Dokumentácia pre Meteorologickú Stanicu

Marek Horecký, Ivana Dukayová, Richard Bielovič, Viliam Vrba, Filip Hlubík

Obsah

1 Úvod a prehľad systému

1.1 Účel

Meteorologická stanica zbiera dáta o vlhkosti, teplote a intenzite slnečného žiarenia pre solárny panel, ktorý poháňa inteligentný skleník Vesna.

1.2 Architektúra systému

- **ESP32-S3**: Základné zariadenie pre zber a odosielanie dát.
- **Senzory**: SHT30, SHT31, SHT45 pre meranie teploty a vlhkosti, LPS35HW pre meranie barometrického tlaku a teploty, fotorezistor pre meranie intenzity slnečného žiarenia.
- **Multiplexer**: Použitý na pripojenie viacerých senzorov s rovnakou I2C adresou.
- **Arduino Cloud**: Platforma pre ukladanie a vizualizáciu nameraných dát.

2 Hardvér

2.1 Použité komponenty

- **ESP32-S3**: Mikroprocesor.
- **SHT30 a SHT31**: Senzory pre vonkajšie meranie teploty a vlhkosti.
- **SHT45**: Senzor pre vnútorné meranie teploty a vlhkosti.
- **LPS35HW**: Senzor pre meranie barometrického tlaku a teploty.
- **Fotorezistor**: Analogový senzor pre meranie intenzity slnečného žiarenia.
- **TCA9548A**: I2C multiplexer.

Sensor	Parameter	Tolerance
SHT30	Temperature Humidity	±0.3°C ±2% RH
SHT31	Temperature Humidity	±0.3°C ±2% RH
SHT45	Temperature Humidity	±0.2°C ±1.5% RH
LPS35HW	Pressure Temperature	±0.5 hPa ±0.8°C
DS18B20	Temperature	±0.5°C

Tabuľka 1: Sensor Accuracy Specifications

Sensor	Response Time (Humidity)	Response Time (Temperature)	Sampling Rate
SHT30	8 seconds	5-30 seconds	Up to 2 measurements/second (0.5 Hz to 2 Hz)
SHT31	8 seconds	5-30 seconds	Up to 2 measurements/second (0.5 Hz to 2 Hz)
SHT45	8 seconds	5-30 seconds	Up to 2 measurements/second (0.5 Hz to 2 Hz)
LPS35HW	N/A	1 millisecond	Up to 25 Hz
DS18B20	N/A	Up to 750 milliseconds	Up to 1 Hz

Tabuľka 2: Sensor Response Times and Sampling Rates

2.2 Schéma zapojenia

1. **ESP32-S3** je zapojené do všetkých senzorov cez I2C zbernicu a jeden analogový pin pre fotorezistor.
2. **Multiplexer TCA9548A** je pripojený k ESP32-S3 a riadi pripojenie jednotlivých senzorov.

   TCA9548A sa využíva na expanziu I2C busu, pr epoužitie viacero senzorov s rovnakou I2C adresou.

   - bus 0: SHT30
   - bus 1: LPS35HW
   - bus 2: SHT4x
   - bus 3: SHT31
3. **DS18B20 temperature sensors** je pripojený k digitálnemu pinu 11.
4. **Fotorezistor** je pripojený k analogovému pinu A15.

3 Použité nástroje

- **Arduino IDE**: Programovacie prostredie.
- **Adafruit knižnice**: Pre komunikáciu so senzormi (Adafruit_LPS35HW, Adafruit_SHT31, Adafruit_SHT4x).
- **Fusion360**: 3D modelovanie ochranného obalu pre ESP32-S3 a senzory.
- **PrusaSlicer**: Príprava modelov na tlač.

4 Komunikácia s Arduino Cloud

4.1 Konfigurácia a nastavenie

1. Vytvorenie účtu na Arduino Cloud.
2. Pridanie nového zariadenia a konfigurácia vlastností (teplota, vlhkosť, tlak, intenzita slnečného žiarenia).
3. Pripojenie ESP32-S3 pomocou Wi-Fi a overenie autentifikácie.

4.2 Formát odosielaných dát

Premenné:

- `float h30`
- `float hfs400`
- `float hlps45`
- `float pressureLPS35hw`
- `float t30`
- `float tfs400`
- `float tLPS35hw`
- `float tlps45`
- `float temp`
- `float temp2`
- `int sensorValue`

Dátové body sú aktualizované každé 5 sekundy a odosielané na cloud.

