

Wilhelm-Ostwald-Schule, Gymnasium der Stadt Leipzig

Dokumentation zur Besonderen Lernleistung

Im Fachbereich: Informatik

Thema:	Programmierung und Bau eines Statuswürfels zur Steuerung eines E-Ink-Displays
Vorgelegt von:	Leopold Peer Hofmann
Schuljahr:	2024/2025
Externer Betreuer:	Philipp Dockhorn Hochschule Merseburg Fachbereich Ingenieur- und Naturwissenschaften LfbA Mikroprozessortechnik/Embedded Systems
Interner Betreuer:	Herr Simon Koch

Kurzreferat

Diese BeLL des Fachgebiets Informatik befasst sich mit dem Bau und der Programmierung eines Statuswürfels. Der Statuswürfel soll ein Tool zur Produktivitätsverbesserung sein. So kann der Nutzer durch einfache Interaktion mit dem Würfel beispielsweise seinen Beschäftigungsstatus ändern, einen Timer stellen oder Smart-Home-Steuerung vornehmen. Im Wesentlichen handelt es sich um eine einfache, haptische Fernbedienung.

Diese Arbeit bezieht sich zunächst auf die Änderung des Beschäftigungsstatus. Dafür werden drei Komponenten entwickelt: der Würfel, ein Server mit REST-API und ein E-Ink-Display.

Der Würfel ermittelt seine Ausrichtung mittels eines ESP32-Microcontrollers, welcher mit einem Beschleunigungssensor und Gyroskop verbunden ist. Mittels des ESP32 wird zudem schon die erste Verarbeitung der Daten vorgenommen bzw. die Rohdaten in einen Rotationszustand umgewandelt. Weiter wird mit dem ESP32 eine Internetverbindung hergestellt, mit welcher der Rotationszustand per WLAN an einen Server gesendet werden kann. Dabei werden die Daten per HTTP-Request an eine REST-API übermittelt.

Die REST-API ist darauf ausgelegt, die übermittelten Daten vorzuhalten und bei Bedarf an andere Systeme weiterzuleiten. Darüber hinaus stellt diese weitere Daten bereit, wie Status-Texte oder Grafiken.

Ein zweiter ESP32, welcher mit einem E-Ink-Display verbunden ist, soll nun in der Lage sein, den korrekten Rotationszustand des Würfels inklusive weiterer Daten, wie ein Bild oder einen Text, per HTTP-Request abzufragen. Daraufhin zeigt das E-Ink-Display je nach empfangenen Daten einen Text oder ein Bild an.

Inhaltsverzeichnis

1 Einleitung	1
2 Grundlagen	2
2.1 Der ESP32	2
2.2 Batterie-Shield	3
2.3 Der MPU-6050	4
2.4 Das I ² C-Protokoll	5
2.5 3D-Druck	6
2.5.1 Modellierung	6
2.5.2 TinkerCAD	7
2.5.3 3D Drucker	8
2.6 API	8
2.7 HTML	9
2.8 IP-Adressen	9
2.9 http-Requests	10
2.10 E-Ink-Display	10
2.11 Software	12
2.11.1 MicroPython	12
2.11.2 Git und GitHub	13
3 Konzeption	14
3.1 Überblick über das Projekt	14
3.2 Netzwerktopologie	14
3.3 Anforderungen an den Statuswürfel	15
3.4 Anforderungen an den Würfeldruck	18
3.5 Anforderungen an den Anzeigebildschirm	19
3.6 Anforderungen an die API	21
3.7 Anforderungen an die Website	22
4 Durchführung	23
4.1 Der Statuswürfel	23
4.1.1 Auslesung der Daten des MPU-6050	23
4.1.2 Integrierung der I ² C Übertragung	24
4.1.3 Auswertung der Daten	26
4.1.4 Übersenden der Daten an die API	29

4.1.5	Genauigkeit der Messungen	30
4.1.6	Modellieren	33
4.1.7	Zusammenbau	36
4.2	Das E-Ink-Display	37
4.3	Die API	41
4.4	Website	43
5	Zusammenfassung und Ausblick	45
5.1	Zusammenfassung	45
5.2	Ausblick	46

1 Einleitung

Sowohl in Kanzleien als auch in vielen Behörden wird veraltet immer noch viel zu viel Zettelwirtschaft betrieben. Damit wird insbesondere das Anbringen von Klebezetteln oder Haftnotizen zum Darstellen der derzeit verübten Tätigkeit an der Tür eines Büorraumes beschrieben. Diese Zettelchen erzeugen viele Probleme. Zum einen erzeugen sie durch Erwerb und Entsorgung eine nicht notwendige Kostenquelle und zum anderen wird das Entfernen dieser Papierstücke zu oft auch vergessen, wodurch es zu Fehlkommunikationen in Bürogebäuden kommen kann.

Daher soll das Ziel dieses Projektes sein, diese Zettel zeitgemäß durch ein digitales und ferngesteuertes Display zu ersetzen, welches zudem eine einfache, haptische und intuitive Steuerung durch einen drehbaren und intelligenten Statuswürfel besitzt, welcher seinen Status bzw. seinen Rotationsstatus selbstständig auslesen kann.

Zur Umsetzung dieser Ziele müssen zuerst weitere Feststellungen getroffen werden. Um eine optimale Flexibilität des Statuswürfels zu ermöglichen, sollte dieser keine konstante Energiezufuhr benötigen, sondern optimalerweise kabellos agieren, denn nur so kann der Statuswürfel einfach und dynamisch bedient werden. Folglich muss die Datenübertragung zwischen dem Statuswürfel und dem Display kabellos erfolgen, wozu entweder einer der beiden Komponenten einen Server darstellen muss, oder ein Server zwischen dem Statuswürfel und dem Display stehen muss. Für das Projekt sinnvoller ist die zweite Option, da durch die fehlenden Kabel der Grundkomponenten der eventuell große und dauerhafte Energiebedarf eines Servers nicht dauerhaft oder nur über einen kurzen Zeitraum gedeckt werden kann. Ferner ist das System durch einen zentralen Server auch leichter erweiterbar durch zum Beispiel ein zweites Display.

2 Grundlagen

2.1 Der ESP32

Der ESP32 ist ein Mikrocontroller, welcher von der Firma Espressif preiswert produziert und verkauft wird. Je nach Anbieter und Menge kostet er nur € 3,00 bis € 5,00. [1] Er ist beliebt, da er durch seine Taktfrequenz von 80MHz bis 240MHz einen Grundstein für viele Projekte legen kann. Auch in diesem Projekt bildet der ESP32 eine Kernkomponente, welche nicht nur im Statuswürfel Verwendung findet, sondern auch im Display. Er besitzt 4MB externen Massenspeicher, sowie 512 kB RAM. Auch daher wird der zum Ansteuern des ESP32 nötige Treiber "CH341SER" [2] extern und nicht intern auf dem ESP32 gespeichert. Des Weiteren ist er WLAN und auch Bluetooth-fähig, wodurch er sich für eine drahtlose Datenübertragung eignet. Darüber hinaus besitzt er verschiedene Kommunikationsschnittstellenprotokolle, wie SPI, I²C, oder I²S. In diesem Projekt ist das I²C-Protokoll besonders relevant. [3, 4, 5]

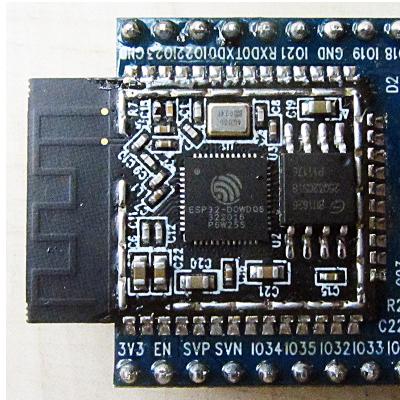


Abbildung 1: Der ESP32 [6]

Der ESP32, dargestellt in Abbildung 1, ist in der Regel auf Platinen angebracht, durch welche die Pins des ESP32 angesteuert werden können. Eine dieser Platinen ist das ESP32 D1-Mini.

Viele Funktionen des ESP32 D1-Mini, wie das I²C-Protokoll, können über die verschiedenen Pins angesteuert werden, wie auch die Abbildung 2 zeigt. Für den Statuswürfel sind dabei die Pins des oberen Teils relevant. Genauer sind der "3V3" Pin, welcher eine Spannungsversorgung von 3,3 V für anzuschließende Geräte darstellt, der "GND"-Pin, welcher den sogenannten "Ground" und somit den Stromrückfluss bildet, sowie die "IO21" und "IO22"-Pins, welche für das I²C-

Protokoll den I²C1 CL oder SCL und I²C1 DA oder SDA-Anschluss bilden, gemeint. [3, 4, 5]

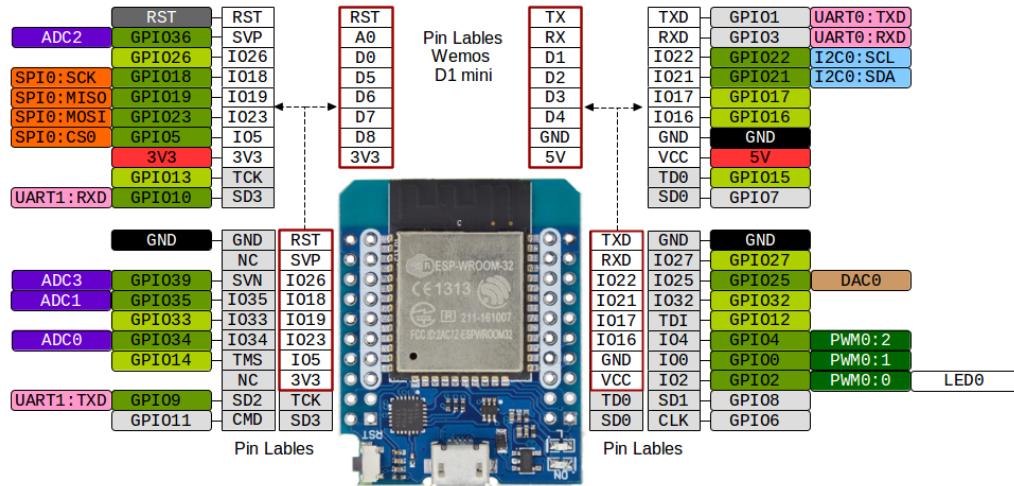


Abbildung 2: ESP32 D1-Mini-Pinout [7]

2.2 Batterie-Shield

Das "AZDelivery Batterie-Shield für Lithium Batterien für D1 Mini", welches in Abbildung 3 dargestellt ist, stellt eine einfache Möglichkeit dar, ein D1-Mini-Board, wie es auch beim ESP32 verbaut wird, mit einem Akkumulator zu verbinden. [8] Dabei erfüllt das "Batterie-Shield" drei Eigenschaften:

1. Ladeschutz: In dem "Batterie-Shield" ist ein TP 5400 Chip verbaut. Dieser Chip besitzt eine Schutzschaltung wodurch ein kurzer maximaler Ladestrom von 1A bzw. 1000 mA ermöglicht wird. Zudem ist eine Ladeautomatik integriert, durch welche der angeschlossene Akkumulator nicht überladen werden kann. [9]
 2. Leichte Bedienbarkeit: In dem "Batterie-Shield" ist ein Mikro-USB Anschluss integriert, an welchem ein einfaches Mikro-USB-Kabel zum Aufladen des Akkumulators angeschlossen werden kann. Das führt zu einer einfachen Bedienbarkeit, da das "Batterie-Shield" mit einer im elektrotechnischem Bereich standardisierten Ladespannung von 5 Volt oder 12 Volt den Akkumulator aufladen kann.

3. Integrierte Montage: Das Board kann einfach über oder unter dem ESP32 D1-Mini angebracht werden, da die beigelegten Stiftreihen auf beiden Boards verlötet werden können. [8]

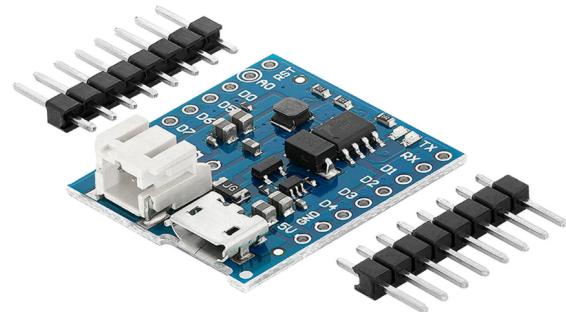


Abbildung 3: Batterie-Shield für das D1-Mini-Board [8]

2.3 Der MPU-6050

Der MPU-6050 (siehe Abbildung 4) ist ein elektronischer Mikrosensor (mikro-elektromechanisches System = MEMS), welcher zugleich als Thermometer, Gyroskop sowie Accelerometer verwendet werden kann. Somit kann er die Temperatur, seine Winkelgeschwindigkeit und seine Ausrichtung durch die auf ihn wirkende Beschleunigung messen. [10, 11]



Abbildung 4: Der MPU-6050 [12]

Das elektronische Thermometer basiert auf dem unterschiedlichen Widerstand von Stoffen bei unterschiedlichen Temperaturen. Wenn zum Beispiel ein Halbleiter verwendet wird, dann sinkt der Widerstand mit steigender Temperatur. Somit kann

die Temperatur indirekt durch den sinkenden Widerstand bestimmt werden. Jedoch sind viele elektronische Thermometer nicht sehr genau, da durch die selbst verursachte Abwärme die zu messende Temperatur erhöht wird. Dadurch kann ein Messfehler entstehen. Dieser ist davon abhängig, wie lange das Thermometer bereits misst, da zu Beginn die Abwärme und damit der Messfehler gering ist. Die Durchlüftung des Sensors mit Umgebungsluft, sowie die erzeugte Abwärme haben starken Einfluss auf den Messfehler. [13]

Ein Gyroskop verwendet einen sich möglichst reibungsfrei rotierenden Kreisel, dessen Trägheit genutzt wird, um Änderungen der Orientierung oder Winkelgeschwindigkeit zu messen. Aufgrund des Drehimpulses des Kreisels bleibt dessen Rotationsachse stabil und bietet eine Referenz, gegen welche Bewegungen des Systems gemessen werden können. Gyroskope erfassen dabei nicht direkt Kräfte, sondern Winkelgeschwindigkeiten, die auf den Kreisel oder entsprechende Messmechanismen einwirken. [14]

Ein Accelerometer, auch Beschleunigungsmesser genannt, ist ein Messgerät zur Ermittlung der Beschleunigung häufig in Abhängigkeit zu der Ausrichtung. In diesem Fall ist es ein elektronisches Messgerät und agiert auf der Basis von Mikro-Silicium-”Federn”, welche nur wenige Mikrometer breit sind. Je nach Abstand dieser ”Federn” zueinander ändert sich die Kapazität des aus den ”Federn” bestehenden Kondensators. Je kleiner der Abstand, desto höher die Kapazität. So kann mit vielen dieser ”Federn” die Beschleunigung indirekt bestimmt werden.

Durch die dreiachsig Aufstellung von drei linearen Accelerometern kann die Beschleunigung auch dreidimensional ermittelt werden. Da durch die Erde konstant eine Beschleunigung g von exakt $g = 9,81 \text{ m/s}^2$ auf das Accelerometer wirkt, kann jederzeit festgestellt werden, welche Seite des Accelerometers oben bzw. unten ist, da dort g oder $-g$ gemessen werden sollte. [15]

2.4 Das I²C-Protokoll

I²C steht für ”Inter-Integrated Circuit” und beschreibt einen seriellen Datenbus, welcher 1982 von Philips Semiconductors entwickelt wurde. Bis heute bleibt der Datenbus ein weit verbreiteter Industriestandard für serielle Datenkommunikation. Hauptsächlich wird er zur geräteinternen Kommunikation zwischen verschiedenen Schaltungsteilen verwendet, wie etwa in Fernsehgeräten oder zwischen einem Mikrocontroller und Peripheriegeräten.

