# SCHÜLERFORUM 2021

INTERNET
OF STUPID
THINGS

Von **Felix Hauptmann**, **Niklas Krieger und Mirko Weih**

aus der **Ludwig-Geißler-Schule in Hanau**

in Betreuung durch

**Herr Dr. Martin Löffler und Herr Fabian Bott**



# Inhaltsverzeichnis



## 1. ABSTRACT

We want to find out what can be digitized in public buildings without having to purchase new equipment. Because of the current Covid-19 pandemic, our project specializes primarily in hygiene. This involves communicating the fill level of paper towel, soap or disinfectant dispensers to a digital system making it is easier to see when and where these dispensers need to be refilled. To do this, we are using microcomputers with various sensors. In the further course of our development, we also want to integrate other analog states such as for example open windows or status lamps. In addition, we are trying to miniaturize our product so that it is versatile and can be used anywhere. We are trying to find ways to improve battery life by making our product as efficient as possible so that it is rarely necessary to change or charge the power source.

## 2. KURZFASSUNG

Mit unserem Projekt wollen wir herausfinden, was man in öffentlichen Gebäuden alles digitalisieren kann, ohne neue Gerätschaften mit entsprechenden Schnittstellen anschaffen zu müssen. Wegen der aktuellen Covid-19-Pandemie haben wir uns dabei vor allem auf die Hygiene in Räumen spezialisiert. Hierbei geht es darum, den Füllstand von Papier- oder Seifen- bzw. Desinfektionsmittelspender an ein elektronisches System weiterzugeben, damit einfacher zu erkennen ist, wann und wo diese nachgefüllt werden müssen. Dazu verwenden wir Mikrocomputer mit verschiedenen Sensoren. Im weiteren Verlauf unserer Entwicklung wollen wir außerdem

auch noch weitere analoge Zustände wie offene Fenster oder Statuslampen in Räumen einbinden. Darüber hinaus versuchen wir unser Produkt bzw. unsere Produkte so zu verkleinern, dass sie überall verwendbar und vielseitig einsetzbar sind und deren Akkubetrieb so effizient wie möglich zu gestalten, damit nur selten ein Akkuwechsel bzw. eine Akkuladung notwendig ist.

## 3. EINLEITUNG

IoT und vernetzte Systeme sind unsere Zukunft. IoT steht als Abkürzung für „Internet of Things“, was übersetzt so viel wie „Internet der Dinge“ heißt. Unter solch einem Netzwerk versteht man die Vernetzung von Gegenständen aus unterschiedlichen Bereichen. Beispiele hierfür sind: Eine Heizung, die mit Hilfe eines intelligenten Thermostates automatisch die Raumtemperatur anpasst, oder ein intelligenter Briefkasten, der eine Statusmeldung auf das Mobiltelefon sendet, sobald er vom Briefträger geöffnet wurde. Häuser und Wohnungen, deren Geräte genau durch solch ein System vernetzt sind, nennt man neudeutsch auch „Smart Home“. Bei der Vernetzung von Haustechnik (z.B. Heizung oder Lichtquellen) und Haushaltsgeräten (bspw. Kühlschrank oder Waschmaschine) sowie von Komponenten der Unterhaltungselektronik strebt man eine Erhöhung der Wohn- und Lebensqualität, der Sicherheit und einer effizienten Energienutzung an.


*Abbildung 1: Anwendungsgebiete von IoT-Systemen im Smart-Home*

### 3.1 Möglichkeiten, IoT-Systeme aufzubauen

Heute gibt es eine Vielzahl von Herstellern, die IoT-Systeme anbieten. Diese Systeme dienen unter anderem zum Schalten von Lampen, Rollläden und Heizungen oder zum Protokollieren und Visualisieren von Ereignissen wie der Temperatur, der Feuchtigkeit und der Stromverbrauch. Hersteller sind zum Beispiel: AVM, Bosch, eQ3, Phillips, Rademacher oder Siemens. Die meisten oben genannten Systeme funktionieren ohne Kabel und lassen sich einfach in bestehende Hausinstallationen integrieren. Des Weiteren gibt es auch leitungsgebundene Systeme (z.B. KNX oder Homematic IP Wired), die allerdings bereits bei der Hausplanung berücksichtigt werden sollten.

### 3.2 Herstellerspezifische IoT-Systeme

Jeder Hersteller nutzt für seine Systeme eigene Protokolle, welche oftmals nicht herstellerübergreifend unterstützt werden. Diese Protokolle bestimmen die Regeln für den Austausch von Daten zwischen den Komponenten der IoT-Lösung und laufen über verschiedenste Schnittstellen. Einige Hersteller setzen bspw. auf das ZigBee-Light-Link-Protokoll. Es belastet das WiFi-Netz nicht mit Datendurchsatz und hat einen geringeren Stromverbrauch als eine WiFi-Anbindung. Im Vergleich zu Kurzstreckenfunktechnologien wie Bluetooth oder ANT ist der Stromverbrauch noch recht hoch. Ein ZigBee-Netzwerk nutzt das 2,4-GHz-Frequenzband. Der in Deutschland weitverbreitete Hersteller eQ3 nutzt für seine Systeme 868,3 MHz als Funk-Frequenz. AVM verwendet für seine Smart-Home-Lösung den DECT-Standard. Der Vorteil einer homogenen Herstellerlösung ist, dass die Integration der herstellerspezifischen Aktoren und Sensoren sowie die Programmierung von Steuerungen recht einfach umzusetzen ist. Einige dieser Lösungen nutzen Cloud-Server, auf denen die Programme ablaufen. Hierbei ist jedoch zu beachten, dass es sich um seriöse Betreiber handeln sollte, die mit den Daten vertrauenswürdig und sicher umgehen.

