



Höhere Technische Bundeslehranstalt  
und Bundesfachschule  
im Hermann Fuchs Bundesschulzentrum

# sunnyHOME

## Diplomarbeit

**Schulautonomer Schwerpunkt Communications**

*ausgeführt im Schuljahr 2018/2019 von:*

Andreas Herz, 5BHELS

*Betreuer:*

Dipl.-Ing. Roland Sageder

9. April 2019

# **Thema: sunnyHOME**

## **Subthemen und Bearbeiter:**

### **Umweltdatenerfassung**

Andreas Herz, 5BHELS

*Betreuer:* Dipl.-Ing. Roland Sageder

### **Basisstation**

Andreas Herz, 5BHELS

*Betreuer:* Dipl.-Ing. Roland Sageder

## DIPLOMARBEIT DOKUMENTATION

**Verfasser/innen** Andreas Herz

**Jahrgang** 5BHELS 2018/2019  
**Schuljahr**

**Thema der  
Diplomarbeit** sunnyHOME

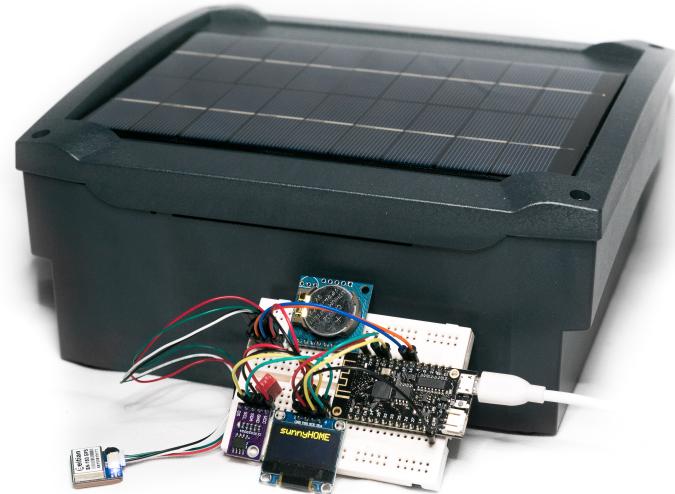
**Aufgabenstellung** Die Aufgabenstellung war, eine Energie-autarke Wetterstation zu entwickeln. Es sollen verschiedene Sensoren, wie zum Beispiel Niederschlags- oder UV-Sensoren verwendet werden. Die Daten der Sensoren sollen dann über einen Mikrocontroller eingelesen und an einen Web-Server gesendet werden. Dieser wertet diese Daten aus, welche dann über eine Website oder eine App visualisiert werden.

**Realisierung** Realisiert wird das Projekt mithilfe des ESP32 Mikrocontrollers. Dieser hat die benötigte Energie-Effizienz und lässt sich leicht an verschiedenste Sensoren anbinden. Mit Hilfe eines GPS-Receiver soll der Standort der Wetterstation bestimmt werden, und mit Hilfe von Solarzellen, soll das ganze System Energie-autark realisiert werden.

**Ergebnisse** Die wichtigsten Ergebnisse sollen sein:

- Konzept fertiggestellt
- Sensoren können messen
- GPS-Signal wird empfangen
- Datenübertragung funktioniert
- Visualisierung funktioniert
- Energie-Versorgung funktioniert
- Diplomarbeit verfasst

**Typische Grafik, Foto etc.  
(mit Erläuterung)**



**Möglichkeiten der  
Einsichtnahme  
in die Arbeit**

Archiv der HTL Braunau, bzw.  
[https://diplomarbeiten.berufsbildendeschulen.  
at/](https://diplomarbeiten.berufsbildendeschulen.at/)

**Approbation (Datum / Unterschrift)**

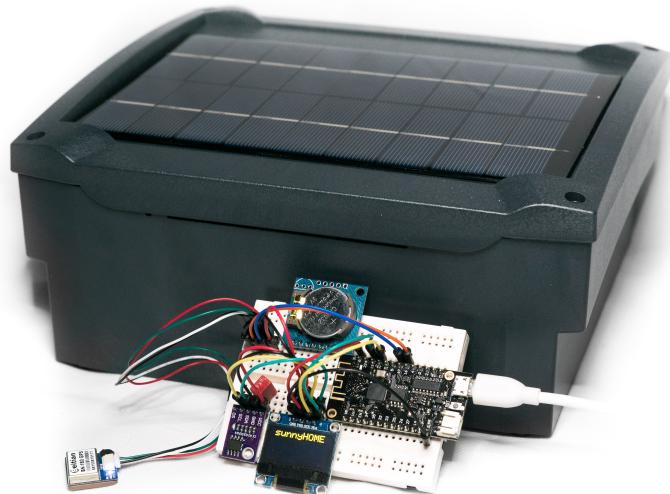
Prüfer/Prüferin:

Direktor/Direktorin  
Abteilungsvorstand/Abteilungsvorständin

## DIPLOMA THESIS Documentation

<b>Author(s)</b>	Andreas Herz
<b>Form</b>	5BHELS 2018/2019
<b>Academic year</b>	
<b>Topic</b>	sunnyHOME
<b>Assignment of tasks</b>	The major task was to develop a self-sufficient weather station. Various sensors should be used, like for example rainfall- and UV-sensors. The measured data should be read from the ESP32, and then published to a web-server. The web-server evaluates this data, which then will be visualized via a website or a mobile application.
<b>Realisation</b>	The realisation will happen with help of the ESP32 microcontroller. The ESP32 has the needed energy-efficiency and can be easily connected to a wide variety of sensors. A GPS-receiver should provide the current location of the weather station, and solar cells should keep the system self-sufficient.
<b>Results</b>	<p>The most important results should be:</p> <ul style="list-style-type: none"><li>• concept is done</li><li>• sensors are able to measure</li><li>• receiving the GPS-signal</li><li>• transferring of data works</li><li>• visualization works</li><li>• power-supply works</li><li>• diploma thesis is written</li></ul>

**Illustrative graph,** The sunnyHOME weather station:  
**photo**  
(incl. explanation)



**Accessibility of  
diploma thesis**

HTL Brauna archive, or  
[https://diplomarbeiten.berufsbildendeschulen.  
at/](https://diplomarbeiten.berufsbildendeschulen.at/)

**Approval (date / signature)**

Examiner

Head of College / Department

# Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als angegebene Quellen und Hilfsmittel nicht direkt benutzt und die benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Braunau/Inn, 09.04.2019

Andreas Herz

*Ort, Datum*

*Verfasser*

*Unterschrift*

---

# Inhaltsverzeichnis

<b>Vorwort</b>	x
<b>Kurzfassung</b>	xi
<b>Abstract</b>	xii
<b>1 Einleitung</b>	1
1.1 Einführung in sunnyHOME . . . . .	1
1.2 Organisation . . . . .	2
1.2.1 Microsoft To-Do . . . . .	2
1.2.2 Excel . . . . .	3
1.2.3 GitHub . . . . .	7
1.2.4 Microsoft Project . . . . .	8
1.3 Zielsetzung . . . . .	9
1.3.1 Prioritäten . . . . .	9
<b>2 Grundlagen</b>	10
2.1 Was ist IoT? . . . . .	10
2.2 Energieeffiziente Systeme . . . . .	11
<b>3 Konzept</b>	12
3.1 Grundlegende Idee . . . . .	12
3.2 Kommunikation . . . . .	12
3.3 Energie-Management . . . . .	13
3.4 Visualisierung . . . . .	14
3.5 Funktionsablauf . . . . .	14
<b>4 Umsetzung</b>	15
4.1 Hardware . . . . .	15
4.1.1 LM75A . . . . .	15
4.1.2 ESP8266 . . . . .	16
4.1.3 ESP32 . . . . .	16
4.1.4 GPS-Modul . . . . .	17
4.1.5 Real-Time Clock . . . . .	17
4.2 Software . . . . .	18
4.2.1 Atom Editor . . . . .	18
4.2.2 PlatformIO . . . . .	19
4.2.3 Ubidots [18] . . . . .	20
4.3 Bibliotheken . . . . .	21
4.3.1 HardwareSerial.h . . . . .	21
4.3.2 Wire.h . . . . .	21
4.3.3 WiFi.h & PubSubClient.h . . . . .	22

4.3.4	DS1307.h . . . . .	22
4.3.5	TinyGPS++.h . . . . .	23
<b>5</b>	<b>Bedienungsanleitung</b>	<b>24</b>
5.1	Suche eines geeigneten Standortes . . . . .	24
5.2	Einstellen von Ubidots . . . . .	25
5.3	Modifizierung des Codes . . . . .	26
<b>6</b>	<b>Fazit und Persönliche Erfahrungen</b>	<b>27</b>
6.1	Fazit . . . . .	27
6.1.1	Zusammenfassung der Ergebnisse . . . . .	27
6.1.2	Mögliche Erweiterungen . . . . .	27
6.2	Persönliche Erfahrungen . . . . .	28
<b>A</b>	<b>Diverse Anhänge</b>	<b>29</b>
A.1	Projekttagebuch . . . . .	29
	<b>Literaturverzeichnis</b>	<b>32</b>
	<b>Abbildungsverzeichnis</b>	<b>34</b>
	<b>Quelltextverzeichnis</b>	<b>35</b>
	<b>Autoren</b>	<b>37</b>

# Vorwort

Sehr geehrte Damen und Herren, liebe Leserschaft,

vor Ihnen findet sich eine Diplomarbeit zum Thema „sunnyHOME – Umweltdatenerfassung“, welche von dem HTL-Braunau Schüler Andreas Herz verfasst wurde. Dieser besuchte an der HTL Braunau den schulautonomen Zweig „Communications“, und schieb auf Grund des Abschlusses die vor Ihnen liegende Diplomarbeit.

Ende des letzten Schuljahres (2017/18), entschloss ich mich (Andreas Herz), mit einem Klassenkameraden, die Themenstellung einer Energie-autarken Wetterstation in Angriff zu nehmen. Da dieser Klassenkamerad jedoch kurzfristig nicht mehr aufsteigen durfte, wurde das Projekt zu einer Einzelarbeit.

Mit Hilfe von DI. Roland Sageder, welcher der zugewiesene Betreuer des Projektes war, konnten wir uns bereits vor den Sommerferien ein grobes Bild des Vorhabens machen. Auch wenn in den Sommerferien 2018 recherchiert wurde, mussten nach der vorgezogenen Matura die Pläne, auf Grund der plötzlichen Einzelarbeit, verändert werden. Das Projekt musste auf die Grundfunktionen begrenzt werden, und Zeitpläne wurden erstellt. Frau Margit Fuchs, welche zur Aushilfe beim Projektmanagement zugewiesen war, half mir mit guten Vorschlägen dabei. Während des Jahres wurde aber immer wieder klar, dass man für Softwareprojekte oft mehr Zeit braucht als angesetzt ist. Immer wieder gab es Hindernisse und Pannen, in welchen ich mir zusätzliches Wissen aneignen musste um weiterzukommen. Im Endeffekt sehe ich dann jedoch ein, dass jede Panne eine lehrreiche Lektion war.

Hiermit möchte ich die Gelegenheit nutzen, um meinen Dank an all diejenigen auszusprechen, die mir bei meinem Projekt geholfen haben. Insbesondere möchte ich dem Herrn Dipl. Ing. Roland Sageder danken. Für seine Geduld mit mir, in Situationen in denen ich vielleicht etwas mehr als nur einmal eine Erklärung gebraucht habe und auch die Motivation, welche er mir immer wieder gegeben hat. Aber auch Frau Margit Fuchs möchte ich danken. Ohne sie hätte ich das Projekt definitiv nicht so gut organisieren können.

Ich wünsche Ihnen nun viel Freunde beim Lesen dieser Abschlussarbeit.

