REPUBLIK INDONESIA
KEMENTERIAN HUKUM DAN HAK ASASI MANUSIA

SURAT PENCATATAN
CIPTAAN

Dalam rangka pelindungan ciptaan di bidang ilmu pengetahuan, seni dan sastra berdasarkan Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta, dengan ini menerangkan:

Nomor dan tanggal permohonan	:	EC00202272545, 6 Oktober 2022
Pencipta		
Nama	:	**Yois Phahirani, Asniar Aliyu, S.T., M.Eng dkk**
Alamat	:	Karang Bajang RT 02/RW 26 Tlogoadi, Mlati, Sleman, Sleman, DI YOGYAKARTA, 55286
Kewarganegaraan	:	Indonesia
Pemegang Hak Cipta		
Nama	:	**LPPMI**
Alamat	:	Jl. Babarsari, Tambak Bayan, Caturtunggal, Kec. Depok, Kabupaten Sleman, Daerah Istimewa Yogyakarta , Sleman, DI YOGYAKARTA, 55281
Kewarganegaraan	:	Indonesia
Jenis Ciptaan	:	**Program Komputer**
Judul Ciptaan	:	**SISTEM KONTROL DAN MONITORING ATAP JEMURAN PAKAIAN MENGGUNAKAN APLIKASI ANDROID**
Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia	:	11 Agustus 2022, di Yogyakarta
Jangka waktu pelindungan	:	Berlaku selama 50 (lima puluh) tahun sejak Ciptaan tersebut pertama kali dilakukan Pengumuman.
Nomor pencatatan	:	000388286

adalah benar berdasarkan keterangan yang diberikan oleh Pemohon.
Surat Pencatatan Hak Cipta atau produk Hak terkait ini sesuai dengan Pasal 72 Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

a.n Menteri Hukum dan Hak Asasi Manusia
Direktur Jenderal Kekayaan Intelektual
u.b.
Direktur Hak Cipta dan Desain Industri

Anggoro Dasananto
NIP.196412081991031002

Disclaimer:
Dalam hal pemohon memberikan keterangan tidak sesuai dengan surat pernyataan, Menteri berwenang untuk mencabut surat pencatatan permohonan.

LAMPIRAN PENCIPTA

No	Nama	Alamat
1	Yois Phahirani	Karang Bajang RT 02/RW 26 Tlogoadi, Mlati, Sleman
2	Asniar Aliyu, S.T., M.Eng	Pogung Kidul RT 01/ RW 49 Sinduadi, Mlati, Sleman
3	Arif Basuki, S.T., M.T.	Basen RT 13/ RW 04 Purbayan, Kotagede, Yogyakarta

DOKUMEN

HAK KEKAYAAN INTELEKTUAL KARYA CIPTA

SISTEM KONTROL DAN MONITORING ATAP JEMURAN PAKAIAN MENGGUNAKAN APLIKASI ANDROID

Pencipta :

Yois Phahirani
Asniar Aliyu, ST. M.Eng
Arif Basuki, ST. MT.

