Wetterstation

Dokumentation zum IoT Projekt 2021

Nick Crisci

TH Köln, nick.cirsci@smail.th-koeln.de

Sebastian Brock

TH Köln, sebastian.brock@smail.th-koeln.de

Paul Fafengut

TH Köln, paul.fafengut@smail.th-koeln.de

1. EINLEITUNG

Im Rahmen des Moduls “Internet of Things” gilt es ein Projekt zum Thema “Smart Environments” mit gewissen Rahmenbedingungen in einem Team á 3 Studierenden umzusetzen. Ziel ist es ein Prototyp für ein innovatives Produkt zu entwickeln. Dieser Prototyp soll zum einen mindestens einen Sensor (physical input) und mindestens einen Aktor (physical output) besitzen, und zum anderen einen Microcontroller enthalten, welcher die Sensoren und Aktoren vernetzt. Außerdem soll das Produkt mit dem Web, einer App oder einem zweiten “Ding” verbunden sein.

* 1. Produktidee

Eine Wetterstation, bestehend aus einem ESP32 Microcontroller, einem Temperatur-, Luftfeuchtigkeit- und Luftdrucksensor und einen Knopf als Sensoren, ein Display für 4 Ziffern als Aktor, sowie eines Solarpanels und eines Solar Power Managers, um die Wetterstation autark funktionieren lassen zu können, wird als Prototyp für dieses Modul entwickelt.

Außerdem wird die Wetterstation stündlich die Messwerte an einen Webserver schicken, welcher die Daten aufbereitet den Nutzern im Web präsentiert.

* 1. Eigene Rahmenbedingungen

Ein wichtiger Punkt bei der Entwicklung des Produktes ist das Einsparen von Strom, damit die Wetterstation nur durch den Akku, der über das Solarpanel geladen wird, betrieben werden kann und so möglichst umweltfreundlich ist. Hierfür wurden verschiedene Maßnahmen ergriffen und es wurde mehr Zeit und Planung in die verwendeten Komponenten gesteckt, da wie auch in den Folgenden Absätzen beschrieben, sich einige Technologien nicht für solch eine Art von Stromversorgung eignen.

1. Technologien
   1. Internetverbindung

Damit die Wetterstation die gemessenen Daten, dem Nutzer im Web präsentieren kann, muss dieser den Webserver erreichen können. Realisiert wird dieses über eine normale W-Lan Verbindung. Um den Stromverbrauch beim Halten der Verbindung zu minimieren, wird der Microcontroller nach Versenden der Daten in den „Deep Sleep Modus“ versetzt. Beim „Deep Sleep Modus“ drosselt der ESP32 seine Funktionen, wie z.B. die Fähigkeit eine Internetverbindung aufzubauen, und somit auch seinen Stromverbrauch. Dadurch kann er nur noch über gewisse Inputs erreicht und wieder aufgeweckt werden. Um den Microcontroller wieder aufzuwecken, wurde eine Funktion implementiert, die die Zeit in Sekunden bis zur nächsten vollen Stunde berechnet und den ESP32 für nur genau diese Zeit in den „Deep Sleep Modus“ versetzt.

Beim Entwickeln der Produktidee, kamen die Bedenken auf, dass eine instabile Internetverbindung durch Platzieren der Wetterstation am Rande oder Außerhalb des W-Lan Signals zu einem Problem führen könnte. Um dieses Problem zu lösen, könnte man statt W-Lan „LoRaWAN“, was kurz für „Long Range Wide Area Network“ steht, verwenden. Auf diese Weise könnten viel größere Entfernungen zum W-Lan Router abgedeckt werden. Allerdings benötigt man für diese Lösung ein Gateway, welcher mit dem Microcontroller kommuniziert, den Payload empfängt und diesen an den Webserver weiterschickt. Dieses weitere Gerät, welches über Strom aus dem normalen Stromnetz betrieben wird, wäre daher hinderlich für unseren nachhaltigen Ansatz.