Der I²C-Bus ist als Master-Slave-Bus konzipiert, wobei ein Controller, auch

”Master“ genannt, die Kommunikation initiiert und die Targets, auch ”Slaves“ genannt, darauf reagieren. Der Bus benötigt dabei nur zwei Signalleitungen. Diese sind eine Taktleitung (”SCL“ bzw. ”serial clock“) und eine Datenleitung (”SDA“ bzw. ”serial data“), welche beide mit Pull-up-Widerständen an die Versorgungsspannung angeschlossen sind. Die Kommunikation erfolgt mit positiver Logik, wobei ein High-Pegel einer logischen „1“ und ein Low-Pegel einer „0“ entspricht. Es ist daher aufgrund seiner Einfachheit, Flexibilität und seines dadurch geringen Stromverbrauchs ein beliebtes Protokoll in vielen Branchen. [16, 17]

Ursprünglich sah das I²C Protokoll eine Maximalgeschwindigkeit von 100kbit/s vor. Jedoch wurden 1998 ”Fast Mode“ und 2007 ”Fast Mode Plus“ vorgestellt, welche Geschwindigkeiten von 400kbit/s und sogar 1.000kbit/s erreichen. Zuletzt wurde im Jahr 2012 der ”Ultra Fast Mode“ entwickelt, welcher eine Geschwindigkeit von bis zu 5.000kbit/s unterstützt, jedoch nur mit Einschränkungen, wie zum Beispiel, dass die Daten nur einseitig gesendet werden oder dass es nur ein ”Master“-Gerät geben kann. [18, 19]

2.5 3D-Druck

Der 3D-Druck ist eine neuartige Form der Herstellung von feinen, verschiedenartigen und einzigartigen Objekten. Um jedoch ein gewünschtes Objekt im 3D-Druckverfahren drucken zu können, muss es vorerst für die Maschine bzw. den Drucker verständlich modelliert werden. [20]

2.5.1 Modellierung

Modellieren beschreibt im physikalischen und auch informatischen Sprachgebrauch das Bilden eines Modells bzw. einer Abbildung der Realität, zur Vereinfachung eines komplexen Vorgangs.

In diesem Fall jedoch beschreibt Modellierung das Erstellen eines Modells, welches die Realität möglichst genau abbildet. Genauer wird hierbei beim 3D-Druck das Erstellen eines für den 3D-Drucker verständlichen Modells bezeichnet. Somit gibt es auch einige, für den 3D-Druck wichtige und zu beachtende Faktoren.

Der 3D-Drucker kann nicht ”ins Nichts“ drucken bzw. unter jeder Fläche benötigt der 3D-Drucker einen geeigneten Unterbau. Somit können verschiedenste hohle Strukturen bzw. Strukturen mit Überhang auf verschiedene Arten und Weisen gedruckt werden. Zum einen kann versucht werden, auf waagerechte Überhänge zu verzichten, indem stattdessen Schrägen mit einem Neigungswinkel von bis zu

60°, gemessen von der Vertikalen, benutzt werden. Je nach Schichtdicke, Material und Schichtbreite unterscheidet sich der maximale Neigungswinkel eines 3D-Druckers. Des Weiteren können sehr gute 3D-Drucker je nach Druckeinstellungen, sowie Filamentart "Lücken" bzw. vertikale Brücken mit einem Abstand von 8 cm bis 13 cm drucken. Zusätzlich können Überhänge durch ein temporäres, meist sehr schmales und dünnes, leicht zu entfernendes Fundament bzw. Stütze gedruckt werden.

Die einfachste Möglichkeit jedoch einen Hohlraum zu drucken, ist das Zerteilen des Objekts in zwei Teile, wodurch die Decke nun auf dem Druckbett bzw. Druckboden aufliegt und deshalb die Decke bzw. der Hohlraum mit Leichtigkeit gedruckt werden kann. Bei dieser Methode muss nur beachtet werden, dass eine Art Zusammensteckmechanismus modelliert wird, welcher entweder an sich verkantet ist und zusammenhängt oder leicht verklebt werden kann. [21]

Die bekanntesten zum Modellieren für den 3D-Druck sind TinkerCAD, Blender, FreeCAD, OpenSCAD, Cura oder Zbrush. Das CAD in vielen dieser Programme steht für "Computer-Aided Design" oder zu deutsch "rechnerunterstütztes Konstruieren".

2.5.2 TinkerCAD

TinkerCAD ist ein weit verbreitetes Programm zur Erstellung von 3D-Modellen. Es ist online und kostenfrei verfügbar, wodurch es an Beliebtheit erlangt hat. Doch das Wichtigste ist, dass die von TinkerCAD bereitgestellte Nutzeroberfläche sich intuitiv und nutzerfreundlich bedienen lässt. So lassen sich in das erstellte Modell schnell Objekte, wie einen Würfel, einfügen, sowie diesen beliebig auf der Arbeitsfläche verschieben. Dabei können verschiedenste Objekte, wie beispielsweise Prismen, Kugeln oder auch komplexe Formen, wie Sterne, Ikosaeder, beliebige Texte oder auch sehr verschiedene vorgefertigte Strukturen, zum Beispiel ein "Cartoon eye-left" oder ein "weihnachtlicher Tannenbaum" verwendet werden.

Weiter können sämtliche Formen "merged", zusammengeführt oder gruppiert werden, womit sie als eine einzige Struktur betrachtet werden können. Das ist sehr nützlich, da sonst die modellierte Ergebnis sehr kompliziert und inkohärent erscheinen kann.

Zuletzt und das ist eine der wichtigsten Funktionen, können Formen invertiert und somit miteinander an der Intersektion ausgelöscht werden. Mit diesen Funktionen ist es mit TinkerCAD möglich, alle denkbaren und notwendigen Formen zu modellieren. [22]

2.5.3 3D Drucker

Um nun eine modellierte Form drucken zu können, wird ein 3D-Drucker benötigt. Doch das Drucken selbst ist auch eine schwierige Aufgabe, denn unter Modellierung wurde schon das Problem des Druckens über der Leere besprochen, jedoch gibt es noch weitere.

Durch die große Druckerspitze muss der 3D-Drucker von unten nach oben drucken und benötigt somit einen "Fahrplan" bezüglich seiner Bewegungslinie. Darüber hinaus muss der Druckuntergrund erwärmt und mit der Druckerspitze synchronisiert sein.

Weiterhin werden solide Wände, oft um Material zu sparen, nicht vollständig solide, sondern eine Füllstruktur gedruckt, welche bis zu 90% aus Luft besteht und trotzdem stabil ist, da in der Regel regelmäßige Formen in dieser Füllstruktur verbaut werden, wie gleichmäßige Dreiecke oder Quadrate. [20]

2.6 API

Das Wort "API" ist eine Abkürzung für "Application Programming Interface" oder übersetzt "Programmierschnittstelle". Somit ist eine API eine Schnittstelle zwischen zwei mit der API verbundenen informatischen Systemen. Diese Schnittstelle, durch welche Daten übertragen werden, wird von Programmierern genutzt, um eine Datenübertragung zu kontrollieren, indem Daten abgespeichert und manipuliert werden. Zudem können mit einer API mehr als nur zwei informative Systeme verbunden werden und die Datenübermittlung bei drei oder mehr Systemen gesteuert werden. Zusätzlich haben API's oft schon vorprogrammierte vereinfachte Funktionen, welche die Programmierung oder grafische Darstellung stark vereinfachen können. [23]

Eine solche programmierbare REST-API bietet "FastAPI". "FastAPI" nutzt dabei "uvicorn", um mittels Python in einer nicht synchronen "async"-Funktion Daten einzulesen, diese zu verarbeiten und damit diese zu speichern, zu manipulieren sowie danach weiterzusenden. [24, 25]

REST steht dabei für "Representational State Transfer" und beschreibt eine heute gängige Kommunikationsart mittels zum Beispiel http-Protokollen, bei welchen die Kommunikation in sich einmalig und abgeschlossen ist, sowie keine Informationen vom Datentransfer-Service selbst einbehalten werden. [26]

2.7 HTML

HTML steht für "Hypertext Markup Language" oder auf deutsch "Hypertext-Auszeichnungssprache". Das ist ein Dateityp, welcher dabei helfen soll einen Text zu strukturieren und auch teilweise zu formatieren. Dazu zählt die Strukturierung in "head" und "body" oder in Tabellen, sowie die Formatierung von zum Beispiel der Schriftgröße. Zusätzlich können auch Metadaten, wie beispielsweise der Titel, leicht in HTML unter dem "head" eingefügt werden.

Diese HTML-Dokumente sind eine Grundlage des heutigen "World-Wide-Web" und werden von gängigen Browsern wie Safari, Opera, Firefox, Microsoft Edge oder Google Chrome unterstützt sowie dargestellt. Jedoch ist HTML ein Dateityp beziehungsweise eine Art Programmiersprache, welche nicht für eine Formatierung ausgelegt wurde, sondern für die semantische Strukturierung. Für die Formatierung wird daher häufig CSS verwendet, welches dann in Kombination mit dem HTML-Dokument vom Browser dargestellt wird. [27]

2.8 IP-Adressen

Vorab sollte klargestellt werden, dass es in Bezug auf LAN private und öffentliche IP-Adressen gibt. Öffentliche IP-Adressen werden Netzwerken im "World-Wide-Web" zugewiesen. In diesen Netzwerken wird den dort verbundenen Geräten eine lokale und private IP-Adresse zugewiesen.

Hier soll es nur um private IP-Adressen gehen. Diese identifizieren ein Gerät eines lokalem Netzwerkes und bestehen aus vier 8-Bit Zahlen von 0 bis 255, getrennt mit je einem ". ". Ein Gerät eines lokalen Netzwerkes kann ein anderes Gerät des selben lokalen Netzwerkes mit seiner lokalen IP-Adresse adressieren und ansteuern sowie Informationen übersenden.

Des Weiteren gibt es Ports, welche mit einem ":" an die IP-Adresse angehängt werden können. Dieser besteht in der Regel aus einer 16-Bit Zahl (0 bis 65.535). Durch den Port können unterschiedliche Funktionen eines Gerätes des Netzwerkes parallel angesteuert und einfacher unterschieden werden. [28]

2.9 http-Requests

Das http-Protokoll, oder "Hypertext Transfer Protocol", bezeichnet ein Protokoll zur Datenübertragung zwischen zwei informatischen Systemen. Anwendung findet das http-Protokoll großteils im "World-Wide-Web", wo es die Informationsübergabe, genauer das Herunterladen von Websites auf den Browser regelt. Jedoch geschieht dies heutzutage meist mit einer Fortentwicklung des http-Protokolls, da dieses als veraltet und unsicher angesehen wird. Dessen Nachfolger, die https Datenübertragung ist sehr ähnlich, jedoch verwendet sie eine digitale Signatur mittels SSL, um die Datenübertragung sicherer zu machen. [29, 30]

Das http-Protokoll hat die folgenden zwei Hauptbefehle, mit welchen Daten gesendet oder auch gefordert werden können: den POST-Befehl und den GET-Befehl. Diese beiden Befehle stellen die Grundlage für die http-Requests dar.

Eine GET-Request besteht in der Regel aus dem Befehl "GET", gefolgt von der url der Informationsquelle, welche heruntergeladen werden soll. Es ist dabei nötig, alle Informationen, wie zum Beispiel einen Suchbegriff, in der url zu versenden. Der Nachteil dessen ist jedoch, dass viele Browser die url sichern und persönliche Daten damit gespeichert werden könnten. Daher können mit der POST-Request Daten abseits der url in einem "body" mitgesandt werden. Diese werden dann dementsprechend nicht in der url zwischengespeichert. [31]

2.10 E-Ink-Display

Ein E-Ink-Display oder "Electronic-Ink-Display" ist eine Art Bildschirms, welcher nicht auf leuchtenden LED-Pixeln basiert, sondern auf Farbe, welche in den einzelnen Pixeln platziert wird und von dem Umgebungslicht angeleuchtet wird.

Ein klassischer E-Ink-Display-Pixel besteht dabei aus einer Flüssigkeit, in welcher positiv und negativ geladene Farbpartikel vorliegen. Die Partikel können sich dabei frei in der Flüssigkeit bewegen. Auf beiden Seiten des Pixels, das heißt hinter und vor dem Pixel aus der Sicht auf das Display, liegen durchsichtige Elektroden an, welche verschieden geladen sind.

Je nach Ladung der Elektroden bewegen sich dann die unterschiedlich geladenen Farbpartikel in den Vordergrund des Pixels und in den Hintergrund des Pixels. Sie sind dadurch separiert und die geladenen Farbpartikel, welche sich nach vorne bewegt haben, bilden eine homogene Farbschicht, welche von einem auf das Display schauenden Beobachter gesehen werden kann. Nun ist es möglich die an den Pixel anliegenden Elektroden umzupolen. Dies führt zu einer Verschie-

bung der geladenen Farbpartikel, sodass auf die vorher vorne vorliegenden Partikel durch das elektrische Feld eine Kraft nach hinten wirkt und die Partikel nach hinten verschoben werden, während die vorher hinten vorliegenden Partikel nach vorne bewegt werden. [32]



Abbildung 5: Das 400x300 three-color E-Ink display module von Waveshare [33]

Somit ist es möglich, mittels dieser einfachen Technik ein Display mit zwei verschiedenen Farben, häufig schwarz und weiß, zu bauen. Der Vorteil dessen gegenüber eines normalen Displays ist, dass sobald die Farbe einmal eingestellt ist keine weitere Energie mehr benötigt wird, um das Bild anzuzeigen. Dadurch muss die Stromversorgung nicht konsistent sein und das E-Ink-Display hat ohne Bildänderung keinen Energiebedarf. Des Weiteren weist das Display eine höhere Ähnlichkeit zu Papier auf als ein herkömmliches LED-Display und ist auch bei starkem Sonnenlicht gut lesbar, was je nach Anwendungsbereich vorteilhaft sein kann. Jedoch hat ein E-Ink-Display auch Nachteile. Entsprechend leuchtet es nicht von selbst und ist daher im Vergleich zu üblichen Displays dunkel und vor allem in einer dunklen Umgebung kann ein E-Ink-Display kaum bis gar nicht lesbar sein, da es nur von Licht aus der Umgebung beleuchtet wird. Zudem ist ein E-Ink-Display durch die physischen Pixel in der Bildwiederholrate eingeschränkt. Filme können kaum angeschaut werden und bei höherer Bildwiederholrate sinkt auch die Energieeffizienz, der eigentliche Vorteil des E-Ink-Displays. Daher lässt es sich optimal in Tätigkeiten anwenden, welche dauerhaft oder für lange Zeit einen Text ausgeben sollen, wie Preisschilder in Läden oder in E-books. [32, 34]

In neueren E-Ink-Displays lassen sich mehr als nur zwei Farben darstellen. Mit der Kaleido-Bauweise zum Beispiel wird ein RGB-Farbfilter über die Pixel gelegt, wodurch Farben darstellt werden können. Auch sind auf ähnliche Art und Weise E-Ink-Displays mit den Farben weiß, rot und schwarz möglich, welche besonders

bei Ladenschildern Anwendung finden, da mit der Farbe rot ein Angebot im Laden dargestellt werden kann. [35]

Ein Beispiel eines solchen E-Ink-Displays, welches schwarz, weiß und rot darstellen kann, ist das "400x300, 4.2inch E-Ink display module, three-color, SPI interface" von Waveshare, welches auch in der Abbildung 5 dargestellt ist. Um dieses anzusteuern, sind die in Tabelle 1 aufgelisteten Kabelanschlüsse vorgesehen. [33, 36]

Anschluss	Beschreibung
VCC	3.3V 5V
GND	Ground
DIN	SPI MOSI pin
CLK	SPI SCK pin
CS	SPI chip selection, low active
DC	Data/Command selection (high for data, low for command)
RST	External reset, low active
BUSY	Busy status output, low active

Tabelle 1: Kabelanschlüsse für das E-Ink-Display [33]

2.11 Software

2.11.1 MicroPython

MicroPython ist eine, wie der Name schon sagt, sehr an Python angelehnte Programmiersprache, welche im Gegensatz zu Python primär für die Programmierung von Mikro-Controllern verwendet wird. In der Realität ist sogar Python 3 mit vielen Standard-Bibliotheken, wie unter anderem "time", implementiert. Ferner enthält MicroPython eine interaktive Eingabeaufforderung (REPL), über welche einfach Befehle ausgeführt und sein Code somit getestet werden kann. MicroPython selber besteht aus einem Python-Compiler für Bytecode und einem Laufzeitinterpreter dieses Bytecodes. [37, 38]

Zudem gibt es Entwicklungsumgebungen wie "Thonny", welche den Einstieg mit "MicroPython" weiter erleichtern können. Der Vorteil bei "Thonny" ist, dass es kostenlos, sowie Open-Source ist und Anfängern durch Intuition das Programmieren erleichtert. [39]

