Schweizerische
Fachschule
TEKO

Abbildung 1 Titelbild

Automatisches Pflanzenbewässerungssystem für deine Zimmerpflanzen

Diplomarbeit

Auftraggeberschaft: Schweizerische Fachschule Teko

Autor: Thomas Meier

Dozent: Christian Meier

Ort, Datum: Rümlang, 21.10.2023

Automatisches Pflanzenbewässerungssystem für Zimmerpflanzen

Autor
Thomas Meier
Oberdorfstrasse 3
8153 Rümlang
079 882 92 27
thomasmeier89@gmx.ch

Dozent
Christian Meier
Fachhochschule Nordwestschweiz
christian.meier@edu.teko.ch

Auftraggeberschaft
Schweizerische Fachschule Teko
Europa-Strasse 18
8152 Opfikon
043 305 23 37
zuerich@teko.ch

Teko, 21.10.2023

Ehrenwörtliche Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der im Literaturverzeichnis angegebenen Quellen und Hilfsmittel angefertigt habe.

Die wörtlich oder inhaltlich den im Literaturverzeichnis aufgeführten Quellen und Hilfsmitteln entnommenen Stellen sind in der Arbeit als Zitat bzw. Paraphrase kenntlich gemacht.

Diese Bachelor-Thesis / Projektarbeit / Studienarbeit ist noch nicht veröffentlicht worden. Sie ist somit weder anderen Interessenten zugänglich gemacht noch einer anderen Prüfungsbehörde vorgelegt worden.

Rümlang, 21.10.2023

Thomas Meier

Vorwort

Diese Diplomarbeit ist das Ergebnis eines faszinierenden Projekts, welche sich dem Ziel verschrieben hat, das Leben von Zimmerpflanzen zu verbessern. Die heutige Welt ist von Innovation und Automatisierung geprägt. Nun ist es Zeit das unsere Zimmerpflanzen von diesem Fortschritten profitieren.

In der schnelllebigen Welt, in der wir uns heute befinden, sind Zeit und Aufmerksamkeit kostbare Ressourcen. Dabei ist es leicht, die Bedürfnisse unserer geliebten Zimmerpflanzen aus den Augen zu verlieren. Wer kennt nicht das Gefühl der Sorge, wen man auf Reisen ist oder viel viel beschäftigt zu sein und sich fragt, ob unsere grünen Mitbewohner zu Hause genug Wasser bekommen? Nach einer dreiwöchigen Reise nach Ägypten machte ich die tragische Erfahrung zwei von drei Zimmerpflanzen aufgrund von Wassermangel verloren zu haben. Dies inspirierte mich dazu, ein automatisches Pflanzenbewässerungssystem für Zimmerpflanzen zu entwickeln.

Mein Ziel ist es, nicht nur die Grundbedürfnisse von Zimmerpflanzen zu erfüllen, sondern auch einen Schritt weiterzugehen: Dieses Bewässerungssystem kombiniert Sensortechnologie mit einer intelligenten Steuerung, um die optimale Pflege der Pflanzen zu garantieren. Mit Hilfe von Sensoren für die Bodenfeuchtigkeit, Temperatur sowie Luftfeuchtigkeit als auch einem Füllstandsensor für den Wassertank wird die Pflanze in Echtzeit überwacht.

Ein Mikrocontroller ist das Herz dieses Systems, welcher alle Daten verarbeitet und entscheidet, wann und wie viel Wasser die Pflanze benötigt. Dabei soll jeder Pflanze ein optimales Wachstumsumfeld geboten werden. Der Zustand der Pflanze wird jederzeit auf einem LCD-Display angezeigt und kann sogar auf Ihrem Handy nachgeschaut werden. So hat man jederzeit die volle Gewissheit über das Wohlbefinden der Pflanze.

Diese Diplomarbeit dokumentiert den gesamten Entwicklungsprozess von der Ideenfindung über die Konzeption bis hin zur Implementierung und Evaluierung des automatischen Pflanzenbewässerungssystems. Sie zeigt, wie Technologie dazu beitragen kann, die Schönheit und Gesundheit von Zimmerpflanzen zu bewahren, ohne dass man sich Sorgen muss, wer sich in der eigenen Abwesenheit um sie kümmert.