INSTITUT TEKNOLOGI NASIONAL YOGYAKARTA

2022

SISTEM KONTROL DAN MONITORING ATAP JEMURAN PAKAIAN MENGGUNAKAN APLIKASI ANDROID

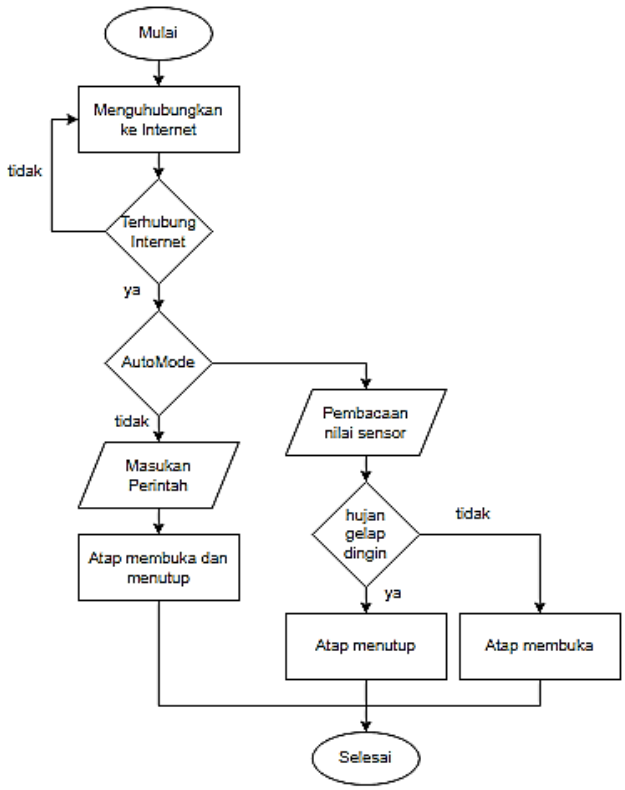
Sistem kontrol dan monitoring atap jemuran pakaian menggunakan aplikasi android merupakan sebuah prototipe berbasis IoT yang dapat bekerja dengan mode otomatis maupun mode manual. Sistem otomatis akan bekerja berdasarkan kondisi lingkungan yang dibaca oleh sensor hujan, sensor LDR, dan sensor DHT11. Dalam pemrogramannya melalui Arduino IDE diatur batas nilai atau *treshold* yang menjadi penentu apakah atap jemuran harus membuka atau menutup. Atap jemuran membuka dan menutup dengan cara menggulung dan digerakkan oleh motor DC. Motor DC akan berhenti berputar ketika ujung atap telah menyentuh *limitswitch.* Data hasil pembacaan sensor dapat dimonitor melalui *firebase* dan aplikasi pada *smartphone* yang dibuat dengan Kodular dan telah dihubungkan dengan ESP32. Mode otomatis sistem aktif pada pukul 07.00 WIB dan atap akan menutup pada pukul 14.00 WIB kemudian beralih ke mode manual. Tetapi ketika sistem masih dalam mode otomatis kemudian ditekan tombol buka-tutup manual maka untuk kembali ke mode otomatis lagi harus menunggu hingga jam yang telah ditentukan yaitu pukul 07.00 WIB. Dalam mode manual atap jemuran dapat dibuka-tutup menggunakan *push button* yang telah dipasang pada rangka jemuran dan dapat menggunakan tombol buka-tutup melalui aplikasi pada *smartphone.* Ketika tombol buka-tutup ditekan maka aplikasi akan mengirimkan perintah ke *firebase* kemudian diteruskan ke ESP32 dan menghasilkan luaran berupa putaran motor DC.

1. Prinsip Kerja

Prinsip kerja sistem ini adalah ketika sensor hujan, sensor cahaya, sensor suhu dan kelembaban mendeteksi adanya perubahan kondisi, maka sensor akan memberikan sinyal yang dikirimkan melalui ESP32 berupa notifikasi ke aplikasi. Luarannya adalah dalam bentuk perintah pada motor DC untuk membuka atau menutup gulungan yang berfungsi sebagai atap jemuran.

2. Agoritma Pemrograman

Algoritma pemrograman yang ada pada sistem ini seperti ditunjukkan pada diagram alir berikut.