2.11.2 Git und GitHub

Git ist eine kostenlose und frei zugängliche Software zur Versionsverwaltung von Daten. Auch wenn sie ursprünglich für das Linus-Betriebssystem und deren Versionsverwaltung erstellt wurde, wird Git heutzutage sehr häufig für die Speicherung und Verwaltung von Versionen einzelner (Informatik-) Projekte genutzt. So findet sich auf Git eine sehr diverse Reihe verschiedenster, teilweise öffentlich zugänglicher Projekte, genannt Git-“Repository’s”. [40]

Für die Verwaltung stellt Git einige Befehle für die einzelnen Nutzer bereit. Zunächst muss klargestellt werden, dass Git mit zwei verschiedenen Instanzen arbeitet. Dem “Remote”, bzw. den in Git abgespeicherten Daten, und den “Working Files” bzw. den bei dem Nutzer abgespeicherten Daten. Zwischen diesen beiden Instanzen sind nun verschiedene Befehle möglich. Die wichtigsten Befehle sind der “pull” sowie der “push”. Durch einen “pull” werden die im Git abgespeicherten Daten zu dem Nutzer kopiert. Der “push” andererseits kopiert die Daten des Nutzers auf die im Git gespeicherten Daten. Es wird bei einem “push” nur die Änderung “gepusht” und auf diese, welche mittels eines “add” festgestellt wird, wird mit einem “commit” eine Änderungsüberschrift sowie ein Änderungsprotokoll angehängt. So werden im Git nicht nur die einzelnen Dateien gespeichert, sondern auch die historischen Änderungen und somit die früheren Versionen der Dateien. Dadurch wiederum können unvorhergesehene negative Änderungen der Dateien annulliert werden. [41]

Es gibt zudem viele verschiedene Softwares, mit welcher der Zugang zu dem Git-System erleichtert wird. Die größten und bekanntesten Beispiele dazu sind Gitlab und GitHub, von welchen GitHub oft als Programm für Privatpersonen und Gitlab häufig als Programm für Firmen oder Geschäftsleute angesehen wird. In dieser BeLL wurde ein Git-Repository mittels GitHub erstellt. In diesem ist die gesamte BeLL abgespeichert. Der Link dazu lautet “<https://GitHub.com/Typiano/Git>”. [42]

3 Konzeption

3.1 Überblick über das Projekt

Das Projekt beinhaltet die Programmierung und den Bau eines Statuswürfels. Der Statuswürfel soll ein Würfel sein, welcher seinen Rotationszustand, relevant ist seine obere Seite, an eine API sendet, wo die Daten weiterverarbeitet werden. Er soll dabei zudem seine Temperatur, bzw. die Umgebungstemperatur um den Statuswürfel auslesen.

Die empfangenen Daten werden von der API gespeichert und manipuliert, bzw. weitere Daten angehängt und darauf an das E-Ink-Display weitergegeben. Auf dem E-Ink-Display sollen die Daten dann angezeigt werden. Dabei insbesondere auch ein von der API mitgegebener Text.

Diese Textweitergabe sowie die API sind nötig, da die Daten flexibel über eine weitere Instanz geändert werden sollen. Diese Instanz soll eine Website sein. Von dieser aus sollen die Daten der API derart verändert werden können, dass sie auf dem E-Ink-Display anders dargestellt werden. Dadurch können auch neuartige ursprünglich unvorhergesehene Zustände der arbeitenden Person einfach integriert werden.

3.2 Netzwerktopologie

Der Begriff Netzwerktopologie bezeichnet die physische und logische Struktur von Knotenpunkten und Verbindungen in einem Netzwerk. Als die Knoten werden in der Regel Netzwerkgeräte wie Switches, Router und Software mit Switch- und Router-Funktionen bezeichnet. Netzwerktopologien werden meist in Form von Diagrammen visualisiert. [43]

Dadurch kann die Struktur eines Netzwerkes als Graph einfach visualisiert werden, wodurch das logische Verständnis der praktischen Funktionsweise des Netzwerkes vereinfacht wird. In der folgenden Abbildung 6 ist die Topologie der in diesem Projekt geplanten Anwendung dargestellt. Dabei sind die Knoten bzw. Instanzen als gelbe Rechtecke mit Beschriftung dargestellt und die Kanten/ Pfeile stellen die Verbindungen beziehungsweise Beziehungen zwischen den einzelnen Instanzen dar. [44]

So hat der Statuswürfel einen Einfluss auf die API, da er dieser Daten sendet und die API diese speichert. Dem E-Ink-Display sollen ebenfalls Daten übersandt werden, jedoch ist das E-Ink-Display kein Server und nicht dauerhaft aktiv. Daher

muss es die Daten erst von der API anfordern um sie daraufhin empfangen zu können. Deswegen ist in Abbildung 6 ein Doppelpfeil zwischen API und E-Ink-Display zu erkennen. Die Beziehung der Website zu der API ist insofern eine Kombination aus den Beziehungen des Statuswürfels und des E-Ink-Displays zu der API, da die Website sowohl Daten anfordern, als auch senden soll.

Die einzelnen Komponenten bzw. Instanzen unter sich, genauer der Statuswürfel, die Website und das E-Ink-Display besitzen keine Kanten oder Verbindungen untereinander, da sie nicht miteinander kommunizieren können und sämtliche Kommunikation über die API ausgetragen wird.

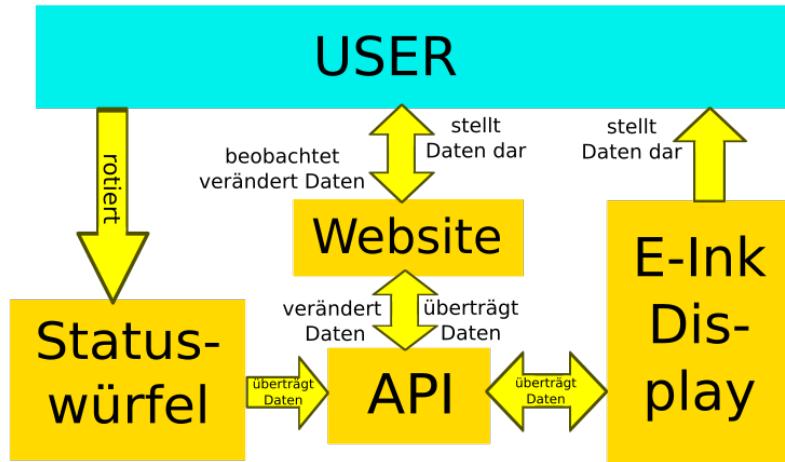


Abbildung 6: grobe Darstellung der Netzwerktopologie

Der Benutzer, hier “USER”, hat ebenfalls eine besondere Rolle in dem Netzwerk. Er kann sowohl den Statuswürfel, als auch die Website beeinflussen und sie kontrollieren, bzw. ihnen Daten geben, indem er den Statuswürfel dreht oder die Website benutzt. Von dem Nutzer gehen jedoch nicht nur die Pfeile aus, denn bei ihm münden sowohl Pfeile der Website, als auch vom E-Ink-Display, da beide Instanzen Daten an den Nutzer übermitteln können. Da der Nutzer keinen maschinellen Teil des Netzwerkes darstellt, ist er auch nicht gelb gekennzeichnet, sondern cyan.

3.3 Anforderungen an den Statuswürfel

Der Statuswürfel ist das Herzstück des Projekts und sollte verschiedene Funktionen erfüllen. Es muss ein Gerät sein, welches sowohl Rotationsdaten ermittelt,

diese in einen Rotationszustand umwandelt und diesen daraufhin drahtlos weitergibt. Der Apparat soll zudem die Außentemperatur ermitteln können und sie ebenfalls kabellos über WLAN übermitteln. Das System muss weiter ohne konstante kabelgebundene Stromquelle agieren können. Somit ist ein Akkumulator unverzichtbar.

Diese Anforderungen können mit dem richtigen Aufbau überwunden werden. Der ESP32 D1-Mini ist hierbei essentiell, da er zum einen WLAN-fähig ist, wodurch eine drahtlose Datenübertragung ermöglicht wird, und zum anderen aber auch einen leistungsfähigen Prozessor sowie ein integriertes I²C-Protokoll besitzt. Das ist relevant, da der Sensor, der MPU-6050, der hier verwendet werden soll auch integriert durch ein I²C-Protokoll adressiert werden kann. Er ist mit 2 cm x 1,5 cm klein und kann schnell und präzise Rotationsdaten sowie Temperaturdaten auslesen.

Die Verbindung zwischen ESP32 D1-Mini und dem Sensor geschieht über vier Kabel: Einem zur Spannungsversorgung von 3,3 V und einen für den "Ground". Zwei werden zudem für das I²C-Protokoll benötigt.

Der Sensor erzeugt nun 3 Rohdaten, welche im kartesischen Koordinatensystem die Beschleunigung auf der X, Y und Z Achse beschreiben. Da der Sensor jedoch auf der Fallbeschleunigung und Gewichtskraft basiert, kann ein starkes Bewegen und Krafteinwirken auf den Sensor zu Fehlmessungen führen. Daher soll aus drei Messreihen der Durchschnitt gebildet werden, sodass ein falscher Messwert durch eine Mittelung bereinigt werden kann. Zudem sind die Rohdaten ohne Einheit in einem 16-Bit Zweierkomplement angegeben, wodurch sie zum besseren Verständnis erst in eine Dezimalzahl umgewandelt werden sollen. Im Idealfall wird die Beschleunigung dann in alle Richtungen von 1 bis -1 angegeben, wobei $1 \equiv g$ gilt. Dies sollte nach dem Datenblatt circa der Fall sein, wenn die Beschleunigung $2^{14} = 16.384$ ausgegeben wird. Wenn der Sensor korrekt ausgerichtet ist und somit die Richtungsvektoren der Beschleunigungsdaten den Normalvektoren der Seitenflächen des Statuswürfels entsprechen, ergibt sich für jeden Rotationszustand des Würfels eine Kombination aus den Werten X, Y und Z. Dabei entsprechen je zwei Werte ca. einer 0, während ein Wert entweder ungefähr einer 1 oder einer -1 entspricht. Somit kann der Rotationszustand des Statuswürfels ermittelt werden.

Es muss zudem beachtet werden, dass der Würfel unter Umständen nicht auf einer ebenen Fläche liegt, weshalb die auf dem Sensor wirkende Gewichtskraft in der Richtung der Wirkungslinie deutlich geringer ausfallen kann. Daher sollte

allgemein eine Toleranz von 0,25 implementiert werden. Das entspricht mit der Formel $\cos(\alpha) = F_{Wirk}/F_g$ und $F_{Wirk} = 0,75$ sowie $F_g = 1$ einem Toleranzwinkel von ungefähr $\alpha \approx 40^\circ$. Die Kraft "F" ist direkt proportional zu der Beschleunigung "a" bei konstanter Masse "m". Deshalb kann die Berechnung, welche in Bezug auf die Kraft üblich ist, auf die Beschleunigung "a" übertragen werden.

Nun sollte ein Rotationszustand mit den Daten ermittelt werden können. Diesem Rotationszustand kann wie bei einem Würfel ein Wert zwischen 1 und 6 zugeordnet werden. Es ist theoretisch denkbar, dass durch Fehlmessung kein Rotationszustand festgestellt werden kann. Dies kann durch intensives Schütteln oder einem um 45° geneigten Statuswürfel geschehen. In diesem Fall soll ein weiteres Mal gemessen werden und die Daten sollen nicht übersandt oder gespeichert werden.

Weiter muss eine Temperatur beim Statuswürfel gemessen werden. Die Temperatur kann ebenfalls mit dem MPU-6050 gemessen werden, da es gleichzeitig auch als Thermometer fungiert. Die Temperatur kann so von dem ESP32 D1-Mini ausgelesen werden. Es gilt zu beachten, dass wahrscheinlich durch einen Widerstand in der Elektronik, welcher zu Abwärme führt, der Betrag der Temperatur zu hoch sein wird.

Die ermittelten Daten sollen nun mit einer POST- oder GET-Request an die API in Form von zwei Zahlen gesandt werden. Um Energie und Strom zu sparen, sollen die Daten nur übermittelt werden, wenn es neue Daten gibt. Daher müssen die Daten auch auf dem ESP32 immer wieder abgespeichert werden, um zu kontrollieren, ob der nun ermittelte Rotationszustand neu ist. Zudem sollen bei dem Rotationszustand 1 schon neue Daten übersandt werden, auch wenn sich nur die Temperatur ändert. Das geschieht, da für den Status 1 eine temperaturabhängige Ausgabe auf dem E-Ink-Display geplant ist.

Um den Statuswürfel kabellos zu gestalten muss zudem ein Akkumulator, sowie eine Kabelverbindung zwischen dem Akkumulator und dem ESP32 D1-Mini eingebaut werden. Jedoch ist die Verbindung zu einem Akkumulator nicht im ESP32 D1-Mini integriert, weshalb auf den ESP32 D1-Mini ein Aufsatz aufgelötet sein soll, welcher einen Akkumulator unterstützt. Hier bietet sich das "AZDelivery Batterie Shield für Lithium Batterien für D1 Mini" an, da es eine einfache Bedienbarkeit besitzt.

Da der Statuswürfel kabellos mit einem Akkumulator betrieben wird, sollte auch so wenig Energie wie möglich verbraucht werden. Der ESP32 hat dabei eine besondere Funktion mit der Bezeichnung "deepsleep", welche den ESP32

vorübergehend für eine bestimmte Zeit deaktiviert. So kann zwischen dem Messen des Status, bzw. der Temperatur, immer für eine bestimmte Zeit der "deepsleep" durchgeführt werden, wodurch die Energieeffizienz gesteigert werden soll. Dabei soll der ESP32 alle 5 Sekunden seinen Status messen und somit zwischen jeder Messung 5 Sekunden lang den "deepsleep" vollführen. Dazu soll für besondere Energieeffizienz ein Status existieren, welcher den "deepsleep" besonders lange für sehr gute Energieeffizienz durchführt. Die Zeit soll hier 5 Minuten betragen und dem Rotationszustand soll der Status 6 zugeordnet sein.

3.4 Anforderungen an den Würfeldruck

Damit nun auch noch ein Würfel entsteht, sollen alle technischen Komponenten in einen 3D-gedruckten Würfel eingesetzt werden.

Damit der Statuswürfel nun auch flexibel und intuitiv gedreht werden kann, muss auch ein dementsprechend gut in der Hand liegender Würfel produziert werden, in welchem sich dann die verschiedenen Komponenten befinden. Um eine derart feine und komplexe Form herzustellen, wurde sich für ein 3D-Druck-Verfahren entschieden, da durch dieses fein und mit viel Präzision gearbeitet werden kann, sowie schnell eine Herstellung erneut durchgeführt werden kann, falls der Würfel nicht exakt modelliert wurde.

Zum Modellieren können verschiedenste Programme bzw. Software verwendet werden. In diesem Fall wurde sich für TinkerCAD entschieden, da es online und kostenlos eine breite und sehr einfach zu bedienende Nutzeroberfläche bietet, durch welche mit Leichtigkeit einfache Formen, wie Würfel, Zylinder, Quader allgemein oder Prismen mit einer n-Eckigen Grundfläche, in sämtlichen Varianten zusammengefügt werden können, wie es auch für den Statuswürfel benötigt wird. [45]

Die Optimalgröße für einen handlichen Statuswürfel orientiert sich bei der Größe eines üblichen Zauberwürfels oder umgangssprachlich "Rubik's-Cube", da dieser mit seinen 5,7 cm Kantenlänge eine sehr handliche Form darstellt. Beim Modellieren müssen zudem einige Eigenschaften des Würfels beachtet werden. Das größte Bauelement des Statuswürfels ist der Akkumulator, welcher mit einer Länge von 6 cm, an seinen elektrischen Spannungsquellen und seinem Kabelursprung sogar 6,5 cm, einer Breite von 5 cm und einer Dicke von 0,5 cm die Maße des Statuswürfels Maßgeblich bestimmen. Da der Statuswürfel ein Würfel sein soll sowie eine Wanddicke nach außen von in der Regel 0,5 cm, jedoch minimal 0,2

cm haben soll, ist eine allgemeine Kantenlänge im äußereren Bereich von 7 cm angebracht. 7 cm sind zwar 1,3 cm länger, als ein handelsüblicher Zauberwürfel, jedoch ist auch das noch eine sehr handliche Form. Weitere in der Modellierung zu berücksichtigende Objekte sind neben dem ESP32 und dem MPU-6050, für welche jeweils eine passende Plattform bzw. passende Einkerbungen modelliert wurden, auch eine Öffnung für Stromzufuhr für den Akkumulator bzw. eine Öffnung um die Software des ESP32 auch im Nachhinein noch verändern zu können. Weiterhin muss noch ein Loch für einen Reset-Taster gelassen werden, da dieser bei unerwartet akzidentell auftretenden Fehlern gebraucht werden könnte.