Andreas Herz

4963 St. Peter am Hart, 9. April 2019

# Kurzfassung

In dieser Diplomarbeit geht es hauptsächlich um die Arduino-Programmierung eines ESP32 im Zusammenhang mit IoT (Internet of Things) und Energy-Harvesting. Für den Endbenutzer soll eine Wetterstation geschaffen werden, welche robust, unabhängig von einer Stromversorgung und leicht zu modifizieren ist.

„sunnyHOME“ ist ein Gerät, welches Wetterdaten sammelt und kabellos ins Internet publiziert. Es können verschiedene Sensoren, wie zum Beispiel Niederschlags- oder UV-Sensoren, verwendet werden. Die Daten der Sensoren werden dann über einen Mikrocontroller eingelesen und an einen Web-Server gesendet. Dieser wertet die Daten aus, welche dann über eine Website oder eine App visualisiert werden. Somit kann man von überall auf der Welt nachschauen, ob zum Beispiel die Blumen im Garten genug Wasser bekommen oder nicht.

# **Abstract**

This diploma thesis focuses on the Arduino programming of an ESP32 in combination with IoT (Internet of Things) and energy harvesting. For the consumer it should be a weather station, which is robust, independent of power supply and it should be easy to modify.

“sunnyHOME“ is a device, which collects weather-information, and publishes it wirelessly to the internet. Various sensors can be used, such as rainfall- or UV-sensors. The data from the sensors is then read by a microcontroller and sent to a web-application. There gets the data evaluated, and then visualized via a website or a mobile-application. Therefore, everyone can check from anywhere in the world, if for example the flowers get enough water in the own garden.

# 1 Einleitung

## 1.1 Einführung in sunnyHOME

Um im Urlaub zu sehen, wie das Wetter zurzeit zu Hause ist, musste man bisher immer bei öffentlichen Wetterberichten nachschauen. Diese sind jedoch nicht immer präzise. Das soll nun geändert werden, indem eine Hardware entwickelt werden wird, welche Wetterdaten sammelt und versendet.



Abbildung 1.1: sunnyHOME-Logo

sunnyHOME ist ein Gerät, welches Wetterdaten sammelt und kabellos an einen Empfänger weiter sendet.

Es können verschiedene Sensoren, wie zum Beispiel Niederschlags- und UV-Sensoren verwendet werden. Die Daten der Sensoren werden dann über einen Mikrocontroller eingelesen und an einen Web-Server gesendet. Dieser wertet die Daten aus, die dann über eine Website oder eine App visualisiert werden. Somit kann man von überall auf der Welt nachschauen, ob die Blumen im Garten genug Wasser bekommen oder nicht.

## 1.2 Organisation

Das Projekt “sunnyHOME” wurde mit Hilfe von verschiedenen Tools organisiert. Die grundlegendsten sind in den folgenden Seiten beschrieben.

### 1.2.1 Microsoft To-Do

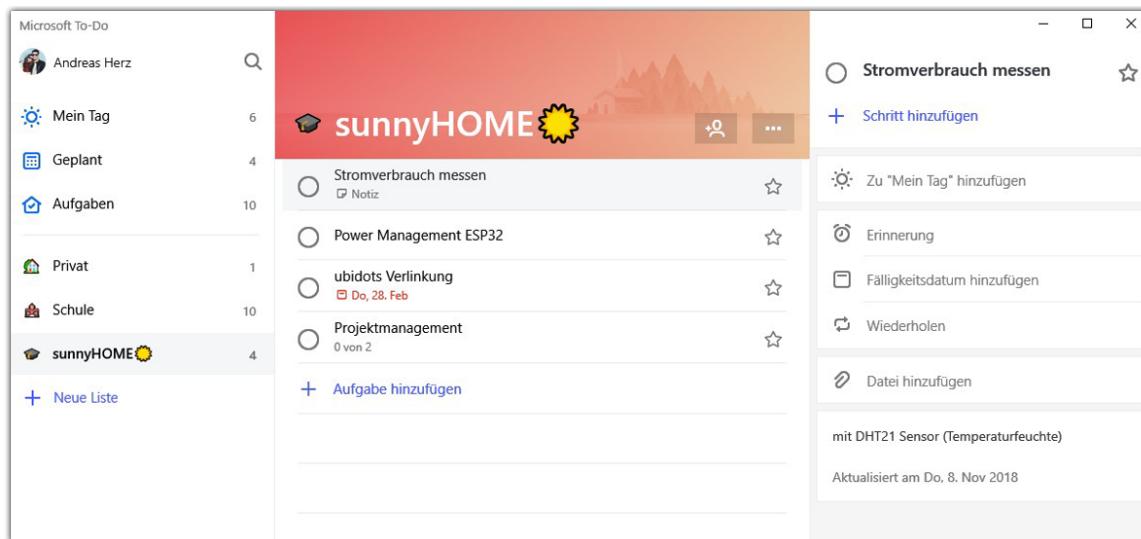


Abbildung 1.2: Microsoft To-Do Screenshot

Microsoft To-Do ist eine übliche To-Do-Listen Applikation. Sie ist für Windows, Android sowie iOS verfügbar. Man kann zwischen Aufgabenbereichen trennen, und den einzelnen Aufgaben Termine zuweisen. Erstellte Termine werden über das Microsoft-Konto synchronisiert, und um einen guten Überblick über die kommende Agenda zu haben, im Kalender angezeigt.

### 1.2.2 Excel

Mit Hilfe von Microsoft Excel, sind die Arbeitszeiten und die durchgeführten Arbeitsschritte protokolliert worden. Man kann angeben, zu wie viel Prozent ein Arbeitsschritt fertiggestellt worden ist und ebenfalls Beschreibungen hinzufügen, was genau durchgeführt worden ist, wo sich Probleme ergeben haben, und was das nächste Mal gemacht werden sollte.

AUFGABENLISTE										
KONTO	MEINE AUFGABEN	DATUM	UHRZEIT ANFANG	UHRZEIT ENDE	PAUSEN	SUMME UHRZEIT	ERLEDIGT	FERTIG	NOTIZEN	
SCHULE	Layout	13.09.18	9:40	13:30		3:50	100 %	●	Besprechung des weiteren Vorgehens	
SCHULE	Zeitpläne	20.09.18	8:00	17:00	1:00	8:00	100 %	●	Erstellung eines GitHub-Projekts, Objekt-Struktur-Plans, Zeitplan, Arduino und GitHub für Atom installiert	
SCHULE	Projektsteckbrief	27.09.18	8:00	11:00		3:00	100 %	●	Ausfüllen eines Steckbriefs. Es fehlen noch persönliche Daten	
SCHULE	Testprogramm mit PlatformIO	27.09.18	11:00	16:30		5:30	100 %	●	Es wird nun mit Atom und einem Package namens PlatformIO programmiert.	
FREIZEIT	Organisation	30.09.18	14:30	16:00		1:30	100 %	●	Zeitmanagement, Installation der Software auf Heim-PC	
FREIZEIT	Testprogramm mit PlatformIO	30.09.18	16:00	22:00	2:00	4:00	100 %	●	GPS-Daten abruf funktioniert. Grafische Anzeige auf OLED nicht. OLED wahrscheinlich kaputt	
FREIZEIT	Testprogramm mit PlatformIO	01.10.18	11:00	12:00	0:00	1:00	100 %	●	Fehler wurde entdeckt. Testprogramm funktioniert	
SCHULE	GPS_to_Display Testprogramm	04.10.18	8:00	17:00	1:15	7:45	100 %	●		
FREIZEIT	Logo Design	07.10.18	20:00	21:30	0:00	1:30	100 %	●	verschiedene Beispiele angefertigt. Favorit ausgesucht	
SCHULE	Umbau auf ESP32 und Testprogramm	11.10.18	8:00	17:00	1:15	7:45	100 %	●	Erfahren des ESP32, Befassung mit Thematik und Umbau hardware- und softwareseitig	
FREIZEIT	Organisation	17.10.18	21:30	23:30	0:15	1:45	100 %	●		
SCHULE	GPS_to_Display Testprogramm	18.10.18	8:00	17:00	1:00	8:00	100 %	●	Display gibt nur Anzahl der Satelliten, Alt., Spd. Und Destination aus	

Abbildung 1.3: Protokoll Screenshot

Gerade bei größeren Projekten ist es sehr wichtig, Arbeitsschritte aus der Vergangenheit zurückverfolgen zu können. Sobald eine Woche nicht mehr gearbeitet wird, sind oft wichtige Ziele oder nicht vervollständigte Schritte vergessen.

## Zeitplan

Zeitplanung Maturaprojekt			
	Datum	KW	Andreas Herz
Wintersemester			
1.	10.09. – 14.09.18	37	Planung: strukturierung des Projekts /der Ideen; MO, 10.09.: Schulbeginn DI, 11.09.: Eröffnungskonferenz
2.	17.09. – 21.09.18	38	Erstellen eines GitHub-Projekts; Erstellen des Zeitplans und des Objektstrukturplans; Auseinandersetzung mit der Software DI, 18.09.: RDP - vorgezogen 5BDHELS MI, 19.09.: RDP - vorgezogen 5CHELS MI, 19.09.: SRDP - 8:00 Herbsttermin
3.	24.09. – 28.09.18	39	Aufnahme von GPS-Daten mit ESP32
4.	01.10. – 05.10.18	40	GPS lesen
5.	08.10. – 12.10.18	41	Logo Design, GPS lesen
6.	15.10. – 19.10.18	42	Anschluss des Transievers und Versuch Daten zu übertragen. MO-MI Abschlussfahrt Graz DO, 25.10.: schulautonom frei FR, 26.10.: Nationalfeiertag
7.	22.10. – 26.10.18	43	FREI
8.	29.10. – 02.11.18	44	Lösungssuche für Stromversorgung bezüglich Energy-Harvesting DO, 01.11.: Allerheiligen FR, 02.11.: Allerseelen
9.	05.11. – 09.11.18	45	MO, 05.11.: schulautonom frei

Abbildung 1.4: Zeitplan Screenshot

Der in Excel erstellte Zeitplan ist quasi das Gegenstück zum Protokoll (siehe: 1.2.2). Er visualisiert einem die kommenden Termine, und welche Arbeitsschritte in welcher Woche zu erledigen sind. Der Plan ist in Kalenderwochen strukturiert, wobei nur Arbeitstage gezählt werden. Außerdem werden einem zu beachtende Termine, Feiertage und Events neben den einzelnen Wochen angezeigt.

## Objektstrukturplan

Ebenfalls in Excel wurde ein Objektstrukturplan erstellt. Dieser orientiert sich an Projektergebnissen. Diese Projektergebnisse werden dann in Teilergebnisse aufgeteilt und hierarchisch untereinander gegliedert.