Gambar 1. Diagram-alir algoritma pemrograman

3. Listing Program Sistem Kontrol Dan Monitoring Atap Jemuran Pakaian Menggunakan Aplikasi Android

```
#include <WiFi.h>
#include <FirebaseESP32.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include "DHT.h"

#include <addons/TokenHelper.h>
#include <addons/RTDBHelper.h>

#define WIFI_SSID     "YoPha"
#define WIFI_PASSWORD "1122334455"

#define API_KEY       "AIzaSyBymeJBE1-rMahqe2raKSH6PtPMPd40A1M"
#define DATABASE_URL  "atap-otomatis-c2829-default-rtdb.firebaseio.com"

#define USER_EMAIL    "fahiraniyois@gmail.com"
#define USER_PASSWORD "atap12345"

#define DHTPIN 21          // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11      // DHT 11

#define LEDPin       2

#define LDRSensor    34   // Assign the LDR Sensor to pin 12
#define RainSensor   35   // Assign the Rain Sensor to pin 13

#define switchPin_1   15   // Assign the COM Pin Switch 1 to pin 15
#define switchPin_2   4    // Assign the COM Pin Switch 1 to pin 4
#define switchPin_3   32   // Assign the COM Pin Switch 1 to pin 12
#define switchPin_4   33   // Assign the COM Pin Switch 1 to pin 13

#define DriverPin_1   27   // Assign the Driver Pin 1 to pin 27
#define DriverPin_2   26   // Assign the Driver Pin 2 to pin 26
#define DriverPin_EN  14   // Assign the Driver Enable Pin to pin 14

String currentDate;
String currentTime;

String uid;                // Variable to save USER UID
String roofModePath;
String restartDevicePath;
String readIntervalPath;
String deviceUpdatePath;

String closeRoofPath;
String openRoofPath;
String commandUpdatePath;

String sensorReadingPath;
String humiPath     = "/DHT11/humidity";
String tempPath     = "/DHT11/temperature";
String rainPath     = "/RainSensor/analogVal";
String rPercentPath = "/RainSensor/percentage";
String lightPath    = "/LDR/analogVal";
String lPercentPath = "/LDR/percentage";
String timePath     = "/timeStamp/lastUpdateTime";
String datePath     = "/timeStamp/lastUpdateDate";
```

```
// Variables will change:
bool roofMode      = false;
bool restartDevice = false;
bool deviceUpdate  = false;

bool roofClose     = false;
bool roofOpen      = false;
bool commandUpdate = false;

bool openTheRoof   = false;
bool closeTheRoof  = false;

int readInterval  = 5;

bool switchState1 = false;          // the current state of the output pin
int stateSwitch1;                   // the current reading from the input pin
int lastSwitchState1 = LOW;         // the previous reading from the input pin
unsigned long lastDebounceTime1 = 0;  // the last time the output pin was toggled

bool switchState2 = false;          // the current state of the output pin
int stateSwitch2;                   // the current reading from the input pin
int lastSwitchState2 = LOW;         // the previous reading from the input pin
unsigned long lastDebounceTime2 = 0;  // the last time the output pin was toggled

bool switchState3 = false;          // the current state of the output pin
int stateSwitch3;                   // the current reading from the input pin
int lastSwitchState3 = LOW;         // the previous reading from the input pin
unsigned long lastDebounceTime3 = 0;  // the last time the output pin was toggled

bool switchState4 = false;          // the current state of the output pin
int stateSwitch4;                   // the current reading from the input pin
int lastSwitchState4 = LOW;         // the previous reading from the input pin
unsigned long lastDebounceTime4 = 0;  // the last time the output pin was toggled

unsigned long debounceDelay = 50;    // the debounce time; increase if the output flickers

unsigned long readSensorPrevMillis  = 0;
unsigned long sendDataPrevMillis    = 0;
unsigned long requestDataPrevMillis = 0;

// setting PWM properties
const int freq       = 30000;
const int pwmChannel  = 0;
const int resolution  = 8;

int motorSpeed  = 255;

float humi    = 0;
float temp    = 0;

int lightIn   = 0;
int waterRead = 0;

int lightInPercent   = 0;
int waterReadPercent = 0;

int waterThreshold = 2000;
int lightThreshold = 2500;
int tempThreshold  = 27;

bool openStatus    = false;
```

```
bool isRain        = false;
bool isDark        = false;
bool isWarm        = false;

String openTime    = "07:00";
String closeTime   = "14:00";

bool openOnTime    = false;