### 3.3 Herstellerübergreifende IoT-Systeme

Um die Systeme verschiedenster Hersteller kombinieren zu können, gibt es Heimautomatisierungssysteme, die die verschiedenen Protokolle unterstützen und somit herstellerübergreifend die Steuerung von Aktoren und Sensoren ermöglichen. Hierzu gehört beispielsweise FHEM (Akronym für „Freundliche Hausautomation und Energie-Messung“). Dabei handelt es sich um ein Programm, welches zum Beispiel auf einem Raspberry Pi läuft und häufig auftretende Aufgaben herstellerübergreifend automatisiert.

### 3.4 IoT-Sicherheit

So toll und komfortabel eine Hausautomation auch ist, birgt diese leider auch Sicherheitslücken, welche Hackern den Zugriff auf das Smart-Home-System ermöglicht. Das Bundesamt für Sicherheit in der Informationstechnologie (BSI) warnt vor Sicherheitslücken in vielen vernetzten Geräte wie IP-Kameras, smarten Türschlössern oder smarten Stromzählern. Auch Umgebungssensoren, zum Beispiel zur Erfassung von Temperatur und Luftfeuchtigkeit, sowie Lichtsysteme, Netzwerkdrucker und vernetzte Audiosysteme sind gefährdet. Angreifer können über Sicherheitslücken in die Netzwerke eindringen, Daten stehlen und

Computersysteme außer Gefecht setzen oder in ihrem Sinne fernsteuern. Bei einem smarten Türöffner beispielsweise bedeutet das: Der Angreifer kann sich Zugang zu der damit gesicherten Wohnung verschaffen. Dies ist eine große Herausforderung für Hersteller und Nutzer, welche es zu bewältigen gilt.

### 3.5 Praktische IoT-Anwendung

Als Anlass für dieses große Thema haben wir die aktuelle Covid-19-Pandemie genommen. Weltweit kämpft die Bevölkerung gegen das neuartige Coronavirus an. Um die Pandemie eindämmen zu können, sind vor allem die AHA-Regeln (Abbildung 2) für uns alle von großer Bedeutung. Ein gründliches Waschen und Desinfizieren der Hände sind nötig. Damit dies sichergestellt ist, muss überall eine optimale Versorgung mit Seife, Papiertüchern und Desinfektionsmittel gewährleistet sein. Da das leider nicht immer der Fall ist, wollen wir daran forschen, wie eine solche Versorgung und auch anderweitig die Hygiene in Räumen sichergestellt werden kann. Mit Hilfe eines Mikrocomputers versuchen wir, vorhandenes Equipment in öffentlichen Gebäuden zu digitalisieren, sodass keine neuen Gerätschaften mit entsprechenden Schnittstellen angeschafft werden müssen. Ohne alle Räume ständig überprüfen zu müssen, soll die Versorgung gesichert werden, was auch eine große Zeitersparnis darstellt. Hausmeister werden vom System automatisch benachrichtigt, ohne dass sie angerufen oder aufgesucht werden müssen. Die Meldung kann somit nicht verloren gehen und die Arbeitskraft wird effizienter. Da wir uns jedoch zuerst nur einem realisierbaren Ziel widmen wollten, entschieden wir uns dafür, das Problem mit den fehlenden Papiertüchern beim Abtrocknen zu lösen und Papierspender (Abbildung 3) zu digitalisieren. Ein richtiges Abtrocknen der Hände ist nebenbei genauso wichtig wie das Waschen: Durch nasse Hände können Keime leichter übertragen werden, wodurch die Wahrscheinlichkeit, die Umgebung zu kontaminieren viel höher ist. Laut der Bundeszentrale für gesundheitliche Aufklärung (BZgA) können sich Mikroorganismen in einer feuchten Umgebung besser halten und vermehren. Aus diesem Grund rät die BZgA, die Hände direkt nach dem Waschen gründlich abzutrocknen. Darüber hinaus streift man beim Trocknen sämtliche Keime ab, die noch an den Händen haften. Wir besuchen auf der Ludwig-Geißler-Schule im beruflichen Gymnasium die Bereiche Praktische Informatik und Elektrotechnik, weshalb dieses Projekt genau zu unseren Interessen passt. Dadurch können wir unsere bis jetzt erworbenen Fähigkeiten verwenden und uns in unserer Fachrichtung weiterbilden.