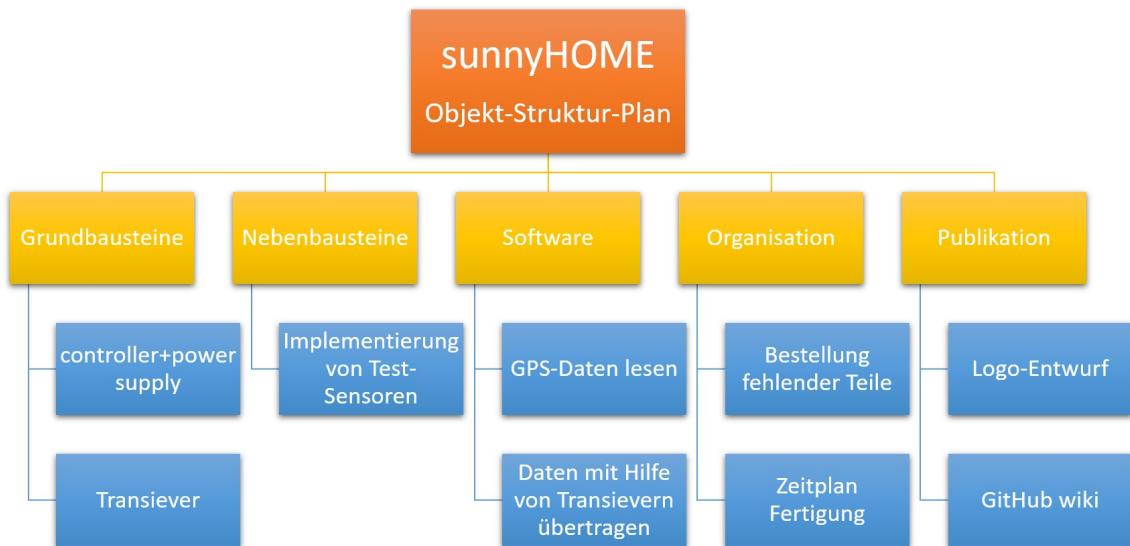


Abbildung 1.5: Objektstrukturplan Screenshot

Vervollständigt wird der Objektstrukturplan durch einen Projektstrukturplan. (siehe 1.2.2)

[1]

## Projektstrukturplan

Ein Projektstrukturplan orientiert sich hingegen einem Objektstrukturplan (siehe: 1.2.2) nicht am Ergebnis, sondern an den Projektphasen. Es ist direkt bemerkbar, dass dieser detaillierter ist als der Objektstrukturplan.

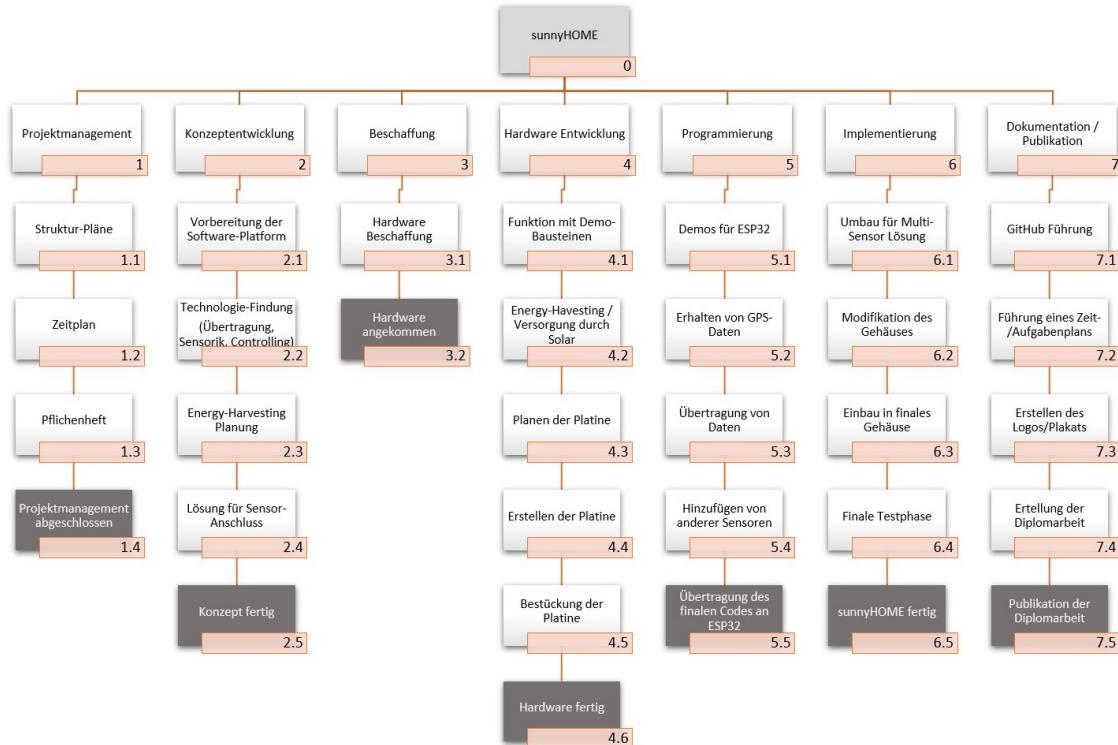


Abbildung 1.6: Projektstrukturplan Screenshot

Die Projektphasen werden von links nach rechts ihrer Durchführungsreihenfolge angeordnet. Lediglich das Projektmanagement bleibt immer links (siehe 1.6: 1), da es von Anfang bis Ende des Projektes gemacht wird. Schlussendlich endet jede Arbeitsphase mit einem Meilenstein, der das Ende der Phase kennzeichnet.

[1]

### 1.2.3 GitHub

Ein großer Teil des Projektmanagements ist über GitHub gelaufen. Mit Hilfe von GitHub können andere Menschen einen kompletten Überblick über das Projekt erhalten. Dafür gibt es eine Main-Page, über welche man das Datei-Verzeichnis ansehen und eine Zusammenfassung des Projektes lesen kann. Quellcodes sind für jeden zugänglich, und sogar vergangene Veränderungen an diesen sind anschaubar. Zudem ist es möglich, ein projekteigenes Wiki zu erstellen. GitHub wurde in diesem Projekt aber hauptsächlich wegen der Online-Synchronisierung verwendet. Um die Synchronisierung leichter zu gestalten, gibt es eine Desktopanwendung, welche Veränderungen hochlädt und neuste Updates herunterlädt. Dadurch lässt es sich von jedem beliebigen PC, am Projekt weiterarbeiten.

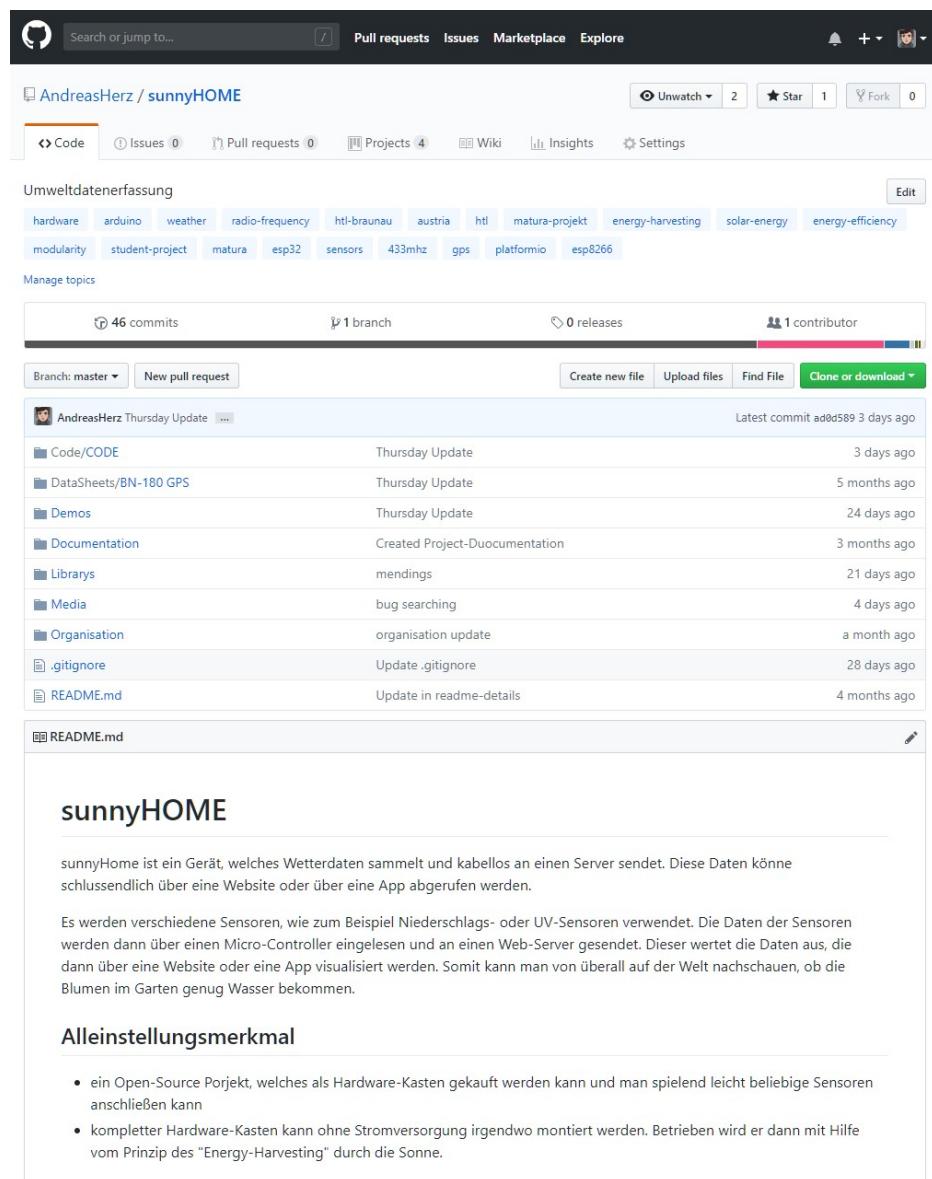


Abbildung 1.7: GitHub Screenshot

### 1.2.4 Microsoft Project

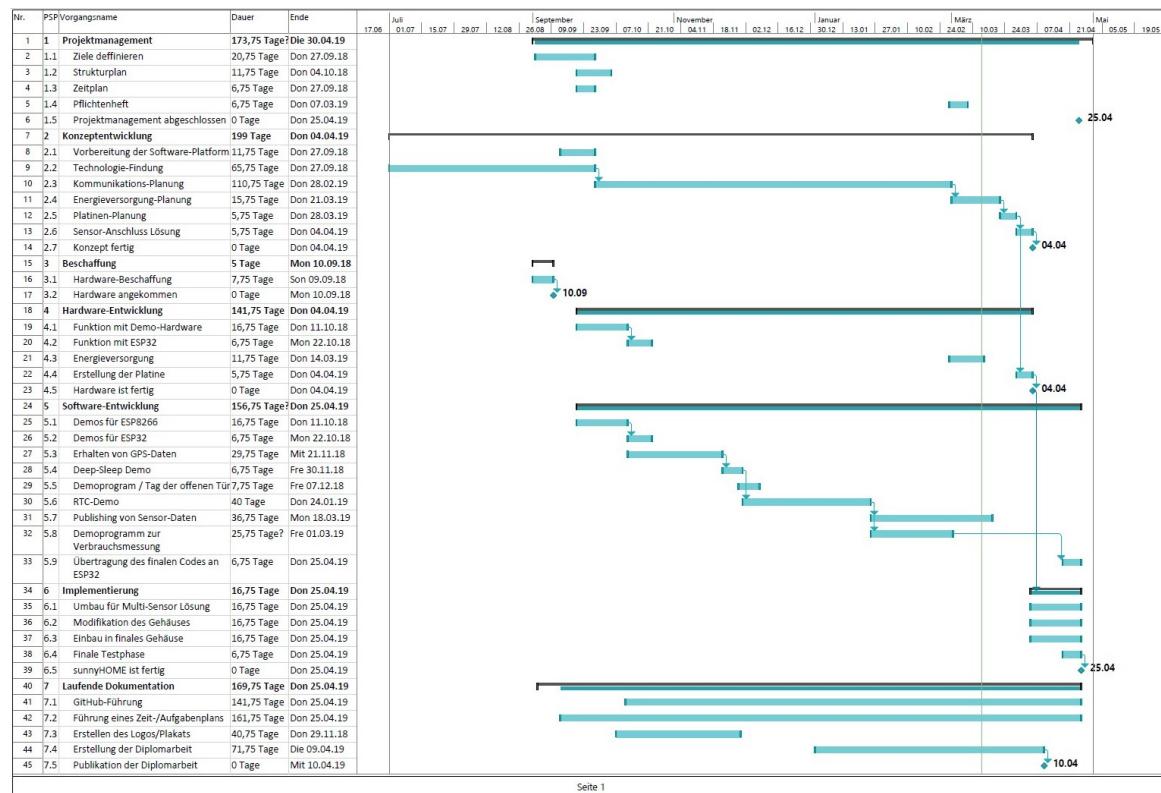


Abbildung 1.8: Mit Microsoft Project erstelltes GANTT-Diagramm

Mit Microsoft Project lassen sich spielend leicht, komplexe GANTT-Diagramme erstellen. Das Diagramm macht es dann möglich, dem Betrachter einen guten Rundumblick über das gegebene Projekt zu bieten. Vor allem für Außenstehende, welche sich über das Projekt und dessen Fortschritt informieren wollen, eignet es sich hervorragend. Man sieht, wann welche Arbeitsschritte durchgeführt wurden, was zurzeit das Thema ist, und welcher Arbeitsschritt der Zukunft, zu welchem Zeitpunkt ansteht. Die angegebenen Arbeitsschritte sind hierbei die selben, welche auch im Projektstrukturplan zu finden sind (siehe 1.2.2).