bool closeOnTime   = false;

WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP);

DHT dht(DHTPIN, DHTTYPE);

FirebaseData fbdo;
FirebaseJson json;

FirebaseAuth auth;
FirebaseConfig config;

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

  pinMode(LDRSensor,   INPUT);
  pinMode(RainSensor,  INPUT);

  pinMode(switchPin_1, INPUT);
  pinMode(switchPin_2, INPUT);
  pinMode(switchPin_3, INPUT);
  pinMode(switchPin_4, INPUT);

  // sets the pins as outputs:
  pinMode(LEDPin, OUTPUT);

  pinMode(DriverPin_1,  OUTPUT);
  pinMode(DriverPin_2,  OUTPUT);
  pinMode(DriverPin_EN, OUTPUT);

  // configure LED PWM functionalitites
  ledcSetup(pwmChannel, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(DriverPin_EN, pwmChannel);

  delay(500);

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");

  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }

  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  // Initialize a NTPClient to get time
  Serial.print("Initialize a NTP Client : ");
```

```
timeClient.begin();

// Set offset time in seconds to adjust for your timezone, for example:
// GMT -1 = -3600
// GMT 0  = 0
// GMT +1 = 3600
timeClient.setTimeOffset(25200);
Serial.println("done!");
Serial.println();

dht.begin();

Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);

config.api_key = API_KEY;

auth.user.email = USER_EMAIL;
auth.user.password = USER_PASSWORD;

config.database_url = DATABASE_URL;
config.token_status_callback = tokenStatusCallback; // see addons/TokenHelper.h

Firebase.begin(&config, &auth);

Serial.println();
Serial.print("Getting User UID"); // Getting the user UID might take a few seconds
while ((auth.token.uid) == "") {
  Serial.print(".");
  delay(500);
}

Serial.println();

// Print user UID
uid = auth.token.uid.c_str();
Serial.print("User UID: ");
Serial.println(uid);
Serial.println();

Firebase.reconnectWiFi(true);

roofModePath      = "/userData/" + uid + "/device/autoMode";
restartDevicePath = "/userData/" + uid + "/device/restartDevice";
readIntervalPath  = "/userData/" + uid + "/device/sensorReadingInterval";
deviceUpdatePath  = "/userData/" + uid + "/device/updateRequest";

closeRoofPath     = "/userData/" + uid + "/commands/roof/closeRoof";
openRoofPath      = "/userData/" + uid + "/commands/roof/openRoof";
commandUpdatePath = "/userData/" + uid + "/commands/updateRequest";

sensorReadingPath = "/userData/" + uid + "/sensorReading";

if (Firebase.setBool(fbdo, deviceUpdatePath.c_str(), true)) {
  Serial.println("Requesting lates data on rtdb..");
}
else {
  Serial.printf("Requesting Data '/device/updateRequest' failed!\n");
  Serial.printf("Reason : [");
  Serial.printf(fbdo.errorReason().c_str());
  Serial.printf("]\n");
  Serial.println();
}
```

```
}

void loop() {
  readSwitches();
  readSensors();

  forceToManual();

  FirebaseRequestData();
  FirebaseSendData();

  if (roofMode == true) {
    if (openOnTime == true && closeOnTime == false) {
      Serial.println("Jam buka sudah terpenuhi");

      if (isRain == false) {
       Serial.println("Tidak Hujan");

        if (isDark == false && isWarm == true) {
          openTheRoof = true;
          digitalWrite(LEDPin, HIGH);
        }
        else {
          closeTheRoof = true;
          digitalWrite(LEDPin, LOW);
        }
      }
      else {
        Serial.println("Hujan");

        closeTheRoof = true;
      }
    }
    else {
      Serial.println ("Jam buka belum terpenuhi");
    }

    if (openOnTime == false && closeOnTime == true) {
      closeTheRoof = true;
    }
  }

  if (roofMode == false) {
    if (roofOpen == true && switchState1 == true) {
      openTheRoof = true;

      roofOpen = false;
      if (Firebase.setBool(fbdo, openRoofPath.c_str(), false)) {
        Serial.printf("Sending Data '/commands/roof/openRoof' success!\n");
      }
      else {
        Serial.printf("Sending Data '/commands/roof/openRoof' failed!\n");
        Serial.printf("Reason : [");
        Serial.printf(fbdo.errorReason().c_str());
        Serial.printf("]\n");
        Serial.println();
      }
    }

    if (roofClose == true && switchState2 == true) {
      closeTheRoof = true;
```

```
      roofClose = false;
      if (Firebase.setBool(fbdo, closeRoofPath.c_str(), false)) {
        Serial.printf("Sending Data '/commands/roof/closeRoof' success!\n");