*Abbildung 2: AHA-Regeln*

*Abbildung 3: Handelsüblicher Papiertuchspender*

## 4. ENTWICKLUNG

Zu Beginn bestand unsere Vorgehensweise darin, verschiedene Ideen und Fragestellungen zu sammeln. Bis wir endlich unser Thema gefunden hatten, dauerte es einige Zeit. Schließlich hat uns aber die Idee gut gefallen, etwas im Bereich „Internet Of Things“ beziehungsweise „Internet Of Stupid Things“ zu entwickeln. Zunächst überlegten wir, welche Materialien wir benötigen, und informierten uns über diese.

### 4.1 Hardware

Unser Gerät sollte möglichst klein und kompakt, damit es zum Beispiel gut in einem Papierspender platzierbar ist, günstig, damit man viele Einheiten davon einsetzen kann, und energieeffizient sein, damit der Hausmeister nicht jeden Tag die Batterien wechseln muss. Also entschieden wir uns, einen Mikrocomputer bzw. Mikrocontroller (Abbildung 4) zu verwenden. Wir haben kein Produkt eines gängigen Smart-Home-Lösungen-Hersteller gefunden, was diese Kriterien erfüllt, zumal wir auch eine lokale Lösung bevorzugt haben, die nicht jedes Ereignis beispielsweise auf einem Server in China sichert. Darüber hinaus merkten wir schnell, dass unsere Geräte nur eine simple Aufgabe haben, wofür wir nicht ein vollständiges IoT-System eines Herstellers benötigen. Deshalb kam für uns nur ein Einplatinencomputer in Frage. Wir entschieden uns schließlich für ein besonders kompaktes Modell des Herstellers Espressif namens „ESP 8266 Plus“, welches im Vergleich zu den Konkurrenten der Raspberry Pi Foundation oder des Arduino-Projekts am besten unsere Erwartungen erfüllt. Von Haus aus ist eine WLAN-Schnittstelle vorhanden, die Kosten sind, besonders bei vielen Einheiten, sehr gering (siehe 4.1.5), der Stromverbrauch kann durch geschickte Programmierung auf bis zu 25µA

*Abbildung 4: Mikrocomputer*

minimiert werden und als Entwicklungsumgebung kommt auch die Arduino IDE in Frage, mit der wir aus dem Unterricht schon Vorerfahrung sammeln durften.

### 4.1.1 Schnittstellen

Es bestand nun die Möglichkeit, unsere Geräte und unser System über kabelgebundene oder kabellose Schnittstellen kommunizieren zu lassen. Obwohl wir in unserer Schule in nahezu jedem Raum einen Ethernet-Anschluss haben, mit dem wir unseren Mikrocomputer ins Netzwerk bringen könnten, schlossen wir eine derartige Verkabelung in jedem Raum aus, da sie zu zeitintensiv gewesen wäre und man die vorhandene Infrastruktur insoweit verändern hätte müssen, dass man Löcher, zum Beispiel am Papierspender, für die Kabel hätte bohren müssen. Durch dieselben Gründe und der Tatsache, dass man in der ganzen Schule hätte Kabel verlegen müssen, um die Geräte in den Räumen zu verbinden, haben wir auch die kostengünstige, serielle Schnittstelle RS232 (Recommended Standard 232) und das zu umfangreiche Bussystem USB (Universal Serial Bus) ausgeschlossen. Zusätzlich hätte mit den erforderlichen Schnittstellen-Steckern ein Mikrocomputer ausgesucht werden müssen, der deutlich an Kompaktheit verloren hätte. Als kabellose Schnittstelle haben wir uns dann gegen ZigBee entschieden, weil wir keine zusätzliche Hardware wie einen ZigBee-Router kaufen wollten, und nahmen die höhere Energieaufnahme von WLAN in Kauf, da wir dafür schon alle benötigten Geräte in der Schule vorfinden können. Dieses Privileg haben die Schüler nun schon seit beinahe fünf Jahren auf der Ludwig-Geißler-Schule. Die IT-Abteilung hat ein äußerst performantes WLAN-Netzwerk aufgebaut, welches den Schülern im Schüler-WLAN sogar in der Sporthalle eine kostenfreie Internetanbindung zur Verfügung stellt. Die Schule setzt hierbei auf ca. 50 WLAN-Access-Points der Firma Ubiquiti, also auf „Zugriffspunkte“, die im Gegensatz zu WLAN-Repeatern (Verstärkern) ihr eigenes WLAN aufbauen, somit nicht nur das vom Router verstärkt und dabei mit ebendiesem per LAN verbunden sind. Bluetooth haben wir darüber hinaus als Schnittstelle ausgeschlossen, da wir wollten, dass man die Statusmeldungen von allen Geräten zu jeder Zeit mit unserem System einsehen kann und nicht nur von denen in der direkten Umgebung. Da das Schulnetzwerk auch von außen erreichbar ist, kann der Hausmeister nebenbei, egal wo er auch ist, sich die Füllstände anzeigen lassen.