Zuletzt sollte bedacht sein, dass alle Komponenten an deren korrekte Position platziert werden müssen und somit eine verschließbare Öffnung für ESP32, MPU-6050 und die Batterie existieren muss und zudem gilt es allgemein zu beachten, dass ein 3D-Drucker nicht wahllos Strukturen in der Luft bzw ohne stabilen Unterbau drucken kann, denn ein handelsüblicher 3D-Drucker ist nur fähig Überhänge mit bis zu 60° Neigung sicher zu drucken.

Daher ist es sinnvoll, den Würfel in zwei circa gleich großen Hälften zu drucken, weil so zum einen das Problem des Druckes mit Überhang als auch das Problem der Platzierung der Komponenten im Statuswürfel geschickt und problemlos umgangen werden kann.

3.5 Anforderungen an den Anzeigebildschirm

Das E-Ink Display kann benutzt werden, um jedes seiner Pixel einzeln rot, weiß oder schwarz erscheinen zu lassen. Grundsätzlich soll das E-Ink-Display den Rotationszustand des Statuswürfels übermittelt bekommen, um so abhängig davon das Display zu verändern. Das Ziel ist schließlich mit den Daten der API den Beschäftigungsstatus auf dem E-Ink-Display darzustellen.

Um dies zu ermöglichen sollte ein ESP32 an das E-Ink-Display angeschlossen werden. Dieser ist dann in der Lage das E-Ink-Display beliebig zu verändern. Des Weiteren kann der ESP32 eine WLAN-Verbindung herstellen und so mittels http-Get-Requests alle nötigen Daten von der API anfordern. Die API ist dabei ein ständig aktiver und adressierbarer Server, während der ESP32 eine energieeffiziente Maschine sein soll, welche nicht ständig aktiv ist. Daher kann die API nicht immer den ESP32 adressieren und erreichen. Der ESP32 jedoch kann jederzeit die API adressieren und mit ihr kommunizieren, da diese durchgehend aktiviert ist.

Um seine Funktion zu erfüllen muss der ESP32, angeschlossen an das E-Ink-Display, zuerst von der API den derzeitigen Rotationszustand des Statuswürfels anfordern. Danach muss je nach empfangenem Rotationszustand das E-Ink-Display eine Statusmeldung anzeigen. Alle verschiedenen Statusmeldungen sind in der nachfolgenden Abbildung 7 grafisch dargestellt.



Abbildung 7: Entwürfe für die Bildschirme des E-Ink-Displays

Die Abbildung 7 stellt alle vier Vorlagen für die Zustände des E-Ink-Displays dar. Links oben ist der Zustand "6" dargestellt. Das E-Ink-Display soll einen schwarzen Hintergrund mit einer dicken weißen Schrift in der Mitte, "DEAKTIVIERT!", zeigen. Das Besondere an diesem Zustand ist, dass sich hier der ESP32, wie auch beim Statuswürfel des gleichen Zustandes, immer kurz in einen "deepsleep" begeben soll und bei dem Status "6" soll auch hier der "deepsleep" mit fünf Minuten statt sonst wie üblich fünf Sekunden deutlich länger sein.

Rechts daneben in der Abbildung 7 ist eine Vorlage für den Status "1" abgebildet. Das E-Ink-Display soll hier die aktuelle Temperatur, die Uhrzeit der letzten Änderung sowie einen kleinen Satz zur derzeitigen Temperatur projizieren. Dazu muss bei dem Status "1" auch bei nur einer Temperaturänderung das E-Ink-Display den Text neu projizieren. Zudem soll der Satz, welcher zentral angezeigt wird, auch veränderbar sein und somit sowohl in der API gespeichert, als auch von dem ESP32 angefordert werden.

Überdies zeigt die Abbildung 7 links unten die Darstellung des Status "2". Diese soll lediglich aus einem Bild bestehen, welches auch in der API gespeichert werden und von dem ESP32 nur angefordert werden soll.

Zuletzt stellt die Abbildung 7 rechts unten beispielsweise eine E-Ink-Display-Vorlage dar, welche von dem Status "3", "4" oder "5" initiiert werden sollen. Hierbei steht links oben eine Temperatur sowie rechts oben die Zeit zum Zeitpunkt der letzten Änderung. Mittig stehen ein von der API bereitgestellter Text, welcher einmal im oberen Teil eine Zustandsbeschreibung sein soll und einmal im unteren Teil eine Anweisung an die lesende Person. Die Anweisung soll zudem von einem roten Rechteck eingerahmt sein, um ihre Wichtigkeit zu unterstreichen.

Um die Richtigkeit des von dem E-Ink-Display dargestellten Status zu gewährleisten, sollte mindestens alle 5 Sekunden geprüft werden, ob die abgebildete Darstellung noch aktuell ist.

3.6 Anforderungen an die API

Die API stellt eine Verbindungsstelle zwischen dem Statuswürfel, dem E-Ink-Display und der Website dar. Wie schon angedeutet, werden zu ihr Daten übersandt. Optimalerweise geschieht dies über WLAN oder Lan, welches an der API und den beiden ESP32 anliegen muss. Damit diese Technik funktionieren kann, müssen sich alle Geräte in einem Netzwerk befinden, über welches die lokale IP-Adresse der API bekannt ist, damit die zwei ESP32 mit der API kommunizieren können und die API auf einem gewöhnlichem Windows Rechner laufen kann.

Die API muss zudem einige Funktionen erfüllen. Zuallererst sollte sie ständig erreichbar sein, damit ihr sowohl der Statuswürfel, das E-Ink-Display als auch die Website ständig Daten senden oder von ihr anfordern können. Spezifiziert werden nun auch die Daten, welche die API speichern sollte. Der Grund der Speicherung ist die mehrfache Nutzung der Daten zwischen den einzelnen Komponenten mit denen die API kommuniziert.

Daher sollte die API den derzeit vom MPU-6050 gemessenen Status sowie die Temperatur festgehalten werden, um die von dem Statuswürfel bestimmten Daten zu speichern, bis neue ermittelt werden. Weiterhin sollten für das E-Ink-Display ebenfalls Temperatur und Status gespeichert werden, jedoch zusätzlich ein Bild, ein Text, welcher je nach Status wie in der Tabelle 2 gezeigt verändert und generell auch im Rahmen der Website anpassbar ist, sowie für die Temperatur eine Einschätzung, ob diese kalt, kühl, angenehm, heiß oder sehr heiß ist. Je

nach Einschätzung der Temperatur soll zudem ein passender Text an das E-Ink-Display weitergegeben werden.

Rotationszustand	Anweisung bzw. Text dazu
1	Wetterseite / Bildschirmschoner
2	hier wird ein voreingestelltes Bild angezeigt
3	Es ist frei, bitte hereintreten
4	In einer Besprechung, bitte nicht stören
5	Gerade intensiv am Arbeiten, bitte nicht stören
6	DEAKTIVIERT

Tabelle 2: Zuordnungen zwischen Rotationszuständen und Texten

3.7 Anforderungen an die Website

Die Website hat eine besondere Funktion in dem Netzwerk, da auch ohne sie der Betrieb des Systems funktionieren soll. Jedoch soll sie dem Nutzer einige nützliche Funktionen ermöglichen, welche die Funktionalität des Statuswürfels entweder erweitern, ändern oder vereinfachen.

In erster Linie soll die Website die Funktion des E-Ink-Displays übernehmen können, indem sie ähnlich zum E-Ink-Display entsprechende Daten wie den Status anfordert und darstellt. Darüber hinaus soll dem Nutzer über die Website die Möglichkeit geboten werden, vorprogrammierte Texte zu verändern. Dazu zählen die Texte "DEAKTIVIERT!" des Status "6", "In einer Besprechung" gefolgt von "Bitte nicht eintreten!" des Status "3" oder auch die Temperaturtexte wie zum Beispiel "Es ist heute angenehm" oder "Es ist kühl heute". Durch diese Funktionen der Website soll eine Personalisierung des Statuswürfels vereinfacht möglich sein.

4 Durchführung

4.1 Der Statuswürfel

Der Statuswürfel besteht grob gesagt aus einem ESP32 D1-Mini in seinem Zentrum, welcher mittels des Sensors MPU-6050 seinen eigenen Rotationszustand auslesen und folglich die ermittelten Rotationsdaten der API übersenden soll. Alle unter dem Kapitel abgebildeten Dateien sind im Anhang im Unterordner "Statuswürfel" vollständig aufzufinden.

4.1.1 Auslesung der Daten des MPU-6050

Um die Sensordaten des MPU-6050 zu erhalten, wurde der folgende in Abbildung 8 verkürzt dargestellte Code mittels Micro-Python geschrieben. Das vollständige Programm liegt unter dem Namen "mpu6050.py" im Anhang anbei.

```
1 class accel():
2     ...
3     def get_raw_values(self):
4         self.iic.start()
5         a = self.iic.readfrom_mem(0x68, 0x3B, 14)
6         self.iic.stop()
7         return a
8
9     def bytes_toint(self, firstbyte, secondbyte):
10        if not firstbyte & 0x80:
11            return firstbyte << 8 | secondbyte
12        return - (((firstbyte ^ 255) << 8) | (secondbyte ^ 255)+1)
13
14     def get_values(self):
15         raw_ints = self.get_raw_values()
16         vals = {}
17         vals["AcX"] = self.bytes_toint(raw_ints[0], raw_ints[1])
18         vals["AcY"] = self.bytes_toint(raw_ints[2], raw_ints[3])
19         vals["AcZ"] = self.bytes_toint(raw_ints[4], raw_ints[5])
20         vals["Tmp"] = self.bytes_toint(raw_ints[6], raw_ints[7])
21             / 340.00 + 36.53
22         ...
23         return vals
```

Abbildung 8: Ausschnitt der Datei "mpu6050.py"

Um das Programm zu verstehen, wird eine strukturelle Abwärtsanalyse vorgenommen. Als Erstes wird in Zeile 3 die Funktion „`self.get_raw_values(self)`“ definiert. Diese initialisiert das I²C-Protokoll, um mit diesem den MPU-6050 in Zeile 5 mittels der Adresse „`0x68`“ zu adressieren, sowie von der Speicherposition „`0x3B`“ des MPU-6050 „`14`“ Bytes auszulesen, welche dann alle durch die temporären Liste „`a`“ in Zeile 7 zurückgegeben werden.

Die darauf folgende Funktion „`self.bytes_toint(self, firstbyte, secondbyte)`“ soll zwei Bytes einlesen, welche zusammen eine Zahl im Zweierkomplement bilden und daraus eine Zahl im Einerkomplement, welche besser lesbar ist, formen. Die Umwandlung erfolgt folgendermaßen: Wenn das erste Bit eine 0 ist, dann werden beide Bytes schlicht aneinandergefügt und bilden eine Zahl. Wenn das erste Bit jedoch eine 1 ist, werden beide Bytes invertiert, zusammengefügt und zudem mit 1 addiert, sowie folglich mit -1 multipliziert, da die 0 nicht berücksichtigt werden muss. Das Ergebnis ist, dass zwei Bytes zu einer Zahl von -32.768 bis 32.767 umgewandelt werden. [46]

Die Funktion, welche letztlich die Sensordaten zurückgeben soll, ist die in Zeile 14 definierte „`get_values(self)`“ Funktion. In dieser werden zuerst die Rohmesswerte, welche durch die Funktion „`self.get_raw_values()`“ erhalten werden, in der Variable „`raw_ints`“ gespeichert. Zwei der 14 Bytes liefern je einen Messwert. Die ersten sechs Bytes ergeben die drei Beschleunigungswerte und die zwei folgenden sind für die Temperatur relevant. Die Restlichen sind nur für den im MPU-6050 verbauten Gyrosensor bzw. das Gyrometer zuständig. Daher werden auch je zwei Bytes, welche zusammen im Zweierkomplement formatiert sind, gemeinsam mit der Funktion „`self.bytes_toint(self, firstbyte, secondbyte)`“ in ein Einerkomplement formatiert. Diese Einerkomplemente werden dann in der Liste „`vals`“ unter den Bezeichnungen „`AcX`“, „`AcY`“, „`AcZ`“ und „`Tmp`“ gespeichert und über einer Liste zurückgegeben.

Somit kann mit der Funktion „`get_values(self)`“ eine Messreihe in Form einer Liste des Sensors ausgelesen werden.

4.1.2 Integrierung der I²C Übertragung

Das I²C-Protokoll ist wie schon erwähnt in "MicroPython" integriert, weshalb die Umsetzung erleichtert verläuft. In den folgenden Abbildungen 9 und 10 ist die Integrierung des I²C-Protokolls, um mit dem MEMS MPU-6050 zu kommunizieren, in den Dateien "main.py" und "mpu6050.py" verkürzt dargestellt.

```

1 from machine import SoftI2C, Pin
2 import mpu6050
3
4 i2c = SoftI2C( scl=Pin(22), sda=Pin(21), freq=100000)
5 mpu= mpu6050.accel(i2c)
6 ...

```

Abbildung 9: Ausschnitt der Datei "main.py"

In der Abbildung 9 wird zunächst die Integration des I²C-Protokolls in die Datei "main.py" des Statuswürfels gezeigt. Zuerst wird von dem Modul "machine" "SoftI2C" importiert. "SoftI2C" verlangt dabei nach einem Pin für die "serial clock" und einem Pin für die "serial data". Die Pinbezeichnungen wurden bereits in Abbildung 2 visualisiert. Des Weiteren fordert "SoftI2C" eine Frequenz, hier 100.000Hz, mit welcher die Übertragung durchgeführt werden soll. "SoftI2C" wird zudem in der Variable "i2c" gespeichert, welche dann in Zeile 5 eine in der Datei "mpu6050.py" abgespeicherte Klasse initialisiert und diese Klasse unter der Variable "mpu" speichert, welche später relevant wird. [47]

```

1 class accel():
2     def __init__(self, i2c, addr=0x68):
3         self.iic = i2c
4         self.addr = addr
5         self.iic.start()
6         self.iic.writeto(self.addr, bytearray([107, 0]))
7         self.iic.stop()
8 ...

```

Abbildung 10: Ausschnitt der Datei "mpu6050.py"

Diese Initialisierung der Klasse "accel" ist in Abbildung 10 dargestellt. Dabei wird über der Variable "i2c", bzw. "iic" die "SoftI2C" Funktion, in welcher zudem die Pin-Informationen gespeichert sind, übertragen. Unter "addr" wird die Adresse des MPU-6050 gespeichert. Daraufhin wird in Zeile 5 die "i2c"-Übertragungsmethode gestartet. In Zeile 6 werden mit der "iic.writeto"-Funktion dem durch "addr" adressierten Gerät die Bytes "107" und "0" übertragen. Diese beiden Bytes führen zum Aufwachen des MPU-6050, damit von diesem später Rotationsdaten ausgelesen werden können.