Management Summary

Diese Diplomarbeit, durchgeführt von Thomas Meier im Auftrag der Schweizerischen Fachschule Teko, konzentrierte sich auf die Entwicklung eines automatischen Bewässerungssystems für Zimmerpflanzen. Das Hauptziel des Projekts bestand darin, ein benutzerfreundliches System zu schaffen, das Zimmerpflanzen automatisch bewässert und gleichzeitig den Zustand der Pflanzen überwacht und auf einem LCD-Display anzeigt. Die Arbeit sollte ausserdem die Möglichkeit bieten, den Zustand der Pflanzen über ein mobiles Gerät abzurufen und das gesamte System in einem 3D-gedruckten Gehäuse unterzubringen. Das Budget für dieses Projekt wurde auf 200 Franken festgelegt.

Die wichtigsten Ergebnisse und Fortschritte des Projekts sind:

Automatische Bewässerung der Zimmerpflanzen: Das entwickelte System ist in der Lage, Zimmerpflanzen zuverlässig zu bewässern. Die Automatisierungsfunktion wurde erfolgreich getestet und hat gezeigt, dass die Pflanzen effektiv und pünktlich bewässert werden.

Anzeige des Pflanzenzustands auf dem LCD-Display: Das LCD-Display zeigt den aktuellen Zustand der Pflanze in Echtzeit an. Dies beinhaltet die Bodenfeuchtigkeit, den Füllstand des Wassertanks, Temperatur und Luftfeuchtigkeit. Dem Benutzer bieten diese Informationen wertvolle Einblicke in den Zustand ihrer Zimmerpflanze.

Trotz des insgesamt erfolgreichen Verlaufs des Projekts gab es einige Herausforderungen:

Budgetüberschreitung: Das Projektbudget von 200 Franken wurde um 3.90 Franken überschritten. Dies war auf unberücksichtigte Lieferkosten und die doppelte Bestellung von Komponenten aus Sicherheitsgründen zurückzuführen, um mögliche defekte Bauteile auszuschliessen. Die Budgetüberschreitung wird durch eine sorgfältigere Kostenschätzung in Zukunft vermieden.

Nicht erreichte Ziele: Das ursprünglich geplante Ziel, den Zugriff auf den Pflanzenzustand über ein mobiles Gerät zu ermöglichen, konnte aufgrund der äusserst aufwendigen Programmierarbeit und Zeitbeschränkungen nicht erreicht werden. Dies wurde bewusst gestrichen, um die verfügbaren Ressourcen auf die Hauptziele zu konzentrieren.

Das Projekt wurde innerhalb des geplanten Zeitrahmens abgeschlossen, wobei die Pufferzeit für die Dokumentation genutzt wurde, die aufgrund zusätzlicher Zeit für 3D-Design und Programmierung benötigt wurde.

In Bezug auf die Qualitätssicherung wurde das Bewässerungssystem ausführlich getestet und für mindestens eine Woche im realen Betrieb erprobt, um sicherzustellen, dass es zuverlässig funktioniert.

Trotz der genannten Herausforderungen ist das entwickelte Bewässerungssystem eine effektive Lösung, um Zimmerpflanzen gesund zu halten. Es bietet eine automatische Bewässerungsfunktion und Echtzeitüberwachung des Pflanzenzustands, was den Bedürfnissen von Pflanzenliebhabern gerecht wird. Die Budgetüberschreitung kann in zukünftigen Projekten durch eine präzisere Kostenkalkulation vermieden werden. (ChatGPT Spellcheck, 2023)

Inhaltsverzeichnis

1 Projektinitialisierung und Planung

1.1 Themenbeschreibung

Die vorliegende Diplomarbeit zielt darauf ab, ein automatisches Pflanzenbewässerungssystem für Zimmerpflanzen zu entwickeln. Dieses System soll die Bodenfeuchtigkeit kontinuierlich überwachen und bei Bedarf die Bewässerung der Pflanze mithilfe einer Wasserpumpe automatisch durchführen. Darüber hinaus werden wichtige Umgebungsparameter wie die aktuelle Temperatur, Luftfeuchtigkeit, Bodenfeuchtigkeit und der Füllstand des Wassertanks auf einem LCD-Display angezeigt. Zusätzlich erhält der Nutzer die Möglichkeit, den aktuellen Zustand seiner Zimmerpflanze über ein mobiles Gerät zu verfolgen.