      }
      else {
        Serial.printf("Sending Data '/commands/roof/closeRoof' failed!\n");
        Serial.printf("Reason : [");
        Serial.printf(fbdo.errorReason().c_str());
        Serial.printf("]\n");
        Serial.println();
      }
    }
  }

  if (openTheRoof == true) {
    openRoof();

    if (switchState1 == false && switchState2 == true) {
      stopRoof();
      openTheRoof = false;

      if (roofMode == false) {
        Serial.println("Roof opened!");
      }
    }
  }

  if (closeTheRoof == true) {
    closeRoof();

    if (switchState1 == true && switchState2 == false) {
      stopRoof();
      closeTheRoof = false;

      if (roofMode == false) {
        Serial.println("Roof closed!");
      }
    }
  }

  deviceRestart();
}

void readSwitches() {
  readSwitch1();
  readSwitch2();
  readSwitch3();
  readSwitch4();
}

void readSwitch1() {
  int reading = digitalRead(switchPin_1);

  if (reading != lastSwitchState1) {
    lastDebounceTime1 = millis();
  }

  if ((millis() - lastDebounceTime1) > debounceDelay) {
    if (reading != stateSwitch1) {
      stateSwitch1 = reading;

      if (stateSwitch1 == HIGH) {
```

```
        // switchState1 = !switchState1;
        switchState1 = true;
        Serial.println("Switch 1 True");
       }
       else {
        switchState1 = false;
        Serial.println("Switch 1 False");
       }
     }
   }
   lastSwitchState1 = reading;
}

void readSwitch2() {
  int reading = digitalRead(switchPin_2);

  if (reading != lastSwitchState2) {
    lastDebounceTime2 = millis();
  }

  if ((millis() - lastDebounceTime2) > debounceDelay) {
    if (reading != stateSwitch2) {
      stateSwitch2 = reading;

      if (stateSwitch2 == HIGH) {
        // switchState2 = !switchState2;
        switchState2 = true;
        Serial.println("Switch 2 True");
       }
       else {
        switchState2 = false;
        Serial.println("Switch 2 False");
       }
     }
   }
   lastSwitchState2 = reading;
}

void readSwitch3() {
  int reading = digitalRead(switchPin_3);

  if (reading != lastSwitchState3) {
    lastDebounceTime3 = millis();
  }

  if ((millis() - lastDebounceTime3) > debounceDelay) {
    if (reading != stateSwitch3) {
      stateSwitch3 = reading;

      if (stateSwitch3 == LOW) {
        // switchState3 = !switchState3;
        switchState3 = true;
        Serial.println("Switch 3 True");
       }
       else {
        switchState3 = false;
        Serial.println("Switch 3 False");
       }
     }
   }
   lastSwitchState3 = reading;
}
```

```
void readSwitch4() {
  int reading = digitalRead(switchPin_4);

  if (reading != lastSwitchState4) {
    lastDebounceTime4 = millis();
  }

  if ((millis() - lastDebounceTime4) > debounceDelay) {
    if (reading != stateSwitch4) {
      stateSwitch4 = reading;

      if (stateSwitch4 == LOW) {
        // switchState4 = !switchState4;
        switchState4 = true;
        Serial.println("Switch 4 True");
      }
      else {
        switchState4 = false;
        Serial.println("Switch 4 False");
      }
    }
  }
  lastSwitchState4 = reading;
}

void readSensors() {
  if (millis() - readSensorPrevMillis > (1000) || readSensorPrevMillis == 0) {
    readSensorPrevMillis = millis();

    DHTSensorReading();
    rainStatusReading();
    lightReading();
    NTPRequestTime();
  }
}

void DHTSensorReading() {
  humi = dht.readHumidity();     // Read Temperature as Celsius (the default)
  temp = dht.readTemperature();  // Read Humidity

  // Check if any reads failed and exit early (to try again).
  if (isnan(humi) || isnan(temp)) {
    Serial.println(F("Failed to read from DHT sensor"));
    return;
  }

  if (temp >= tempThreshold) {
    isWarm = true;
  }
  else {
    isWarm = false;
  }
}

void rainStatusReading() {
  waterRead = analogRead(RainSensor);
  waterReadPercent = map(waterRead, 0, 4092, 100, 0);

  if (waterRead <= waterThreshold) {
    isRain = true;
  }
```

```
  else {
    isRain = false;
  }
}

void lightReading() {
  lightIn = analogRead(LDRSensor);
  lightInPercent = map(lightIn, 0, 4092, 100, 0);

  if (lightIn >= lightThreshold) {
    isDark = true;
  }
  else {
    isDark = false;
  }
}

void NTPRequestTime() {
  /*
    while (!timeClient.update()) {
    timeClient.forceUpdate();
    }