## 1.3 Zielsetzung

Es soll eine Hardware entwickelt werden, die verschiedene Wetterdaten vor Ort sammelt, und über einen Web-Server von überall abgreifbar macht. Ein weiteres Ziel ist es Mithilfe von Website und App Daten übersichtlich zu visualisieren.

### Die wichtigsten 5 Meilensteine:

- Konzept fertig
- Hardware funktioniert
- Software funktioniert
- Datenübertragung funktioniert
- Diplomarbeit verfasst

### 1.3.1 Prioritäten

Zum einen sollte die Wetterstation sich selbst mit Energie versorgen können. Dann sollte sie modular sein, damit Hobby-Bastler leicht zwischen Funktionalitäten wechseln können. Aus diesem Grund ist das Projekt auch Open-Source (siehe: 1.2.3). Es soll eine grafische Benutzeroberfläche entstehen, von der aus man alles gut im Blick hat. Die Box soll dann zusätzlich über GPS lokalisiert werden können. Die folgenden Meilensteine sind nach Priorität sortiert.

1. Ein Energie-effizientes System welches Unabhängig von einer Stromversorgung ist
2. Ein System zur Übertragung der Daten welches Energie-effizient ist
3. Eine Hardware die multifunktional mit verschiedenen Sensoren kompatibel ist
4. Ein einfaches modulares System, um beliebige Sensoren anzubringen
5. Energie-effiziente Sensoren

## 2 Grundlagen

### 2.1 Was ist IoT?

IoT ist heutzutage nicht weg zu denken. Deswegen bedarf der Begriff an dieser Stelle einer Erklärung. Das „Internet of Things“ (IoT oder auch zu Deutsch „Internet der Dinge“) verbindet Physisches und Virtuelles miteinander und publiziert dann im Endeffekt gesammelte Informationen. So sollen zum Beispiel immer kleiner werdende Computer, Menschen bei Tätigkeiten unterstützen. Diese werden sogar in Kleidung eingearbeitet, damit eingebaute Sensoren Daten sammeln können. Die Daten werden im Endschrift über das Internet publiziert, worüber man diese dann von überall auf der Welt einsehen kann.



Abbildung 2.1: IoT Imagebild [2]

Eine weitere Veränderung, welche mit IoT kommt ist, dass veraltete Techniken von der Idee her komplett neu überarbeitet werden. Wie zum Beispiel bei einer normalen Klingel. Früher konnte sie nur eins: klingeln. Heute verfügen Klingeln über Bewegungssensoren, Kameras und Mikrofone, um einem über alles Bescheid zu geben was vor der Tür passiert.

IoT findet sich nicht nur im privaten Gebrauch. Ein Begriff, welcher große Gemeinsamkeiten aufweist wäre „Industrie 4.0“ (oder auch Industrial Internet). Beide Gebiete haben das Ziel, Geräte und Maschinen zu vernetzen.

## 2.2 Energieeffiziente Systeme

Energy-Harvesting ist in dieser Diplomarbeit ebenfalls ein Thema, da eine Energie-autarke Stromversorgung für die Wetterstation nötig ist. Aber dann stellt sich die Frage: Was ist Energy-Harvesting eigentlich?



Abbildung 2.2: Green Energy Imagebild [4]

Beim Energy-Harvesting geht es grundsätzlich darum, die verlorene Energie einer Funktionalität zumindest teilweise wiederzuverwerten. Es ist ja zurzeit so, dass wir Energiequellen nicht vollkommen effektiv nutzen. Man nehme zum Beispiel Kohlekraftwerke: Diese haben heutzutage einen Wirkungsgrad von 40% [5].

Energy-Harvesting ist im speziellen jedoch für kleinere Projekte mit niedrigem Stromverbrauch gedacht. Wie zum Beispiel für eine kleine Wetterstation. Andere Anwendungsbeispiele wären zum Beispiel Uhren, welche durch Erschütterung ihre Energie beziehen oder Sensoren, welche dank Wärmeeinstrahlung funktionieren.

# 3 Konzept

## 3.1 Grundlegende Idee

sunnyHOME sollte eine Wetterstation werden, welche für jeden Hobby-Elektroniker leicht zu bedienen und zu konfigurieren ist. Man sollte sie einfach in den Wald stellen können, ohne sich jegliche Sorgen um Stromversorgung machen zu müssen. Das heißt sie sollte Energie-autark werden.

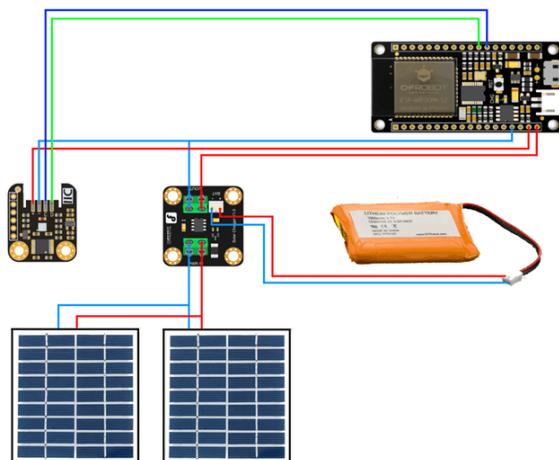


Abbildung 3.1: Aufbau der Wetterstation.[7]

Ebenfalls sollte eine Funkverbindung implementiert werden. Dies wäre nötig, um eine Kommunikation über mehrere Kilometer gewährleisten zu können.

In einem weiteren Schritt sollte eine intuitive Website und App programmiert werden. Über diese sollte man dann die empfangenen Sensordaten auslesen, und den Standort der Wetterstation abrufen können.

## 3.2 Kommunikation

Ein ESP32 (siehe: 4.1.3) hat standardmäßig WLAN und Bluetooth verbaut. Ein Ziel des Projektes war jedoch, die Positionierung der Wetterstation, weit weg von jeglichen WLAN-Netzen und Bluetooth-Receiveern zu positionieren. Um dies zu verwirklichen war zuerst ein Funk-Modul vorgesehen. Im späteren Laufe des Projektes, wurde die Kommunikation über WLAN vorgezogen, da diese keine zusätzliche Hardware für einen MQTT-Broker braucht, und Zeit aufgeholt werden musste.

### 3.3 Energie-Management

Für die Energie-Versorgung ist eine 9V-Solarzelle beschafft worden. Diese wurde anderen Solarzellen vorgezogen, weil sie eine bessere „Low-Light Performance“ bietet. Da sie eine höhere Spannung liefert, können kleinere Ströme wahrgenommen werden, welche dann über ein Energy-Harvesting Modul den Akku speisen.



Abbildung 3.2: 9V Solarzelle [3]

Energy-Harvesting Module können wie Kondensatoren verstanden werden. Sie werden mit niedrigen Strömen aufgeladen, und wenn sie sich dann aufgeladen haben, wird der ganze Strom mit einem Schub in den Akku gespeist. Nötig ist das, da Akkus keine zu niedrigen Ströme aufnehmen können.

Um zusätzlich an Energie zu sparen, gibt es beim ESP32 verschiedene „Power-Modes“:

- **Active Mode:** Der Chip ist voll funktionstüchtig. Er kann Empfangen, Versenden und Zuhören
- **Modem-Sleep Mode:** Der CPU ist betriebsbereit und der Takt lässt sich konfigurieren. Funkverbindungen sind ausgeschalten
- **Light-Sleep Mode:** Der CPU ist pausiert. Der RTC-Speicher und die RTC-Peripherien (RTC ... Real-Time Clock), sowie der ULP Co-Prozessor (ULP ... Ultra-Low-Power) sind am Laufen. Verschiedene Wake-Up Events können das Aufwachen des Chips verursachen.
- **Deep-Sleep Mode:** Nur der RTC-Speicher und die RTC-Peripherien sind eingeschalten. Verbindungsdaten sind im RTC-Speicher gespeichert. ULP Co-Prozessor ist funktionsbereit.
- **Hibernation Mode:** Der interne 8MHz Oszillator und der ULP Co-Prozessor sind ausgeschalten. Nur der RTC-Timer am langsamen Takt und einige RTC GPIOs sind aktiv. Der RTC-Timer oder die RTC-GPIOs können den Chip aus dem Hibernation Mode aufwecken.

[6]

Im Programm wurde der Deep-Sleep Mode gewählt. Es kann ein Intervall eingestellt werden, in dem der ESP immer hochfährt, die Daten sendet, und dann wieder in den Schlaf-Modus zurückkehrt.

### 3.4 Visualisierung

Wie bereits in 3.1 erwähnt, sollte eine App und eine Website realisiert werden. Hauptanliegen dieser Applikationen waren:

- eine umfassende Übersicht über gemessene Sensordaten
- das Tracken der Wetterstation
- eine intuitive, benutzerfreundliche Oberfläche
- die Möglichkeit Einstellungen des ESP zu verwalten

Um das Pensum des Projektes zu kürzen, musste auf die Entwicklung von eigenen Applikationen verzichtet werden. Dafür wurde ein online MQTT-Broker implementiert, welcher es möglich macht, alle obigen Punkte zu realisieren (siehe: 4.2.3).

### 3.5 Funktionsablauf

Für den Funktionsablauf des Programms wurde eine Struktur festgelegt. In der folgenden Abbildung 3.3 sieht man, nach welcher Reihenfolge die Teilaufgaben des Programms geordnet sind.



Abbildung 3.3: Illustration zum Funktionsablauf

Sobald der Controller aus dem Deep-Sleep Mode (siehe: 3.3) aufwacht, soll versucht werden, eine Verbindung zum Internet und zum MQTT-Broker aufzubauen. Ist dies geschehen, wird auf GPS-Daten Empfang gewartet. Sobald sie dann empfangen werden, wird die RTC mit der GPS-Zeit synchronisiert. Danach erfolgen die Sensor-Messungen, welche im nächsten Schritt, zusammen mit den GPS-Daten, an den MQTT-Broker versendet werden sollen. Ist die Routine zu Ende, wird der Deep-Sleep Modus wieder gestartet, und der Timer gestellt, nachdem der Controller wieder aufwachen soll.

# 4 Umsetzung

## 4.1 Hardware

### 4.1.1 LM75A

Der LM75A ist ein Temperatursensor, welcher via I<sup>2</sup>C kommuniziert. In der folgenden Abbildung 4.1 ist er zu sehen.