*Abbildung 5: WLAN-Access-Point aus dem Schulgebäude*

### 4.1.2 Sensorsuche

Im Hinblick auf unser gesetztes Ziel, den Füllstand eines Papiertuchspenders zu messen, gingen wir nun auf die Suche nach einem Sensor, der genau für diese Anwendung geeignet ist. Dazu hatten wir das „Sensorkit X40“ des Herstellers Joy-IT zur Hand. Nach kurzer Recherche ergab sich schnell, dass für unsere Anwendung vor allem der Infrarot-Abstandsensor (KY-032) und der Ultraschall-Abstandssensor (KY-050) in Frage kommt. Nach ausgiebigen Testläufen erkannten wir, dass der Infrarot-Abstandsensor für die Füllstandmessung eines Papierspenders eher ungeeignet ist, da wir gerne den genauen Füllstand in Prozent durch Messen der Entfernung zwischen Spenderdecke und den Papiertüchern ermitteln wollten. Natürlich hätte man auch den Infrarot-Abstandsensor seitlich in einer bestimmten Höhe befestigen können und sobald diese unterschritten ist, würde der Papierspender als leer angezeigt. Unsere Vision, die ermittelten Daten auch auszuwerten und später beispielsweise durch KI bevorstehende Spenderauffüllungen hervorzusagen, im Kopf habend, entschieden wir uns für den Ultraschall-Abstandsensor (Abbildung 6). Natürlich wussten wir, dass er ein wenig teurer ist (siehe 4.1.5), aber bei hohen Stückzahlen und im Hinblick auf die dadurch ermöglichten Funktionen für unser System ist dies durchaus tragbar und sinnvoll. Unsere Testläufe mit dem HC-SR04 (Herstellerbezeichnung des Ultraschall-Abstandsensors) waren allesamt erfolgreich und führten zu einer zuverlässigen Entfernungsermittlung, auch in der Dunkelheit eines geschlossenen Papiertuchspenders.

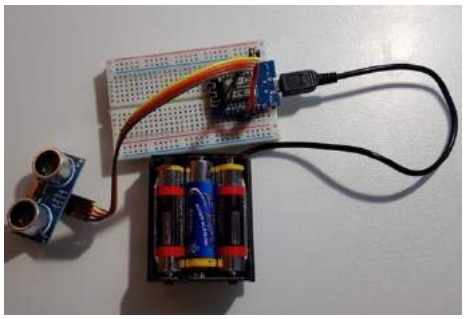

*Abbildung 6: HC-SR04 angeschlossen an durch Akkus versorgten ESP*

### 4.1.3 Spannungsversorgung & Gehäuse

Da wir auch schon ein Kabel für unsere Netzwerkverbindung ausgeschlossen hatten, waren wir uns natürlich einig, dass wir auch hier auf eine kabellose Lösung in Form einer handelsüblichen Batterie bzw. eines Akkumulators (Akku) setzen. Akkus sind besser für die Umwelt, da sie wiederverwertbar sind. Sie kosten zwar grundsätzlich einzeln in der Anschaffung mehr - durch die mögliche Wiederaufladung kann man sie aber viel länger nutzen. Natürlich muss man dann aber auch ein Ladegerät anschaffen und die Zeit aufbringen, die Akkus

damit immer wieder aufzuladen. In unseren Tests haben wir dies so gemacht, aber ob der Hausmeister dies in der Praxis machen wird, ist ungewiss. Unser ESP8266 benötigt eine Eingangsspannung von 3,3V bis 5V. Mit drei Batterien bzw. Akkus kommen wir zusammengerechnet auf ca. 4,5V. Dementsprechend schnitten wir ein microUSB Kabel zurecht und löteten es an eine alte Batteriehaltung (Abbildung 6), die dann mit vollen Batterien oder Akkus genug Spannung und Strom für unseren Mikrocomputer lieferte. Damit unsere Gerätschaften so energieeffizient wie möglich arbeiten, wacht der ESP8266 immer nur kurz auf und geht dann für lange Zeit in den sogenannten Deep Sleep (dt. „Tiefschlaf"), durch welchen wir extrem viel Strom sparen und dem Hausmeister einen seltenen Batterie- oder Akkuwechsel ermöglichen. Um unser inzwischen als „TellMe" getauftes Produkt so klein wie möglich zu halten und es gut im Papiertuchspender befestigen zu können, haben wir für das Konstrukt aus Sensor, Einplatinencomputer und Batterien bzw. Akkumulatoren nun in ein Gehäuse verlagert. Zunächst fertigten wir einen Prototyp mit einer Schachtel aus Pappe und verbesserten diesen einige Zeit später durch ein etwas langlebigeres Gehäuse (Abbildung 7), welches wir mit dem 3D-Drucker unserer Schule erstellen konnten. Das Gehäuse wird dann im Papierspender oben an der Decke im rechten Winkel zur Wand befestigt, damit der Ultraschalsensor senkrecht in Richtung der Papiertücher den Abstand messen kann.

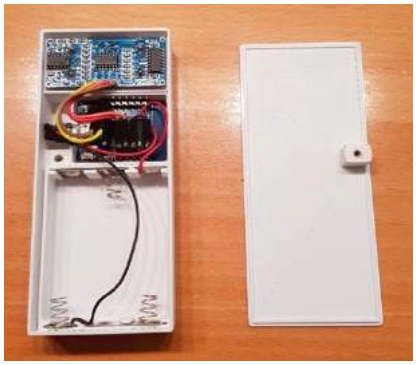

*Abbildung 7: durch 3D-Drucker erstelltes Gehäuse*

### 4.1.4 Schaltplan

Zur Veranschaulichung der Verbindung zwischen Sensor und Mikrocomputer sowie generell nötige Kabelführungen haben wir einen Schaltplan erstellt.

