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.

1. Technologien

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

* 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 Fähig 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.

Quellen

1. …