Die Inspiration zu diesem Projekt entstand aus einem persönlichen Erlebnis von Thomas, nach einem Urlaub in Ägypten. Während einer dreiwöchigen Abwesenheit wurde vergessen, jemanden damit zu beauftragen, sich um die Zimmerpflanzen von Thomas zu kümmern. Dies führte dazu, dass zwei von drei Pflanzen in dieser Zeit verstarben, während eine Pflanze nur knapp gerettet werden konnte. Diese Erfahrung diente als Ausgangspunkt und Motivation für die Entwicklung eines automatischen Bewässerungssystems, um sicherzustellen, dass solche Vorfälle in Zukunft vermieden werden können. Diese Diplomarbeit widmet sich der Schaffung einer Lösung, die die Pflege von Zimmerpflanzen erleichtert und ihnen die optimale Menge an Wasser und die richtigen Umgebungsbedingungen bietet, um ihr Wohlbefinden zu gewährleisten.

1.2 Pflichtenheft

1.2.1 Einleitung

Thomas Meier ist der Auftragnehmer in dieser Diplomarbeit. Der Auftraggeber dieser Diplomarbeit ist die Schweizerische Fachschule Teko. In diesem Projekt wird ein automatisches Bewässerungssystem für Zimmerpflanzen entwickelt, welches in Zukunft das Bewässern für Pflanzenliebhaber/-innen erleichtern soll. Dabei wird mittels Sensoren der aktuelle Zustand der Pflanze stets überwacht. Zudem wird die Pflanze bei Bedarf automatisch bewässert.

1.2.2 Auftrag

Das ganze Bewässerungssystem wird von einem Mikrocontroller gesteuert. Dabei wird die Erdfeuchtigkeit mit einem Bodenfeuchtigkeitssensor gemessen. Sobald die Erdfeuchtigkeit zu tief ist, wird die Pflanze mit einer Wasserpumpe automatisch bewässert. Die Temperatur, Luftfeuchtigkeit und der Füllstand wird ebenfalls mittels Sensoren gemessen. Diese Informationen sollen auf einem LCD-Display angezeigt werden. Zusätzlich kann der Nutzer auf dem Handy den aktuellen Zustand der Pflanze respektive die genannten Daten einsehen. Die Komponenten werden verkabelt sowie zusammengelötet. Das ganze System wird in einem 3D gedruckten Gehäuse, welches einen Wassertank beinhaltet eingebaut.

Die Erfolgskriterien dieser Arbeit sind es eine Zimmerpflanze automatisch zu bewässern, den aktuellen Zustand der Pflanze auf einem LCD-Display anzuzeigen, den Zustand der Pflanze via Handy einzusehen und das Ganze in einem 3D gedruckten Gehäuse unterzubringen. Die Kosten für dieses Projekt sollten nicht höher als 200 Franken sein.

1.2.3 Teams und Schnittstellen

Das Projekt wird vom Studenten Thomas Meier umgesetzt. Dabei ist der Dozent Christian Meier als Diplomcoach an einer Seite, welcher bei allfälligen Fragen zur Unterstützung zur Verfügung steht. Der Diplomexperte Philipp Amacker bewertet diese Arbeit sowie die Präsentation in Zusammenarbeit mit Christian Meier.

1.2.4 Rahmenbedingungen

Die Diplomarbeit beginnt, nachdem die Eingabe des Projekts angenommen wurde. Die Arbeit beginnt am 01.08.2023 und muss bis am 25.10.2023 um 14:00 abgegeben werden. Die Arbeit wird am 13.11.2023 an der Teko Zürich präsentiert.

Das Projekt wird unter der Woche am Abend, sowie am Wochenende umgesetzt. Geplant ist ein Aufwand von mindestens 200 Stunden.

1.2.5 Technische Anforderungen

Zur Umsetzung des Projektes wird ein Mikrocontroller gebraucht. Zudem werden verschiedene Sensoren eingesetzt. Zum einen wird ein Bodenfeuchtigkeits-, Temperatur-, Luftfeuchtigkeit- und Füllstandsensor verwendet. Zudem benötigt man eine Wasserpumpe sowie ein LCD-Display. Damit das Gehäuse 3D gedruckt werden kann benötigt man einen 3D Drucker, sowie eine CAD-Software für die Zeichnung und Gestaltung des Gehäuses. Für die Programmierung wird ein Laptop mit der Software «Visual Studio Code» benötigt.