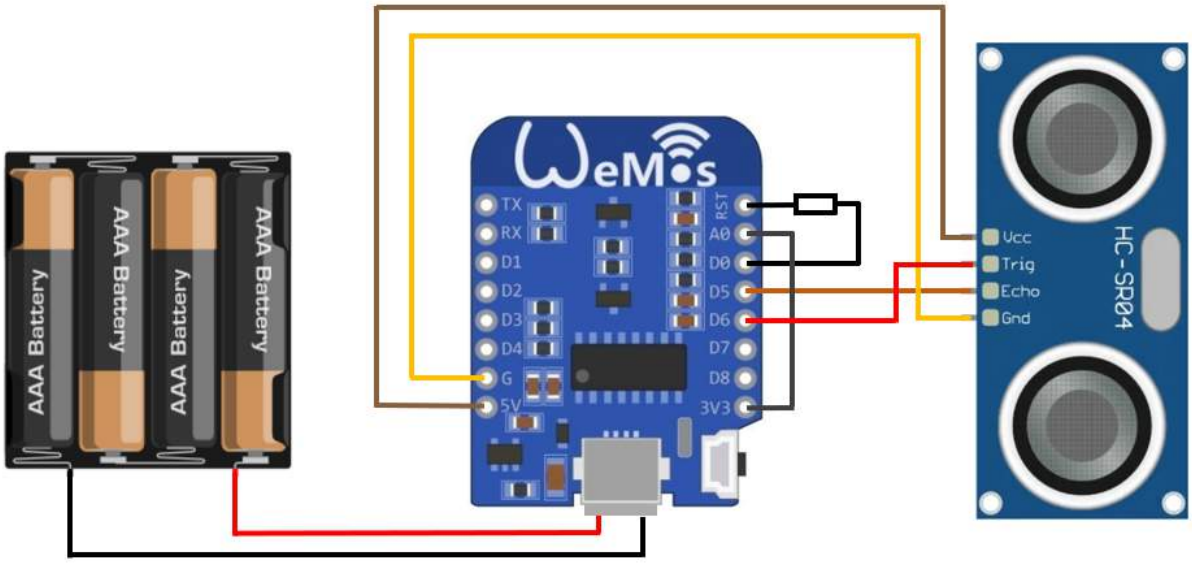


*Abbildung 8: Schaltplan eines TellMes*

Man kann dort die drei Hauptelemente, den ESP8266 auf dem von uns verwendeten Wemos D1 mini Board, die befüllte Batteriehaltung und den Ultraschallabstandsensor HC-SR04, erkennen. Zusätzlich nutzen wir natürlich auch noch mehrere Jumperkabel, um diese Elemente miteinander zu verbinden. Erwähnenswert sind darüber hinaus noch der Widerstand zwischen RST und D0 sowie die Verbindung zwischen 3V3 und A0. Der Widerstand stellt eine Verbindung zwischen RST und D0 her. Diese wird benötigt für den oben beschriebenen Deep Sleep, durch welchen der Mikrocomputer eine gewisse Zeitspanne lang nichts tut und dann einen Impuls auf A0 ausgibt, welcher dann zu einem Reset und somit einem Start bzw. Aufwachen des ESPs führt. Grundsätzlich ist für diese Verbindung ein normales Kabel ausreichend, doch uns fiel auf, dass der Mikrochip beim Start ungewollt einen kleinen Impuls auf A0 gibt, wodurch dann immer wieder ein Reset eingeleitet wurde, was zu massiven Problemen beim Hochladen von Maschinencode führte. Durch den Widerstand konnten wir dieses Problem eliminieren. Die Kabel-Verbindung zwischen 3V3 und A0 wird benötigt, um die Eingangsspannung am Mikrocomputer zu ermitteln. Diese wird benötigt, um eine Aussage über den Füllstand der Batterien oder Akkus zu treffen.

### 4.1.5 Kosten

Ein ESP8266 auf einem Wemos D1 mini Board kostet 2,45€. Der Ultraschallabstandsensor ist mit einem Preis von 1,49€ im Vergleich zum Infrarotabstandsensor, welcher ab ungefähr einem Euro zu haben ist, nicht viel teurer und überzeugt mit einem deutlich größeren Funktionsumfang. Wir verbrauchen acht Jumperkabel, welche zusammen knapp 56ct kosten. Ein 750 Ohm Widerstand kostet 10ct. Unser Gehäuse wiegt 35g - dessen Druckfilament kostet dementsprechend 34ct. Zusammengerechnet kommen wir demensprechend auf ungefähr 6€. Dabei wurden keine Versand- und Stromkosten berücksichtigt. Die drei benötigten AA-Batterien kosten rund 87ct. Drei AA-Akkus kosten rund 3,38€. Zusätzlich wird natürlich auch noch ein Ladegerät gebraucht. Beide Anschaffungen können sich aber auf längere Sicht lohnen. Es wurden nur Einzelpreise genommen, weshalb bei einer größeren Stückzahl durch den Mengenrabatt der Preis noch um einiges verringert werden kann. Darüber hinaus gibt es noch die Möglichkeit alle Komponenten einzeln zu kaufen und unseren TellMe auf einem eigenen Board selbst zu bauen. Bei geringen Stückzahlen ist dies aber auf jeden Fall nicht lohnenswert, da wir dabei auf den doppelten Wert kamen.

