DEPARTMENT OF PHYSICS, UNIVERSITY OF COLOMBO

PH4031 - ENGINEERING PHYSICS LABORATORY II

System to detect variation of migration patterns in animals with environmental conditions

Name : P.A. Seneviratne

Index : s14376

ABSTRACT

The main objective of this project was to create a system to observe how variations in environmental conditions affect the migration patterns of animals. This is important in understanding the impact humans have in the surrounding environments and how this impact affects the habitability of animals in such environments. In order to understand these migration patterns with regard to environmental changes the information collected through the system needs to be collaborated with already existing migration data collected from other methods to draw a solid conclusion. In future iterations of this product, a mic could be introduced into the system which has been trained on recognized animal noises in order to understand animals in the surrounding so that this could act as an isolated system in collecting the density of animals in an area along with the environmental data of the area. The created system takes data from multiple sensors and processes them simultaneously before saving it in a micro SD card so that it could be remotely obtained by a collecting device by connecting to the Wi-Fi network created by the ESP-32 module. The system is successful in achieving the above mentioned objective.

Contents

1 Introduction **1**

2 Theory **2**

2.1 Serial Communication 2

2.2 ESP-32 module 2

2.3 Soil Moisture Sensor 3

2.4 Humidity Sensor 3

2.5 Pressure Sensor 4

3 Methodology **5**

4 Results and Analysis **8**

5 Discussion **9**

6 Conclusions **10**

Refrences **11**

Appendix A **27**

List of Figures

3.0.1 Created circuit schematic for the Environmental Information System 5
3.0.2 Created circuit for the Environmental Information System 6
3.0.3 Solar panels that were attached to power the system 7

4.0.1 Internal Web Server created by the ESP-32 module 8
4.0.2 Data collected by the ESP-32 module 8

Chapter 1

Introduction

In the wild, animals migrate from one location to another frequently. This migration is dependent on various factors such as environmental conditions, reproduction needs and to improve their chances of survival. Scientists and Zoologists have long since attempted to understand the underlying conditions that would affect the migration patterns of animals and the capability to change the patterns of animals to better understand them.

The most common method of monitoring the migration patterns of animals is to use trackers which are fitted onto animals and track their movements over a fixed period of time via satellites, VHF radio tracking and global positioning system tracking. While these methods are suited for animals that travel long distances in order to migrate, animals that move from one location to another depending on the environmental conditions provide less merit when using such expensive methods. The tracking of environmental conditions also allows scientists to understand how the presence of humans affects the environmental conditions of animals causing such migrations to occur or accelerate.

An inexpensive system which is capable of studying surrounding environmental conditions while monitoring the migration patterns of animals in a certain region simultaneously would allow researchers to understand how the variation of animal migration is affected by the environmental condition and factors which affect the animal habitat.

Chapter 2

Theory

2.1 Serial Communication

Serial communication is a method of transmitting data bit by bit over a single communication line. It uses two signals for transmitting and receiving data. Synchronization is crucial to ensure accurate communication. Various serial protocols exist, such as RS-232, SPI, I2C, and UART, each with its own standards. The baud rate determines the speed of data transmission. Data is organized into frames with start, data, parity, and stop bits. Error detection and correction techniques are used to maintain data integrity. Serial communication can operate in half-duplex or full-duplex mode. Hardware interfaces like UARTs and transceivers handle the conversion and electrical signalling. Understanding these principles is important for designing reliable serial communication systems.

2.2 ESP-32 module

ESP32 is a versatile and powerful microcontroller module developed by Espressif Systems. It combines Wi-Fi and Bluetooth connectivity with a dual-core processor, making it suitable for a wide range of applications. The following is a brief summary of the key features and capabilities of the ESP32 module:

- Dual-Core Processor: The ESP32 features two Xtensa LX6 microprocessor cores, which can be independently programmed and controlled. This dual-core architecture allows for efficient multitasking and high-performance computing.
- Wi-Fi and Bluetooth Connectivity: The ESP32 module supports both Wi-Fi 802.11 b/g/n and Bluetooth v4.2, providing wireless connectivity options for Internet of Things (IoT) applications. It can function as a Wi-Fi client or access point, enabling communication with other devices or connecting to the internet.
- Low-Power Capabilities: The ESP32 is designed to optimize power consumption, making it suitable for battery-powered and energy-efficient applications. It offers different sleep modes and power management features to minimize power consumption when idle.
- Rich Set of Input/Output Interfaces: The ESP32 provides a wide range of interfaces and peripherals, including digital and analogue GPIO pins, UART, SPI, I2C, I2S, ADC, DAC, PWM, and more. These interfaces enable seamless integration with various sensors, actuators, and external devices.

- Integrated Development Environment (IDE) Support: The ESP32 can be programmed using the Arduino IDE or Espressif's own ESP-IDF (ESP32 IoT Development Framework). Both options offer extensive libraries and examples to facilitate rapid development.
- Secure Communication: The ESP32 includes features to ensure secure communication over Wi-Fi and Bluetooth connections. It supports encryption and authentication protocols to protect data transmission and device connectivity.
- Real-Time Operating System (RTOS) Support: The ESP32 supports FreeRTOS, a popular open-source real-time operating system, providing a robust framework for multitasking and managing system resources.
- Memory and Storage: The ESP32 module typically has ample program and data memory, allowing for complex applications. It also offers external storage options such as SPI flash memory or SD cards for additional data storage.
- Community Support: The ESP32 has a large and active community of developers and enthusiasts, which means extensive online resources, forums, and tutorials are available to assist with development and problem-solving.