1.2.6 Problemanalyse

Es könnte Probleme bei der Beschaffung der Bauteile geben, da es momentan oft zu Lieferverzögerungen führt. Um dieses Problem zu verhindern, müssen die Bauteile ohne Aufschub bestellt werden. Zudem kann bei Verzögerungen bereits an der Dokumentation sowie der Software gearbeitet werden.

Defekte Bauteile: Ein weiteres Problem könnte sein, das ein Bauteil in defektem Zustand geliefert wird oder diese beim Zusammenbauen beziehungsweise dem Löten kaputt geht. Damit es nicht zu zusätzlichen Verzögerungen kommt wird bei kritischen Komponenten jeweils ein zusätzliches Teil bestellt. Zudem werden die Bauteile, sobald sie erhalten wurden, sofort getestet. Falls zu viele der gleichen Teile defekt geliefert wurden, können diese mit genügend Restzeit im Projekt nachbestellt werden.

Software: Zudem könnte es Probleme in der Software-Entwicklung geben, da sich die Programmierkünste des Studenten auf die grundlegenden Kenntnisse beschränken. Dies kann eine Reihe von Herausforderungen und Einschränkungen mit sich bringen. Zum einen könnte sich die Funktionen begrenzen, da der Student Schwierigkeiten hat komplexere Funktionen und Anforderungen zu implementieren. Dazu könnte der Code ineffizient werden oder schlechtes designe werden, damit dieser in Zukunft schlecht erweiterbar ist. Deshalb muss aufgrund dieser Probleme bei der Software allenfalls mit gewissen Einschränkungen gerechnet werden.

Toleranzen: Es könnte sein, dass die Toleranzen mit dem 3D Drucker nicht erreicht werden können, damit alle Bauteile sauber ins Gehäuse passen. Falls dies der Fall sein wird, könnte man auf Universalgehäuse ausweichen.

Unfall/Krankheit: Ein weiteres Risiko ist, das sich das Projekt, beziehungsweise der Ablauf verschieben oder gar verzögern könnte, falls der Student eine Krankheit oder einen Unfall erleidet.

Damit dies keinen allzu grossen Rückschlag bei der Umsetzung des Projektes mit sich bringt muss am Ende genügend Pufferzeit eingerechnet werden.

1.2.7 Qualität

Zur Prüfung der Qualität und Anforderungen des Bewässerungssystems wird am Ende ein Soll/Ist Abgleich gemacht. Zudem wird das System für mindestens eine Woche getestet, wobei eine Pflanze automatisch bewässert wird.

1.2.8 Projektentwicklung

Als erstes werden alle Bauteile verglichen und anschliessend die Besten ausgewählt, welche zur Realisierung des Projektes benötigt werden. Diese werden im Anschluss bestellt. Sobald die Teile geliefert werden, wird jedes Bauteil mit einem Test Skript (Code) getestet. Anschliessend wird das Ganze auf einem Breadboard zusammengesteckt und die Software geschrieben. Sobald die Software einwandfrei funktioniert, werden alle Teile verkabelt und gelötet. Nun wird alles vermessen, das Gehäuse im CAD entworfen und 3D gedruckt. Danach wird alles im Gehäuse eingebaut und anschliessend auf die Funktionen geprüft. Parallel zum gesamten Projekt wird der gesamte Vorgang dokumentiert. Die detaillierte Planung ist unter dem Punkt «1.4 Ablaufplanung» ersichtlich.

1.3 Zielformulierung / Erfolgskriterien

Das übergeordnete Ziel der Diplomarbeit besteht darin, ein automatisches Bewässerungssystem für Zimmerpflanzen zu entwickeln und zu realisieren. Dieses System wird mit einer Wasserpumpe, einem Bodenfeuchtigkeitssensor, einem Füllstandsensor, sowie Temperatur- und Luftfeuchtigkeitssensor ausgestattet, die alle von einem Mikrocontroller gesteuert werden. Zusätzlich wird der aktuelle Zustand der Pflanze sowohl auf einem LCD-Display als auch über ein mobiles Gerät für den Benutzer leicht zugänglich gemacht, damit er den aktuellen Zustand seiner Pflanze jederzeit anschauen kann.

Die Erfolgskriterien dieser Arbeit umfassen:

Automatische Bewässerung der Zimmerpflanze: Das System muss in der Lage sein, die Bewässerung der Zimmerpflanze basierend auf den Messungen der Sensoren und den vordefinierten Bewässerungsparametern automatisch zu steuern.