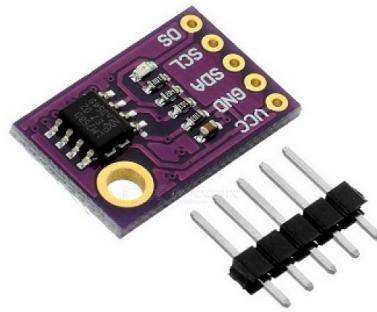


Abbildung 4.1: LM75A[8]

Die Messung der Temperatur ist eine der grundlegendsten Funktionen, welche eine Wetterstation können soll. Zusätzlich ist der LM75A mit einem Durchmesser von 23mm sehr klein und billig in der Anschaffung.

Um Störgrößen zu vermeiden, wurde ein kleiner Kondensator mit  $1 \mu F$ , zwischen Versorgung und Ground verbaut.

#### 4.1.2 ESP8266

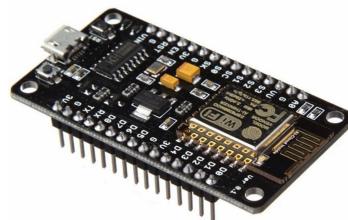


Abbildung 4.2: ESP8266[9]

Angefangen wurde das Projekt mit dem ESP8266. Dieser ist ein Mikrocontroller, welcher unter Arduino spielend leicht programmierbar ist. Positive Aspekte dessen sind, dass es viele stabile Bibliotheken gibt, welche von vielen Nutzern verwendet werden. Nur ist der ESP8266 unbrauchbar geworden, als es um energiesparendere Wege ging. So musste dann ein neuer Mikrocontroller verwendet werden (siehe 4.1.3).

#### 4.1.3 ESP32

Der ESP32 ist ein Mikrocontroller aus dem Hause Espressif. Er eignet sich ideal für IoT-Projekte (siehe: 2.1), da er WLAN und Bluetooth an Bord hat. Außerdem besitzt er einen Dual-Core Prozessor, welcher für erhöhte Energie-Effizienz ausgelegt ist.

[10]

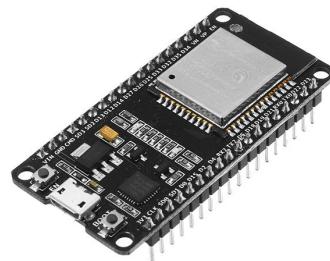


Abbildung 4.3: ESP32 [11]

Nachdem der ESP8266 (siehe: 4.1.2) nicht genügend Energie-effizient war, ist der ESP32 als neuer Controller ausgewählt worden. Dieser hat nämlich einen Deep-Sleep Modus 3.3 integriert. Probleme hat es mit dem ESP32 jedoch auch gegeben. Viele Bibliotheken, welche es für den ESP32 gibt, sind nicht zuverlässig. Oft musste bei diesen für das eigene Projekt nachgebessert werden.

#### 4.1.4 GPS-Modul

Der u-blox M8030-KT ist ein GPS-Empfänger, welcher über Serielle Schnittstelle angesprochen wird. In diesem Projekt ist er in einem Modul von Beitian verbaut (siehe Abbildung 4.4). Über HardwareSerial.h wird der empfangene Datenstrom des GPS-Empfängers ausgelesen. Dieser Datenstrom kann dann mithilfe der Bibliothek „TinyGPS++.h“ verwertet werden.



Abbildung 4.4: Beitian BN-180 im Vergleich zu einer Münze [12]

#### 4.1.5 Real-Time Clock

Die Echtzeituhr DS1307 ist ein Modul zum Zählen der Zeit. Sie hat einen integrierten Speicher und ist sehr Energie-effizient. Wenn die Echtzeituhr nicht über den Controller versorgt wird, gibt es die Möglichkeit, eine Batterie zu platzieren. Die Kommunikation mit dem Controller verläuft über I<sup>2</sup>C.

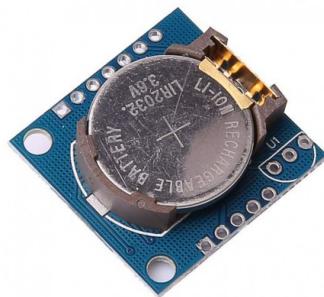


Abbildung 4.5: DS1307 Real-Time Clock [13]

Der ESP32 hat eigentlich bereits eine Real-Time Clock verbaut. Nur ist diese nicht präzise genug und verfälscht die Zeit auf lange Sicht. Deswegen wurde sich für den DS1307 entschieden.

## 4.2 Software

### 4.2.1 Atom Editor

Atom ist ein moderner Texteditor, welcher sich durch verschiedenste Pakete, Bibliotheken und Designs erweitern und modifizieren lässt. Da Atom ein Produkt von GitHub ist, funktioniert die Anbindung daran einwandfrei. Außerdem gibt es eine Menge an nützlichen Tools, wie zum Beispiel die „smarte Autokorrektur“.

[14]

In der folgenden Abbildung (4.6) sieht man einen Screenshot des Atom-Editors.

```

main.cpp — D:\Users\Andi\Documents\GitHub\sunnyHOME — Atom
File Edit View Selection Find Packages Help PlatformIO

Project G: main.cpp G: LM75A.cpp
sunnyHOME
  Code
    include
    lib
    src
      libraries
        WiFi
          DS1307.cpp
          DS1307.h
          LM75A.cpp
          LM75Ah
          LM75Sh
          TinyGPS+.c
          TinyGPS+.h
          main.cpp
          Wire.cpp
          Wire.h
      test
        ignore
        travis.yml
        platformio.ini
        DataSheets
        Demos
        Documentation
        Libraries
        Media
        Organisation
        README.md
        sunnyHOME_GANTT.jpg
  OLED_TEST
    include
    lib
    src
      Adafruit_GFX.cpp
      Adafruit_GFX.h
      Adafruit_SSD1306.cpp
      Adafruit_SSD1306.h
      gfont.h

void setup() {
  Serial.begin(9600);
  Serial_2.begin(9600, SERIAL_8N1, RXD2, TXD2);
  Wire.begin(19,23);
  WiFi.begin(WIFISSID, PASSWORD);

  // by default, we'll generate the high voltage from the 3.3v Line internally! (neat!)
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C); // initialize with the I2C addr 0x3C (for the 128x64, grounded)
  // init done

  Serial.println("\n\n***** Welcome to sunnyHOME *****");
  Serial.println("***** you're personal Weather Station *****");
  // draw scrolling text
  testscrolltext();
  display.clearDisplay();
  display.display();

  // Connecting to WiFi
  Serial.println();
  Serial.print("Wait for WiFi...");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }

  Serial.println("");
  Serial.println("Wifi Connected");
  Serial.println("IP address: ");
}

2 results found for 'VARIABLE_LABEL_LAT'
VARIABLE_LABEL_LAT
Replace in current buffer

```

Abbildung 4.6: Atom Screenshot[15]

Gewählt wurde der Atom-Editor, da man ihn auf den meisten Platformen installieren kann und die Installation ebenfalls keine große Hürde ist. Und wie bereits erwähnt wurde, sind die Funktionalitäten dieses Editors sehr praktisch. Um jedoch einen ESP32 programmieren zu können reicht ein gewöhnlicher Texteditor nicht aus. Aus diesem Grund musste das PlatformIO-Paket installiert werden (siehe 4.2.2).

### 4.2.2 PlatformIO

PlatformIO ist eine IDE<sup>1</sup>, welche im speziellen für das Programmieren von Mikrocontrollern gedacht ist. Es lassen sich die geschriebenen Codes auf Fehler überprüfen, man hat die Möglichkeit die geschriebenen Programme mit Kabel oder kabellos auf den Controller zu laden, und ein Serieller Monitor, über welchen man Nachrichten des Mikrocontrollers empfangen kann, ist ebenfalls verbaut.

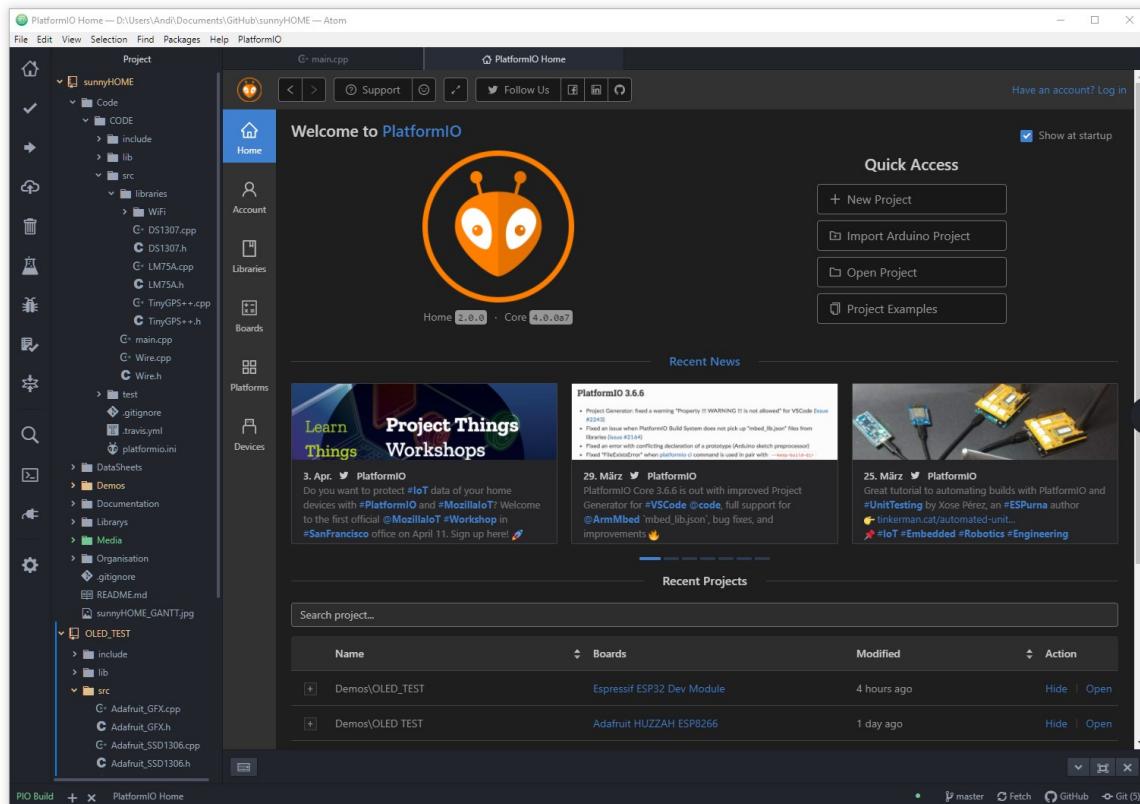


Abbildung 4.7: PlatformIO Screenshot[16]

Entschlossen wurde sich für diese IDE, da diese eine Lösung für die Programmierung mit Atom (siehe: 4.2.1) gewesen ist.

<sup>1</sup>Wikipedia-Eintrag: [https://de.wikipedia.org/wiki/Integrierte\\_Entwicklungsumgebung](https://de.wikipedia.org/wiki/Integrierte_Entwicklungsumgebung), 06. April 2019

#### 4.2.3 Ubidots [18]

Ubidots ist ein MQTT-Broker, welcher wie ein Cloud-Service funktioniert. Mit Hilfe von Ubidots lassen sich die Daten eines Mikrocontrollers spielend leicht im Internet publizieren und visualisieren. Man kann sich beispielsweise Statistiken über Temperaturunterschiede der letzten Wochen anzeigen lassen, oder natürlich ebenfalls auf aktuell aufgenommene Daten wie GPS-Standort zugreifen.