5 Prevádzka a údržba

5.1 Návod na spustenie

1. Pripojte všetky komponenty k zariadeniu.
2. Nastavime settings upload speed, flash mode (OPI 80Mz), flash size (16 MB)
3. Nahrajte kód do ESP32-S3 pomocou Arduino IDE.
4. Po nahratí kódu sa ESP32-S3 automaticky pripojí k Wi-Fi a začne odosielať dáta na Arduino Cloud.

5.2 Monitorovanie a diagnostika

- Sériový monitor (baud rate 115200) poskytuje diagnostické správy a aktuálne hodnoty zo senzorov.
- LED indikátory na ESP32-S3 môžu signalizovať stav zariadenia.

6 Modelovanie a 3D tlač

- **Fusion360**: 3D modelovanie ochranného obalu pre ESP32-S3 a senzory.
- **PrusaSlicer**: Príprava modelov na tlač.

6.1 Bežné problémy a ich riešenie

- **Senzor neodpovedá**: Skontrolujte zapojenie a adresu I2C.
- **Žiadna komunikácia s cloudom**: Overte Wi-Fi pripojenie a autentifikačné údaje.
- **Zapojenie**: Používaním veľkého množstva senzorov a spájacích káblov často dochádzalo k ich vytrnutiu (neúmyselnému). Bolo taktiež obtiažne nájsť dostatočný počet káblov.

7 Prílohy

7.1 Datasheety

- ESP32-S3 Datasheet
- SHT30 a SHT31 Datasheet
- SHT45 Datasheet
- LPS35HW Datasheet
- Schéma doskz