Anzeige des Pflanzenzustands auf dem LCD-Display: Das LCD-Display soll den aktuellen Zustand der Pflanze, einschliesslich Bodenfeuchtigkeit, Füllstand des Wassertanks, Temperatur und Luftfeuchtigkeit, in Echtzeit anzeigen.

Zugriff auf den Pflanzenzustand über ein mobiles Gerät: Der Benutzer soll den Zustand der Pflanze von Überall aus über sein Handy überwachen und gegebenenfalls Einstellungen anpassen können.

Integration in ein 3D-gedrucktes Gehäuse: Das gesamte System, einschliesslich aller Komponenten und Verkabelungen, wird in einem benutzerfreundlichen und ästhetisch ansprechenden 3D-gedruckten Gehäuse untergebracht, das sowohl funktional, praktisch und auch sicher ist.

Die erfolgreiche Umsetzung dieser Erfolgskriterien wird sicherstellen, dass das entwickelte automatische Bewässerungssystem effektiv und benutzerfreundlich ist und die Bedürfnisse von Zimmerpflanzen und ihren Besitzern erfüllt.

1.4 Ablaufplanung

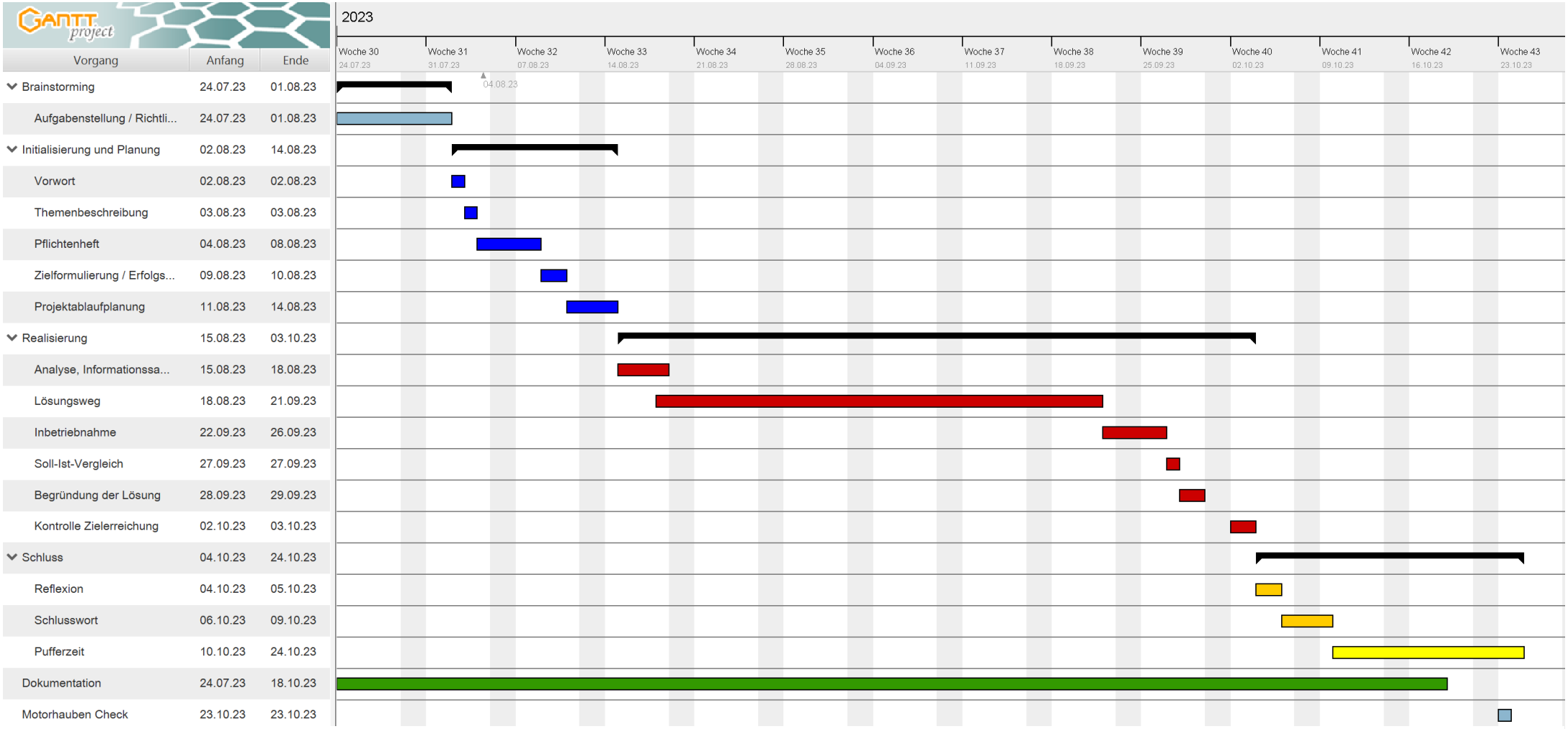


Abbildung 2 Ablaufplan

1.4.1 Beschreibung der Planung

Am Anfang wurde ein Brainstorming zur Ideenfindung gemacht. Zudem wurde die Aufgabenstellung und die Richtlinien gründlich durchgelesen. Anschliessend wurde eine Idee gefunden und diese mit dem Eingabeformular eingereicht. Nachdem das Projekt angenommen wurde, konnte die Arbeit am 01.08.2023 in Angriff genommen werden.