4.1.3 Auswertung der Daten

Die nachfolgende Abbildung 11 projiziert einen verkürzten Ausschnitt aus der Datei "main.py". Die Funktion "getAnswer()" soll zunächst den Zustand des Statuswürfels auslesen, wie es die Abbildung 11 zeigt.

```
1 def getAnswer():
2     wert1 = mpu.get_values()
3     wert2 = mpu.get_values()
4     wert3 = mpu.get_values()
5     x = round((wert1["AcX"]+wert2["AcX"]+wert3["AcX"])/16384/3, 2)
6     y = round((wert1["AcY"]+wert2["AcY"]+wert3["AcY"])/16384/3, 2)
7     z = round((wert1["AcZ"]+wert2["AcZ"]+wert3["AcZ"])/16384/3, 2)
8     Tmp = round(2*(wert1["Tmp"]+wert2["Tmp"]+wert3["Tmp"])/3, 0)/2
9
10    if x >= 0.75 and -0.25 < y < 0.25 and -0.25 < z < 0.25:
11        ausgabewert = "1" # oben ist oben
12    elif x <= -0.75 and -0.25 < y < 0.25 and -0.25 < z < 0.25:
13        ausgabewert = "6" # unten ist oben
14    elif z <= -0.75 and -0.25 < y < 0.25 and -0.25 < x < 0.25:
15        ausgabewert = "4" # links ist oben
16    elif z >= 0.75 and -0.25 < y < 0.25 and -0.25 < x < 0.25:
17        ausgabewert = "3" # rechts ist oben
18    elif y <= -0.75 and -0.25 < x < 0.25 and -0.25 < z < 0.25:
19        ausgabewert = "5" # hinten ist oben
20    elif y >= 0.75 and -0.25 < x < 0.25 and -0.25 < z < 0.25:
21        ausgabewert = "2" # vorne ist oben
22    else:
23        return None #nichts Konkretes ist oben
24    ...
```

Abbildung 11: Ausschnitt der Datei "main.py" des Statuswürfels

In der Abbildung 11 werden zuerst, wie auch schon unter der Teilüberschrift "4.1.1 Auslesung der Daten des MPU-6050" erklärt, Messreihen des Sensors erstellt und in den Variablen "wert1", "wert2" und "wert3" zwischengespeichert. Aus diesen Listen der Messreihen werden dann eine Beschleunigung in x-, y-, und z-Richtung sowie eine Temperatur durch die Bildung eines Mittelwertes aus den jeweils 3 Einzelwerten bestimmt. Zudem wird bei der Beschleunigung der Messwert durch 16.384 geteilt, da das ungefähr der Betrag der Hälfte des Maximalwertes 32.767 und Minimalwertes -32.768 ist und die Erdbeschleunigung g ungefähr die Hälfte des Maximalwertes sein muss. Zudem werden die Beschleuni-

gungen auf eine Nachkommastelle und die Temperatur auf Komma fünf gerundet.

Nach diesen Operationen sollten „ x “, „ y “ und „ z “ einen Wert zwischen -2 und 2 annehmen. Dabei hätte die Erdbeschleunigung g einen Betrag von 1. In der Konzeption wurde zudem errechnet, dass die Fehlerspanne ca. 0,25 betragen sollte, damit bei einem sehr schiefem Untergrund von bis zu 40° oder einem gravierendem Messfehler durch Schütteln oder Ähnlichem der Rotationszustand noch sicher bestimmt werden kann. Um einen Rotationszustand zu erkennen, müsste auf einer Achse die Beschleunigung $a \approx \pm g \approx \pm 1$ und auf den anderen beiden Achsen müsste $a \approx 0$ betragen. Diese Abstufung wird auch in Abbildung 11 ab Zeile 10 in einer „*if*“-Schleife ausgeführt, sodass bei jedem dieser Möglichkeiten der Variable „*ausgabewert*“ eine Zahl zwischen 1 und 6 zugeordnet wird. Falls der Rotationszustand jedoch nicht ermittelt werden konnte, so wird, wie in Zeile 22 ersichtlich, „*None*“ zurückgegeben.

```
1 def getAnswer():
2     ...
3     database = helper.load_data("RotInfo.dat")
4     if database["Zustand"] == ausgabewert:
5         if ausgabewert == "1": # Temperatur nur bei 1 übertragen
6             if Tmp != database["Temperatur"]:
7                 database["Temperatur"] = Tmp
8                 helper.save_data(database, "RotInfo.dat")
9                 return ausgabewert, Tmp
10    else:
11        database["Zustand"] = ausgabewert
12        database["Temperatur"] = Tmp
13        helper.save_data(database, "RotInfo.dat")
14        return ausgabewert, Tmp
15    return None
```

Abbildung 12: Ausschnitt der Datei „main.py“ des Statuswürfels

In Abbildung 12 kann eine Fortführung der Funktion aus Abbildung 11 erkannt werden, denn in der gleichen Funktion wird der erfasste Rotationszustand zudem ausgewertet. Dazu wird zuerst die Datei „RotInfo.dat“, welche alte Rotationsdaten in einem json-Dateiformat speichert, mittels einer „helper.py“-Datei geöffnet und dessen Inhalt in der Variable „*database*“ gespeichert. Der relevante Inhalt der „helper.py“ ist dabei in Abbildung 13 zu finden.

In „*database*“ sind im Stil einer json-Datei zwei Daten, die Temperatur und der (Rotations-)Zustand, gespeichert. Nun wird der derzeit gemessene Rotationszu-

stand mit dem in der "database" vorliegenden Zustand verglichen und wenn diese nicht gleich sind, dann sollen die neue Temperatur und der neue Rotationszustand sowohl zurückgegeben, als auch in der "RotInfo.dat" abgespeichert werden. Dadurch wäre bei der nächsten Messung des Rotationszustandes des Statuswürfels dieser Zustand gleich dem abgespeicherten, alten Zustand.

Wenn der in "RotInfo.dat" abgespeicherte Rotationszustand nun gleich dem ermitteltem Zustand ist, dann soll "None" zurückgegeben werden und die abgespeicherten Daten sollen unverändert bleiben, es sei denn der gemessene Rotationszustand entspricht der "1". In diesem Fall wird geprüft, ob sich die vom Sensor erfasste Temperatur mit der in "RotInfo.dat" abgespeicherten unterscheidet. Wenn dies der Fall sein sollte, dann wird nicht "None" zurückgegeben, sondern es werden der gemessene Rotationsstatus und die gemessene Temperatur zurückgegeben. Die neue Temperatur wird zudem in der Datei "RotInfo.dat" gespeichert. Der Rotationszustand muss nicht gespeichert werden, da sich dieser nicht verändert hat.

```

1 def save_data(data, name):
2     f=open(name,"w") # opens a file for writing
3     f.write(ujson.dumps(data))
4     f.close()
5
6 def load_data(name):
7     f=open(name,"r") # opens a file for reading
8     data = ujson.load(f)
9     f.close()
10    return data
11
12 def do_connect():
13     import network
14     from config import ssid, pw
15     WLAN = network.WLAN(network.STA_IF)
16     if not WLAN.isconnected():
17         WLAN.active(True)
18         WLAN.connect(ssid, pw)
19         while not WLAN.isconnected():
20             pass

```

Abbildung 13: Ausschnitt der Datei "helper.py" des Statuswürfels

4.1.4 Übersenden der Daten an die API

Der durch Abbildung 14 repräsentierte Code schließt das Programm "main.py" des Statuswürfels ab. Zuerst wird durch die Rückgabe, der in den Abbildungen 11 und 12 erklärten Funktion "getAnswer()", die Variable "Antwort" definiert. Wenn diese Antwort "None" ist, dann soll nichts passieren und der ESP32 geht in den energiesparenden "deepsleep" für fünf Sekunden.

```
1 Antwort = getAnswer() # Antwort = Rotationszustand
2 if Antwort != None:
3     helper.do_connect()
4     import config
5     url1 = config.Link + "/stat/" + str(Antwort[0])
6     url2 = config.Link + "/tempstat/" + str(Antwort[1])
7     urequests.post(url1)
8     urequests.post(url2)
9     if Antwort[0] == "6":
10         Zeit = 300000 # = 300s = 5 min
11     else:
12         Zeit = 5000
13     deepsleep(Zeit)
14 deepsleep(5000)
```

Abbildung 14: Ausschnitt der Datei "main.py" des Statuswürfels

Wenn jedoch die Antwort ungleich "None" ist, dann soll sich der ESP32 mittels einer Funktion "do_connect()" aus "helper.py", wie in Abbildung 13 ersichtlich, mit einem WLAN verbinden. Dazu wird die Bibliothek "network" verwendet, welche alle dazu nötigen Funktionen bereitstellt. In Zeile 16 der Abbildung 13 wird zunächst geprüft, ob eine Verbindung besteht. Wenn dies nicht der Fall ist, dann wird mittels der in "config.py", in Abbildung 15 sichtbar, gespeicherten Variablen die SSID und das Passwort in der Funktion "connect(ssid, password)" verwendet um sich mit dem Netzwerk zu verbinden. Darauf wird in Zeile 19f noch gewartet, bis die Funktion abgeschlossen ist, bzw. Verbindung besteht. [48]

```
1 ssid = 'Netzwerkname'
2 pw = 'Passwort'
3 Link ="http://192.168.1.190:8000"
```

Abbildung 15: "config.py" des Statuswürfels

Verbunden mit einem WLAN kann nun die Übertragung der neuen Daten an die API beginnen. Dazu führt der Statuswürfel zwei http-POST-Requests (mittels der "urequests"-Datei) aus, welche jeweils aus drei Teilen bestehen. Der erste Teil ist der in "config.py" gespeicherte "*Link*", welcher den Server, bzw. die API adressiert. Das kann zum Beispiel die lokale IP-Adresse zusammen mit einem zugewiesenen Port sein, über welche die API auffindbar ist. Darauf folgt, getrennt mit einem "/" eine Information darüber, welche Daten übersandt werden. In diesem Fall steht "*stat*" für (Rotations-)Status und "*tempstat*" für Temperatur-Status, bzw. nur Temperatur. Zuletzt folgt nach einem weiterem "/" die Information, welcher Status neu ist bzw. übergeben werden soll. Diese Funktionen der API sind auch in Tabelle 3 ersichtlich. [49]

Nachdem diese http-Push-Requests in Zeile 7f der Abbildung 14 übersandt wurden, soll die Dauer des "deepsleep" festgelegt werden. Dieser soll fünf Sekunden lang andauern, es sei denn der Zustand "6", welcher den Statuswürfel energieeffizient in einen Ruhemodus versetzen soll, ist aktiviert. In diesem Fall soll der "deepsleep" fünf Minuten lang andauern.

Nach dem "deepsleep" startet sich der Mikro-Controller automatisch neu, wodurch zuerst die "boot.py" ausgeführt wird, welche leer ist, gefolgt von der Datei "main.py". Daher ist der ESP32 in einer ständigen Schleife gefangen, in welcher er den Sensor ausliest und neue Daten an die API sendet.

Die Datei "config.py", welche auch in Abbildung 15 wiedergegeben ist, soll einzelne veränderbare Daten, wie den Link speichern, damit diese im Nachhinein einfach in dieser Datei geändert werden können, anstatt die exakte Stelle der Verwendung in dem "MicroPython"-Code zu suchen.

4.1.5 Genauigkeit der Messungen

Die Genauigkeit der Messungen des MPU-6050 ist für die Funktionsweise des Statuswürfels essenziell. In der Durchführung wird von drei Werten der Mittelwert gebildet, um sicher einen eindeutigen Status zu erhalten. Diese Mitteilung aus drei Werten geschieht, da ein falsches Messergebnis oder eine klare Fehl-messung nicht das Ergebnis beeinflussen sollte. Zusätzlich ist bereits eine große Messtoleranz für die einzelnen Status implementiert, jedoch nicht nur für einen Messfehler, sondern auch für den Fall eines ungeraden Untergrundes und eine Messung im Wackeln, da der Sensor die Beschleunigung misst. Daher soll nun untersucht werden, ob auch ein signifikanter Messfehler in der Datenauswertung berücksichtigt werden müsste.

Um die Genauigkeit des Sensors zu ermitteln, sollten möglichst viele Messungen bei gleichen Bedingungen gemacht und daraufhin festgestellt werden, wie hoch die durchschnittliche Abweichung und die maximale Abweichung ist. Dabei wird zuerst festgestellt, dass die beschriebenen Messungen eine x-Beschleunigung, eine y-Beschleunigung und eine z-Beschleunigung darstellen. Diese werden von 3 gleichen, jedoch in der Achse gedrehten Sensoren gemessen. Da die Sensoren die gleichen sind, ist es nur nötig in einer Bewegungsrichtung den Messfehler zu messen.

Dazu wurde ein neues Programm mittels Micro-Python geschrieben, welches in Abbildung 16 dargestellt ist. Das in Abbildung 16 sichtbare Programm beschreibt eine ähnliche Datenerhebung wie es zu den Programmen in Abbildungen 8, 9 und 11 schon erklärt wurde. Zuerst werden in Abbildung 16 bis Zeile 8 für den ESP32 und den MPU-6050 Metadaten definiert. Danach wird eine leere Liste (Zeile 10) erstellt, welcher nun in einer Schleife 1.000 Mal ein gemessener Wert der Beschleunigung in x-Achsen-Richtung angefügt wird. Schließlich wird die Liste, nun bestehend aus 1.000 Elementen ausgegeben.

```
1 import time
2 from machine import SoftI2C
3 from machine import Pin
4 import MPU-6050
5
6 i2c = SoftI2C( scl=Pin(22), sda=Pin(21), freq=100000)
7 #initializing the I2C method for ESP32
8 mpu= MPU-6050.accel(i2c)
9
10 Liste = []
11
12 for i in range(1000):
13     xwert = mpu.get_values()["AcX"]
14     wert = xwert/16384
15     Liste.append(wert)
16
17 print(Liste)
```

Abbildung 16: Micro-Python Code für das Aufnehmen der Messwerte

Diese Messreihe, welcher im Anhang im Ordner "Schriftlicher Teil" aufzufinden ist, kann daraufhin in ein Tabellenkalkulationsprogramm eingegeben werden. Hier wurde Geogebra verwendet, welches die Messungen in ein Boxplot-Diagramm umwandelt. Dieses ist in der nachfolgenden Abbildung 17 abgebildet. Ein Boxplot Diagramm ist ein Diagrammtyp, welches häufig zur Datenauswertung genutzt wird, da er die Häufigkeit von Datenmengen anschaulich darstellt. Bei der Darstellung steht eine Box in der Mitte repräsentativ für 50% der Messwerte und der zur Skala senkrechte Strich in der Mitte der Box stellt den Median der Messwerte dar. Überdies stellen die einem "T" ähnlichen Ausstülpungen aus der zentralen Box je weitere 25% der Messwerte dar. Schließlich geben einige Punkte abseits des zentralen Grafen Ausreißer der Messreihe an.

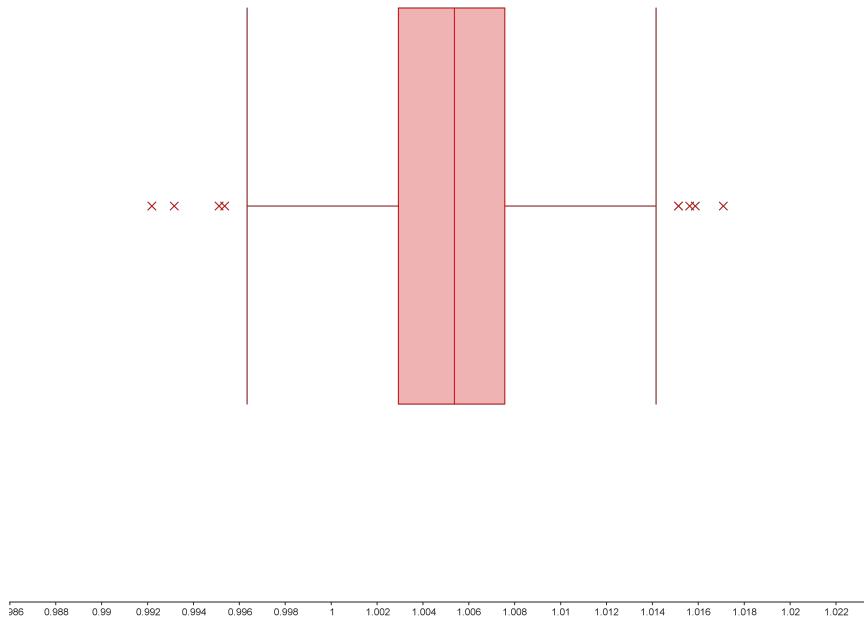


Abbildung 17: Boxplot-Diagramm der Messreihe

In den hier vorliegenden Daten, dargestellt in Abbildung 17, befindet sich der Median bei ungefähr 1,0055. Das ist wahrscheinlich der in diesem Fall ungefähr richtige Messwert. Die mittleren 50% liegen zwischen 1,003 und 1,0075. Das heißt, zu 50% liegt der Fehlerbereich bei nur $\frac{|1,003 - 1,0055|}{1,0055} \approx 0,25\%$ bzw. $\frac{|1,0075 - 1,0055|}{1,0055} \approx 0,2\%$. Die unteren 25% der Messwerte reichen bis 0,9965 und die oberen 25% reichen bis 1,014. Daher liegt der Fehlerbereich bei fast 100% der Messungen

bei $\frac{|0,9965 - 1,0055|}{1,0055} \approx 0,9\%$ bzw. $\frac{|1,014 - 1,0055|}{1,0055} \approx 0,85\%$. Der größte Ausreißer nach unten ist bei 0,992 und der größte Ausreißer nach oben liegt bei 1,017. Damit liegt der maximal Messfehler von 1.000 Messungen bei $\frac{|0,992 - 1,0055|}{1,0055} \approx 1,3\%$ bzw. $\frac{|1,017 - 1,0055|}{1,0055} \approx 1,1\%$.