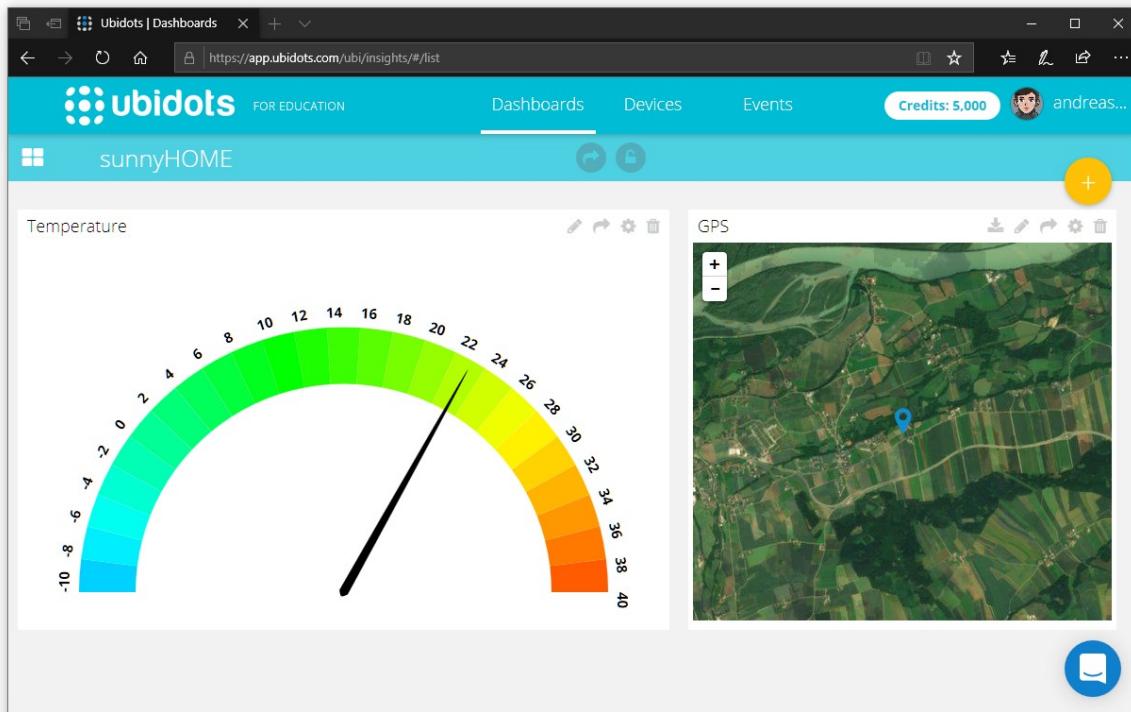


Abbildung 4.8: Ubidots-Dashboard Screenshot[18]

Zu Beginn des Matura-Projektes, ist noch das Erstellen einer eigenen Website und einer eigenen App geplant gewesen. Da das Projekt schlussendlich doch alleine durchgeführt wurde, musste eine einfachere Lösung gefunden werden. So wurde dann Ubidots als Darstellungs-Plattform gewählt.

## 4.3 Bibliotheken

### 4.3.1 HardwareSerial.h

Die „HardwareSerial-Bibliothek“ behandelt die Hardware-Schnittstellen (UART) des ESP32. Grundsätzlich sind am Mikrocontroller drei Schnittstellen verbaut. Die standardmäßige Pinbelegung der Schnittstellen kann jedoch problematisch werden, da dieselben Pins zum Beispiel die Ansteuerung des internen Flashspeichers oder Reseteingänge belegen. Dafür bietet der ESP32 aber eine einfache Lösung: Mit Hilfe der HardwareSerial Bibliothek lassen sich die Hardware-Schnittstellen mit beinahe jedem IO-Pin verbinden.

[19]

In Listing 4.1 wird die Implementierung der seriellen Schnittstelle des GPS-Moduls gezeigt. Über diese werden im eingeschalteten Modus dauerhaft GPS-Daten an den Controller geliefert.

```

1 //////////////////////////////////////////////////////////////////
2 // GPS-Reseiver
3 //////////////////////////////////////////////////////////////////
4 HardwareSerial gps(2); // (2) weil UART2
5
6 void setup() {
7   // Datenrate: 9600 Bit/s, Rx-Pin: 16, Tx-Pin: 17
8   gps.begin(9600, SERIAL_8N1, 16, 17);
9 }
```

Listing 4.1: HardwareSerial Initialisierung

### 4.3.2 Wire.h

Die „Wire-Bibliothek“ erlaubt es einem, via I<sup>2</sup>C zu kommunizieren. Sie hat eine ähnliche Funktionalität wie die „HardwareSerial-Bibliothek“. In dem man die gewünschten Pins in „Wire.begin()“ einfügt, lässt sich die I<sup>2</sup>C-Kommunikation einstellen. Wenn nichts eingefügt wird, werden die Standard Pins gewählt.

Nachdem ein ESP32-Devboard während des Jahres kaputt gegangen ist, musste bei dem nächsten ESP32-Board auf die Standard Pin-Belegung verzichtet werden. Aus diesem Grund mussten, wie in Listing 4.2 beschrieben, die Pins neu definiert werden.

```

1 #include <Wire.h>
2
3 void setup() {
4   // SDA: 19, SCL: 23
5   Wire.begin(19,23);
6 }
```

Listing 4.2: I<sup>2</sup>C Konfiguration

Somit läuft die komplette I<sup>2</sup>C-Kommunikation über die ausgewählten Pins. Werden jedoch Bibliotheken, welche man für I<sup>2</sup>C-Bausteine eingebunden hat verwendet, muss aufgepasst werden, dass diese Pin-Belegung nicht verändert wird.

### 4.3.3 WiFi.h & PubSubClient.h

Die ESP32-Bibliothek „WiFi.h“ ist für die Verbindung zum WLAN-Netz verantwortlich. Im folgenden Listing 4.3 ist der Code für die Verbindung zu einem WLAN-Netzwerk zu sehen. Es wird nur das einfügen des WLAN-Namen und WLAN-Passwortes benötigt.

```

1 #include <WiFi.h>
2
3 void setup() {
4     WiFi.begin(WIFI_SSID, PASSWORD);
5 }
```

Listing 4.3: WLAN Konfiguration

Um eine Verbindung zu einem MQTT-Broker herzustellen braucht es aber noch mehr. Dazu wird die Bibliothek „PubSubClient.h“(Publish-Subscribe Client) benötigt. Diese ermöglicht es einem, MQTT-Topics zu versenden. Listing 4.4 zeigt die Publikation des Temperatur-Topics.

```

1 //***** Temperatur-Publikation *****/
2 // hier wird die ID des Ubidots-Gerätes eingetragen
3 sprintf(topic, "%s%s", "/v1.6/devices/", DEVICE_LABEL);
4 // payload wird gereinigt
5 sprintf(payload, "%s", "");
6 // hier wird die ID der Ubidots-Variable eingetragen
7 sprintf(payload, "{\"%s\"::", VARIABLE_LABEL_TEMP);
8 // hier wird der String mit der Temperatur eingetragen
9 sprintf(payload, "%s {"value": %s}", payload, str_temperature);
10 // Datenpaket wird versendet
11 client.publish(topic, payload);
```

Listing 4.4: Erstellen der Payload und Publikation der Temperatur

Das verschickte Datenpaket (die Payload) sieht folgendermaßen aus:

```
{"temperature": {"value": 26.37}}
```

### 4.3.4 DS1307.h

Die Bibliothek „DS1307.h“ ist für die RTC (Real-Time Clock, siehe: 4.1.5) zuständig. Um die aktuelle Zeit einzutragen, wird die GPS-Zeit mit der RTC-Zeit periodisch synchronisiert.

In Listing 4.5 ist das Eintragen der GPS-Zeit und des GPS-Datums in die Real-Time Clock zu sehen.

```

1 #include <DS1307.h>
2 RTC_DS1307 rtc;
3
4 void setup() {
5     rtc.begin()
6     // TinyGPSTime &d ... GPS-Datum, TinyGPSTime &t ... GPS-Zeit
7     rtc.adjust(DateTime(d.year(), d.month(), d.day(), t.hour(), t.minute(), t.
8         second()));}
```

Listing 4.5: Synchronisierung der GPS-Zeit mit der RTC-Zeit

### 4.3.5 TinyGPS++.h

Die Bibliothek „TinyGPS++.h“ ist vor allem für eine Sache zuständig: das Übersetzen des ankommenden seriellen Datenstroms des GPS-Receiver (siehe: 4.1.4).

Es gibt jedoch auch einige nützliche Funktionen, welche die Bedienung des GPS-Receiver vereinfachen (siehe: Listing4.6).

```

1 //***** GPS Functions *****/
2 // Entschlüsselt Datenstrom
3 static void smartDelay(unsigned long ms);
4 // Gibt gewünschte Daten als Float im Seriellen Monitor aus
5 static void printFloat(float val, bool valid, int len, int prec);
6 // Gibt gewünschte Daten als Integer im Seriellen Monitor aus
7 static void printInt(unsigned long val, bool valid, int len);
8 // Gibt aktuelles Datum und Uhrzeit im Seriellen Monitor aus
9 static void printDateTime(TinyGPSSDate &d, TinyGPSTime &t);

```

Listing 4.6: TinyGPS++ Funktionen

Die wichtigste der Funktionen ist die „smartDelay“-Funktion. Sie entschlüsselt den ankommenden seriellen GPS-Datenstrom und packt die entschlüsselten Informationen in die „TinyGPSPlus“-Variable:

```

1 static void smartDelay(unsigned long ms)
2 {
3     unsigned long start = millis();
4     do
5     {
6 // Solange ein serieller Datenstrom ankommt,
7 // soll dieser entschlüsselt werden
8     while (HWSerialGPS.available())
9 // gps ist eine TinyGPSPlus-Variable
10     gps.encode(HWSerialGPS.read());
11 } while (millis() - start < ms);
12 }

```

Listing 4.7: „smartDelay“-Funktion

Damit die Daten über MQTT versendet werden können, müssen sie als Strings gespeichert werden. Es gibt jedoch keine Möglichkeit, die Koordinaten als Strings zu erhalten. Dazu muss die folgende Wandlung gemacht werden:

```

1 // 1. Parameter: Breitengrad in float
2 // 2. Parameter: Zahl darf 9 Stellen lang sein
3 // 3. Parameter: mit maximal 6 Nachkommastellen
4 // 4. Parameter: wird in folgende String gewandelt
5 dtostrf(gps.location.lat(), 9, 6, str_lat); // für MQTT
6 dtostrf(gps.location.lng(), 9, 6, str_long); // für MQTT

```

Listing 4.8: Float zu String Wandlung für MQTT

## 5 Bedienungsanleitung

### 5.1 Suche eines geeigneten Standortes

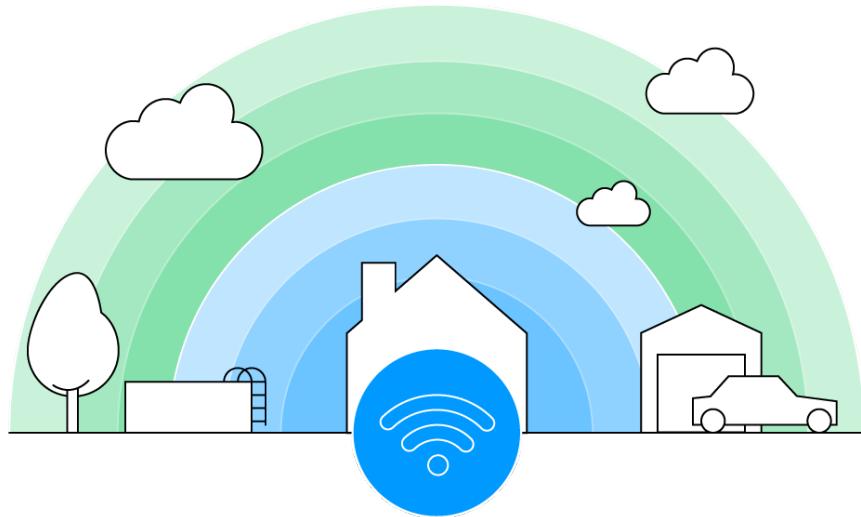


Abbildung 5.1: Atom Screenshot[20]

Damit sunnyHOME überhaupt Daten publizieren kann, braucht die Wetterstation einen Ort mit Verbindung zu einem WLAN-Netz. Handelsübliche WLAN-Netze geben eine mögliche Reichweite von 46 Metern in Gebäuden und 92m im Freien an [21]. Diese Werte sind meist jedoch sehr optimistisch. Hat man sunnyHOME im Garten stehen, sollte man mit einer Reichweite von 35-45 Metern rechnen.