## 4.2 SOFTWARE

Da wir uns gegen die verschiedenen IoT-Lösungen namhafter Hersteller entschieden haben, war uns klar, dass durch die Nutzung eines flexiblen und offenen ESP8266 einiges an Programmierung notwendig sein wird. Dementsprechend wurde die meiste Zeit in die Entwicklung einer Software gesteckt, dem TellMe-System.
Unser System besteht aus mehreren Komponenten: der Firmware des Mikrocomputers, der REST-API, der Datenbank und dem Webinterface.

### 4.2.1 Firmware des Mikrocomputers

Die Firmware des Mikrocomputers, den Client, haben wir mit der Arduino IDE in C++ geschrieben. Ziel hierbei war es, nur die zwingend notwendigen Funktionen auf dem ESP auszuführen, um eine so kurze Rechenzeit zu haben wie nur möglich. Beim ersten Start des unkonfigurierten Einplatinencomputers wird temporär ein WLAN-Hot-Spot erzeugt, über welchen man im Captive Portal die SSID und das Passwort, also die Anmeldedaten des örtlichen WLAN-Netzwerkes, sowie die IP-Adresse und Port der REST-Server eingeben muss. Ist dies geschafft, schickt der Mikrochip die erste Anfrage an die REST-API und bittet um eine Identifikationsnummer. Nach dem Erhalt dieser werden all diese Parameter im Flash-Speicher, also dem nicht flüchtigen Speicher, des ESPs gespeichert, damit sie auch nach einem Reset noch verfügbar sind. Beim nächsten Start werden alle Parameter wieder aus dem Flash-Speicher gelesen und anschließend eine Verbindung zum Netzwerk hergestellt. Ist dies erfolgreich, schickt der Mikrocomputer der REST-API seine Identifikationsnummer und bittet um seine Konfiguration. Diese enthält die Dauer des Tiefschlafs, ob der TellMe sich resetten soll und den Abstand zwischen sowie die Menge an einzelnen Messungen mithilfe derer das jeweilige Messergebnis berechnet wird. Die letzten beiden Parameter werden für die Messungen benötigt, die im Anschluss erstellt werden. Um die Fehlerquote unserer Messung zu verringern, machen wir mehrere Messungen in einem gewissen Abstand (gute Werte erhielten wird bei sieben Messungen in einem Abstand von 250ms) und bilden von diesen dann den Mittelwert, welcher anschließend in einem Report, zusammen mit der zuvor gemessenen Eingangsspannung und der Identifikationsnummer an die REST-API weitergeben wird. Danach geht der Einplatinencomputer so lange in den Tiefschlaf, wie es beim Laden der Konfiguration durch die REST API vorgegeben wurde.


*Abbildung 9: Arduino IDE*

### 4.2.2 REST-API

Eine REST-API (Representational State Transfer – Application Programming Interface) ist eine Programmierschnittstelle, die den Austausch von Informationen zwischen unterschiedlichen Systemen ermöglicht. Der Client kann dabei über HTTP-Requests Informationen vom Server anfordern oder solche an den

Server schicken (Abbildung 10). Wir nutzen das kompakte und einfach lesbare Datenformat JSON (JavaScript Object Notation), mit den mehrere Parametern auf einmal unkompliziert übermittelt werden können. Der Mikrocomputer schickt also immer POST-Requests bzw. GET-Requests zum Senden bzw. Anfordern von Informationen an die REST-API. Der REST-Server interpretiert diese – wir nutzen dazu Java – und speichert sowie liest sie in bzw. aus der Datenbank. Zusätzlich berechnet der REST-Server beispielsweise die Länge des Tiefschlafs um Funktionen wie eine Zeitsteuerung unserer TellMes zu ermöglichen.

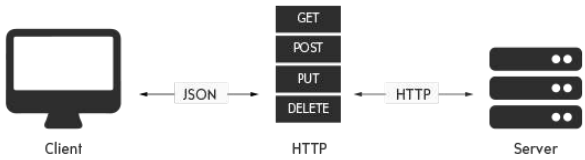


*Abbildung 10: Schema einer REST-API*

### 4.2.3 Datenbank

In der Datenbank speichern wir die über die REST-API übergebenen Daten. Darüber hinaus werden dort natürlich aber auch die verschiedenen Einstellungen für die jeweiligen TellMes gespeichert, welche der Administrator im Webinterface eingestellt hat. Wir nutzen hierfür die dokumentenorientiere Datenbank MongoDB (Abbildung 11).

