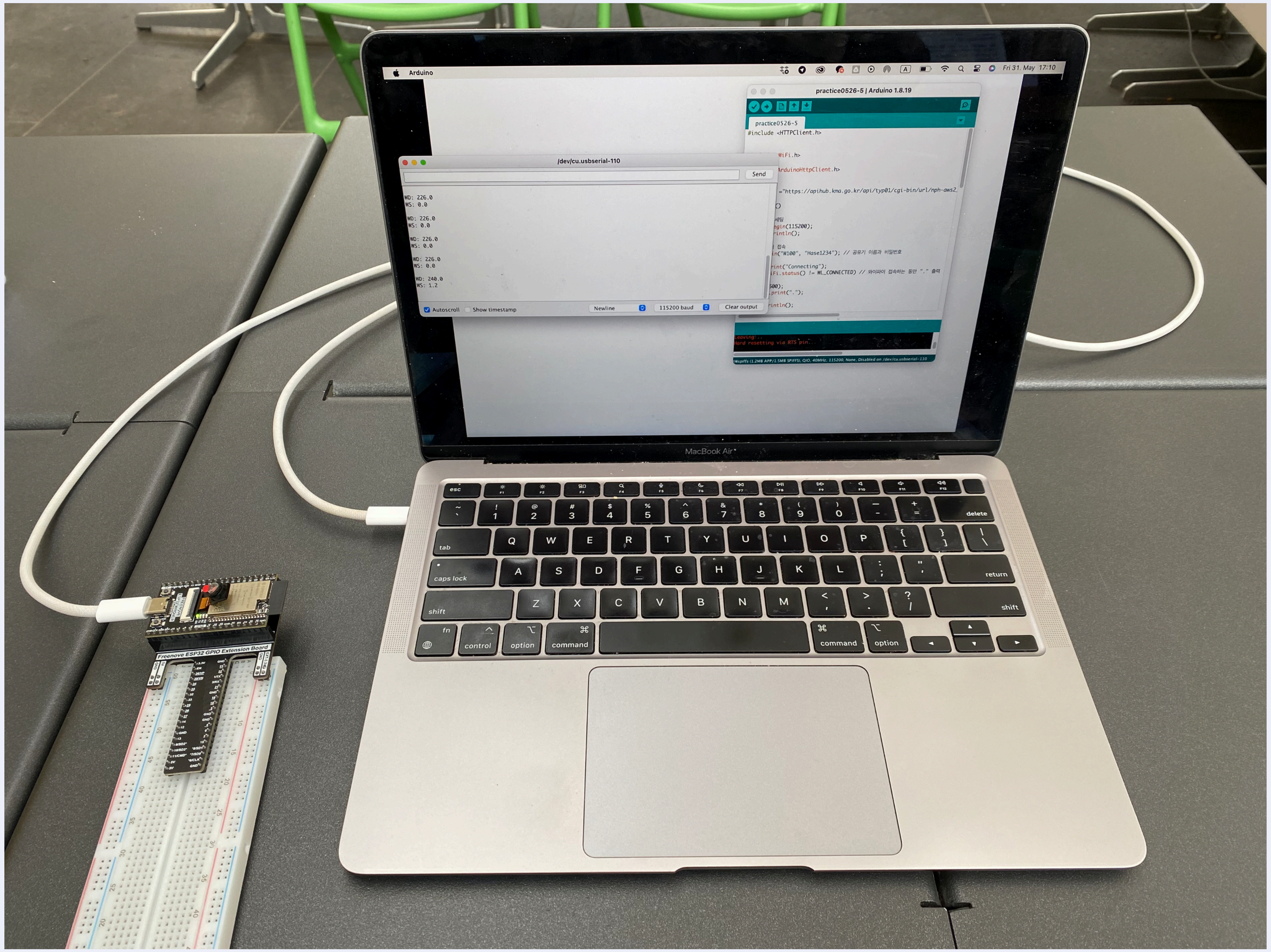
ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth



ESP 32 _ Freenove Wrover

Scientific/Technical

Physical Computing



```
WD: 226.0
WS: 0.0

WD: 226.0
WS: 0.0

WD: 226.0
WS: 0.0

WD: 226.0
WS: 0.0

WD: 240.0
WS: 1.2
```

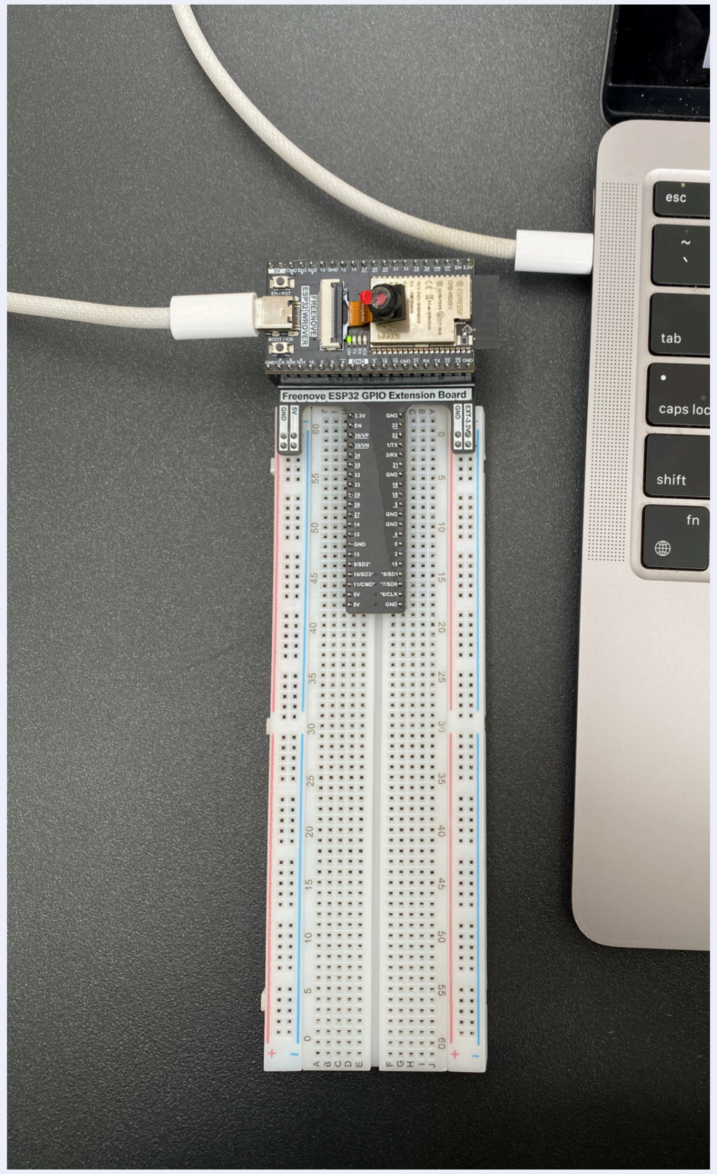
```
#include <HTTPClient.h>

WiFi.h>

ArduinoHttpClient.h>

="https://apihub.kma.go.kr/api/typ01/cgi-bin/url/nph-aws2_

()

세팅
egin(115200);
rintln();

 접속
in("W100", "Hase1234"); // 공유기 이름과 비밀번호

rint("Connecting");
iFi.status() != WL_CONNECTED) // 와이파이 접속하는 동안 "." 출력

500);
.print(".");

rintln();
```