Ein Ort mit guter Sonneneinstrahlung ist vorteilhaft.

## 5.2 Einstellen von Ubidots

Für das Publizieren der gesammelten Daten wird ein Ubidots-Konto benötigt (siehe: 4.2.3). Das lässt sich auf ihrer Website [www.ubidots.com](http://www.ubidots.com) machen. Ist man noch Student, kann man sich über [www.ubidots.com/education/](http://www.ubidots.com/education/) ein kostenloses „Education-Konto“ erstellen.

Ist das Konto erstellt worden, muss ein „Device<sup>1</sup>“ erstellt werden. Dann braucht es noch „Variables<sup>2</sup>“ und eine „Dashboard<sup>3</sup>“. Sind diese Schritte gemacht, kann das Device wie in Screenshot 5.2 aussehen.

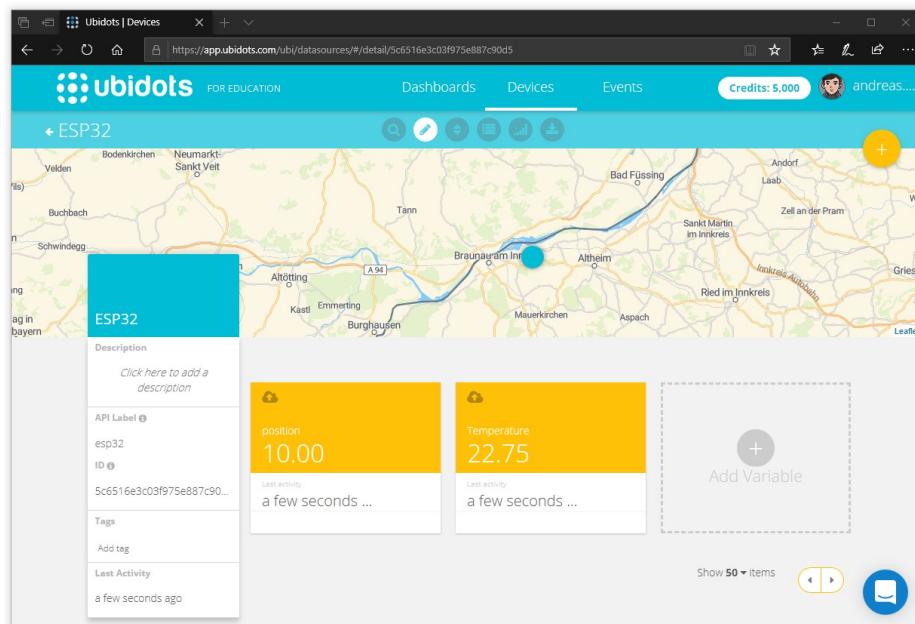


Abbildung 5.2: ubidots-Device Screenshot[18]

<sup>1</sup> Anleitung „Devices“: <https://help.ubidots.com/user-guides/creating-devices-in-ubidots>, 07.04.2019

<sup>2</sup> Was sind „Variables“: <https://help.ubidots.com/faqs-and-troubleshooting/what-are-variables>, 07.04.2019

<sup>3</sup> Anleitung zu „Dashboards“: <https://help.ubidots.com/user-guides/create-dashboards-and-widgets>, 07.04.19

## 5.3 Modifizierung des Codes

Der letzte Schritt bei der Einrichtung von sunnyHOME, ist das modifizieren des Codes.

Sobald man das Programm in einem passenden Editor (siehe: 4.2.1) bzw. in einer passende IDE (siehe: 4.2.2) geöffnet hat, sollte die Verbindung zu einem WLAN-Netzwerk hergestellt werden. Im Quellcode befinden sich zwei globale Konstanten, in welche man den Namen seines Netzwerkes und den Schlüssel des Netzwerkes einfügt. In Listing 5.1 sind diese Konstanten zu sehen.

```
1 // WLAN Konstanten
2 #define WIFISSID "meinInternet"          // hier WLAN-SSID eintragen
3 #define PASSWORD "meinGeheimesPasswort"   // hier WLAN-Passwort eintragen
4 // Ubidots Konstanten
5 #define TOKEN "meinUbidotsToken"          // hier Ubidots-Token eintragen
```

Listing 5.1: Verbindung mit WLAN-Netzwerk und Ubidots

Im nächsten Schritt gilt es die Verbindung zum Ubidots-Broker herzustellen. Wenn man sich ubidots öffnet, findet sich ein Menü-Punkt namens „API Credentials“ (siehe: 5.3). Von dort kopiert man sich den „Default token“, welcher ebenfalls ins Programm eingefügt wird (siehe: 5.1).

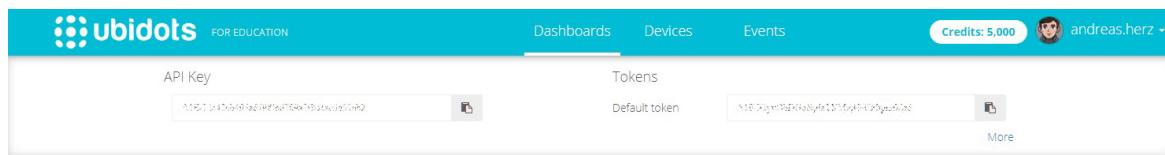


Abbildung 5.3: Ubidots-token Screenshot[18]

Jetzt fehlen nur noch die Labels. Diese sind Etiketten, welche es für jede Messung, welche über MQTT übertragen werden soll gibt. Für die Labels gibt es ebenfalls globale Konstanten (siehe: 5.2).

```
1 // Labels für MQTT
2 #define VARIABLE_LABEL_TEMP "temperature"      // Hier Temperatur-Label eintragen
3 #define VARIABLE_LABEL_GPS "gps"                // Hier GPS-Label eintragen
4 #define DEVICE_LABEL "esp32"                    // Hier Device-Label eintragen
```

Listing 5.2: MQTT-Labels

Wenn die eingetragenen Labels mit den in Ubidots eingetragenen API-Labels identisch sind, sollte die Übertragung der Daten per MQTT nun funktionieren.

# 6 Fazit und Persönliche Erfahrungen

## 6.1 Fazit

### 6.1.1 Zusammenfassung der Ergebnisse

Im Großen und Ganzen wurden die wichtigsten Themenstellungen des Projektes behandelt und fertiggestellt. Einzig die Energie-Versorgung ist nicht fertig realisiert worden.

Die Software funktioniert und der Funktionsablauf des Codes (siehe: 3.5) ist fertiggestellt. Die Verbindung zu Ubidots (siehe: 4.2.3 ist stabil und es lassen sich die gewünschten Messwerte im Internet betrachten. Der Mikrocontroller wechselt ebenfalls nach dem Erfüllen seiner Aufgaben in den Deep-Sleep Modus, um dadurch Energie zu sparen.

### 6.1.2 Mögliche Erweiterungen

Damit die Wetterstation wirklich fertiggestellt ist, muss die Energie-Versorgung durch die Solarzellen fertiggestellt werden. Dafür werden Energieverbrauchs-Messungen am ESP32 benötigt, dann kann man erst den Energy-Harvesting Modul bestimmen. Und in Abhängigkeit des gewählten Moduls, bräuchte es noch einen passenden Akkumulator. Dazu ist bisher ein Lithium-Eisenphosphat-Akkumulator geplant gewesen, da dieser eine lange Lebensdauer hat, und der Aufladeprozess effizient ist.

Ein weiterer Ansatzpunkt wäre die Realisierung einer Platine, auf der man den ESP32 ohne jeglicher Peripherie setzen könnte. Dadurch könnte man noch mehr an Energie sparen und eine längerfristigere Lösung für die Wetterstation schaffen.

Um die Wetterstation als Produkt anbieten zu können, bedarf es auch noch an einer eigenen Software. Eine mobile Applikation für Smartphone-Anwender wäre die ideale Lösung zur Visualisierung der Messdaten.

Möchte man sunnyHOME in der freien Wildnis nutzen, würde es sich anbieten, eine Funkverbindung zu einer zusätzlichen Receiver-Hardware aufzubauen, welche dann die Daten ins Internet publiziert. Ein anderer Ansatz könnte eine Verbindung mit einem mobilen Funknetz sein. Dies würde jedoch stark auf die Kosten der Energie-Effizienz gehen.

## 6.2 Persönliche Erfahrungen

„Ohne Fleiß, kein Preis“. Dieses Sprichwort hat Recht. Wenn ambitionierte Ziele gesetzt werden, muss auch ambitioniert gearbeitet werden. Ein Problem bei der Sache kann jedoch sehr schnell eine falsche Einschätzung werden. Das man zum Beispiel im letzten Jahr genug Zeit hat, um das Projekt ausführlich zu realisieren. So hatte auch ich das Problem, kommende Schwierigkeiten und volle Terminkalender nicht vorauszusehen.

Softwareentwicklung lässt sich schwer einschätzen. Es läuft einmal alles glatt, und beim nächsten Start des Programms, funktioniert es aus mysteriösen Gründen auf einmal nicht mehr. Mal findet sich der Fehler binnen einer Stunde, und mal braucht es mehrere Wochen, bis es wieder weiter gehen kann. Es waren oft Fehler, welche davon kamen, dass ich mich mit der Thematik zu wenig auskannte. In solchen Situationen muss man einfach durchhalten und ausdauernd weitersuchen. Dadurch eignet man sich die Fähigkeiten dann an, und sammelt Erfahrungen, welche einem zum Schluss dann unbezahlbar sind. Das muss ganz klar gesagt werden: Meine schönsten Momente bei diesem Projekt hatte ich immer dann, wenn der Fehler gefunden wurde und dieser spezielle „Aha“-Moment aufgetreten ist.

Schlussendlich bin ich sehr dankbar, dass ich in meinem Abschlussjahr die Möglichkeit hatte, ein Projekt wie dieses durchzuführen. Nicht nur für meine heutigen Arduino-Kenntnisse, welche meiner Meinung nach recht gut sind, sondern auch für die Erfahrungen im Bezug auf die Organisation und des Zeitmanagements eines solchen Projektes.