Es lässt sich zuerst feststellen, dass der Fehler nach oben generell geringer ist als der Fehler nach unten. Des Weiteren liegt der gemessene Median-Wert nicht bei exakt 1, sondern knapp darüber. Das könnte an Konstruktionsfehlern des Sensors MPU-6050 liegen, aber auch daran, dass die Erdbeschleunigung auf der Erde nicht überall gleich ist, sondern an den Polen circa 0,5% größer ist. Der gemessene Median-Wert liegt nur $\frac{|1,0055 - 1|}{1} \approx 0,5\%$ über dem theoretischen Optimalwert. Somit ist der Median ein realistischer gemessener Wert der Beschleunigung. Zuletzt sollte noch dokumentiert werden, dass unter 1.000 Messungen der maximale Fehler bei gerade einmal 1,3% liegt. Somit ist der unter 1.000 Messungen registrierte maximale Fehler nur 2,5 mal so groß wie der reale Unterschied der Erdbeschleunigung zwischen den Polen und dem Äquator. Daher kann festgestellt werden, dass der Sensor MPU-6050 sehr genau die Beschleunigung misst und der Fehler in der Verarbeitung des Messwertes kaum bis gar nicht berücksichtigt werden muss, besonders, da schon eine Abweichung von 25% integriert ist.

4.1.6 Modellieren

In den nachfolgenden Abbildungen 18 und 19 ist ein Modell des Statuswürfels in TinkerCAD zu erkennen, bei welchen zudem relevante Stellen mit den Nummern 1 bis 9 markiert wurden. Eine auf den Abbildungen 18 und 19 erkennbare Skaleneinheit des Bodens, welcher durch die schlechte Auflösung schwer erkennbar ist, steht für einen Millimeter. In Abbildung 20 sind zudem die zu beachtenden Komponenten, welche in den Statuswürfel eingebaut werden müssen, dargestellt. Des Weiteren kann das Modell unter einem im Anhang im Ordner Statuswürfel zu findenden Link aufgerufen werden.

Die Zahl 1 in den Abbildungen 18 und 19 markiert zuerst die Position, in welcher der ESP32, sichtbar in Abbildung 20, platziert werden soll. In Abbildung 19 können dazu ein Podest sowie links und rechts daneben kleine Einkerbungen erkannt werden, in welche die unten herausragenden Stifte der Pins des ESP32 hereinragen können. Der Graben musste dafür 0,5cm tief sein und, da der Boden insgesamt nur 0,5cm dick ist, musste ein Podest für den ESP32 modelliert werden, sodass Podest und Graben zusammen eine Höhe von 0,5cm bilden. In Abbildung 18 ist jedoch die Sicht auf den Podest durch die vordere Seitenwand

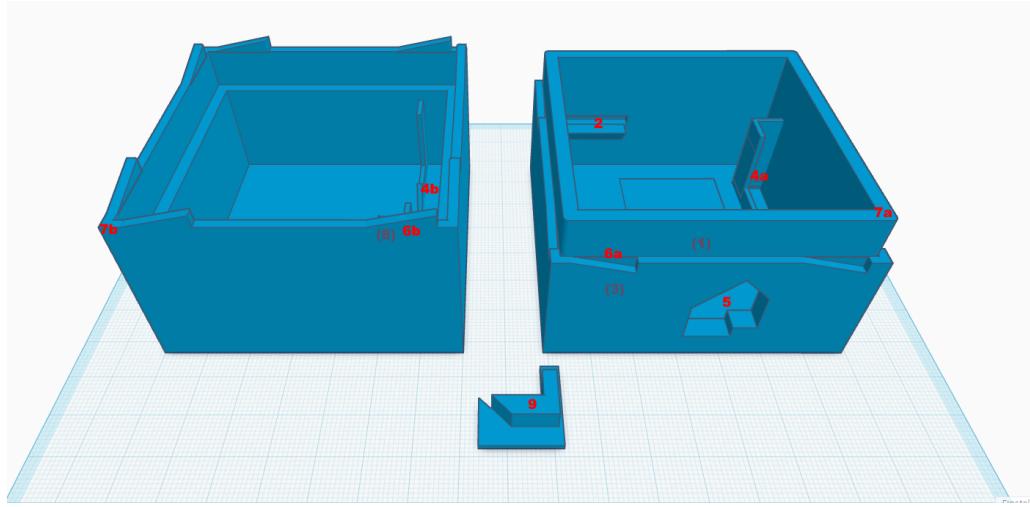


Abbildung 18: Modell des Statuswürfels in TinkerCAD

verdeckt. Immer wenn dies geschieht, ist die Zahl für die Nummer zum einen in Klammern und zum anderen teils transparent geschrieben.

Die Zahl 2 in den Abbildungen 18 und 19 kennzeichnet den Ort, in welchen der MPU-6050, sichtbar in Abbildung 20, eingefügt werden soll. Dieser zeichnet sich durch einen Überhang mit einer Breite von 4mm und einer Höhe von mindestens 6mm aus. Dadurch haben die nach unten zeigenden Stifte freien Entfaltungsraum. Die Stifte haben zudem eine Ausdehnung in posteriore Richtung des MPU-6050 von ca. 2mm. Daher wurde eine Kuhle modelliert, welche eine Tiefe von 3mm besitzt. Es kann zudem angemerkt werden, dass die Neigungen der Schrägen sowohl beim Überhang, als auch bei der Kuhle ungefähr 45° betragen, damit die Schrägen gedruckt werden können.

Die Zahl 3 ist die Einkerbung für einen Knopf, welcher auch in der Abbildung 20 sichtbar ist. Der Knopf selber hat dabei einen Durchmesser von 4mm, bei einer Höhe von 1mm und das System mit Kasten dahinter hat einen Durchmesser von fast 7mm. Diese Maße wurden dabei in das Modell eingearbeitet. Es musste nur beachtet werden, dass der Knopf von außen sichtbar und bedienbar ist, jedoch nicht herausragt und somit nicht unbeabsichtigt gedrückt wird.

Die Zahl 4 in den Abbildungen 18 und 19 markiert den Ort für den Akkumulator, welcher in Abbildung 20 beobachtet werden kann. Die Unterteilung in 4a und 4b zeigt bloß, dass dieses Element, wie hier der Akkumulator, in beiden Einzelteilen des Modells berücksichtigt werden musste. Der Akkumulator besitzt die Ausmaße 0,5cm x 6cm x 5cm und am Kabelursprung sogar eine Länge von fast 6,5cm. Um Toleranz zu erschaffen wurde der Boden bei 4a um 1mm abgesenkt,

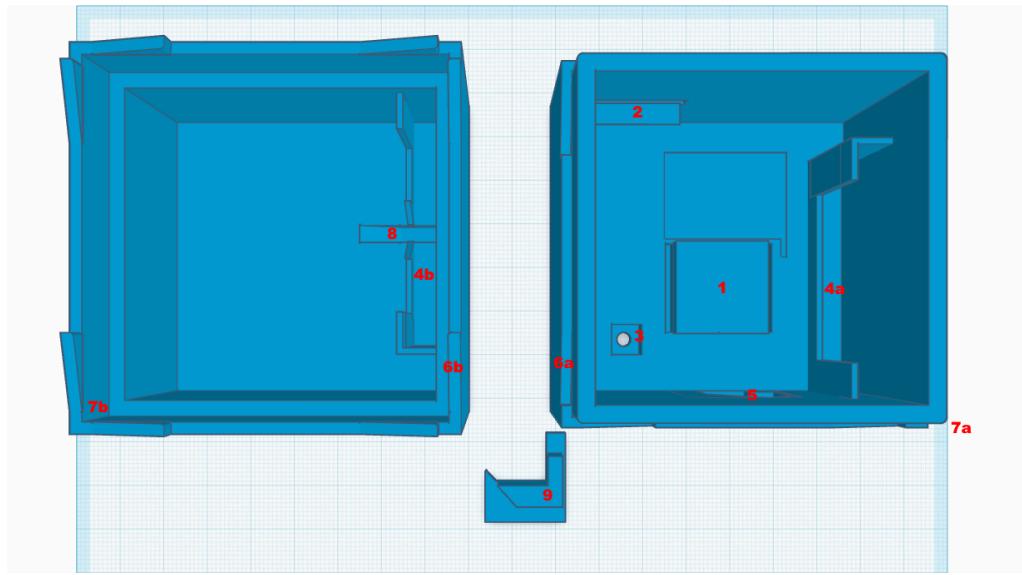


Abbildung 19: Modell des Statuswürfels in TinkerCAD von oben

damit eine Höhe von 6,1cm entsteht. Zudem ist der Akkumulator durch 1mm dicke Plastikwände an seinem Platz gesichert.

Dazu zeigt die Zahl 8 das Modell für die Einkerbung des Kabelursprunges an dem Akkumulator. Diese Einkerbung ist 3mm tief, damit das Kabel derart gebogen werden kann, dass es durch die Einkerbung den ESP32, bzw. dessen "Batterie-Shield", erreichen kann.

Die Zahl 5 in den Abbildungen 18 und 19 zeigt die Öffnung des Statuswürfels für das Verbinden von Mikro-USB Kabeln mit dem ESP32 D1-Mini oder dem diesem anhängendem "Batterie-Shield", welche beide auch in Abbildung 20 dargestellt sind. Die Kabel dafür haben in der Regel eine Breite von 1cm, welche die Größe der Öffnung bestimmt. Zudem wurde wieder darauf geachtet, dass keine zu großen Überhänge modelliert werden, indem das obere Ende der Öffnung durch zwei Schrägen von ca. 45° abgeschlossen wird.

Die Zahl 9 zeigt dazu eine Kappe, welche in die Öffnung eingesetzt werden soll, damit kein Kabelzugang von Außen zum ESP32 besteht, sondern nur zu dem "Batterie-Shield", welcher zum Aufladen des Akkumulators benutzt werden soll. Das ist besonders für den Endnutzer relevant, welcher keinen Zugriff auf die Programme des ESP32 haben soll.

Durch die in 6a und 6b modellierten Ausstülpungen bzw. Einkerbungen ist sichergestellt, dass der Statuswürfel nicht falsch rotiert zusammengesteckt werden kann, da sonst die bei 6b herausragenden Ausstülpungen nicht in die bei 6a sichtbare Gegenform passen.

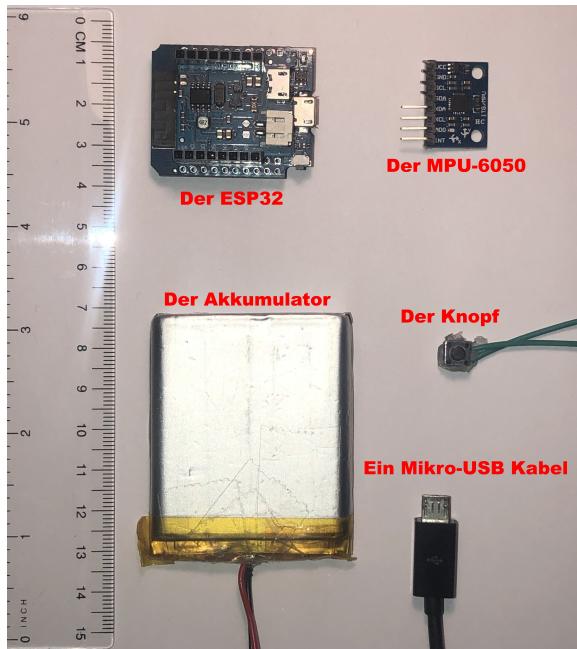


Abbildung 20: Die entscheidenden Komponenten des Statuswürfels

Zuletzt zeigt die Zahl 7a noch, dass die Kanten abgerundet sind, da sonst durch die Druckereigenschaften bei 7b eine Rundung, statt einer glatten Ecke, nach innen entsteht, wodurch das Zusammenstecken des Statuswürfels fast unmöglich wäre.

4.1.7 Zusammenbau

In den gedruckten Würfel wurden daraufhin die einzelnen Teile des Statuswürfels eingeklebt, wie auch in Abbildung 21 ersichtlich ist. In dieser Abbildung ist nochmals der ESP32 mit einer 1 markiert, der Sensor MPU-6050 mit einer 2, der Akkumulator mit einer 3 und zu guter Letzt ist die von außen erreichbare Reset-Taste mit einer 4 gekennzeichnet.

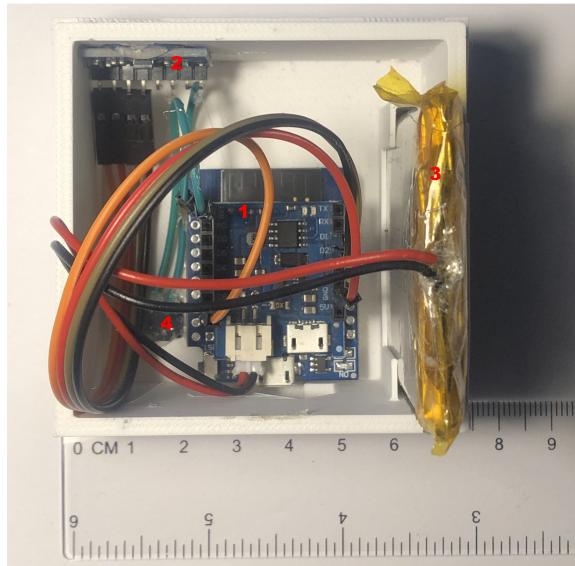


Abbildung 21: Der Statuswürfel mit eingebauten Komponenten

4.2 Das E-Ink-Display

Das E-Ink-Display ist mittels der 8 Anschlüsse, die "VCC", "GND", "DIN", "CLK", "CS", "DC", "RST" und "BUSY" benannt sind, mit den Pins des ESP32 der Bezeichnungen "3.3V", "GND", "IO27", "IO25", "IO32", "TDI", "IO4" und "IO0" verbunden.

Alle zu dem E-Ink-Display gehörenden Dateien sind im Anhang unter dem Ordner "E-Ink-Display" einsehbar. In der Dokumentation werden nur unvollständige und gekürzte Ausschnitte gezeigt.

Der in Abbildung 22 wiedergegebenen Code ist ähnlich zu dem für den Statuswürfel geschriebenen und in den Abbildungen 12 und 14 repräsentierten Code. Auch die hier benutzten Dateien "helper.py" und "config.py" fungieren sehr ähnlich zu den schon in den Abbildungen 13 und 15 dargestellten Dateien.

Zuerst verbindet sich der ESP32, wieder mittels der "helper.py", mit einem WLAN. Daraufhin werden mittels der "urequest.py", dort mittels der "usocket"-Bibliothek, zwei http-GET-Requests mit dem in "config.py" zu findenden Link, mit welcher die API adressiert wird, und jeweils der Spezifikationen "/statustext" oder "/tempstat", ausgeführt. Diese Requests fordern Daten von der API an und die erhaltenen Daten werden in den Variablen "newdata" sowie "newtemp" gespeichert. Die erhaltenen Daten liegen im json-Dateiformat vor, wobei "newdata" die Eigenschaften "text" sowie "status" und "newtemp" die Eigenschaften "Temperatur" sowie "TempZusatz" besitzt. Das Attribut "status" soll der aktuell ge-

```

1 helper.do_connect()
2 newdata = urequests.get(Link + "/statustext",
3                         headers = {'Content-Type':'text/json'}).json()
4 newtemp = urequests.get(Link + "/tempstat",
5                         headers = {'Content-Type':'text/json'}).json()
6 data=helper.load_data("data.dat")
7 temp = helper.load_data("Tempdata.dat")
8 if data["status"] != newdata["status"]:
9     data = newdata
10    temp = newtemp
11    NeuesBild(data, temp)
12 elif newdata["status"] == 1 and
13     temp["Temperatur"] != newtemp["Temperatur"]:
14     tempdata = tempdatenlage # falls sich nur die Temperatur ändert
15     NeuesBild(data, temp)
16 if newdata["status"] == 6:
17     machine.deepsleep(300000)
18 machine.deepsleep(5000)

```

Abbildung 22: Ausschnitt aus "main.py" des E-Ink-Displays

messene Rotationsstatus und das Attribut "*Temperatur*" die aktuell gemessene Temperatur sein. Die weiteren Eigenschaften werden für die Darstellung auf dem E-Ink-Display benötigt. [50, 49]

In Zeile 6f der Abbildung 22 werden die Inhalte der Dateien "data.dat" und "Tempdata.dat" in den Variablen "data" und "temp" gespeichert. Das sind die alten Daten.