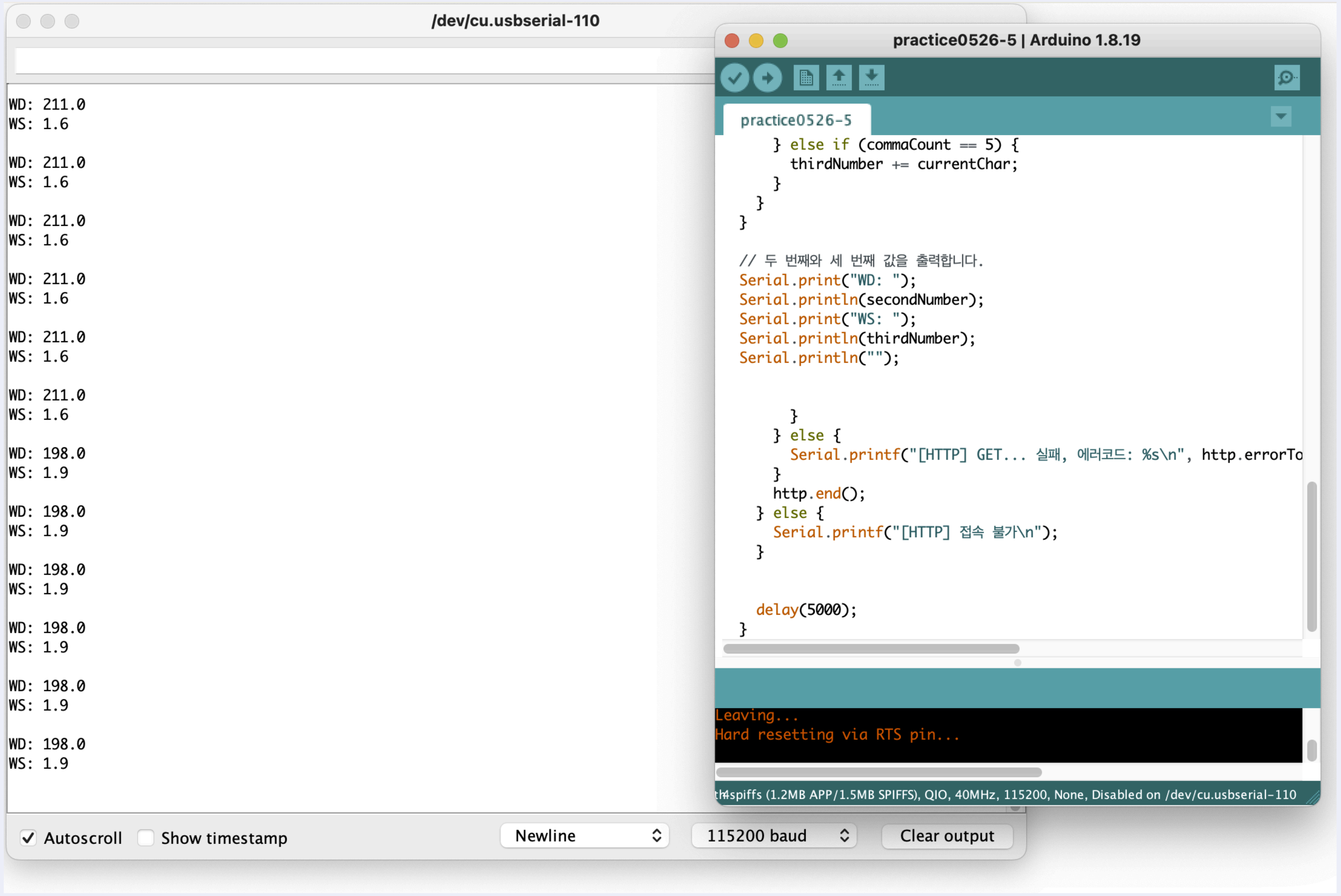
```
Leaving...
Hard resetting via RTS pin...
```

```
spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 40MHz, 115200, None, Disabled on /dev/cu.usbserial-110
```

Scientific/Technical

Physical Computing





/dev/cu.usbserial-110

```
WD: 211.0
WS: 1.6

WD: 211.0
WS: 1.6

WD: 211.0
WS: 1.6

WD: 211.0
WS: 1.6

WD: 211.0
WS: 1.6

WD: 211.0
WS: 1.6

WD: 198.0
WS: 1.9

WD: 198.0
WS: 1.9

WD: 198.0
WS: 1.9

WD: 198.0
WS: 1.9

WD: 198.0
WS: 1.9

WD: 198.0
WS: 1.9
```