# A Diverse Anhänge

## A.1 Projekttagebuch

DATUM	SCHULE	FREIZEIT	TÄTIGKEIT
13.09.2018	4:30		Layout Planung.
20.09.2018	9:00		Zeitpläne.
27.09.2018	9:00		Projektsteckbrief, Testprogramm mit PlatformIO.
30.09.2018		6:00	Organisation, Testprogramm mit PlatformIO.
01.10.2018		1:00	Fehler gefunden -> Behebung
04.10.2018	9:00		GPSToDisplay Demo
07.10.2018		2:00	Logo Design, Beispiele angefertigt
11.10.2018	9:00		Umbau auf ESP32 und Testprogramm
17.10.2018		2:00	Organisatorisches
18.10.2018	9:00		GPSToDisplay Demo
05.11.2018		2:30	GPSToDisplay Demo
08.11.2018	9:00		Planung weiterer Schritte
20.11.2018		2:00	Plakat Erstellung
21.11.2018	6:00		Problembehebung: Konektivität zu Rechner
25.11.2018		2:00	Problem gelöst
29.11.2018	9:00		Deep-Sleep Demo
05.12.2018		3:00	Demoprogramm für Tag der offenen Tür

DATUM	SCHULE	FREIZEIT	TÄTIGKEIT
06.12.2018	9:00		Demoprogramm für Tag der offenen Tür
12.12.2018		1:00	Einrichtung von MS Project
13.12.2018	9:00		Einstellen der RTC, Planung von MQTT
19.12.2018		1:00	Einstellen der RTC
29.12.2018		2:00	Einstellen der RTC und Doku
10.01.2019	9:00		TimeToDisplay RTC
17.01.2019	6:00		Implementierung von RTC-Modul
20.01.2019		2:00	Fehlersuche
24.01.2019	9:00		Demo zur Energieverbrauchsmessung
30.01.2019		3:00	Raspberry Pi Setup
07.02.2019	6:00		Demo zur Energieverbrauchsmessung
10.02.2019		2:30	Demo zur Energieverbrauchsmessung
11.02.2019		4:30	Studieninformationsörse
11.02.2019		1:30	Demo zur Energieverbrauchsmessung
13.02.2019		4:00	Demo zur Energieverbrauchsmessung
14.02.2019	9:00		Demo zur Energieverbrauchsmessung
15.02.2019		5:30	HTL Informationsnachmittag
17.02.2019		1:00	Demo zur Energieverbrauchsmessung
18.02.2019		2:30	Demo und DA
19.02.2019		1:00	Organisatorisches
20.02.2019		2:00	Diplomarbeit
21.02.2019		2:00	Diplomarbeit

DATUM	SCHULE	FREIZEIT	TÄTIGKEIT
26.02.2019		0:30	Diplomarbeit
27.02.2019		2:00	Demo zur Energieverbrauchsmessung: Fehlerbehebung
28.02.2019	9:30		Demo zur Energieverbrauchsmessung: Fehlerbehebung
28.02.2019		2:30	Demo zur Energieverbrauchsmessung: Fehlerbehebung
02.03.2019		1:00	Befassung mit Solarversorgung
03.03.2019		2:00	Befassung mit Solarversorgung
05.03.2019		3:30	DA und Präsentation
07.03.2019	7:30		Fehlersuche, DA und Präsentation
07.03.2019		3:30	GANTT und Pflichtenheft
09.03.2019		2:00	Präsentation
10.03.2019		4:30	Präsentation
11.03.2019		5:00	Präsentation
12.03.2019		9:00	Jobbörse
14.03.2019	8:30		Diplomarbeit
17.03.2019		3:00	Diplomarbeit
21.03.2019	9:00		Diplomarbeit
23.03.2019		2:00	Diplomarbeit
27.03.2019		3:00	Demo zur Energieverbrauchsmessung: Fehlersuche
28.03.2019	9:00		Diplomarbeit und Fehlersuche
28.03.2019		3:00	Diplomarbeit
31.03.2019		6:30	Diplomarbeit
01.04.2019		2:00	Diplomarbeit

DATUM	SCHULE	FREIZEIT	TÄTIGKEIT
02.04.2019		3:00	Diplomarbeit
03.04.2019		3:00	Diplomarbeit
04.04.2019		4:30	Diplomarbeit
04.04.2019	9:00		Diplomarbeit
05.04.2019		6:00	Diplomarbeit
06.04.2019		3:00	Finaler Code & Diplomarbeit
07.04.2019		12:00	Diplomarbeit
08.04.2019		2:00	Diplomarbeit, letzter Schliff

Summe Schulstunden: 183:00

Summe Freizeitstunden: 144:30

# Literaturverzeichnis

- [1] Projektnachwuchs, *Projektstrukturplan und Objektstrukturplan*,  
<http://projektnachwuchs.de/projektstrukturplan-und-objektstrukturplan/>, projektnachwuchs.de, (2019)
- [2] Michael Moore, *IoT adoption is rising in the enterprise*,  
<https://betanews.com/2017/09/28/iot-adoption-rising-enterprise/>, betanews.com, (2017)
- [3] KnapEnergy, *3w 9v 195mm125mm DIY Monocrystalline Silicon Solar Panel*,  
<https://www.amazon.com/3w-9v-195mm125mm-Monocrystalline-Silicon-dp/B0165BW7SK>, amazon.com, (2019)
- [4] Designed by janno028 / Freepik, *Green Energy Imagepicture*,  
[https://www.freepik.es/foto-gratis/bombilla-arbol-adentro\\_1007989.htm](https://www.freepik.es/foto-gratis/bombilla-arbol-adentro_1007989.htm), freepik.com, (2019)
- [5] *ESP32 Datasheet*,  
<https://www.energie-lexikon.info/kohlekraftwerk.html>, energie-lexikon.info, (2019)
- [6] Espressif Systems, *ESP32 Datasheet*,  
[https://www.espressif.com/sites/default/files/documentation/esp32\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf), Espressif Systems, (2019)
- [7] Hugo Gomes, *ESP32 Solar Weather Station*,  
<https://www.hackster.io/Tiobel/esp32-solar-weather-station-bf9c23>, hackster.io, (2017)
- [8] *LM75A*,  
[https://www.xcluma.com/lm75a-temperature-sensor-development\board-module-i2c-interface-hi-q](https://www.xcluma.com/lm75a-temperature-sensor-development-board-module-i2c-interface-hi-q), xcluma.com, (2019)
- [9] *NodeMCU*,  
<https://store.fut-electronics.com/products/nodemcu-esp8266-programming-and-development-kit>, Future Electronics Egypt, (2019)
- [10] Dr. Claus Kühnel, *Erweiterung der Arduino-Familie: ESP32 – wie gut ist es?*,  
<https://www.elektroniknet.de/design-elektronik/embedded/erweiterung-der-arduino-familie-esp32-wie-gut-ist-es-160294.html>, elektroniknet.de, (2018)
- [11] *Geekcreit® ESP32 Development Board*,  
[https://www.banggood.com/ESP32-Development-Board-WiFiBluetooth\](https://www.banggood.com/ESP32-Development-Board-WiFiBluetooth)

- Ultra-Low-Power-Consumption-Dual-Cores-ESP-32-ESP-32S-Board\ -p-1109512.html?cur\_warehouse=USA, Banggood.com, (2019)
- [12] *BN-180 Micro Double GPS Antenna Module*,  
[https://www.banggood.com/Smallest-Mini-Dual-GLONASSGPS-BN-180-Micro-Double-GPS-Antenna-Module-UART-TTL-For-CC3D-F3-p-1208587.html?akmClientCountry=AT&currency=AUD&createTmp=1&utm\\_source=commissionfactory&utm\\_medium=aff&utm\\_content=31940&cfclick=52859512c3a140e4a916c1311c90d01b&cur\\_warehouse=CN](https://www.banggood.com/Smallest-Mini-Dual-GLONASSGPS-BN-180-Micro-Double-GPS-Antenna-Module-UART-TTL-For-CC3D-F3-p-1208587.html?akmClientCountry=AT&currency=AUD&createTmp=1&utm_source=commissionfactory&utm_medium=aff&utm_content=31940&cfclick=52859512c3a140e4a916c1311c90d01b&cur_warehouse=CN), Banggood.com, (2019)
- [13] *DS1307 RTC*,  
<https://hacktronics.co.in/rtc-eeprom-memory-storage/ds1307-rtc-at24c32-eeprom-i2c-real-time-clock-module-board>, Hacktronics India, (2019)
- [14] Thomas Peham *Atom: Das kann der Code-Editor von GitHub*,  
<https://usersnap.com/de/blog/atom-tipps/>, Usersnap GmbH, (2019)
- [15] *Atom Screenshot*,  
<https://atom.io/>, (2019)
- [16] *PlatformIO Screenshot*,  
<https://platformio.org/>, (2019)
- [17] Arduino, *Arduino Logo*,  
[https://de.wikipedia.org/wiki/Datei:Arduino\\_Logo.svg](https://de.wikipedia.org/wiki/Datei:Arduino_Logo.svg), (2019)
- [18] *Ubidots Website*,  
<https://ubidots.com/>, (2019)
- [19] Gerald Lechner, *ESP-32 Lora alle seriellen Schnittstellen nutzen*,  
<https://www.az-delivery.de/blogs/azdelivery-blog-fur-arduino-und-raspberry-pi/esp-32-lora-alle-seriellen-schnittstellen-nutzen?ls=de&cache=false>, AZ-Delivery, (2018)
- [20] Connectify, *The Range of a Typical Wi-Fi Network*,  
<https://www.lifewire.com/range-of-typical-wifi-network-816564>, lifewire.com, (2018)
- [21] Bradley Mitchell, *The Range of a Typical Wi-Fi Network*,  
<https://www.lifewire.com/range-of-typical-wifi-network-816564>, lifewire.com, (2018)

# Abbildungsverzeichnis

1.1	sunnyHOME-Logo . . . . .	1
1.2	Microsoft To-Do Screenshot . . . . .	2
1.3	Protokoll Screenshot . . . . .	3
1.4	Zeitplan Screenshot . . . . .	4
1.5	Objektstrukturplan Screenshot . . . . .	5
1.6	Projektstrukturplan Screenshot . . . . .	6
1.7	GitHub Screenshot . . . . .	7
1.8	Mit Microsoft Project erstelltes GANTT-Diagramm . . . . .	8
2.1	IoT Imagebild [2] . . . . .	10
2.2	Green Energy Imagebild [4] . . . . .	11
3.1	Aufbau der Wetterstation.[7] . . . . .	12
3.2	9V Solarzelle [3] . . . . .	13
3.3	Illustration zum Funktionsablauf . . . . .	14
4.1	LM75A[8] . . . . .	15
4.2	ESP8266[9] . . . . .	16
4.3	ESP32 [11] . . . . .	16
4.4	Beitian BN-180 im Vergleich zu einer Münze [12] . . . . .	17
4.5	DS1307 Real-Time Clock [13] . . . . .	17
4.6	Atom Screenshot[15] . . . . .	18
4.7	PlatformIO Screenshot[16] . . . . .	19
4.8	Ubidots-Dashboard Screenshot[18] . . . . .	20
5.1	Atom Screenshot[20] . . . . .	24
5.2	ubidots-Device Screenshot[18] . . . . .	25
5.3	Ubidots-token Screenshot[18] . . . . .	26

# Listings

4.1	HardwareSerial Initialisierung . . . . .	21
4.2	I2C Konfiguration . . . . .	21
4.3	WLAN Konfiguration . . . . .	22
4.4	Erstellen der Payload und Publikation der Temperatur . . . . .	22
4.5	Synchronisierung der GPS-Zeit mit der RTC-Zeit . . . . .	22
4.6	TinyGPS++ Funktionen . . . . .	23
4.7	„smartDelay“-Funktion . . . . .	23
4.8	Float zu String Wandlung für MQTT . . . . .	23
5.1	Verbindung mit WLAN-Netzwerk und Ubidots . . . . .	26
5.2	MQTT-Labels . . . . .	26

# Autoren

## Andreas Herz

*Geburtstag, Geburtsort:* 16.02.1998, Nürnberg, DE

*Schulbildung:* Grundschule Oberasbach-Altenberg, DE  
Hauptschule Bogenhofen, AT  
HTL Braunau, AT

*Praktika:* 2015: Hammerer Aluminium Industries GmbH  
2017: Hammerer Aluminium Industries GmbH  
2018: conova communications GmbH

*Anschrift:* Heitzenberg 6  
4963, St. Peter am Hart  
Österreich

*E-Mail:* andreas.herz@outlook.com