In summary, the ESP32 is a versatile micro-controller module that combines Wi-Fi and Bluetooth connectivity with a dual-core processor. It offers a rich set of interfaces, low-power capabilities, and secure communication features. With extensive community support and multiple programming options, the ESP32 is a popular choice for a wide range of IoT and embedded system projects.

2.3 Soil Moisture Sensor

The capacitive soil moisture sensor measures soil moisture by detecting changes in capacitance between two electrodes in contact with the soil. The moisture content affects the dielectric constant of the soil, which in turn alters the capacitance value. Calibration is necessary to establish the relationship between capacitance and moisture content. Environmental factors can influence accuracy, and installation considerations are important for reliable readings. These sensors are used in various fields, including agriculture and environmental monitoring.

2.4 Humidity Sensor

Humidity sensors, also called hygrometers, are devices that measure and monitor the moisture content or relative humidity (RH) in the air. They use different sensing principles like capacitive, resistive, or thermal to detect changes in humidity. Capacitive sensors are widely used, measuring electrical

capacitance variations due to moisture absorption. These sensors have applications in weather monitoring, HVAC systems, indoor air quality control, and industrial processes. They are essential for maintaining optimal conditions in various environments.

2.5 Pressure Sensor

Pressure sensors are devices used to measure and monitor the pressure of gases or liquids. They utilize different sensing principles like piezoresistive, capacitive, or piezoelectric to detect pressure changes. The sensors convert physical pressure into an electrical signal for measurement. Calibration is necessary to establish accuracy by comparing sensor readings with known pressure references. Pressure sensors have specific measurement ranges and resolutions, and their performance can be influenced by environmental factors. They are used in various industries for pressure monitoring, control systems, and quality assurance processes.

Chapter 3

Methodology

The first step in creating this system was to understand the environmental conditions that need to be observed in order to get a good understanding of the changes that occur in the given region over a period of time. The sensors used in order to achieve this were a Humidity sensor, pressure sensor, temperature sensor and soil water moisture sensor. The continuous observation of these sensors would allow the user to achieve a solid idea regarding the changes occurring in the environment around the system. Once the sensors were decided upon, the next step was to create a schematic with all the circuitry correctly attached to each order for its intended operation.