practice0526-5 | Arduino 1.8.19

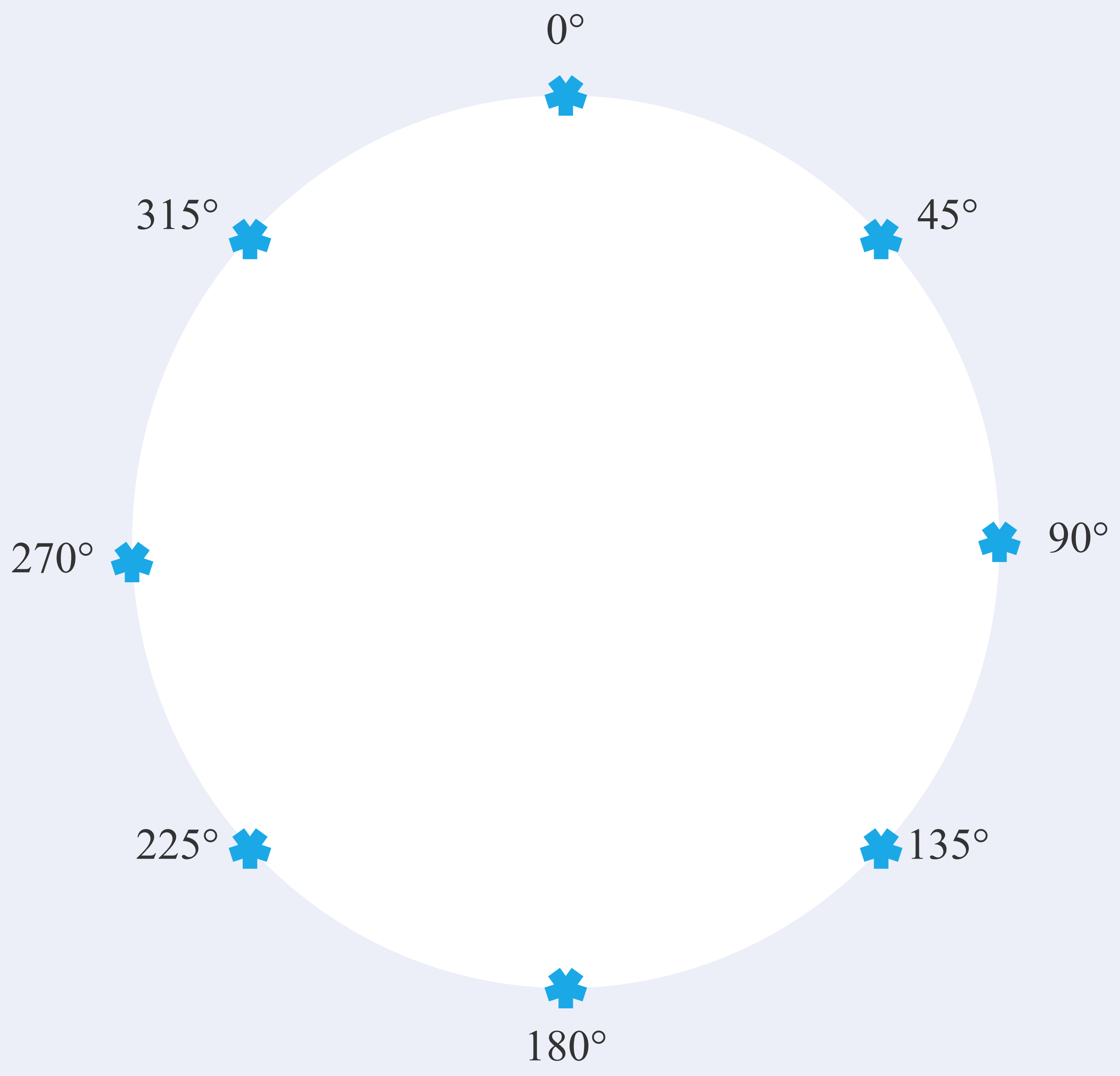
practice0526-5

```
    } else if (commaCount == 5) {
      thirdNumber += currentChar;
    }
  }
}

// 두 번째와 세 번째 값을 출력합니다.
Serial.print("WD: ");
Serial.println(secondNumber);
Serial.print("WS: ");
Serial.println(thirdNumber);
Serial.println("");


      }
    } else {
      Serial.printf("[HTTP] GET... 실패, 에러코드: %s\n", http.errorTo
    }
    http.end();
  } else {
    Serial.printf("[HTTP] 접속 불가\n");
  }


  delay(5000);
}
```

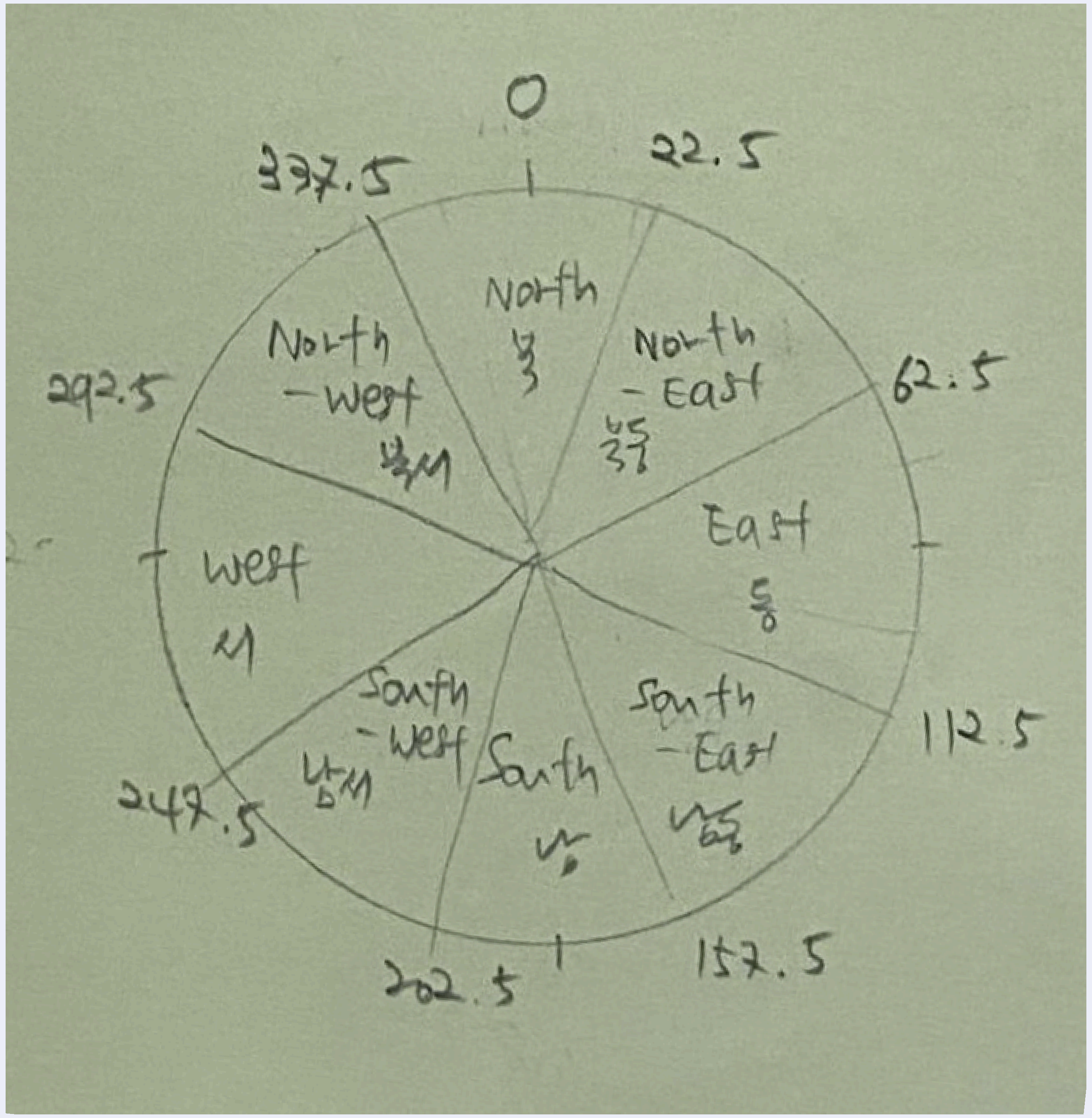
```
Leaving...
Hard resetting via RTS pin...
```

th4spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 40MHz, 115200, None, Disabled on /dev/cu.usbserial-110