Ein weiteres Problem, das uns bezüglich einer mangelnden Internetverbindung in den Sinn kam, ist das Fehlschlagen des Uploads des Payloads. Um den Stromverbrauch zu begrenzen, wird der ESP32, wie bereits beschrieben, kurze Zeit nach dem Aufwecken wieder in den „Deep Sleep Modus“ versetzt. Daher kann der Microcontroller nicht durchgehend versuchen die Daten hochzuladen, bis es irgendwann funktioniert. Aus diesem Grund werden den Daten, die im JSON-Format an den Webserver geschickt werden, ein Zeitstempel verpasst, sodass der Webserver auch Daten, die eigentlich schon vor einer Stunde hätten ankommen sollen, einordnen und korrekt darstellen kann. Eine weitere Möglichkeit dieses Problem zu lösen wäre mit Hilfe von MQTT, was kurz für „Message Queuing Telemetry Transport“ steht. Bei MQTT werden die Daten mit Hilfe des Publish-Subscribe-Prinzips versendet und weitergeleitet. Der Microcontroller würde daher seine Daten an einen Broker schicken, welcher dann für die Weiterleitung der Daten zuständig ist und der Microcontroller kann in den „Deep Sleep Modus“ wechseln. Allerdings tritt hier erneut das Problem der weiteren Geräte und damit verbrauch von Strom ein, denn sowohl der Broker muss mit Strom versorgt werden als auch ein weiteres Gerät mit besserer Internetverbindung, welches den Payload dann final an den Webserver schickt.

1. Prototyp

Zur Realisierung des Produkts wird ein IoT-Kit von Grove bereitgestellt.

* 1. Komponenten

Der Prototyp besteht aus den folgenden Bauteilen:

* Grove Base Shield v1.0

Das Grove Base Shield dient als Hauptschnittstelle zwischen den Komponenten und ist für den ESP32 ausgelegt.

Durch die einfachen zugänglichen Steckplätze für die 4-Pin Universal Kabel von Grove, lassen sich die anderen Bauteile einfach verbinden.

* ESP32

Der ESP32 empfängt die Daten der Sensoren, schickt die Befehle an die Aktoren, stellt eine Verbindung zum Internet her und lädt die Daten auf den Webserver hoch.

* Grove Temperature and Humidity Sensor v1.2

Dieser Sensor misst die Temperatur und die Luftfeuchtigkeit und schickt sie an den ESP32

* Grove IMU 10DOF v2.1

Der IMU 10DOF dient in diesem Projekt zur Messung des Luftdrucks, auch wenn er noch mehr Funktionalität besitzt, wie das Anzeigen von magnetischen Flussdichten, das Bestimmen der Position und Beschleunigung in 3 Achsen1.

* Grove Button

Durch den Button wird ein manuelles Starten der Messung ermöglicht

* Grove 4-Digit Display v1.0

Wurde eine manuelle Messung angefordert, wird diese auf dem Display angezeigt.

Außerdem werden auf diesem Display Fehlercodes angezeigt. Eine Auflistung der Fehlercodes ist im Wiki und am Ende der Dokumentation zu finden.

* Waveshare Solar Power Manager

Der Solar Power Manger dient zur Verwaltung der Stromversorgung des ESP32. Hier wird sowohl die Batterie als Akku angeschlossen, als auch das Solar Panel, um den Akku zu laden.

* XCell IFR 14500 (600mAh, 3.2V)

XCell IFR 14500 ist der Akku und damit die Stromversorgung für unsere Wetterstation

* DEBO Solar 5W Entwicklerboard

Das Entwicklerboard von DEBO bietet den richtigen Anschluss und Watt Werte für den Solar Power Manager und lädt den Akku im laufe des Tages wieder auf.

* 4x Grove 4-Pin Universal Kabel

Mit Hilfe der Universal Kabel von Grove lassen sie die Komponenten mit dem Base Shield verbinden.