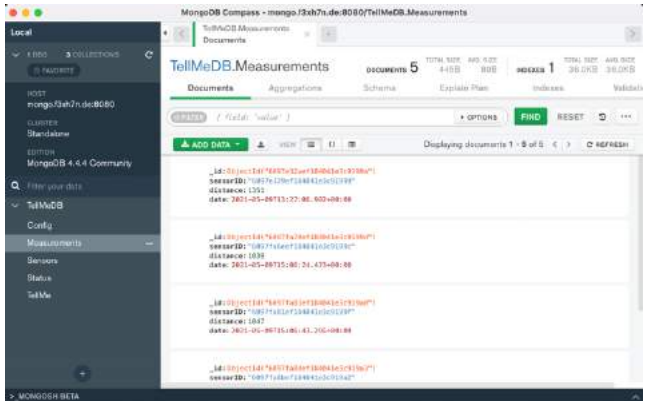


*Abbildung 11: MongoDB Compass (MongoDB GUI)*

### 4.2.4 Webinterface

In unserem Webinterface können wir aktuell nur die übertragenen Informationen, also die Messungen und die jeweilige Eingangsspannung anzeigen und interpretieren, da wir uns in unserer Entwicklung vorerst vor allem auf eine sehr gut funktionierende und generell funktionelle REST-API sowie dessen Datenbankanbindung und eine nahezu fertige Version der Firmware für den Mikrocomputer fokussiert haben.

## 4.3 Ausblick

Da man leider nie genug Zeit hat, um all seine Ideen in die Tat umzusetzen, möchten wir im Folgenden einen Ausblick auf unsere weiteren Vorhaben und Ziele für unser Projekt geben.

### 4.3.1 Weiterentwicklung des Webinterfaces

Unser Webinterface soll visuell an den Raumplan des jeweiligen (öffentlichen) Gebäudes angelehnt sein. Jedem Standort bzw. Raum des Gebäudes kann man dann einen oder auch mehrere TellMes für jeweilige Aufgaben zuweisen. Man soll die Sensoren verwalten können, indem man sie dort konfigurieren und beispielsweise den Messintervall, also in welchem zeitlichen Abstand die Messungen erfolgen, und Inaktivitätszeiten (Nacht, Wochenende usw.) eintragen kann. Darüber hinaus werden die Messdaten interpretiert und visuell dargestellt, um geeignete Statistiken erstellen zu können. Zusätzlich sollen die Messdaten auch (durch ein neuronales Netz) ausgewertet werden, um zum Beispiel die Auffüllung eines Papiertuchspenders zu planen bzw. vorherzusagen. Es soll dem Nutzer außerdem durch Betätigen eines Tasters möglich sein, den entsprechenden TellMe im Webinterface zu identifizieren.

### 4.3.2 Sicherheit des Systems

Aktuell identifizieren sich die TellMes nur durch ihre Identifikationsnummer, mithilfe welcher ihnen Messdaten und ihre Konfiguration zugeordnet werden können. Zum Zweck der aktuell nicht vorhandenen Sicherheit möchten wir die Übertragung durch eine Authentifizierung erweitern. Außerdem soll auch der generelle Datenaustausch mit der REST-API per HTTPS verschlüsselt ablaufen.

### 4.3.3 Akkukonzept

Im Hinblick auf die Klimaneutralität wollen wir es schaffen, dass wir nur Akkumulatoren, am besten Akku-Packs, für die Stromversorgung nutzen. Es ist noch ein wenig ungewiss, wie ein Akkuwechsel ablaufen soll. Aktuell muss der Hausmeister einzelne AA-Batterien bzw. -Akkus mit sich tragen, den Papiertuchspender öffnen und unser Gehäuse aufschrauben. Ein externer Akku wäre auf jeden Fall deutlich einfacher zu wechseln.

### 4.3.4 Eigenes Board

Wie schon in der Kostenaufstellung (4.1.5) angedeutet, gäbe es die Möglichkeit anstelle des Wemos D1 mini ein eigenes PCB (Printed Circuit Board) zu entwickeln und alle Einzelteile dafür zu kaufen sowie zusammenbauen zu lassen. Dies würde unseren TellMe auf jeden Fall noch energieeffizienter und kompakter machen, da wir nicht

genutzte Funktionen nicht einbauen müssen. Außerdem könnten wir so bei hohen Stückzahlen geringere Produktionskosten erzielen.

### 4.3.5 Andere Sensormodule für andere Zwecke

*Abbildung 12: Anwendungsentwurf*

Was wir schon einige Male haben anklingen lassen und was man sicher auch schnell bemerken wird, ist, dass man unser System auch mithilfe anderer Sensoren auf andere Einsatzzwecken anwenden könnte.
Konkret schweben uns folgende Projekte für die Zukunft vor: Wir könnten einen TellMe entwickeln, der an Fensterrahmen befestigt wird, um den Zustand der entsprechenden Fenster zu übermitteln, sodass offene Fenster, beispielsweise über das Wochenende, der Vergangenheit angehören. Ein weiteres Produkt wäre ein TellMe, der Statuslampen in Räumen in das System einbindet. Zum Beispiel könnten so die Lampen, die in den Fachräumen für Biologie, Physik oder Chemie den Zustand der Gaszufuhr, bzw. der Stromversorgung signalisieren, digitalisiert werden. Darüber hinaus wäre in den aktuellen Zeiten natürlich auch ein TellMe sehr sinnvoll, der den Füllstand von Seifen- und Desinfektionsmittelspendern misst. Dies ist aber im Vergleich zur Digitalisierung von Papiertuchspendern um einiges schwerer, da man unser Gerät nicht einfach in den Innenraum verbauen kann, da dieser zu klein ist und dort natürlich auch nicht die besten Bedingungen für Elektronik herrschen. Dementsprechend müsste man den Füllstand von außen, durch die Gehäusewand des Spenders hindurch, messen. Eine äußere und somit sichtbare Anbringung erhöht jedoch das Risiko eines Diebstahls oder einer Beschädigung von außen. In Abbildung 12 sieht man, dass durch die Lage des Sensors entschieden werden kann, wie früh der TellMe dem Hausmeister einen niedrigen Füllstand mitteilt. An der ersten Position erfolgt eine frühe und an der zweiten Position eine späte Mitteilung.