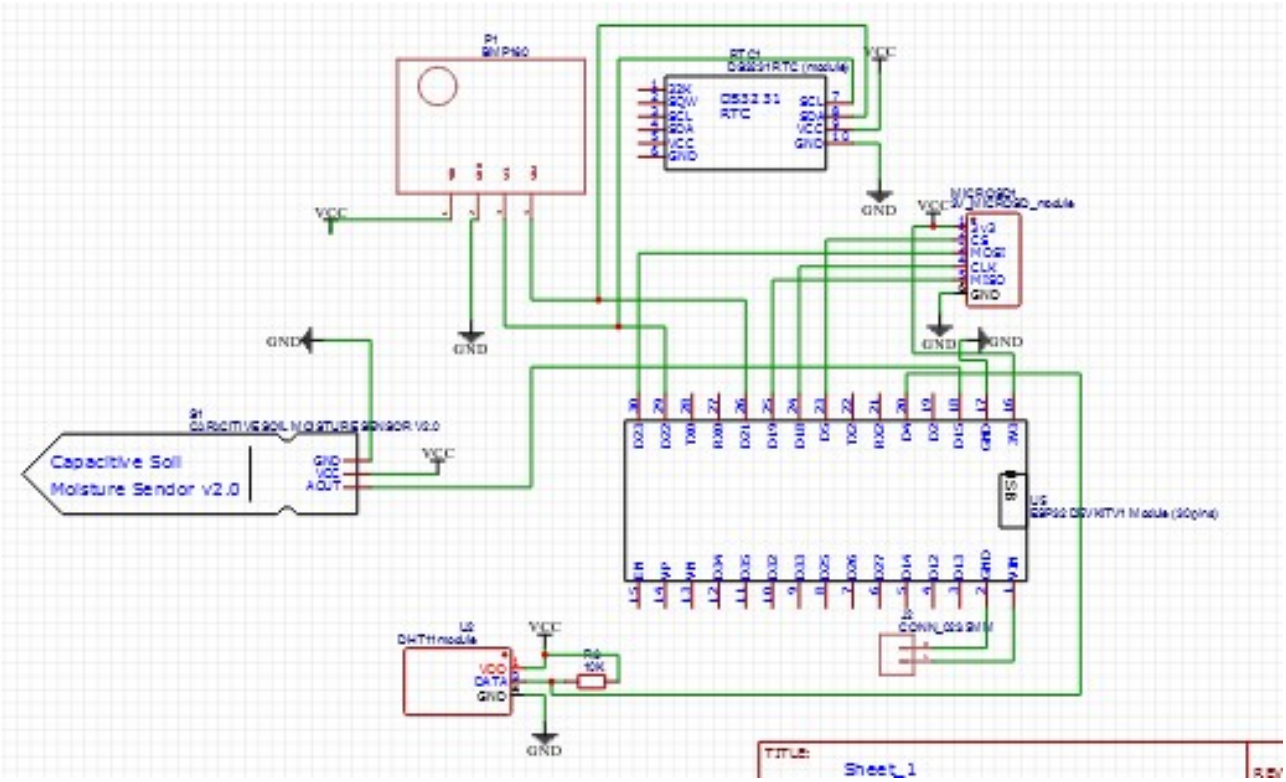

Figure 3.0.1: Created circuit schematic for the Environmental Information System

Once the schematic was completed, the next step was to create the circuit on a dot board. Initially, the ESP-32 module was soldered onto the board. The sensors and the micro SD card module were subsequently added onto the dot board and connected with the ESP-32 module. In addition to the above-mentioned sensors, a real-time clock module was added as well the time at which the data was recorded is important as well to understand migration patterns. The circuitry was put into a waterproof electric box in order to protect the circuitry from external environmental conditions. [1]

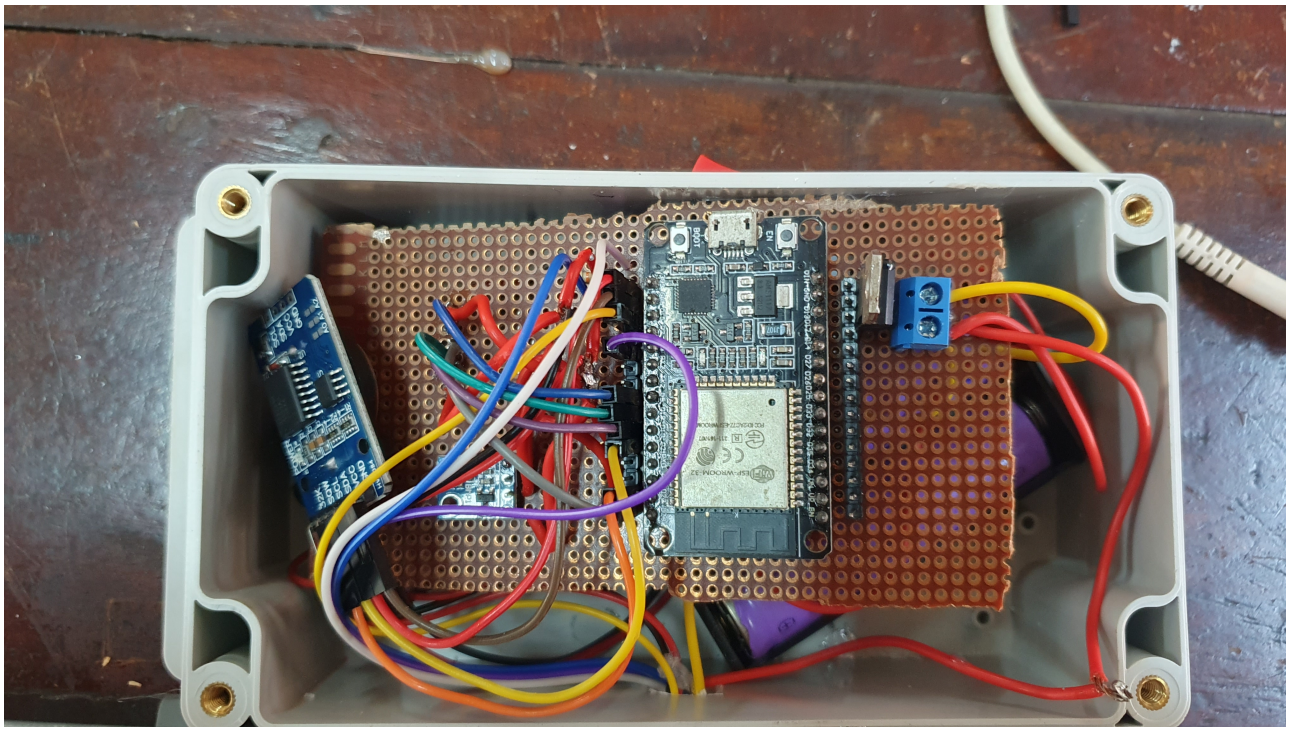

Figure 3.0.2: Created circuit for the Environmental Information System

Subsequently, the code necessary for the correct operation of the system was developed. This code was created in the Arduino internal development environment. This code allowed data to be collected from all the sensors and be processed before being saved in a micro SD card continuously. Once the data has been recorded for a specific period of time it can be remotely collected using a Wi-Fi connection via a separate remote data collection device directly. Once data has been downloaded by the remote data collection device, the system deletes the collected data and restarts the process of data collection. [2]

In order to increase the longevity of the device to operate in an isolated manner, solar panels were added which are capable of charging the batteries while the device operates. This would act to reduce the times the batteries need to be physically changed by the users.