In den restlichen Zeilen wird geprüft, ob sich das angezeigte Bild des E-Ink-Displays ändern soll. Das geschieht, wenn sich entweder die Eigenschaft "status" ändert oder wenn sich beim "status" "1" die Temperatur verändert hat. Die Änderung des Bildes wird in der Funktion "NeuesBild(data, temp)" umgesetzt. Dabei werden die neu erhaltenen Daten der Funktion "NeuesBild(data, temp)", welche in den nachfolgenden Abbildungen 23 und 24 stark verkürzt dargestellt ist, übergeben.

Nach der Ausführung der Funktion "NeuesBild(data, temp)" geht der ESP32 in den "deepsleep" über, um Energie zu sparen. Wie schon bei dem Statuswürfel beträgt die Länge 5 Sekunden, außer der Status "6" ist ausgewählt, worauf der "deepsleep" 5 Minuten lang durchgeführt wird.

```

1 def NeuesBild(newdata, newtemp):
2     import framebuffer
3     e = epaper4in2.EPD(spi, cs, dc, rst, busy)
4     e.init()
5     if newdata["status"] != 2:
6         buf = bytearray(w * h // 8)
7         fb = framebuffer.FrameBuffer(buf, w, h, framebuffer.MONO_HLSB)
8         black = 0
9         white = 1
10        fb.fill(white)
11        e.set_display_frame(buf,buf)# Hintergrund wird weiß gefärbt
```

Abbildung 23: Teil 1: Funktion "NeuesBild()" aus "main.py" des E-Ink-Displays

Die Funktion "*NeuesBild(newdata, newtemp)*" verknüpft zunächst die Pins des ESP32 mit den entsprechenden Funktionen. Daraufhin wird, hier in Abbildung 23 in Zeile 3f ersichtlich, durch die "epaper4in2" -Datei die Datenübertragung an das E-Ink-Display eingeleitet. Dabei werden auch die Pins übergeben. Alles nun folgende geschieht nur, wenn der neu gemessene Status nicht "2" ist, da sonst ein Bild angezeigt werden soll.

Nun wird ein Bytearray mit so vielen Bits (8 Bits sind ein Byte) erstellt, wie es Pixel auf dem E-Ink-Display gibt (hier $400 * 300 = 120.000$; $\frac{120.000}{8} = 15.000$). Auf diesen Bytes kann auch die "FrameBuffer"-Funktion der "framebuf"-Bibliothek zugreifen. Diese füllt (siehe Zeile 10 der Abbildung 23) das gesamte Bytearray mit Einsen auf, worauf die "epaper4in2"-Funktion alle Pixel des E-Ink-Displays mit den Bytes auffüllt. Dabei kann jeder Pixel zwei Bits speichern und wird daher zwei mal befüllt, da es einen Wert für die Schwarzfärbung und einen Wert für die Rotfärbung gibt. Der erste "buf" in der Funktion "e.set_display_frame(buf,buf)" ist dabei die potenzielle Schwarzfärbung und der zweite "buf" die potenzielle Rotfärbung, wobei der Wert für den Pixel 0 sein müsste, um zu einer Färbung zu führen. [51]

In der Abbildung 24 wird die Fortsetzung der Funktion "*NeuesBild(newdata, newtemp)*" wiedergegeben. Es wird geprüft, ob der Status eine "1" oder eine "6" ist. Ist dies nicht der Fall, muss der Status eine "3", "4" oder "5" sein. In diesen Fällen passiert dasselbe, jedoch mit unterschiedlichen Texten. Dabei wird zuerst der in "newdata" gespeicherte Text an dem Zeilenumbruch "
" geteilt und die entstehenden Texte in "tnorm1" und "tnorm2" gespeichert. Danach werden noch in Zeile 8 und 9f die Texte für die Temperatur und die Uhrzeit zusammengestellt.

```

1 def NeuesBild(newdata, newtemp):
2     if newdata["status"] == 1:
3         elif newdata["status"] == 6:
4             else:
5                 tnorm1, tnorm2 = newdata["text"].split("<br>", 1)
6                 tnormtemp = str(newtemp["Temperatur"]) + " Grad Celsius"
7                 rtc = machine.RTC().datetime()
8                 tnormuhr = str(rtc[4]) + ":" + str(rtc[5]) + " Uhr"
9                 helper.text_wrap(fb,str(tnorm1),round(w/2 -
10                     helper.hufuregel(tnorm1)),50, black, w=300, h=100)
11                 helper.text_wrap(fb,str(tnormuhr),round(w -
12                     helper.hufuregel(tnormuhr)*2-10), 10,black, w=300, h=100)
13                 helper.text_wrap(fb,str(tnormtemp),10,10,black, w=300, h=100)
14                 e.set_display_frame(buf, None)
15                 fb.fill(white)
16                 fb.rect(round(w/2-helper.hufuregel(tnorm2)-4), 170,
17                     helper.hufuregel(tnorm2)*2+8, 18, black)
18                 helper.text_wrap(fb,str(tnorm2),round(w/2 -
19                     helper.hufuregel(tnorm2)),175, black, w=300, h=100)
20                 e.set_display_frame(None, buf)
21                 e.show_display_frame()
22                 e.sleep()

```

Abbildung 24: Teil 2: Funktion "NeuesBild()" aus "main.py" des E-Ink-Displays

Nun werden die einzelnen Texte in den "buf" hineingeschrieben, indem die entsprechenden Bits von einer 1 zu einer 0 geändert werden, wobei die 0 durch die Variable "black" ersetzt wurde. Es werden vorerst die schwarzen Texte, gefolgt von den roten Texten in das E-Ink-Display geschrieben. So werden "tnorm1", der Text für die Temperatur und der Text für die Uhrzeit so in das Framebuf geschrieben, sodass an den Stellen, wo der Pixel schwarz sein soll, eine 0 steht. Der Text für die Temperatur soll dabei links oben, an der Stelle (10 | 10) beginnen (der Pixel [0 | 0] befindet sich links oben auf dem Display). Der Text für die Uhrzeit soll bei (400-10 | 10) enden, wodurch die Funktion erschwert wird, da nur er Anfangspunkt angegeben werden kann, und der Text "tnorm1" soll zentriert, was die Funktion deutlich verkompliziert, bei dem Pixel 50 (gemessen von oben) beginnen. Diese Bytes werden nun in Zeile 15 auf das E-Ink-Display auf jeden Pixel in den Bit für die Schwarzfärbung übertragen, während für die Rotfärbung nichts ("None") übertragen wird. Darauf wird gesamte Bytearray in Zeile 16 mit Einsen gefüllt und der zweite Text "tnorm2" wird zentriert auf Höhe 175 in Form von Ein-

sen in das Bytearray geschrieben. Zudem wird die einen Pixel breite Umrandung eines Rechteckes in „buf“ derart geschrieben, dass die Umrandung an allen Seiten vier Pixel von dem Text entfernt ist. Darauf werden auch diese Bytes in die Pixel des Display übertragen, sodass der Bit für die Rotfärbung geändert wird.

Zuletzt wird in Zeile 22 das E-Ink-Display aktiviert, wodurch es die neuen Pixel umfärbt. In Zeile 23 geht das E-Ink-Display schließlich wieder in den Ruhemodus über.

4.3 Die API

In der Abbildung 25 wird ein Ausschnitt der Python Datei „main.py“ gezeigt, welcher die Funktionalität der API verdeutlicht. Auch dieser Ausschnitt ist verkürzt und die gesamte Datei kann im Anhang im Ordner „API“ observiert werden. In der Abbildung 25 wird als erstes eine Instanz der „FastAPI()“-Anwendung unter der Variable „app“ gespeichert. In den darauf folgenden Zeilen wird diese Instanz genauer in der Funktionsweise definiert. Schließlich in Zeile 23f wird durch die „uvicorn“-Bibliothek die Instanz „app“ auf dem Gerät gehostet. Unter „host“ wird zudem spezifiziert, dass alle Geräte aus dem lokalem Netzwerk zugreifen können und „port“ gibt an, dass der zugewiesene Port „8000“ ist. Somit kann mit der lokalen IP des Gerätes, auf welchem die API ausgeführt wird, und dem Port mit der API in einem üblichen Browser kommunizieren.

In Zeile 7ff des in Abbildung 25 wiedergegebenen Programmes, werden die in der API gespeicherten Daten geladen. Das geschieht wieder mit einer „help.py“, welche sehr ähnlich zu der „helper.py“ beim Statuswürfel, abgebildet in Abbildung 13, operiert. Die gespeicherten Daten enthalten den derzeit ermittelten Status, sowie die derzeit gemessene Temperatur. Dazu wird zu jedem möglichen Status ein Text gespeichert. Bei Status „1“ und „2“ ist dieser nicht relevant, bei Status „6“ wird der Text dargestellt und bei Status „3“, „4“ und „5“ beinhaltet der Text einen Zeilenumbruch bei „
“, damit der Text in den oberen und unteren Teil auf dem E-Ink-Display geteilt und daraufhin dargestellt werden kann. Zuletzt speichert die API noch die „TempZusatznachricht“, welche für die jeweilige Temperatur eine extra Nachricht bietet, welche in Zustand „1“ auf dem E-Ink-Display je nach gemessener Temperatur angezeigt werden soll. So existiert eine für weniger als 0 °C, eine für weniger als 18 °C, eine für weniger als 33 °C, eine für weniger als 50 °C und eine für ab 50 °C.

```

1  from fastapi import FastAPI, Path
2  from typing_extensions import Annotated
3  import sys, help, uvicorn
4
5  app = FastAPI()
6
7  Startdaten = help.load_data("Daten.dat")
8  Temperatur = Startdaten["Temperatur"]
9  status = Startdaten["status"]
10 TempZusatznachricht = Startdaten["TempZusatznachricht"]
11 text = Startdaten["text"]
12
13 @app.get("/stat/{val}")
14 @app.post("/stat/{val}")
15 async def set_status(val: Annotated[int, Path(ge=1, le=6)]):
16     global status, Startdaten
17     status = val
18     Startdaten["status"] = status
19     help.save_data(Startdaten, "Daten.dat")
20     return {"status": status, "message" : "status changed to " +
21             str(status)}
22
23 if __name__ == "__main__":
24     uvicorn.run("main:app", host="0.0.0.0", port=8000)

```

Abbildung 25: Ausschnitt der Datei "main.py" der API

In der Abbildung 25 kann beispielsweise eine Funktion der API in Zeile 13ff eingesehen werden. Dabei soll der API der aktuelle Status des Statuswürfels mit einer "GET" oder "POST"-Request übertragen werden. Das wird beispielsweise mit der url "http://192.168.178.74:8000/stat/4" umgesetzt. "192.168.178.74:8000" ist dabei die lokale IP der API, gefolgt von dem Port. "/stat/" führt zu der in Zeile 13ff dargestellten Funktion "set_status()". Der Wert "4" ist die in Zeile 15 unter der Funktion "set_status()" mit "val" deklarierte Variable. Diese muss wie in Zeile 15 beschrieben zwischen ("ge = 1") 1 und ("le = 6") 6 sein, da der Status nur zwischen 1 und 6 definiert ist. Darauf wird der neue Status in den Variablen "status" und "Startdaten" geändert und auch die gesicherte Datei "Daten.dat", durch welche anfangs die Daten initialisiert wurden, wird mit dem neuen Rotationszustand überschrieben. Zudem gibt die Funktion (in diesem Beispiel) die Antwort: "{\"status": 4, \"message\": \"status changed to 4\"}" zurück.

Auf diese Art und Weise sind sämtliche Funktionen der API geschrieben. Diese sind zudem in der nachfolgenden Tabelle 3 zusammengefasst. Die in geschweiften Klammern geschriebenen Begriffe sind dabei Variablen, welche als Teil der restlichen url eingelesen und als Variable mit Wert verarbeitet werden.

Pfad der url	Beschreibung der Funktion
“/stat”	Ausgeben des Status
“/stat/{val}”	Einlesen eines Status ” <i>val</i> ”
“/tempstat”	Ausgeben der Temperatur und des Temperatur-Textes
“/tempstat/{temp}”	Einlesen einer Temperatur ” <i>temp</i> ”
“/aenderstatustext{statusae}/-{Ntext}”	Änderung des Textes von Status ” <i>statusae</i> ” zu dem Text ” <i>Ntext</i> ”
“/aenderWettertext{Wstatus}/-{NWtext}”	Änderung des Textes der Temperatur ” <i>Wstatus</i> ” zu dem Text ” <i>NWtext</i> ”
“/statustext”	Ausgabe des Status inklusive Text des Status
“/text/{nmbr}”	Ausgabe des Textes zu Status ” <i>nmbr</i> ”

Tabelle 3: Funktionen der API

4.4 Website

In der Abbildung 26 sind 3 Websites dargestellt, von welchen die zugehörigen html-Dateien im Anhang im Unterordner ”HTML-Website” zu finden sind. Genauer ist im oberen Teil der Abbildung 26 die Datei ”Bearbeitung.html”, links unten die Datei ”main.html” und rechts unten die Datei ”index.html” visualisiert. Allgemein soll die Website alle Informationen der API abrufen und gegebenenfalls ändern können. Dazu gehören der derzeitige Status, sowie die Texte für den Status und die Temperatur.

In der ”main.html”, sichtbar links unten in der Abbildung 26, soll der derzeitige Status des Statuswürfels dargestellt sein und somit den Status mit dem dazu gehörigen Text zeigen, sowie im Status 1 die Temperatur mit dem Temperaturtext zeigen. Zudem kann der Benutzer von der Seite ”main.html” unter Eingabe des richtigen Benutzernamens und Passwortes zu der Seite ”Bearbeitung.html” weitergeleitet werden. Dadurch ist der Zugriff auf die ”Bearbeitung.html” auf die entsprechenden Administratoren begrenzt.

BEARBEITUNGSSEITE

welcher Status soll geändert werden?

hier kann der Text dann geändert werden

**WEBSITE
des
STATUSWÜRFELS**

Derzeit:
Status: Hier steht eigentlich der Status
Text: Hier steht eigentlich der Text
/Temperatur: nicht auf Status 1!
/Temperaturtext: nicht auf Status f /

Bild:
Kein Bild, da falscher Status

wait

N.sichtbar

Abbildung 26: Die 3 Websites für die API

In der "Bearbeitung.html", erkennbar oben in der Abbildung 26, kann jeweils ein Status und ein neuer Text für den entsprechenden Status eingegeben werden. Daraufhin wird der entsprechende Text für den jeweiligen Status in der API geändert. Zudem kann der bisherige Text eines Status ausgegeben werden, indem bei dem Eingabefeld für den Status "Enter" gedrückt wird. Daraufhin wird in dem unterem Eingabefeld der Text des eingegebenen Status eingefügt, welcher so einfacher bearbeitet werden kann.

Zuletzt ist die "index.html", wahrnehmbar rechts unten in der Abbildung 26, für das Setzen eines neuen Status zuständig. Dabei kann ein neuer Status in die Eingabezeile eingegeben werden, worauf auf den "Button", "Click Me TO set state", geklicked werden kann, worauf der derzeitig gespeicherte Rotationszustand überschrieben wird.

5 Zusammenfassung und Ausblick

5.1 Zusammenfassung

In der Konzeption wurde intensiv geplant und fast alles davon wurde auch erreicht. Dazu gehört die Programmierung und der Bau eines Akkumulator-betriebenen Statuswürfels, welcher sowohl seinen eigenen Rotationszustand erkennt, als auch die Außentemperatur misst. Dazu sendet dieser, jedoch nur wenn sich entweder der Rotationszustand oder die Temperatur beim Status "1" verändern, die neuen Messungen an eine API.

Dazu wurde eine API programmiert, welche in der Lage ist die Messungen des Statuswürfels entgegenzunehmen und diese zu speichern. Zudem speichert diese einen Text zu jedem Status, sowie fünf verschiedene Texte für den Wert der Temperatur, je nachdem wie kalt oder warm es ist. Mittels Anfragen können all diese Daten der API zudem geändert werden.