* USB-Kabel  
  Ein USB-Kabel wird benötigt, um den ESP32 mit dem PC zu verbinden und so die Software aufzuspielen und später um den ESP32 für die Stromversorgung mit dem Solar Power Manager zu verbinden.
  1. Funktionalität
     1. Deep Sleep Mode

Der Deep Sleep Mode dient dem Einsparen von Strom, um die Wetterstation nur mit Hilfe von Solarenergie betreiben zu können. Im Deep Sleep werden alle Komponenten des ESP32 bis auf den RTC (Real Time Clock) Speicher und Peripherie heruntergefahren und verbrauchen keinen Strom mehr. Neben dem Deep Sleep kann sich der ESP32 in dem Modus „Active“, „Modem-Sleep“, „Light-Sleep“ oder „Hibernation“ befinden2. In der Tabelle 1 „ESP32 power modes“ sind die verschiedenen Modi und deren aktiven Komponenten dargestellt.

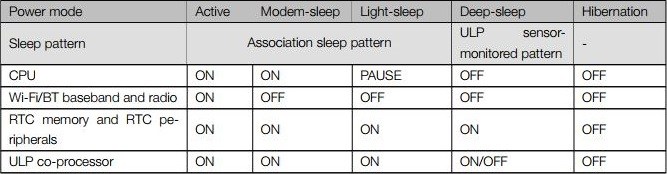


Tabelle 1: ESP32 power modes

Damit der ESP32 von allein zur vollen Stunde sich in den Active Mode begibt, wurden zwei Funktionen geschrieben, *getTimeTillNextHour* und *goToDeepSleep*. Es gibt mehrere Möglichkeiten den ESP32 aus dem Deep Sleep zu holen. Eines davon ist das Aufwecken nach einer bestimmten Zeit. Hierfür ist die Funktion *getTimeTillNextHour* zuständig. Es holt sich die aktuelle Uhrzeit, berechnet die Differenz in Sekunden bis zur nächsten vollen Stunde und gibt diese zurück.

Sobald der ESP32 die Messwerte der Sensoren verarbeitet hat, geht dieser mit Hilfe der Funktion *goToDeepSleep* in den Deep Sleep Mode. Die vorherige Funktion wird aufgerufen, damit die Zeit in Sekunden berechnet wird. Diese wird nun in Mikrosekunden umgerechnet und damit der Timer zum Aufwecken des ESP32 gestellt.

Neben der Möglichkeit den ESP32 zur vollen Stunde zu wecken, wurde das Aufwecken durch Betätigen des angeschlossenen Knopfs aktiviert. Auf diese Weise hat der Nutzer die Möglichkeit die aktuellen Daten sich anzeigen zulassen, ohne auf die volle Stunde zu warten.

Sind beide Funktionen zum Aufwecken aktiviert, wird der Deep Sleep Mode des ESP32 gestartet und alle stromintensiven Komponenten werden runtergefahren.

* + 1. Validieren der Messwerte

Obwohl die eingebauten Sensoren auf höchste Genauigkeit ausgerichtet sind, kann es dennoch vorkommen, dass Messfehler entstehen. Sollte beispielsweise der Temperatur Sensor der prallen Sonne ausgesetzt sein, und sich somit starker als gewollt erhitzen, könnte dieser falsche Werte liefern. Solange diese Abweichungen gering sind, stellt dies kein Problem dar. Sobald jedoch regelrechte “Ausreißer” entstehen, welche sich deutlich von den übrigen Messwerten absetzen, so liegt es im Interesse des Nutzers, dass eben solche Messwerte vorher überprüft und ggf. korrigiert werden.

Genau hier kommt die Validierung der Messwerte ins Spiel. Indem ein Service zwischen die Wetterstation und dem Frontend eingefügt wird, welcher die Messwerte bevor diese weitergegeben werden validiert, können fehlerhafte Messwerte validiert und / oder korrigiert werden.