8 Motors Positioning



Define the wind angle into 8 wind directions

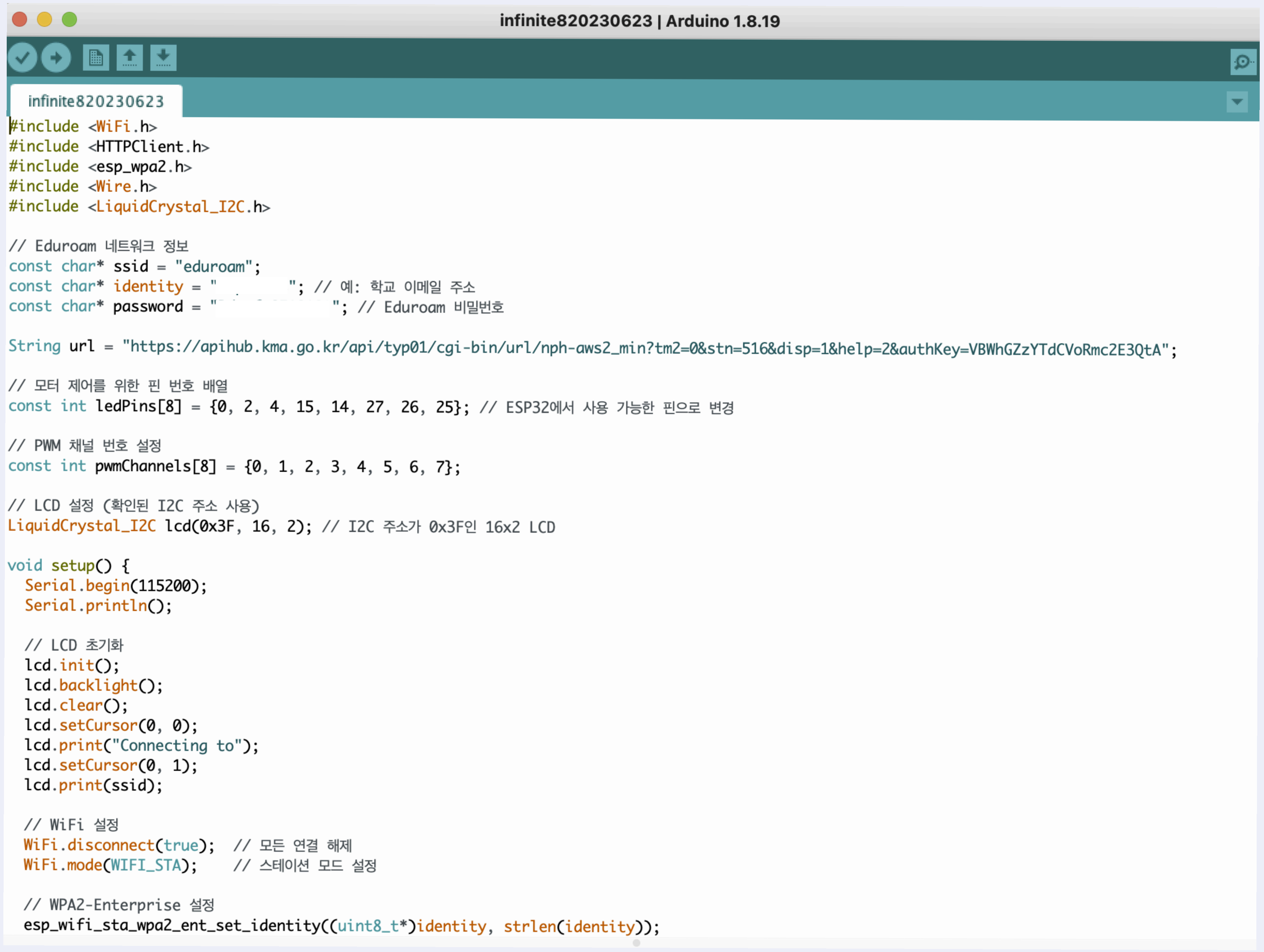


- 337.5 - 22.5 : North
- 22.5 - 62.5 : North-East
- 62.5-112.5 : East
- 112.5 - 157.5 : South-East
- 157.5 - 202.5 : South
- 202.5- 247.5 : South -West
- 247.5 - 292.5: West
- 292.5 - 337.5 : Northh -West