7.2 Odkazy na knižnice a zdroje

- Adafruit_LPS35HW
- Adafruit_SHT31
- Adafruit_SHT4x

8 Kód a jeho štruktúra

- **initProperties()**: Inicializuje vlastnosti pre komunikáciu s Arduino Cloud.
- **setup()**: Inicializácia sériovej komunikácie, I2C zbernice, multiplexeru a jednotlivých senzorov.
- **loop()**: Hlavná slučka, ktorá pravidelne odčíta dáta zo senzorov a odosiela ich na Arduino Cloud.

```
// Adafruit LPS35HW - Version: Latest

/*
  Sketch generated by the Arduino IoT Cloud Thing "Untitled 3"
  https://create.arduino.cc/cloud/things/c46ee676-fc85-4fea-89cd-f953125d83e9

  Arduino IoT Cloud Variables description

  The following variables are automatically generated and updated when
     changes are made to the Thing

  float h30;
  float hfs400;
  float hlps45;
  float latitude;
  float longitude;
  float pressureLPS35hw;
  float t30;
  float temp;
  float temp2;
  float tfs400;
  float tLPS35hw;
  float tlps45;
  int sensorValue;

  Variables which are marked as READ/WRITE in the Cloud Thing will
     also have functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end of
     this sketch.
*/

#include "thingProperties.h"
#include <Wire.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Adafruit_LPS35HW.h>
#include <Adafruit_SHT31.h>
#include <Adafruit_SHT4x.h>
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

// Create instances for the sensors
Adafruit_SHT31 sht30; // Sensor SHT30 for soil
Adafruit_LPS35HW lps35hw; // Sensor LPS35HW
Adafruit_SHT4x sht4; // Sensor SHT4x
Adafruit_SHT31 sht31; // Another instance of SHT31
```

```
const byte muxAddress = 0x70;
#define ONE_WIRE_BUS 11
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature dallasTempSensor(&oneWire);

// GPS
static const int TXPin = 4, RXPin = 3;
static const uint32_t GPSBaud = 9600;
TinyGPSPlus gps;
SoftwareSerial ss(RXPin, TXPin);

void TCA9548A(uint8_t bus) {
  Wire.beginTransmission(muxAddress);  // Start I2C transmission to
     TCA9548A
  Wire.write(1 << bus);                // Send byte to select bus
  Wire.endTransmission();              // Stop I2C transmission
}

void setup() {
  Serial.begin(115200);
  delay(1500);  // Wait for Serial Monitor

  initProperties(); // Initialize Arduino IoT Cloud properties
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();

  Wire.begin(); // Initialize I2C communication

  // Initialize all sensors on their respective buses
  TCA9548A(0);
  if (!sht30.begin(0x44)) {
    Serial.println("Could not find a valid SHT30 sensor on bus 0,
       check wiring!");
    while (1);
  }

  TCA9548A(1);
  if (!lps35hw.begin_I2C(0x5d)) {
    Serial.println("Could not find a valid LPS35HW sensor on bus 1,
       check wiring!");
    while (1);
  }

  TCA9548A(2);
  if (!sht4.begin()) {
    Serial.println("Could not find a valid SHT4 sensor on bus 2, check
       wiring!");
    while (1);
  }

  TCA9548A(3);
  if (!sht31.begin(0x44)) {
    Serial.println("Could not find a valid SHT31 sensor on bus 3,
       check wiring!");
    while (1);
  }
```

```
  dallasTempSensor.begin(); // Initialize Dallas DS18B20 temperature
      sensor

  ss.begin(GPSBaud); // Initialize GPS
  Serial.println(F("DeviceExample.ino"));
  Serial.println(F("A simple demonstration of TinyGPS++ with an
      attached GPS module"));
  Serial.print(F("Testing TinyGPS++ library v. "));
      Serial.println(TinyGPSPlus::libraryVersion());
  Serial.println(F("by Mikal Hart"));
  Serial.println();
}

void loop() {
  ArduinoCloud.update();  // Handle cloud communication

  sensorValue = analogRead(15); // Read from an analog pin
  Serial.print("Analog sensor value: ");
  Serial.println(sensorValue);

  // Request temperature from Dallas DS18B20 sensors
  dallasTempSensor.requestTemperatures();
  temp = dallasTempSensor.getTempCByIndex(0);  // Update global
      variable for the first sensor
  temp2 = dallasTempSensor.getTempCByIndex(1);  // Update global
      variable for the second sensor
  Serial.print("Dallas temperature sensor 1 value: ");
  Serial.println(temp);
  Serial.print("Dallas temperature sensor 2 value: ");
  Serial.println(temp2);

  // Read and update from all sensors, make sure to update the cloud
      variables
  TCA9548A(0);
  t30 = sht30.readTemperature();
  h30 = sht30.readHumidity();
  printSensorData("SHT30", t30, h30);

  TCA9548A(1);
  tLPS35hw = lps35hw.readTemperature();
  pressureLPS35hw = lps35hw.readPressure();
  printSensorData("LPS35HW", tLPS35hw, pressureLPS35hw);

  TCA9548A(2);
  sensors_event_t hum_event, temp_event;
  sht4.getEvent(&hum_event, &temp_event); // Getting temperature and
      humidity
  tlps45 = temp_event.temperature;
  hlps45 = hum_event.relative_humidity;
  printSensorData("SHT4x", tlps45, hlps45);

  TCA9548A(3);
  tfs400 = sht31.readTemperature();
  hfs400 = sht31.readHumidity();
  printSensorData("SHT31", tfs400, hfs400);

  while (ss.available() > 0) {
```

```
    if (gps.encode(ss.read())) {
      displayGPSInfo();
    }
  }

  if (millis() > 5000 && gps.charsProcessed() < 10) {
    Serial.println(F("No GPS detected: check wiring."));
    while (true);
  }

  delay(2000); // Delay to reduce the frequency of cloud updates
}

void printSensorData(const String& sensorName, float temperature,
   float humidity) {
  if (!isnan(temperature)) {
    Serial.print(sensorName + " Temperature: ");
    Serial.print(temperature);
    Serial.println(" *C");
  } else {
    Serial.println("Failed to read temperature from " + sensorName);
  }

  if (!isnan(humidity)) {
    Serial.print(sensorName + " Humidity: ");
    Serial.print(humidity);
    Serial.println(" %");
  } else {
    Serial.println("Failed to read humidity from " + sensorName);
  }
}

void displayGPSInfo() {
  Serial.print(F("Location: "));
  if (gps.location.isValid()) {
    latitude = gps.location.lat();
    longitude = gps.location.lng();
    Serial.print(latitude, 6);
    Serial.print(F(","));
    Serial.print(longitude, 6);
  } else {
    Serial.print(F("INVALID"));
  }

  Serial.println();
}
```

Listing 1: Zdrojový kód pre meteorologickú stanicu