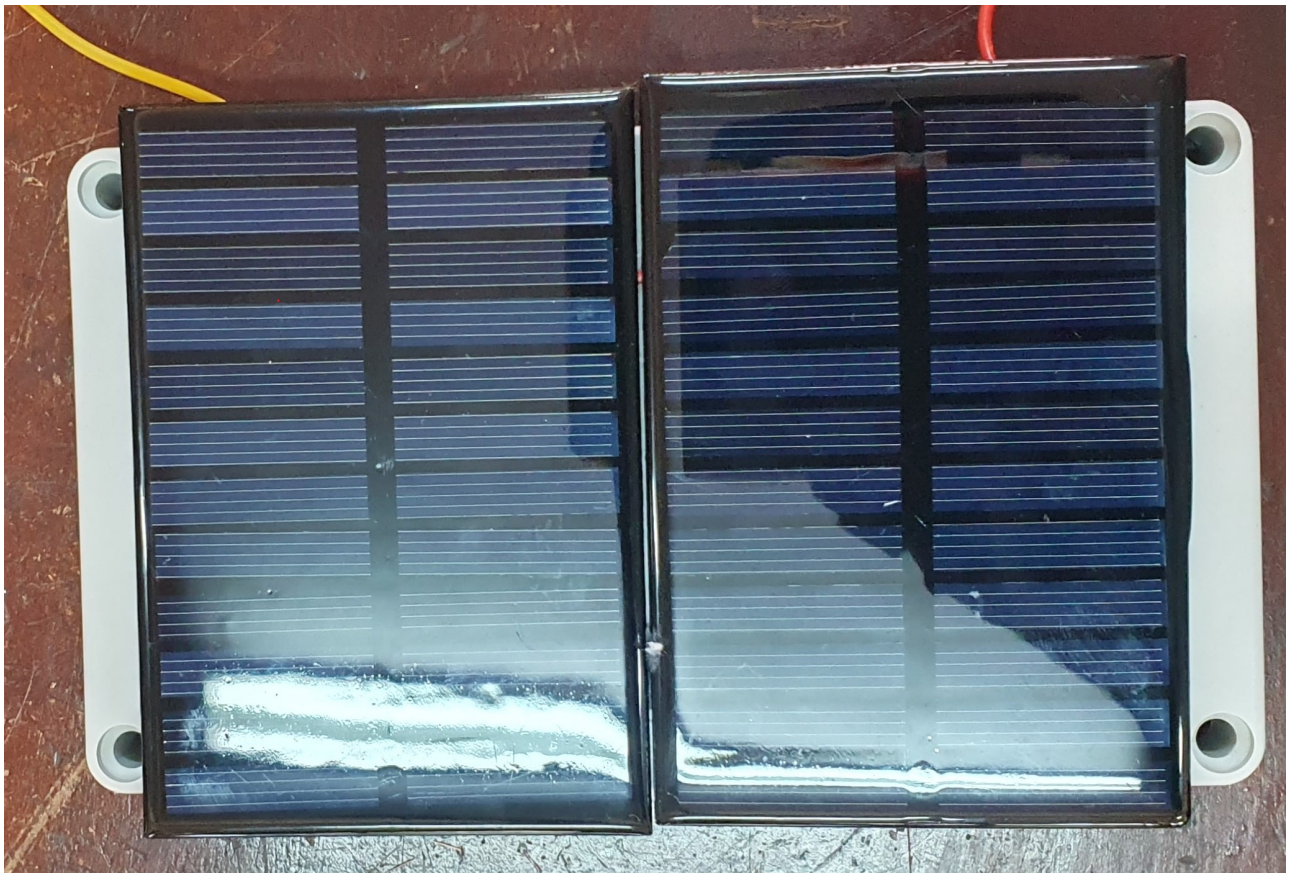

Figure 3.0.3: Solar panels that were attached to power the system

The code was then created for the ESP-32 module so that it would be capable to obtain data from the sensors and save it in the micro SD module. Also, an internal web server was created in the ESP-32 module so that the data stored in the micro SD card could be wirelessly transmitted to an external collecting device. Once the data is collected, the system restarts and starts the process of data collection once again. This system reset in order to erase the data present in the micro SD card and thereby save space. [3]

Chapter 4

Results and Analysis

The created web-server created has the capability for any device that connects to it to download the file remotely. The web server looks as follows.

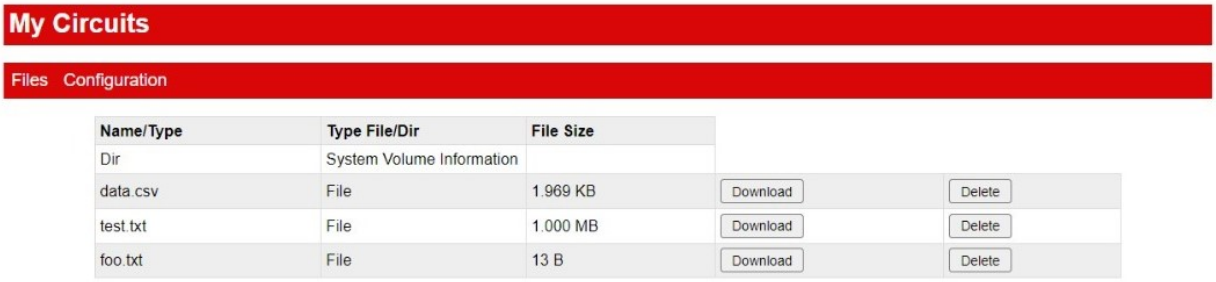

Figure 4.0.1: Internal Web Server created by the ESP-32 module

The data collected can be seen as the ”data.csv” file in the above image. The collected data can be seen in the image shown below.