Scientific/Technical

Mapping

Arduino patch details



infinite820230623 | Arduino 1.8.19

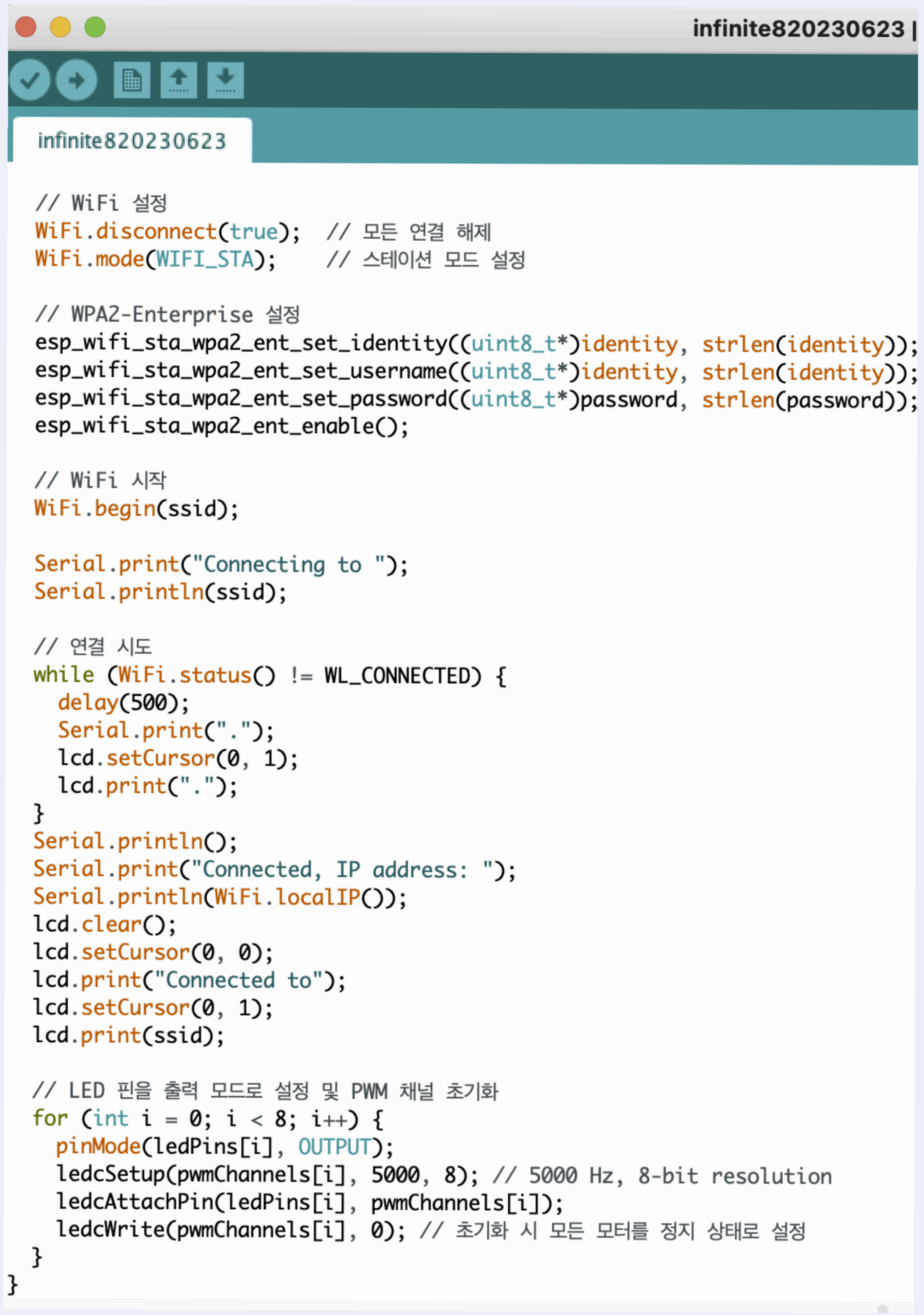
infinite820230623

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <esp_wpa2.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Eduroam 네트워크 정보
const char* ssid = "eduroam";
const char* identity = "[redacted]"; // 예: 학교 이메일 주소
const char* password = "[redacted]"; // Eduroam 비밀번호

String url = "https://apihub.kma.go.kr/api/typ01/cgi-bin/url/nph-aws2_min?tm2=0&stn=516&disp=1&help=2&authKey=VBWhGZzYTdCVoRmc2E3QtA";

// 모터 제어를 위한 핀 번호 배열
const int ledPins[8] = {0, 2, 4, 15, 14, 27, 26, 25}; // ESP32에서 사용 가능한 핀으로 변경

// PWM 채널 번호 설정
const int pwmChannels[8] = {0, 1, 2, 3, 4, 5, 6, 7};

// LCD 설정 (확인된 I2C 주소 사용)
LiquidCrystal_I2C lcd(0x3F, 16, 2); // I2C 주소가 0x3F인 16x2 LCD

void setup() {
  Serial.begin(115200);
  Serial.println();

  // LCD 초기화
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Connecting to");
  lcd.setCursor(0, 1);
  lcd.print(ssid);

  // WiFi 설정
  WiFi.disconnect(true);  // 모든 연결 해제
  WiFi.mode(WIFI_STA);    // 스테이션 모드 설정