  */

  timeClient.update();

  time_t epochTime  = timeClient.getEpochTime();

  int currentHour   = timeClient.getHours();
  int currentMinute = timeClient.getMinutes();

  //Get a time structure
  struct tm *ptm          = gmtime ((time_t *)&epochTime);
  int monthDay            = ptm->tm_mday;
  int currentMonth        = ptm->tm_mon + 1;
  int currentYear         = ptm->tm_year + 1900;

  char Time[] = "  :  ";
  Time[4]     = currentMinute % 10 + 48;
  Time[3]     = currentMinute / 10 + 48;
  Time[1]     = currentHour   % 10 + 48;
  Time[0]     = currentHour   / 10 + 48;

  currentDate = String(monthDay) + "/" + String(currentMonth) + "/" + String(currentYear);
  currentTime = String(Time);

  // Serial.print("Date: ");
  // Serial.println(currentDate);

  // Serial.print("Time: ");
  // Serial.println(currentTime);

  // Serial.println();

  if (currentTime == openTime) {
    openOnTime  = true;
    closeOnTime = false;
  }

  if (currentTime == closeTime) {
    openOnTime  = false;
    closeOnTime = true;
```

```
  }
}

void forceToManual() {
  if (roofMode == true && (switchState3 == true || switchState4 == true)) {
    if (Firebase.setBool(fbdo, roofModePath.c_str(), false)) {
      roofMode = false;
      Serial.printf("Sending Data '/device/autoMode' success!\n");
    }
    else {
      Serial.printf("Sending Data '/device/autoMode' failed!\n");
      Serial.printf("Reason : [");
      Serial.printf(fbdo.errorReason().c_str());
      Serial.printf("]\n");
      Serial.println();
    }
  }

  if (switchState3 == true) {
    openTheRoof = true;
  }

  if (switchState4 == true) {
    closeTheRoof = true;
  }
}

void FirebaseRequestData() {
  if (Firebase.ready() && (millis() - requestDataPrevMillis > 1000 || requestDataPrevMillis == 0)) {
    requestDataPrevMillis = millis();

    if (Firebase.getBool(fbdo, deviceUpdatePath.c_str(), &deviceUpdate)) {
      Serial.print("/device/updateRequest   : ");
      Serial.println((deviceUpdate) ? "true" : "false");
    }
    else {
      Serial.printf("Requesting Data '/device/updateRequest' failed!\n");
      Serial.printf("Reason : [");
      Serial.printf(fbdo.errorReason().c_str());
      Serial.printf("]\n");
      Serial.println();
    }

    if (deviceUpdate == true) {
      Serial.println();
      Serial.println("================================================");
      Serial.println();

      if (Firebase.getBool(fbdo, roofModePath.c_str(), &roofMode)) {
        Serial.print("/device/autoMode              : ");
        Serial.println((roofMode) ? "true" : "false");
      }
      else {
        Serial.printf("Requesting Data '/device/autoMode' failed!\n");
        Serial.printf("Reason : [");
        Serial.printf(fbdo.errorReason().c_str());
        Serial.printf("]\n");
        Serial.println();
      }

      if (Firebase.getBool(fbdo, restartDevicePath.c_str(), &restartDevice)) {
        Serial.print("/device/restartDevice         : ");
```

```
    Serial.println((restartDevice) ? "true" : "false");
   }
   else {
    Serial.printf("Requesting Data '/device/restartDevice' failed!\n");
    Serial.printf("Reason : [");
    Serial.printf(fbdo.errorReason().c_str());
    Serial.printf("]\n");
    Serial.println();
   }

   if (Firebase.getInt(fbdo, readIntervalPath.c_str(), &readInterval)) {
    Serial.print("/device/sensorReadingInterval : ");
    Serial.println(readInterval);
   }
   else {
    Serial.printf("Requesting Data '/device/sensorReadingInterval' failed!\n");
    Serial.printf("Reason : [");
    Serial.printf(fbdo.errorReason().c_str());
    Serial.printf("]\n");
    Serial.println();
   }

   deviceUpdate = false;
   if (Firebase.setBool(fbdo, deviceUpdatePath.c_str(), false)) {
    Serial.println();
    Serial.println("End of data group.");
   }
   else {
    Serial.printf("Sending Data '/device/updateRequest' failed!\n");
    Serial.printf("Reason : [");
    Serial.printf(fbdo.errorReason().c_str());
    Serial.printf("]\n");
    Serial.println();
   }

   Serial.println();
   Serial.println("================================================");
   Serial.println();
  }

  if (roofMode == false) {
   if (Firebase.getBool(fbdo, commandUpdatePath.c_str(), &commandUpdate)) {
    Serial.print("/commands/updateRequest : ");
    Serial.println((commandUpdate) ? "true" : "false");