Humidity (%)	Temperature	Heat index	Moisture value	Pressure (Pa)	Time
87	31.8	46.68	0	100895	16:9:34-Sunday-28-5-2023
86	31.8	46.23	0	100887	16:9:36-Sunday-28-5-2023
86	31.8	46.23	0	100886	16:9:38-Sunday-28-5-2023
86	31.8	46.23	0	100889	16:9:40-Sunday-28-5-2023
87	31.8	46.68	0	100891	16:9:42-Sunday-28-5-2023
87	31.8	46.68	0	100893	16:9:44-Sunday-28-5-2023
87	31.8	46.68	0	100892	16:9:46-Sunday-28-5-2023
87	31.8	46.68	0	100885	16:9:48-Sunday-28-5-2023
87	31.8	46.68	0	100889	16:9:50-Sunday-28-5-2023
87	31.8	46.68	0	100888	16:9:52-Sunday-28-5-2023
87	31.8	46.68	0	100885	16:9:54-Sunday-28-5-2023
87	31.8	46.68	0	100907	16:9:57-Sunday-28-5-2023
87	31.8	46.68	0	100895	16:9:59-Sunday-28-5-2023
87	31.8	46.68	0	100896	16:10:1-Sunday-28-5-2023
87	31.8	46.68	0	100902	16:10:3-Sunday-28-5-2023
87	31.8	46.68	0	100899	16:10:5-Sunday-28-5-2023
87	31.8	46.68	0	100897	16:10:7-Sunday-28-5-2023
87	31.8	46.68	0	100889	16:10:9-Sunday-28-5-2023
87	31.8	46.68	0	100889	16:10:11-Sunday-28-5-2023

Figure 4.0.2: Data collected by the ESP-32 module

Chapter 5

Discussion

The main objective of this project was to create a system to observe how variations in environmental conditions affect the migration patterns of animals. This is important in understanding the impact humans have in the surrounding environments and how this impact affects the habitability of animals in such environments. In order to understand these migration patterns with regard to environmental changes the information collected through the system needs to be collaborated with already existing migration data collected from other methods to draw a solid conclusion. In future iterations of this product, a mic could be introduced into the system which has been trained on recognized animal noises in order to understand animals in the surrounding so that this could act as an isolated system in collecting the density of animals in an area along with the environmental data of the area.

In creating this device, the controlling component was the ESP-32 module due to its capability to create an internal web server and has two cores which allow the module to process larger amounts of data without overloading. However, in the testing phase, the ESP-32 module did overload due to the types of variables in which the data was presented. This was resolved after changing all the variable types to ”string”. Based on the ESP-32 module, a schematic was initially created using the EasyEDA website where all the components were connected to each other. Once the components were correctly connected, it was then recreated in a dot board. The sensors which were added were a pressure sensor, a humidity sensor and a capacitive soil moisture detector. To save the data, a micro SD card module was added as well. In order to prolong the battery life of the system, solar panels were added to the system as well.

In future iterations, an antenna could be added to the system so that the receiving autonomous device can receive the data from much further away. Another valuable addition to the system is the addition of a weather shield which would protect the sensors from external environmental factors.

Chapter 6

Conclusions

The main objective of this project was to create a system to observe how variations in environmental conditions affect the migration patterns of animals. This is important in understanding the impact humans have in the surrounding environments and how this impact affects the habitability of animals in such environments. This system was successfully created by implementing sensors and monitoring the environmental factors simultaneously. The main operating module was the ESP32 module and the peripheral sensors used were the Humidity sensor, Pressure sensor and Capacitive soil moisture sensor. The system is capable of simultaneously recording data from various sensors and then transmitting them to a remote receiving device. The system is powered by solar power for prolonged battery life and overall the device is successful in achieving the above-mentioned objectives.

References

[1] IP Ratings Explained.

[2] An Introduction to 5GHz Technology, 2021.

[3] Espressif. ESP32 Series Datasheet. *Espressif Systems*, pages 1–69, 2022.