  // WPA2-Enterprise 설정
  esp_wifi_sta_wpa2_ent_set_identity((uint8_t*)identity, strlen(identity));
```

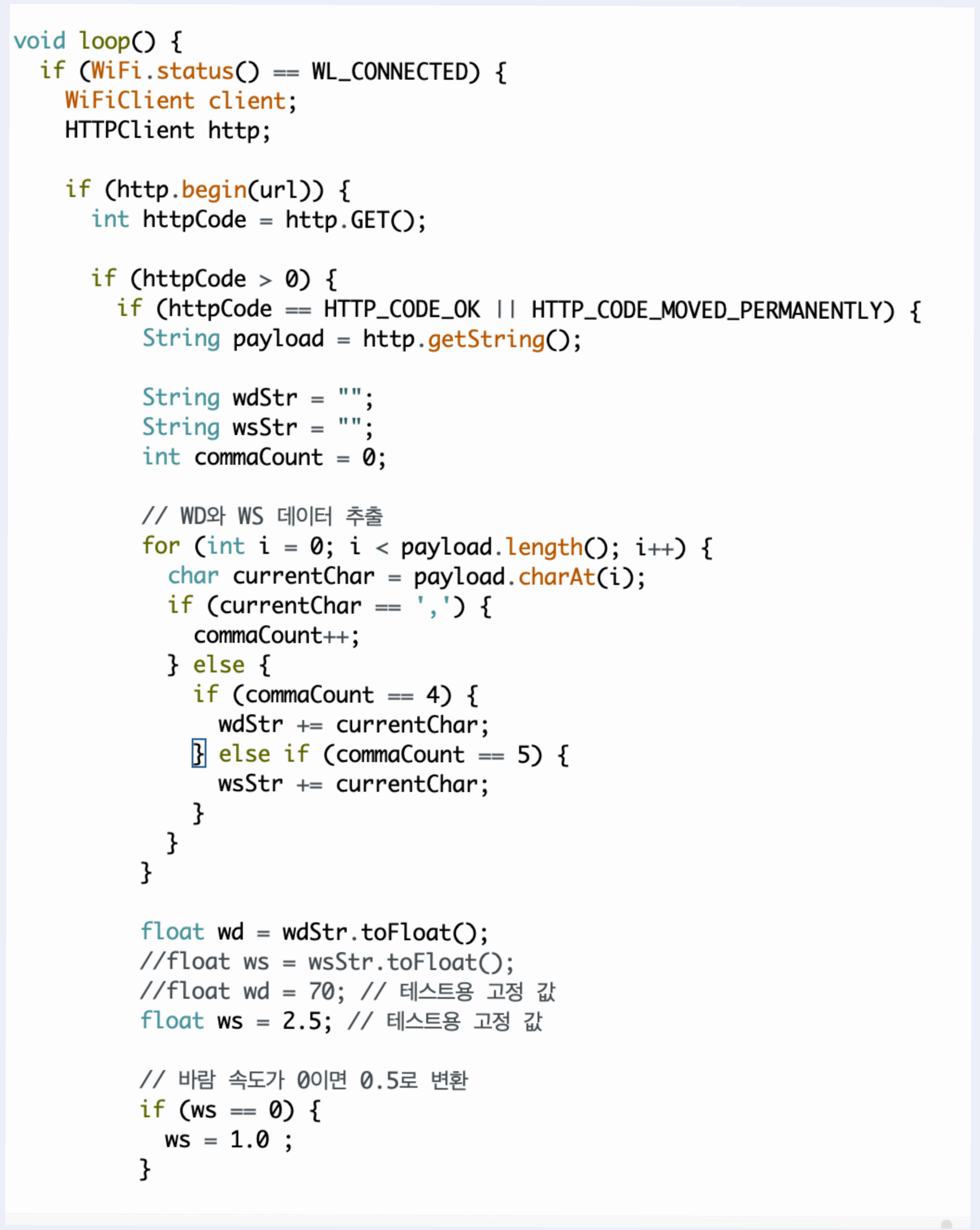
Scientific/Technical

Mapping

Arduino patch details

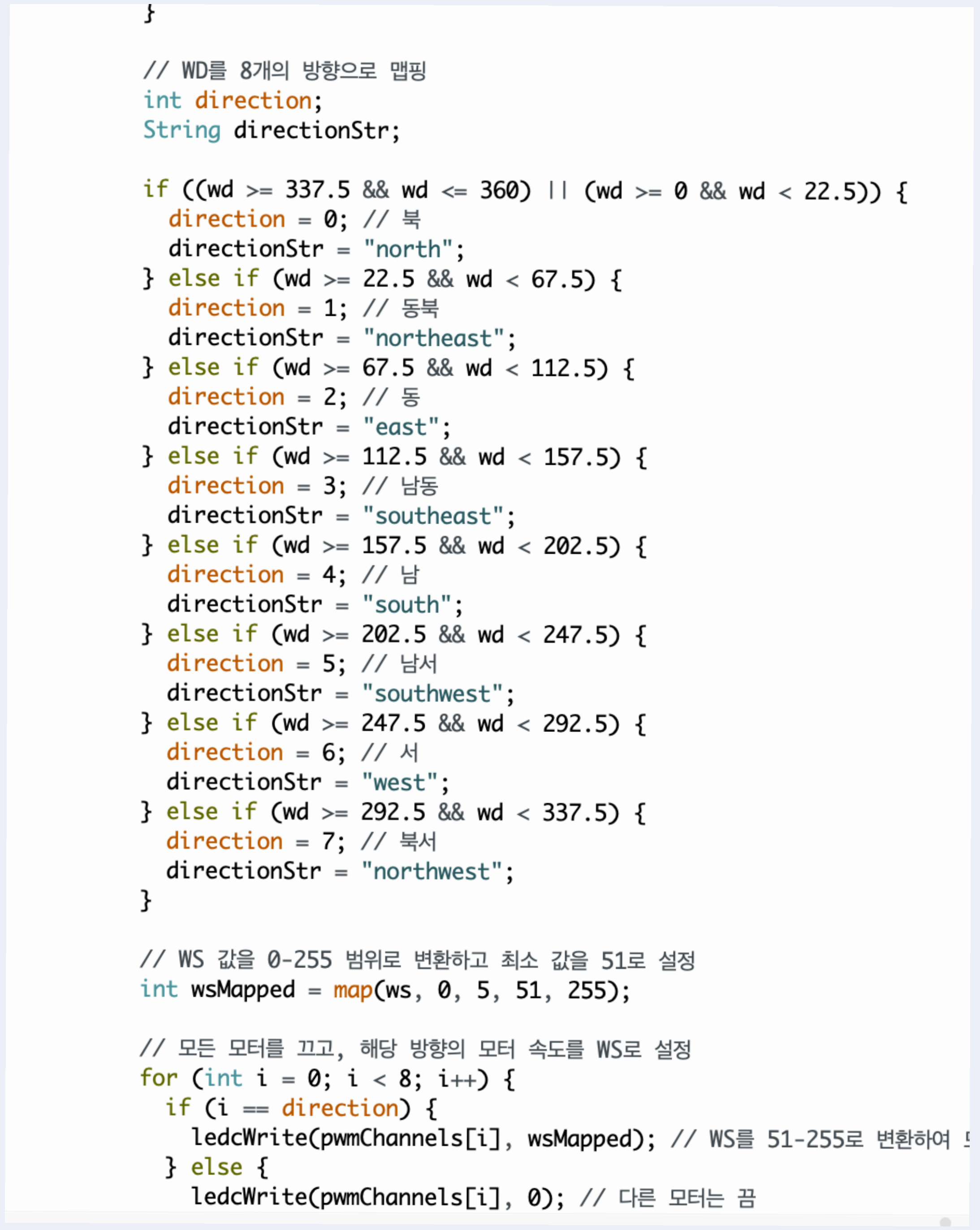


```
  // WiFi 설정
  WiFi.disconnect(true);  // 모든 연결 해제
  WiFi.mode(WIFI_STA);    // 스테이션 모드 설정

  // WPA2-Enterprise 설정
  esp_wifi_sta_wpa2_ent_set_identity((uint8_t*)identity, strlen(identity));
  esp_wifi_sta_wpa2_ent_set_username((uint8_t*)identity, strlen(identity));
  esp_wifi_sta_wpa2_ent_set_password((uint8_t*)password, strlen(password));
  esp_wifi_sta_wpa2_ent_enable();

  // WiFi 시작
  WiFi.begin(ssid);

  Serial.print("Connecting to ");
  Serial.println(ssid);

  // 연결 시도
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    lcd.setCursor(0, 1);
    lcd.print(".");
  }
  Serial.println();
  Serial.print("Connected, IP address: ");
  Serial.println(WiFi.localIP());
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Connected to");
  lcd.setCursor(0, 1);
  lcd.print(ssid);