   }
   else {
    Serial.printf("Requesting Data '/commands/updateRequest' failed!\n");
    Serial.printf("Reason : [");
    Serial.printf(fbdo.errorReason().c_str());
    Serial.printf("]\n");
    Serial.println();
   }
  }

  if (commandUpdate == true) {
   Serial.println();
   Serial.println("================================================");
   Serial.println();

   if (Firebase.getBool(fbdo, closeRoofPath.c_str(), &roofClose)) {
    Serial.print("/commands/roof/closeRoof : ");
    Serial.println((roofClose) ? "true" : "false");
```

```
    }
    else {
      Serial.printf("Requesting Data '/commands/roof/closeRoof' failed!\n");
      Serial.printf("Reason : [");
      Serial.printf(fbdo.errorReason().c_str());
      Serial.printf("]\n");
      Serial.println();
    }

    if (Firebase.getBool(fbdo, openRoofPath.c_str(), &roofOpen)) {
      Serial.print("/commands/roof/openRoof  : ");
      Serial.println((roofOpen) ? "true" : "false");
    }
    else {
      Serial.printf("Requesting Data '/commands/roof/openRoof' failed!\n");
      Serial.printf("Reason : [");
      Serial.printf(fbdo.errorReason().c_str());
      Serial.printf("]\n");
      Serial.println();
    }

    commandUpdate = false;
    if (Firebase.setBool(fbdo, commandUpdatePath.c_str(), false)) {
      Serial.println();
      Serial.println("End of data group.");
    }
    else {
      Serial.printf("Sending Data '/commands/updateRequest' failed!\n");
      Serial.printf("Reason : [");
      Serial.printf(fbdo.errorReason().c_str());
      Serial.printf("]\n");
      Serial.println();
    }

    Serial.println();
    Serial.println("==================================================");
    Serial.println();
   }
  }
}

void FirebaseSendData() {
  if (Firebase.ready() && (millis() - sendDataPrevMillis > (readInterval * 1000) || sendDataPrevMillis == 0))
{
   sendDataPrevMillis = millis();

   json.set(humiPath,     int    (humi));
   json.set(tempPath,     float  (temp));
   json.set(rainPath,     int    (waterRead));
   json.set(rPercentPath, int    (waterReadPercent));
   json.set(lightPath,    int    (lightIn));
   json.set(lPercentPath, int    (lightInPercent));
   json.set(timePath,     String (currentTime));
   json.set(datePath,     String (currentDate));

   if (Firebase.RTDB.setJSON(&fbdo, sensorReadingPath.c_str(), &json)) {
     Serial.println();
     Serial.printf("Sending Data '/sensorReading' success!\n");
   }
   else {
     Serial.printf("Sending Data '/sensorReading' failed!\n");
     Serial.printf("Reason : [");
```

```
      Serial.printf(fbdo.errorReason().c_str());
      Serial.printf("]\n");
      Serial.println();
    }

    Serial.printf("Humidity    : ");
    Serial.print(humi);
    Serial.printf(" % \n");

    Serial.printf("Temperature : ");
    Serial.print(temp);
    Serial.printf(" C \n");

    Serial.printf("Rain        : ");
    Serial.print(waterRead);
    Serial.printf("[");
    Serial.print(waterReadPercent);
    Serial.printf("]");
    Serial.printf("\n");

    Serial.printf("Light       : ");
    Serial.print(lightIn);
    Serial.printf("[");
    Serial.print(lightInPercent);
    Serial.printf("]");
    Serial.printf("\n");

    Serial.println();
  }
}

void openRoof() {
  // Control the motor's direction in clockwise
  digitalWrite(DriverPin_1, LOW);
  digitalWrite(DriverPin_2, HIGH);

  // analogWrite(DriverPin_EN, motorSpeed);
  ledcWrite(pwmChannel, motorSpeed);
}

void closeRoof() {
  // Control the motor's direction in counter-clockwise
  digitalWrite(DriverPin_1, HIGH);
  digitalWrite(DriverPin_2, LOW);

  // analogWrite(DriverPin_EN, motorSpeed);
  ledcWrite(pwmChannel, motorSpeed);
}

void stopRoof() {
  // Control the motor's to soft braking
  digitalWrite(DriverPin_1, LOW);
  digitalWrite(DriverPin_2, LOW);

  // analogWrite(DriverPin_EN, 0);
  ledcWrite(pwmChannel, 0);
}

void deviceRestart() {
  if (restartDevice == true) {
    if (Firebase.setBool(fbdo, restartDevicePath.c_str(), false)) {
      Serial.println("Waiting device to restart..");
```

```
  Serial.println();

  ESP.restart();
}
else {
  Serial.printf("Sending Data '/device/restartDevice' failed!\n");
  Serial.printf("Reason : [");
  Serial.printf(fbdo.errorReason().c_str());
  Serial.printf("]\n");
  Serial.println();
}
```