Appendix-A

```
#include <WiFi.h>
#include <ESP32WebServer.h>
#include <ESPmDNS.h>

#include "CSS.h"
#include <SD.h>
#include <SPI.h>

#include <Wire.h>
#include <Adafruit_GFX.h>
#include "RTClib.h"
#include "DHT.h"
#include <Adafruit_BMP085.h>
#include "FS.h"
#include "SD.h"
#include "SPI.h"

ESP32WebServer server(80);

const int ledPin = 13;

#define servername "research"
#define SD_pin 16

bool   SD_present = false;

Adafruit_BMP085 bmp;
RTC_DS3231 rtc;

String data = "";

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday"};

#define DHTPIN 4
#define AOUT_PIN 15


#define DHTTYPE DHT11   // DHT 11

DHT dht(DHTPIN, DHTTYPE);
```

```
void listDir(fs::FS &fs, const char * dirname, uint8_t levels){
    Serial.printf("Listing directory: %s\n", dirname);

    File root = fs.open(dirname);
    if(!root){
        Serial.println("Failed to open directory");
        return;
    }
    if(!root.isDirectory()){
        Serial.println("Not a directory");
        return;
    }

    File file = root.openNextFile();
    while(file){
        if(file.isDirectory()){
            Serial.print("  DIR : ");
            Serial.println(file.name());
            if(levels){
                listDir(fs, file.path(), levels -1);
            }
        } else {
            Serial.print("  FILE: ");
            Serial.print(file.name());
            Serial.print("  SIZE: ");
            Serial.println(file.size());
        }
        file = root.openNextFile();
    }
}

void createDir(fs::FS &fs, const char * path){
    Serial.printf("Creating Dir: %s\n", path);
    if(fs.mkdir(path)){
        Serial.println("Dir created");
    } else {
        Serial.println("mkdir failed");
    }
}
```

```
void removeDir(fs::FS &fs, const char * path){
    Serial.printf("Removing Dir: %s\n", path);
    if(fs.rmdir(path)){
        Serial.println("Dir removed");
    } else {
        Serial.println("rmdir failed");
    }
}

void readFile(fs::FS &fs, const char * path){
    Serial.printf("Reading file: %s\n", path);

    File file = fs.open(path);
    if(!file){
        Serial.println("Failed to open file for reading");
        return;
    }

    Serial.print("Read from file: ");
    while(file.available()){
        Serial.write(file.read());
    }
    file.close();
}

void writeFile(fs::FS &fs, const char * path, const char * message){
    Serial.printf("Writing file: %s\n", path);

    File file = fs.open(path, FILE_WRITE);
    if(!file){
        Serial.println("Failed to open file for writing");
        return;
    }
    if(file.print(message)){
        Serial.println("File written");
    } else {
        Serial.println("Write failed");
    }
    file.close();
}
```

```
void appendFile(fs::FS &fs, const char * path, const char * message){
    Serial.printf("Appending to file: %s\n", path);

    File file = fs.open(path, FILE_APPEND);
    if(!file){
        Serial.println("Failed to open file for appending");
        return;
    }
    if(file.print(message)){
        Serial.println("Message appended");
    } else {
        Serial.println("Append failed");
    }
    file.close();
}

void renameFile(fs::FS &fs, const char * path1, const char * path2){
    Serial.printf("Renaming file %s to %s\n", path1, path2);
    if (fs.rename(path1, path2)) {
        Serial.println("File renamed");
    } else {
        Serial.println("Rename failed");
    }
}

void deleteFile(fs::FS &fs, const char * path){
    Serial.printf("Deleting file: %s\n", path);
    if(fs.remove(path)){
        Serial.println("File deleted");
    } else {
        Serial.println("Delete failed");
    }
}
```

```
void testFileIO(fs::FS &fs, const char * path){
    File file = fs.open(path);
    static uint8_t buf[512];
    size_t len = 0;
    uint32_t start = millis();
    uint32_t end = start;
    if(file){
        len = file.size();
        size_t flen = len;
        start = millis();
        while(len){
            size_t toRead = len;
            if(toRead > 512){
                toRead = 512;
            }
            file.read(buf, toRead);
            len -= toRead;
        }
        end = millis() - start;
        Serial.printf("%u bytes read for %u ms\n", flen, end);
        file.close();
    } else {
        Serial.println("Failed to open file for reading");
    }


    file = fs.open(path, FILE_WRITE);
    if(!file){
        Serial.println("Failed to open file for writing");
        return;
    }

    size_t i;
    start = millis();
    for(i=0; i<2048; i++){
        file.write(buf, 512);
    }
    end = millis() - start;
    Serial.printf("%u bytes written for %u ms\n", 2048 * 512, end);
    file.close();
}
```

```
void setup() {
  Serial.begin(9600);
  digitalWrite(ledPin, HIGH);
  dht.begin();
  if (!bmp.begin())
  {
    Serial.println("BMP180 Sensor not found ! ! !");
    while (1)
    {

    }
  }
  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    while (1);
  }
  rtc.adjust(DateTime(__DATE__, __TIME__));
  if(!SD.begin()){