## 5. SUMMARY

After we had decided on the feasible goal for us, an automatic level measuring system for paper towel dispensers (Fig. 3), we developed our product called "TellMe". On the hardware side, we decided to use an ESP8266 (Fig. 4), which we selected because of its low power consumption and low cost. The distance between the top of the paper towel dispenser and the paper towels gets measured with the help of an HC-SR04 ultrasonic sensor (Fig. 6). It enables us to determine the exact level of paper towels accurate just a few milimeters. We use WLAN to transmit the data. The power is supplied by batteries or accumulators, which only need to be changed very rarely, since the microcomputer only wakes up briefly for its measurement and is otherwise always in so-called Deep Sleep. This saves an extremely large amount of power. Our TellMe with its 3D-printed housing (Fig. 7) costs only about 7€. With higher quantities and a custom designed PCB, these costs could be reduced even further. The software consists of the firmware of the microcomputer, the REST-API (Fig. 9), the database and the web interface. Out of several distance measurements the most accurate one gets selected by the microcomputer and, along with the current input voltage, is then sent to the database via the REST-API. We use the object-oriented database system MongoDB as our database. The database stores the configuration values of various TellMes and the data passed from the REST-API. Our web interface currently only displays and interprets the measurements and respective input voltages. In the future we want to develop this further and use a building plan to visually align each sensor to its physical location within the building. Furthermore, this web interface is going to enable the user to manage and configure each TellMe. The measurement data will be interpreted and visually displayed and could even be evaluated by neural networks to allow planning further ahead by predicting when a paper towel dispenser is going to be empty. In the future, the data exchange shall be secured by means of authentication. Apart from that we are going to switch over from HTTP to encrypted HTTPS. In addition, we want to use the TellMe for other purposes by equipping it with other sensors.

## 6. QUELLENVERZEICHNIS

https://www.infektionsschutz.de/haendewaschen/ - *abgerufen 02.05.21*

https://www.freundin.de/gesundheit-haende-richtig-abtrocknen-so-gehts - *abgerufen 02.05.21*

https://www.telekom.de/smarte-produkte/iot - *abgerufen 03.05.21*

https://fhem.de/fhem_DE.html - *abgerufen 03.05.21*

https://www.elektronikpraxis.vogel.de/von-ant-bis-zigbee-die-wichtigsten-funkstandards-im-smart-home-a-796103/ - *abgerufen 04.05.21*

https://www.computerwoche.de/a/sicherheitsluecken-bedrohen-millionen-iot-geraete,3550302 - *abgerufen 04.05.21*

https://www.giga.de/extra/android-spezials/specials/apn-access-point-name-was-ist-das/ - *abgerufen 04.05.21*

https://www.mikrocontroller.net/articles/ESP8266 - *abgerufen 04.05.21*

https://sensorkit.en.joy-it.net/index.php?title=Main_Page - *abgerufen 05.05.21*

https://www.heinzinger.de/glossar/akkumulator/ - *abgerufen 05.05.21*

https://de.freepik.com/vektoren-premium/smart-home-iot-internet-der-dinge-steuern-komfort-und-sicherheit-isometrische-infografik-poster-abstrakt_12089696.htm - *abgerufen 07.05.21*

https://www.bundesgesundheitsministerium.de/fileadmin/Dateien/5_Publikationen/Gesundheit/Flyer_Poster_etc/Corona/AHA-Formel_Blau-Rot_barr_V2.pdf - *abgerufen 08.05.21*

https://static.praxisdienst.com/out/pictures/generated/product/1/800_800_90/marplast_handtuchspender_134940_1.jpg - *abgerufen 08.05.21*

https://media.nbb-cdn.de/images/products/250000/257732/Ubiquiti_UAP-AC-PRO_Front.jpg?size=2800 - *abgerufen 09.05.21*

https://www.vario-software.de/wp-content/uploads/2019/10/rest-api-min.png - *abgerufen 09.05.21*

https://de.ryte.com/wiki/REST-API - *abgerufen 09.05.21*

https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266HTTPClient - *abgerufen 10.05.21*

https://github.com/xoseperez/eeprom_rotate - *abgerufen 10.05.21*

https://arduinojson.org/v6/doc/ - *abgerufen 10.05.21*

https://github.com/bblanchon/ArduinoJson - *abgerufen 10.05.21*