In der Initialisierungs- und Planungsphase wird das ganze Projekt beschrieben und ein Pflichtenheft erstellt. In diesem Pflichtenheft wird definiert, was das Bewässerungssystem können muss und welche Bedingungen in Bezug auf diese Eigenschaften erfüllt sein müssen. Danach wurden messbare Ziele und Erfolgskriterien definiert und eine Ablaufplanung erstellt

Die Realisierungsphase beginnt mit einer Analyse und Informationssammlung über die Bauteile, welche gebraucht werden. Anschliessend werden alle Teile bestellt und eine Testsoftware geschrieben, damit die Teile nach der Anlieferung sofort getestet werden können. So werden fehlerhafte Teile ohne Verzögerung aussortiert und ersetz. Nach dem Ausschliessen von Defekten wird ein komplettes Schema erstellt und die einzelnen Bauteile zusammengebaut. Nachdem alles zusammengebaut wurde, wird die Software erstellt und getestet. Sobald das Bewässerungssystem läuft, wird ein Gehäuse 3D designt, anschliessend gedruckt und das System in diesem verbaut. Daraufhin wird das Gerät in Betrieb genommen und ein Soll/Ist Abgleich soll vorgenommen werden. Zum Abschluss der Realisierungsphase wird die Lösung begründet und hinterfragt, ob alle Ziele erreicht wurden.

Den Schluss der Arbeit wird eine Reflexion des Projektes, ein Schlusswort sowie das Management Summary bilden. Parallel wird über das ganze Projekt hinweg, fortlaufend eine Dokumentation mitgeführt und aktualisiert.

Am Ende des Projektes wird ein Motorhauben Check mit dem Diplomcoach durchgeführt, damit die Qualität der Arbeit überprüft und allenfalls korrigiert werden kann.