        Serial.println("Card Mount Failed");
        return;
    }
    uint8_t cardType = SD.cardType();

    if(cardType == CARD_NONE){
        Serial.println("No SD card attached");
        return;
    }

    Serial.print("SD Card Type: ");
    if(cardType == CARD_MMC){
        Serial.println("MMC");
    } else if(cardType == CARD_SD){
        Serial.println("SDSC");
    } else if(cardType == CARD_SDHC){
        Serial.println("SDHC");
    } else {
        Serial.println("UNKNOWN");
    }
```

```
    Serial.begin(9600);
  WiFi.softAP("MyCircuits", "12345678");

  if (!MDNS.begin(servername))
  {
    Serial.println(F("Error setting up MDNS responder!"));
    ESP.restart();
  }

  Serial.print(F("Initializing SD card..."));

  if (!SD.begin(SD_pin))
  {
    Serial.println(F("Card failed or not present, no SD Card data logging possible...
    SD_present = false;
  }
  else
  {
    Serial.println(F("Card initialised... file access enabled..."));
    SD_present = true;
  }

  /*********  Server Commands  **********/
  server.on("/",         SD_dir);
  server.on("/upload",   File_Upload);
  server.on("/fupload",  HTTP_POST,[](){ server.send(200);}, handleFileUpload);

  server.begin();

  Serial.println("HTTP server started");

    uint64_t cardSize = SD.cardSize() / (1024 * 1024);
    Serial.printf("SD Card Size: %lluMB\n", cardSize);
    deleteFile(SD, "/data.csv");
    appendFile(SD, "/data.csv", "Humidity,Temperature,Heat index,
    Moisture value,Pressure,Time\n");
}
```

```
void loop() {
  int value = analogRead(AOUT_PIN);
  DateTime now = rtc.now();

  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);

  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  float hif = dht.computeHeatIndex(f, h);
  float hic = dht.computeHeatIndex(t, h, false);

  data = String(h)+(",")+String(t)+(",")+String(hic)+(",")+
  String(value)+(",")+String(bmp.readPressure())+(",")+
  String(now.hour(), DEC)+(":")+String(now.minute(),
  DEC)+(":")+String(now.second(), DEC)+("-")+
  String(daysOfTheWeek[now.dayOfTheWeek()])+
  ("-")+String(now.day(), DEC)+("-")+String(now.month(),
  DEC)+("-")+String(now.year(), DEC)+"\n";
  Serial.println(data);
  appendFile(SD, "/data.csv", data.c_str());
  delay(2000);
  server.handleClient();
}
```

```
void SD_dir()
{
  if (SD_present)
  {
    if (server.args() > 0 )
    {
      Serial.println(server.arg(0));

      String Order = server.arg(0);
      Serial.println(Order);

      if (Order.indexOf("download_")>=0)
      {
        Order.remove(0,9);
        SD_file_download(Order);
        Serial.println(Order);
      }

      if ((server.arg(0)).indexOf("delete_")>=0)
      {
        Order.remove(0,7);
        SD_file_delete(Order);
        Serial.println(Order);
      }
    }

    File root = SD.open("/");
    if (root) {
      root.rewindDirectory();
      SendHTML_Header();
      webpage += F("<table align='center'>");
      webpage += F("<tr><th>Name/Type</th><th style='width:20%'>
      Type File/Dir</th><th>File Size</th></tr>");
      printDirectory("/",0);
      webpage += F("</table>");
      SendHTML_Content();
      root.close();
    }
    else
    {
```

```
      SendHTML_Header();
      webpage += F("<h3>No Files Found</h3>");
    }
    append_page_footer();
    SendHTML_Content();
    SendHTML_Stop();   //Stop is needed because no content length was sent
  } else ReportSDNotPresent();
}

//Upload a file to the SD
void File_Upload()
{
  append_page_header();
  webpage += F("<h3>Select File to Upload</h3>");
  webpage += F("<FORM action='/fupload' method='post'
  enctype='multipart/form-data'>");
  webpage += F("<input class='buttons' style='width:25%' type='file'
  name='fupload' id = 'fupload' value=''>");
  webpage += F("<button class='buttons' style='width:10%'
  type='submit'>Upload File</button><br><br>");
  webpage += F("<a href='/'>[Back]</a><br><br>");
  append_page_footer();
  server.send(200, "text/html",webpage);
}

void printDirectory(const char * dirname, uint8_t levels)
{

  File root = SD.open(dirname);

  if(!root){
    return;
  }
  if(!root.isDirectory()){
    return;
  }
  File file = root.openNextFile();

  int i = 0;
  while(file){
    if (webpage.length() > 1000) {
      SendHTML_Content();
    }
```

```
    if(file.isDirectory()){
      webpage += "<tr><td>"+String(file.isDirectory()?"Dir":"File")+"
      </td><td>"+String(file.name())+"</td><td></td></tr>";
      printDirectory(file.name(), levels-1);
    }
    else
    {
      webpage += "<tr><td>"+String(file.name())+"</td>";
      webpage += "<td>"+String(file.isDirectory()?"Dir":"File")+"</td>";
      int bytes = file.size();
      String fsize = "";
      if (bytes < 1024)                     fsize = String(bytes)+" B";
      else if(bytes < (1024 * 1024))  fsize = String(bytes/1024.0,3)+" KB";
      else if(bytes < (1024 * 1024 * 1024))
      fsize = String(bytes/1024.0/1024.0,3)+" MB";
      else    fsize = String(bytes/1024.0/1024.0/1024.0,3)+" GB";