  // LED 핀을 출력 모드로 설정 및 PWM 채널 초기화
  for (int i = 0; i < 8; i++) {
    pinMode(ledPins[i], OUTPUT);
    ledcSetup(pwmChannels[i], 5000, 8); // 5000 Hz, 8-bit resolution
    ledcAttachPin(ledPins[i], pwmChannels[i]);
    ledcWrite(pwmChannels[i], 0); // 초기화 시 모든 모터를 정지 상태로 설정
  }
}
```

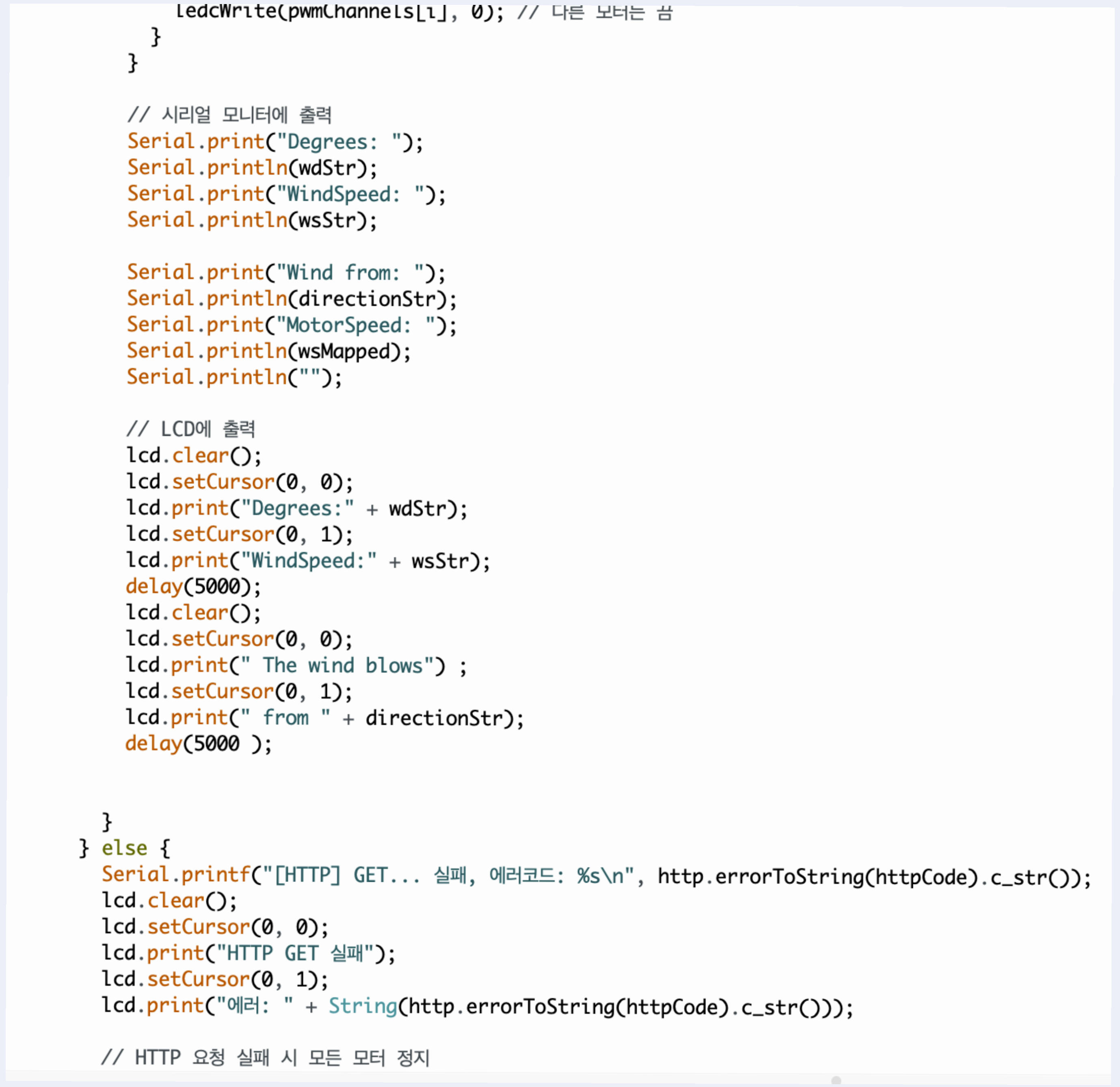


```
void loop() {
  if (WiFi.status() == WL_CONNECTED) {
    WiFiClient client;
    HTTPClient http;

    if (http.begin(url)) {
      int httpCode = http.GET();

      if (httpCode > 0) {
        if (httpCode == HTTP_CODE_OK || HTTP_CODE_MOVED_PERMANENTLY) {
          String payload = http.getString();

          String wdStr = "";
          String wsStr = "";
          int commaCount = 0;

          // WD와 WS 데이터 추출
          for (int i = 0; i < payload.length(); i++) {
            char currentChar = payload.charAt(i);
            if (currentChar == ',') {
              commaCount++;
            } else {
              if (commaCount == 4) {
                wdStr += currentChar;
              } else if (commaCount == 5) {
                wsStr += currentChar;
              }
            }
          }

          float wd = wdStr.toFloat();
          //float ws = wsStr.toFloat();
          //float wd = 70; // 테스트용 고정 값
          float ws = 2.5; // 테스트용 고정 값

          // 바람 속도가 0이면 0.5로 변환
          if (ws == 0) {
            ws = 1.0 ;
          }
```

Arduino patch details



```
}

// WD를 8개의 방향으로 맵핑
int direction;
String directionStr;

if ((wd >= 337.5 && wd <= 360) || (wd >= 0 && wd < 22.5)) {
  direction = 0; // 북
  directionStr = "north";
} else if (wd >= 22.5 && wd < 67.5) {
  direction = 1; // 동북
  directionStr = "northeast";
} else if (wd >= 67.5 && wd < 112.5) {
  direction = 2; // 동
  directionStr = "east";
} else if (wd >= 112.5 && wd < 157.5) {
  direction = 3; // 남동
  directionStr = "southeast";
} else if (wd >= 157.5 && wd < 202.5) {
  direction = 4; // 남
  directionStr = "south";
} else if (wd >= 202.5 && wd < 247.5) {
  direction = 5; // 남서
  directionStr = "southwest";
} else if (wd >= 247.5 && wd < 292.5) {
  direction = 6; // 서
  directionStr = "west";
} else if (wd >= 292.5 && wd < 337.5) {
  direction = 7; // 북서
  directionStr = "northwest";
}

// WS 값을 0-255 범위로 변환하고 최소 값을 51로 설정
int wsMapped = map(ws, 0, 5, 51, 255);