2 Realisierung

2.1 Analyse, Informationssammlung

2.1.1 Mikrocontroller

2.1.1.1 ESP32

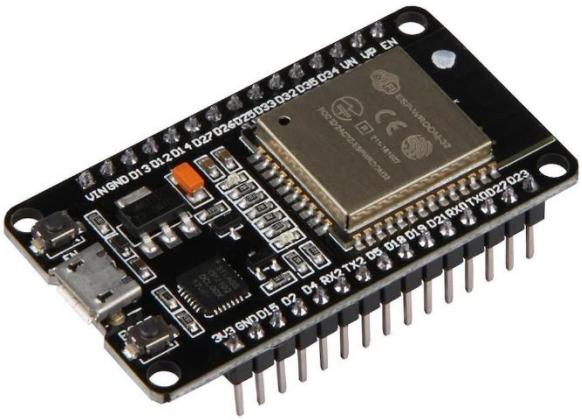


Abbildung 3 ESP32

Der ESP32 ist ein Wifi fähiger Mikrocontroller. Dies ist für mein Projekt von grosser Bedeutung, da ich das Gerät mit dem Wifi verbinden muss, um den Zustand meiner Pflanze mit dem Handy einsehen zu können. Zudem ist er Bluetooth kompatibel, welches benötigt wird, um das Gerät via Smartphone mit dem Wifi verbinden zu können. Er hat eine Taktrate von 240MHz, 2 Kerne und einen Flashspeicher von 4000 Kilobyte. Somit sollte genug Speicher vorhanden sein, damit das Projekt problemlos umsetzen werden kann. Zudem verfügt das Teil über 448 kB ROM und 520 kB SRAM. Der Chip hat 25 digitale Ein- oder Ausgänge, welche im Code als solche definiert werden. Das sind genug Anschlüsse, damit man all die digitalen Sensoren sowie das Relais ansteuern kann. Zudem können 16 von den digitalen Kontakten ebenfalls als analoge Ein- oder Ausgänge verwendet werden. Für diese Projekt benötigt man mindestens einen analogen Eingang, damit der Erdfeuchtigkeitssensor ausgelesen werden kann. Das ganze ESP32 Entwicklerboard ist auf Digitec zum Zeitpunkt der Bestellung als Aktion erhältlich und mit 12.95.- vergleichsweise kostengünstig. (Digitec ESP32, 2023)

2.1.1.2 ESP8266

Abbildung 4 ESP8266

Der ESP8266 ist ebenfalls ein Wifi fähiger Mikrocontroller, welches dringlichst benötigt wird, um die Projektziele zu erreichen. Er ist leider nicht Bluetooth fähig, was einen klaren Nachteil gegenüber dem ESP32 darstellt. Der Chip hat 32 kB SRAM und 512 kB bis 1MB ROM, wobei dieses Detail abhängig vom Hersteller ist. Mit einem Flashspeicher von 512 kB bis 4000 kB ist auch bei diesem Chip genügen Speicher vorhanden. Er verfügt über 13 digitale Ein- oder Ausgänge und hat lediglich einen analogen Eingang, was von Nachteil ist, wenn das Projekt in Zukunft erweitert werden soll. Das ESP 8266 Entwicklerboard ist ebenfalls auf Digitec erhältlich und ist für 25.90.- im Vergleich zum ESP32 deutlich teurer, da es weniger Ein- und Ausgänge als auch analoge

Anschlüsse hat. Da dieser Chip gegenüber dem ESP32 klar im Nachteil ist, kommt er für dieses Projekt also nicht in Frage. (Digitec ESP8266, 2023)

2.1.1.3 Arduino Mega 2560

Der Arduino Mega 2560 ist kostenlos, da er bereits vom Funduino Kit enthalten ist, welches für den Unterricht an der Teko verwendet wurde. Er ist jedoch nicht Wifi und Bluetooth kompatibel. Dafür müssten zusätzlich extra Module gekauft werden. Dieses Bauteil hat 4 kB ROM, 8 kB SRAM sowie einen Flashspeicher von 258 kB. Diese Kapazitäten sollte für das Projekt ebenfalls ausreichen, könnte jedoch bei einer unsauberen Programmierung knapp werden. Da der Arduino Mega 2560 aber leider nicht mit Bluetooth oder Wifi kompatibel ist, kommt dieser Mikrocontroller im Rahmen dieses Projekts leider auch nicht in Frage. (Arduino, 2023)

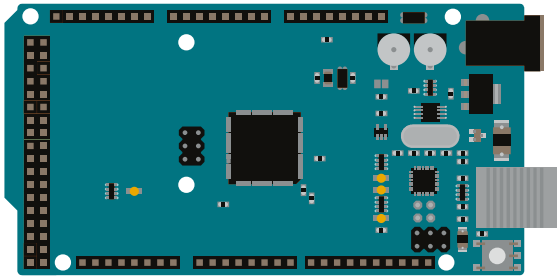


Abbildung 5 Arduino Mega 2560

2.1.1.4 Fazit Mikrocontroller

Beim Vergleichen der verschiedenen Mikrocontroller ESP32, ESP 8266 und Arduino Mega 2560 ist der ESP32 mit Abstand der geeignetste, da er Bluetooth und Wifi kompatibel ist und genügen digitale Ein- und Ausgänge hat. Mit diesem Mikrocontroller ist es ebenfalls möglich, eine Web-App für das Handy zu erstellen, welches für dieses Projekt gebraucht wird. Zudem hat er mehrere analoge Eingänge, welche sich in Zukunft als günstig erweisen könnten, falls das Projekt erweitert werden soll.