      webpage += "<td>"+fsize+"</td>";
      webpage += "<td>";
      webpage += F("<FORM action='/' method='post'>");
      webpage += F("<button type='submit' name='download'");
      webpage += F("' value='"); webpage +="download_"+String(file.name());
      webpage +=F("'>Download</button>");
      webpage += "</td>";
      webpage += "<td>";
      webpage += F("<FORM action='/' method='post'>");
      webpage += F("<button type='submit' name='delete'");
      webpage += F("' value='"); webpage +="delete_"+String(file.name());
      webpage +=F("'>Delete</button>");
      webpage += "</td>";
      webpage += "</tr>";

    }
    file = root.openNextFile();
    i++;
  }
  file.close();


}
```

```
void SD_file_download(String filename)
{
  if (SD_present)
  {
    File download = SD.open("/"+filename);
    if (download)
    {
      server.sendHeader("Content-Type", "text/text");
      server.sendHeader("Content-Disposition", "attachment;
      filename="+filename);
      server.sendHeader("Connection", "close");
      server.streamFile(download, "application/octet-stream");
      download.close();
      delay(1000);
      digitalWrite(ledPin, LOW);
      delay(1000);
      digitalWrite(ledPin, HIGH);
    } else ReportFileNotPresent("download");
  } else ReportSDNotPresent();
}

File UploadFile;
void handleFileUpload()
{
  HTTPUpload& uploadfile = server.upload();
  if(uploadfile.status == UPLOAD_FILE_START)
  {
    String filename = uploadfile.filename;
    if(!filename.startsWith("/")) filename = "/"+filename;
    Serial.print("Upload File Name: "); Serial.println(filename);
    SD.remove(filename);
    UploadFile = SD.open(filename, FILE_WRITE);
    filename = String();
  }
  else if (uploadfile.status == UPLOAD_FILE_WRITE)
  {
    if(UploadFile) UploadFile.write(uploadfile.buf, uploadfile.currentSize);
  }
```

```
  else if (uploadfile.status == UPLOAD_FILE_END)
  {
    if(UploadFile)
    {
      UploadFile.close();
      Serial.print("Upload Size: "); Serial.println(uploadfile.totalSize);
      webpage = "";
      append_page_header();
      webpage += F("<h3>File was successfully uploaded</h3>");
      webpage += F("<h2>Uploaded File Name: "); webpage +=
      uploadfile.filename+"</h2>";
      webpage += F("<h2>File Size: "); webpage +=
      file_size(uploadfile.totalSize) + "</h2><br><br>";
      webpage += F("<a href='/'>[Back]</a><br><br>");
      append_page_footer();
      server.send(200,"text/html",webpage);
    }
    else
    {
      ReportCouldNotCreateFile("upload");
    }
  }
}

void SD_file_delete(String filename)
{
  if (SD_present) {
    SendHTML_Header();
    File dataFile = SD.open("/"+filename, FILE_READ);
    if (dataFile)
    {
      if (SD.remove("/"+filename)) {
        Serial.println(F("File deleted successfully"));
        webpage += "<h3>File '"+filename+"' has been erased</h3>";
        webpage += F("<a href='/'>[Back]</a><br><br>");
      }
      else
      {
        webpage += F("<h3>File was not deleted - error</h3>");
        webpage += F("<a href='/'>[Back]</a><br><br>");
      }
```

```
    } else ReportFileNotPresent("delete");
    append_page_footer();
    SendHTML_Content();
    SendHTML_Stop();
  } else ReportSDNotPresent();
}

//SendHTML_Header
void SendHTML_Header()
{
  server.sendHeader("Cache-Control", "no-cache, no-store, must-revalidate");
  server.sendHeader("Pragma", "no-cache");
  server.sendHeader("Expires", "-1");
  server.setContentLength(CONTENT_LENGTH_UNKNOWN);
  server.send(200, "text/html", "");
  append_page_header();
  server.sendContent(webpage);
  webpage = "";
}


void SendHTML_Content()
{
  server.sendContent(webpage);
  webpage = "";
}

void SendHTML_Stop()
{
  server.sendContent("");
  server.client().stop(); //Stop is needed because no content length was sent
}

void ReportSDNotPresent()
{
  SendHTML_Header();
  webpage += F("<h3>No SD Card present</h3>");
  webpage += F("<a href='/'>[Back]</a><br><br>");
  append_page_footer();
  SendHTML_Content();
  SendHTML_Stop();
}
```

```
//ReportFileNotPresent
void ReportFileNotPresent(String target)
{
  SendHTML_Header();
  webpage += F("<h3>File does not exist</h3>");
  webpage += F("<a href='/"); webpage += target + "'>[Back]</a><br><br>";
  append_page_footer();
  SendHTML_Content();
  SendHTML_Stop();
}

void ReportCouldNotCreateFile(String target)
{
  SendHTML_Header();
  webpage += F("<h3>Could Not Create Uploaded File (write-protected?)</h3>");
  webpage += F("<a href='/"); webpage += target + "'>[Back]</a><br><br>";
  append_page_footer();
  SendHTML_Content();
  SendHTML_Stop();
}

String file_size(int bytes)
{
  String fsize = "";
  if (bytes < 1024)                 fsize = String(bytes)+" B";
  else if(bytes < (1024*1024))      fsize = String(bytes/1024.0,3)+" KB";
  else if(bytes < (1024*1024*1024))
  fsize = String(bytes/1024.0/1024.0,3)+" MB";
  else              fsize = String(bytes/1024.0/1024.0/1024.0,3)+" GB";
  return fsize;
}
```