// 모든 모터를 끄고, 해당 방향의 모터 속도를 WS로 설정
for (int i = 0; i < 8; i++) {
  if (i == direction) {
    ledcWrite(pwmChannels[i], wsMapped); // WS를 51-255로 변환하여
  } else {
    ledcWrite(pwmChannels[i], 0); // 다른 모터는 끔
```

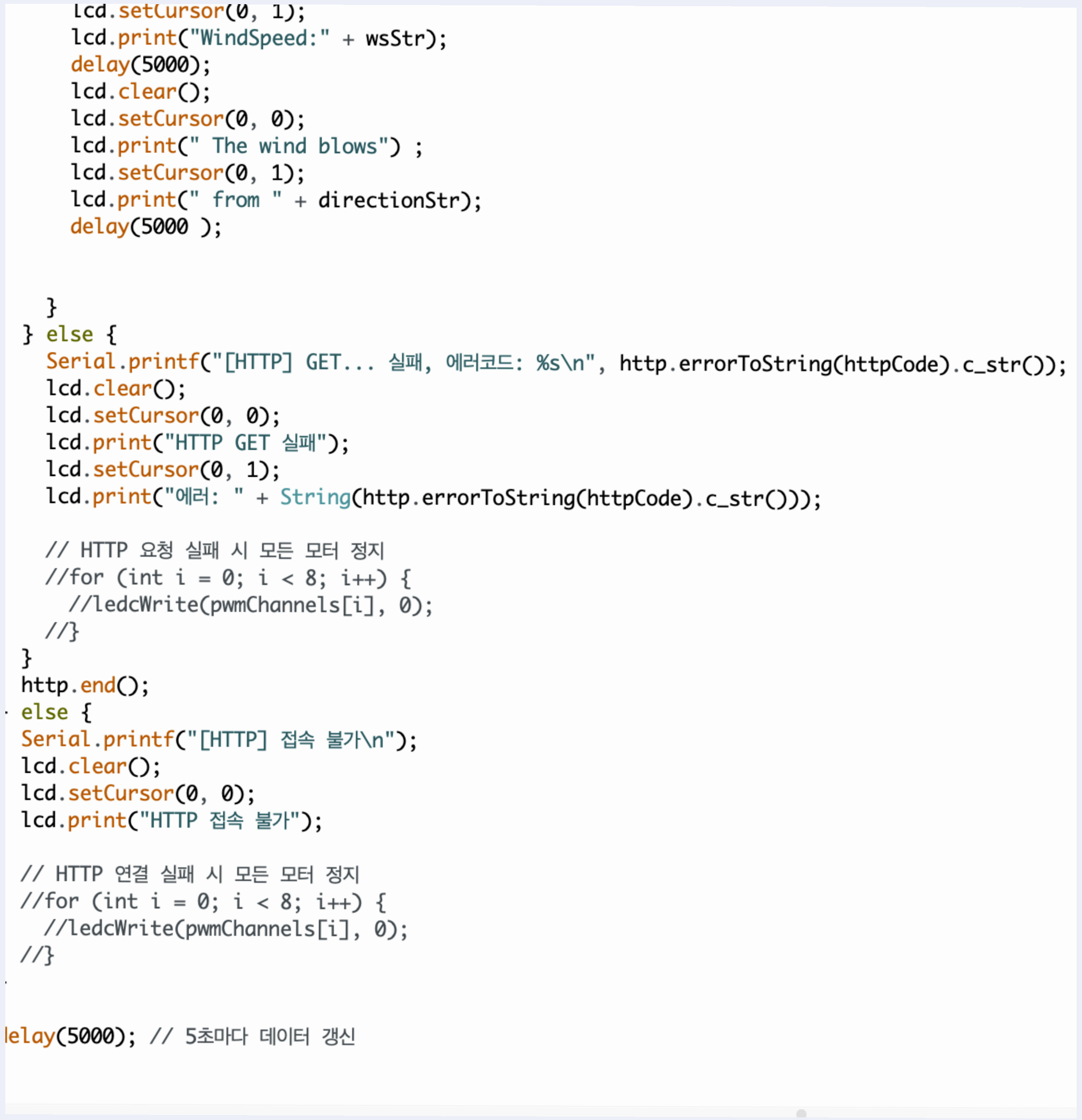


```
      ledcWrite(pwmChannels[i], 0); // 다른 모터는 끔
    }
  }

  // 시리얼 모니터에 출력
  Serial.print("Degrees: ");
  Serial.println(wdStr);
  Serial.print("WindSpeed: ");
  Serial.println(wsStr);

  Serial.print("Wind from: ");
  Serial.println(directionStr);
  Serial.print("MotorSpeed: ");
  Serial.println(wsMapped);
  Serial.println("");

  // LCD에 출력
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Degrees:" + wdStr);
  lcd.setCursor(0, 1);
  lcd.print("WindSpeed:" + wsStr);
  delay(5000);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(" The wind blows") ;
  lcd.setCursor(0, 1);
  lcd.print(" from " + directionStr);
  delay(5000 );


}
} else {
  Serial.printf("[HTTP] GET... 실패, 에러코드: %s\n", http.errorToString(httpCode).c_str());
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("HTTP GET 실패");
  lcd.setCursor(0, 1);
  lcd.print("에러: " + String(http.errorToString(httpCode).c_str()));

  // HTTP 요청 실패 시 모든 모터 정지
```

Scientific/Technical

Mapping

Arduino patch details



```
      lcd.setCursor(0, 1);
      lcd.print("WindSpeed:" + wsStr);
      delay(5000);
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print(" The wind blows") ;
      lcd.setCursor(0, 1);
      lcd.print(" from " + directionStr);
      delay(5000 );


    }
  } else {
    Serial.printf("[HTTP] GET... 실패, 에러코드: %s\n", http.errorToString(httpCode).c_str());
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("HTTP GET 실패");
    lcd.setCursor(0, 1);
    lcd.print("에러: " + String(http.errorToString(httpCode).c_str()));

    // HTTP 요청 실패 시 모든 모터 정지
    //for (int i = 0; i < 8; i++) {
      //ledcWrite(pwmChannels[i], 0);
    //}
  }
  http.end();
  else {
  Serial.printf("[HTTP] 접속 불가\n");
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("HTTP 접속 불가");

  // HTTP 연결 실패 시 모든 모터 정지
  //for (int i = 0; i < 8; i++) {
    //ledcWrite(pwmChannels[i], 0);
  //}


lelay(5000); // 5초마다 데이터 갱신
```

Epilogue

VIDEO 1

VIDEO 2