Ein Tool um diese Daten, insbesondere den Text zu den einzelnen Rotationszuständen, einfach ändern zu können ist die Website, welche zum einen den derzeitigen Status, inklusive dazugehörigem Text, darstellen kann, als auch den Status selber ändern kann. Eine Funktion, welche in der Website jedoch noch nicht integriert ist, ist die Änderung des Textes für die Temperatur. Das könnte ähnlich zu der Änderung der Texte zu dem Status erfolgen.

Zuletzt gibt es noch ein E-Ink-Display, welches mit der Hilfe eines ESP32 die Temperatur und Rotationsdaten, inklusive der dazugehörigen Texte, bei der API anfragt und dadurch zugesandt bekommt. Je nach empfangenen Daten bildet das E-Ink-Display darauf einen Text ab.

Somit kann ein Mitarbeiter in seinem Büroraum den Statuswürfel drehen, worauf innerhalb von 10 Sekunden (da der "deepsleep" je, sowohl beim Statuswürfel, als auch beim E-Ink-Display, maximal 5 Sekunden andauert) das E-Ink-Display den neuen Arbeitsstatus des Mitarbeiters darstellt.

Insgesamt wurde fast alles erreicht. Ein kleiner Teil in der Website fehlt und die Option ein Bild einzufügen fehlt völlig. Somit ist es derzeit weder möglich ein Bild über die Website hochzuladen, noch dieses über die API an dem E-Ink-Display projizieren zu lassen, was die Funktionalität und des Sinn des Projektes jedoch kaum einschränkt.

5.2 Ausblick

Eine Funktion, welche perspektivisch integriert werden sollte, ist, wie schon erwähnt, das Bild und das Ändern des Temperatur-Textes. Überdies ist das Laden des Statuswürfels durch das Kabel manchmal in der Funktion unpraktisch, da die Rotation durch das Kabel eingeschränkt ist. Besser wäre hier Induktionsladen, durch welches sich der Statuswürfel nur an der richtigen Position über dem Ladegerät befinden muss, um zu einer Ladung zu führen.

Weiterhin sollte das Passwort für das WLAN für den ESP32 leichter anpassbar sein, anstatt immer eine Datei im ESP32 verändern zu müssen. Das ist bei wechselndem WLAN sehr aufwendig. Zudem ändert sich die lokale IP-Adresse der API häufig, weshalb auch hier immer die Programme der einzelnen Komponenten verändert werden müssen. Auf dieses Problem sollte auch eingegangen werden.

Darüber hinaus ist es für den Nutzer bisher noch unklar, welche Seite welchen Status darstellt, da die einzelnen Rotationsseiten des Statuswürfels ähnlich aussehen. Weil der Datensatz verändert werden soll, könnte der anzuzeigende Text auf jeder Seite auf einem kleinem E-Ink-Display angezeigt werden um eine einfachere Benutzbarkeit für den Endnutzer zu ermöglichen.

Quellenverzeichnis

- [1] Shenzhen Rainbowsemi Electronics Co. Ltd. D1 mini esp32 esp-32 wifi wireless modul. https://www.alibaba.com/product-detail/D1-mini-ESP32-ESP-32-WIFI_60783959842.html, o.D. (18-12-2024).
- [2] NanjingQinhengMicroelectronics. Ch341ser.exe. https://www.wch.cn/download/ch341ser_exe.html, o.D. (18-12-2024).
- [3] AZ-Delivery Vertrieb GmbH. Esp32 mit integriertem wlan und bluetooth. <https://www.az-delivery.de/collections/esp32>, o.D. (18-12-2024).
- [4] Autoren: <https://xtools.wmcloud.org/authorship/de.wikipedia.org/ESP32?uselang=de>. Esp32. <https://de.wikipedia.org/wiki/ESP32>, 13.09.2024. (18-12-2024).
- [5] Anna Kalinowsky. Esp32: Was ist das? was kann das? <https://www.heise.de/tipps-tricks/ESP32-Was-ist-das-Was-kann-das-4471527.html>, 22.05.2020. (18-12-2024).
- [6] Brian Krent. Esp32-prozessor (bildmitte) auf einer leiterplatte bestückt. links die gedruckte wlan-antenne. https://de.wikipedia.org/wiki/ESP32#/media/Datei:Espressif_ESP-WROOM-32_Wi-Fi_&_Bluetooth_Module.jpg, 05.04.2017. (18-12-2024).
- [7] Codierknecht. -. <https://forum.iobroker.net/topic/69300/probleme-mit-max7210-led-display-festbeleuchtung/20>, 18.10.2023. (18-12-2024).
- [8] AZ-Delivery Vertrieb GmbH. Battery shield for lithium batteries for d1 mini. <https://www.az-delivery.de/en/products/batterie-shield-fuer-lithium-batterien-fuer-d1-mini>, o.D. (18-12-2024).
- [9] Alldatasheet.com. Tp5400 datasheet(pdf) 1 page - billiton ltd. <https://www.alldatasheet.com/html-pdf/1140419/ASIC/TP5400/110/1/TP5400.html>, o.D. (18-12-2024).
- [10] Elektronik-Kompendium.de. Mems - micro-electro-mechanical systems. <https://www.elektronik-kompendium.de/sites/bau/1503041.htm>, o.D. (18-12-2024).

- [11] InvenSense Inc. Mpu-6000 and mpu-6050 product specification revision 3.4. <https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>, 19.08.2013. (18-12-2024).
- [12] Nevit Dilmen. Datei:gy-521 mpu-6050 module 3 axis gyroscope + accelerometer 0487.jpg. https://de.m.wikipedia.org/wiki/Datei:GY-521_MPUS-6050_Module_3_Axis_Gyroscope_+_Accelerometer_0487.jpg, 2014. (18-12-2024).
- [13] Günther GmbH. Günther gmbh temperaturmesstechnik — technische informationen zu widerstandsthermometern. <https://www.mts.ch/produkte-messtechnik-schaffhausen-gmbh/guenther-gmbh-temperaturmesstechnik-thermoelemente\protect\discretionary{\char\hyphenchar}font\-\widerstandsthermometer/technische-informationen/technische\protect\discretionary{\char\hyphenchar}font\-\informationen-zu-widerstandsthermometern/1-aufbau-und-funktionsweise-von-widerstandsthermometern.html/> 280, o.D. (18-12-2024).
- [14] Farnell GmbH. Gyroskope. <https://de.farnell.com/sensor-gyroscope-technology>, o.D. (18-12-2024).
- [15] John. Mems accelerometer. <https://www.instrumentationtoday.com/mems-accelerometer/2011/08/>, 17.08.2011. (18-12-2024).
- [16] HEINEN Elektronik GmbH. I2c. <https://heinen-elektronik.de/glossar/I2C/>, o.D. (18-12-2024).
- [17] Digital Matter 2024. I2c sensorüberwachung. <https://sense.digitalmatter.com/de/blog/I2C-sensor-monitoring>, 20.08.2024. (18-12-2024).
- [18] telos Systementwicklung GmbH. Fast mode. <https://de.i2c-bus.org/fast-mode/>, o.D. (18-12-2024).
- [19] telos Systementwicklung GmbH. Ultra fast mode ufm. <https://www.i2c-bus.org/ultra-fast-mode-ufm/>, o.D. (18-12-2024).
- [20] FKM Sintertechnik GmbH. Was ist der 3d-druck? <https://www.fkm.net/was-ist-der-3d-druck/>, o.D. (18-12-2024).

- [21] Prusa Research a.s. Modellierung mit blick auf 3d-druck. https://help.prusa3d.com/de/article/modellierung-mit-blick-auf-3d-druck_164135, zuletzt Aktualisiert: 2020. (18-12-2024).
- [22] Inc. Autodesk. Tinkercad. <https://www.tinkercad.com>, o.D. (18-12-2024).
- [23] Talend Germany GmbH. Api (application programming interface) — definition und vorteile. <https://www.talend.com/de/resources/was-ist-eine-api/>, o.D. (18-12-2024).
- [24] fastapi. Fastapi. <https://fastapi.tiangolo.com/>, o.D. (18-12-2024).
- [25] uvicorn. uvicorn. <https://www.uvicorn.org>, o.D. (18-12-2024).
- [26] appleute GmbH. Was ist eine rest-api? <https://www.appleute.de/app-entwickler-bibliothek/was-ist-rest-api/>, o.D. (18-12-2024).
- [27] Tim Aschermann. Was ist html? - verständlich erklärt. https://praxistipps.chip.de/was-ist-html-verstaendlich-erklaert_40979,16.05.2015. (18-12-2024).
- [28] Kaspersky Labs GmbH. Ip-adresse – definition und erläuterung. <https://www.kaspersky.de/resource-center/definitions/what-is-an-ip-address>, o.D. (18-12-2024).
- [29] Xovi GmbH. Was bedeutet http? <https://www.xovi.de/was-bedeutet-http/>, o.D. (18-12-2024).
- [30] DigiCert Inc. Was sind ssl, tls und https? <https://www.digicert.com/de/what-is-ssl-tls-and-https>, o.D. (18-12-2024).
- [31] IONOS Redaktion. Get vs. post – die beiden wichtigsten http-requests im vergleich. <https://www.ionos.de/digitalguide/websites/web-entwicklung/get-vs-post/>, 06.04.2020. (18-12-2024).
- [32] Achim Hannemann. E-paper: Unterschätzte displaytechnologie? <https://www.professional-system.de/basics/e-paper-unterschaetzte-displaytechnologie/>, 11.03.2022. (18-12-2024).

- [33] Waveshare. 400x300, 4.2inch e-ink display module, three-color. <https://www.waveshare.com/4.2inch-e-paper-module-b.htm>, o.D. (18-12-2024).
- [34] Dr. Guido Schryen. Dipl.-Kfm. Jürgen Karla. Elektronisches papier displaytechnologie mit weitem anwendungsspektrum. <https://epub.uni-regensburg.de/21255/>, o.D. (18-12-2024).
- [35] Anna Rybalko. E ink: Die evolution der farbtechnologie. <https://www.elektroniknet.de/optoelektronik/displays/e-ink-die-evolution-der-farbtechnologie.215070.html>, 08.03.2024. (18-12-2024).
- [36] waveshare. 4.2inch-e-paper-specification.pdf. <https://www.waveshare.com/w/upload/6/6a/4.2inch-e-paper-specification.pdf>, o.D. (18-12-2024).
- [37] George Robotics Limited. Micropython. <https://micropython.org>, o.D. (18-12-2024).
- [38] SunFounder. 1.1 einföhrung in micropython. https://docs.sunfounder.com/projects/kepler-kit/de/latest/pyproject/python_start/introduction_micropython.html, o.D. (18-12-2024).
- [39] Thonny. Thonny-python ide for beginners. <https://thonny.org>, o.D. (18-12-2024).
- [40] Atlassian Trust Center. Was ist git? <https://www.atlassian.com/de/git/tutorials/what-is-git>, o.D. (18-12-2024).
- [41] Atlassian Trust Center. Befehle. <https://www.atlassian.com/de/git/glossary#commands>, o.D. (18-12-2024).
- [42] Inc. GitHub. Home. <https://github.com>, o.D. (18-12-2024).
- [43] Alexander S. Gillis. Netzwerktopologie. <https://www.computerweekly.com/de/definition/Netzwerktopologie>, o.D. (18-12-2024).
- [44] Dennis Kerzig. Graphentheorie. <https://hpi.de/friedrich/teaching/units/graphentheorie.html>, o.D. (18-12-2024).

- [45] PTC. What is cad? <https://www.ptc.com/en/technologies/cad>, o.D. (18-12-2024).
- [46] Thorsten Thormählen. Technische informatik rechnerinterne zahlenformate. https://www.mathematik.uni-marburg.de/~thormae/lectures/ti1/ti_2_2_ger_web.html, 27.10.2022. (18-12-2024).
- [47] MicroPython. machine — functions related to the hardware. <https://docs.micropython.org/en/latest/library/machine.html>, o.D. (18-12-2024).
- [48] MicroPython. 4. network basics. https://docs.micropython.org/en/latest/esp8266/tutorial/network_basics.html, o.D. (18-12-2024).
- [49] Damien P. George. Paul Sokolovsky. Yan Minge. urequests — network request module. https://makeblock-micropython-api.readthedocs.io/en/latest/public_library/Third-party-libraries/urequests.html, o.D. (18-12-2024).
- [50] Damien P. George. Paul Sokolovsky. usocket – socket module. <https://docs.micropython.org/en/v1.14/library/usocket.html>, o.D. (18-12-2024).
- [51] MicroPython. framebuffer — frame buffer manipulation. <https://docs.micropython.org/en/latest/library/framebuf.html>, o.D. (18-12-2024).

Abbildungsverzeichnis

1	Der ESP32 [6]	2
2	ESP32 D1-Mini-Pinout [7]	3
3	Batterie-Shield für das D1-Mini-Board [8]	4
4	Der MPU-6050 [12]	4
5	Das 400x300 three-color E-Ink display module von Waveshare [33]	11
6	grobe Darstellung der Netzwerktopologie	15
7	Entwürfe für die Bildschirme des E-Ink-Displays	20
8	Ausschnitt der Datei "mpu6050.py"	23
9	Ausschnitt der Datei "main.py"	25
10	Ausschnitt der Datei "mpu6050.py"	25
11	Ausschnitt der Datei "main.py" des Statuswürfels	26
12	Ausschnitt der Datei "main.py" des Statuswürfels	27
13	Ausschnitt der Datei "helper.py" des Statuswürfels	28
14	Ausschnitt der Datei "main.py" des Statuswürfels	29
15	"config.py" des Statuswürfels	29
16	Micro-Python Code für das Aufnehmen der Messwerte	31
17	Boxplot-Diagramm der Messreihe	32
18	Modell des Statuswürfels in TinkerCAD	34
19	Modell des Statuswürfels in TinkerCAD von oben	35
20	Die entscheidenden Komponenten des Statuswürfels	36
21	Der Statuswürfel mit eingebauten Komponenten	37
22	Ausschnitt aus "main.py" des E-Ink-Displays	38
23	Teil 1: Funktion "NeuesBild()" aus "main.py" des E-Ink-Displays	39
24	Teil 2: Funktion "NeuesBild()" aus "main.py" des E-Ink-Displays	40
25	Ausschnitt der Datei "main.py" der API	42
26	Die 3 Websites für die API	44
27	QR-Code zu dem GitHub-Repository	53

Tabellenverzeichnis

1	Kabelanschlüsse für das E-Ink-Display [33]	12
2	Zuordnungen zwischen Rotationszuständen und Texten	22
3	Funktionen der API	43

Anlagen

In den Anlagen befinden sich nochmals alle Dateien, welche in der BeLL erwähnt werden. Dazu gehören sämtliche Programme zu dem Statuswürfel, der API, dem E-Ink-Display und der Website, welche in gleichnamigen Ordnern aufgefunden werden können.

Allgemein liegen die anhängenden Dokumente in der Form einer CD-Rom anbei. Jedoch können alle Dateien auch in einem für die BeLL angelegtem öffentlichen GitHub-Repository unter dem Link <https://GitHub.com/Typiano/Git> aufgefunden werden. Das Repository kann auch durch den in Abbildung 27 dargestellten QR-Code aufgerufen werden.



Abbildung 27: QR-Code zu dem GitHub-Repository

Danksagung

Zuerst gilt ein besonderer Dank meinem externen Betreuer Philipp Dockhorn, welcher mir zuerst schon bei Themenwahl geholfen hat und darauf mir sowohl sämtliches Arbeitsmaterial bzw. sämtliche Hardware zur Verfügung gestellt hat, als auch mir jederzeit sämtliche inhaltliche Fragen zur BeLL beantwortet hat. Zudem hat er bei der Gliederung und dem grobem Aufbau der BeLL maßgeblich geholfen.

Nur durch seine Aktive Unterstützung und wertvollen Hinweise für die BeLL, konnte diese beendet werden.

Mein weiterer Dank gilt der HTWK, welche mir auch durch Herrn Dockhorn einen Arbeitsraum zum effizienten Erarbeiten der BeLL zur Verfügung stellte.

Zuletzt möchte ich mich noch bei Herrn Simon Koch bedanken, welcher mir als mein interner Betreuer stets für Fragen zur Verfügung stand und diese jederzeit umfangreich beantwortet hat.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne unerlaubte fremde Hilfe angefertigt, keine anderen als die angegebenen Quellen und Hilfsmittel verwendet und die den verwendeten Quellen und Hilfsmitteln wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Zusätzlich stimme ich zu, dass diese Arbeit schulintern verwendet werden darf.

Leopold Peer Hofmann, Leipzig, 19. Dezember 2024