Hierfür können statistische Funktionen herangezogen werden um beispielsweise den neu gemessenen Wert gegenüber der Standardabweichung der zuletzt gemessenen Werte zu validieren.

Des Weiteren können eben solche Funktionen auch den Fehler korriegieren, indem ein Fehlerhafter Wert durch den Durchschnitt der zuletzt gemessenen Werte ersetzt wird.

Dem Nutzer werden nach der Validierung also nur noch Werte angezeigt, welche die wahren Begebenheiten der Umwelt zu dem entsprechenden Zeitpunkt wiederspiegeln.

* + 1. Wettervorhersage

Wenn ein Nutzer schon eine hauseigene Wetterstation besitzt, dann würde eine persönliche Wettervorhersage die Qualität des Services der Wetterstation noch weiter steigern.

Dies lässt sich mithilfe des verbauten Luftdruck-Sensors in einfacher Form realisieren.

Es ist ein wenig Wetterkunde von Nöten, um zu verstehen, weshalb eine einfache Wettervorhersage mittels des Luftdrucksensors möglich ist.

Ist der aktuelle Luftdruck erhöht, so spricht man von einem Hochdruckgebiet. Solche Hochdruckgebiete sind in der Regel mit gutem Wetter verbunden4. Sinkt hingegen der Druck über die Zeit ab, so ist dies ein Indiez auf ein nahendes Tief, welches wahrscheinlich Regen, Wind oder sogar Gewitter mit sich bringen kann.

Da die Wetterstation stündlich den Luftdruck abfragt, kann mithilfe dieser Werte eine einfache Wettervorhersage mit einer Genauigkeit von etwa 60-70% implementiert werden, die anhand der Änderungen des Luftdrucks berechnen kann, ob das Wetter eher besser oder schlechter wird.

* 1. Kommunikation

Die Wetterstation kommuniziert im Grunde lediglich mit einem weiteren „Ding“ oder Service. Der Gedanke hierbei war der, dass die Wetterstation möglichst wenig kommunizieren sollte, um so Energie einzusparen.

Wie in 2.1 und 3.2.1 erwähnt, wurden verschiedene Möglichkeiten der Kommunikation besprochen und abgewägt. Die Entscheidung fiel auf eine einfache Kommunikation über die hauseigene W-Lan Verbindung. Um bei der Kommunikation die Last möglichst gering zu halten, wird das JSON Format verwendet, welches dafür sorgt, dass die zu sendenen Daten eine sehr geringe Größe besitzen.

Diese Daten werden von der Wetterstation, nach dem Aufwachen aus dem Deep Sleep, per HTTP Post Request an den entsprechenden Webserver gesendet.

Die Verwendung des HTTP Protokolls rührt daher, dass hierfür kein großer Konfigurationsaufwand anfällt, und keine weiteren Komponenten von Nöten sind.

Der Gedanke bei der Kommunikation das UDP Protokoll zu verwenden, kam zwar kurz auf, da die Wetterstation in diesem Falle nur senden und nicht auch eine Antwort empfangen würde, jedoch ist der Nutzen an dieser Stelle zu gering.

Denn das TCP Protokoll bietet die Möglichkeit, zu überprüfen, ob die Daten von der Wetterstation auch tatsächlich ihr Ziel erreichen.

Sollte bei der Kommunikation ein Problem auftreten und die Daten erreichen den Webserver nicht, so lässt sich dieser Status mittels des HTTP Response Objektes abfragen, und entsprechend darauf reagieren.

Quellen

1. <https://wiki.seeedstudio.com/Grove-IMU_10DOF/> [11.09.2021]
2. <https://randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources/> [19.09.2021]
3. <https://www.wetterstation.net/wetter-vorhersagen/> [19.09.2021]
4. <https://www.lernhelfer.de/schuelerlexikon/physik/artikel/luftdruck-und-wetter> [18.09.2021]
5. <https://www.wetteronline.de/wetterlexikon/hochdruckgebiet> [18.09.2021]