2.1.2 Mikroprozessor

2.1.2.1 Raspberry Pi 4

Der Raspberry Pi 4 ist einer der neusten Computerplatinen von Raspberry Pi. Er hat einen Arbeitsspeicher von 2 GB und eine Taktrate von 1500 MHz. Zudem ist er Wifi und Bluetooth kompatibel, welches zur Realisierung des Projektes benötigt wird. Er verfügt über genügend digitale Ein- und Ausgänge. Leider hat er keine analogen Eingänge. Da diese zentral für den Erdfeuchtigkeitssensor sind, stellt dieser Umstand ein grosses Problem dar. Man könnte einen Analog/Digitalwandler kaufen, damit dieses Problem umgangen wird. Jedoch ist der Raspberry Pi der falsche Controller für dieses Projekt, da man ihn als kleinen Server

Abbildung 7 Raspberry Pi 4

Abbildung 6 Analog/Digital

beziehungsweise Computer aufsetzen könnte und diese Eigenschaft für die Realisierung nicht benötigt wird. Zudem ist der Raspberry Pi 4 mit einem Preis von 76.90.- sehr teuer. Ein weiterer Nachteil wäre, dass ein Analog/Digitalwandler gekauft werden müsste, welcher zusätzlich 13.- kostet. (Digitec Analog/Digital Wandler, 2023)

2.1.2.2 Raspberry Pi Zero

Der Raspberry Pi Zero ist eine weitere Computerplatine von Raspberry Pi. Er ist Wifi und Bluetooth kompatibel, welches ihm einen klaren Vorteil im Vergleich zum Raspberry Pi 4 verschafft. Er hat 512 MB RAM und eine Taktrate von 1000 MHz. Zudem verfügt er genügend D¨digitale Ein- und Ausgänge. Jedoch besitzt auch dieser Controller keinen analogen Eingang, welcher für dieses Projekt benötigt wird. Man müsste auch für dieses Bauteil zusätzlich einen Analog/Digitalwandler kaufen, damit man die Daten vom Erdfeuchtigkeitssensor auslesen kann. Somit ist der Raspberry Pi Zero ebenfalls nicht für dieses Projekt geeignet. Diesen Controller ist für 26.90-. auf Digitec erhältlich. (Digitec Raspberry Pi Zero, 2023) Zudem müsste der Analog/Digitalwandler gekauft werden, welcher zusätzlich 13.- kostet. (Digitec Analog/Digital Wandler, 2023)

Abbildung 8 Raspberry Pi Zero

2.1.2.3 Banana Pi

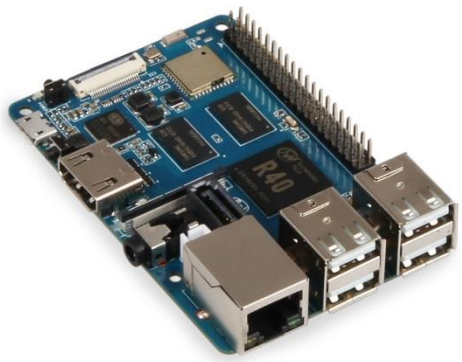


Abbildung 9 Banana Pi

Der Banana Pi ist ein kleiner Single Board Computer. Er ist ebenfalls, wie der Raspberry Pi Zero, Bluetooth und Wifi kompatibel. Er besitzt 1 GB RAM und eine Taktrate von 1200 MHz. Zudem verfügt er genug digitale Ein- und Ausgänge. Leider hat auch er wie der Raspberry Pi 4 und Raspberry Pi Zero keine analogen Eingänge. Hier müsste ebenfalls ein zusätzliches Modul gekauft werden, um das Signal von einem analogen in ein digitales umzuwandeln. Dieser ist ebenfalls nicht für das Projekt geeignet, da er keine analogen Eingänge besitzt und er, wie der Raspberry Pi 4, auch als kleiner Server oder Computer verwendet werden kann. Man könnte diesen Controller für 65.95-. auf Digitec kaufen, was für dieses Projekt teuer ist. (Digitec Banana Pi, 2023) Zudem muss

auch hier der Analog/Digitalwandler gekauft werden, welcher zusätzlich 13.- kosten würde. (Digitec Analog/Digital Wandler, 2023)

2.1.2.4 Fazit Mikroprozessoren

Bei den Mikroprozessoren ist der Raspberry Pi Zero am besten für dieses Projekt geeignet. Er ist kostengünstiger als die anderen genannten Controller und zudem Wifi und Bluetooth fähig, was ein zentrales Kriterium darstellt. Jedoch ist bei allen Modellen das Problem, das sie über keinen analogen Eingang verfügen.

2.1.3 Mikrocontroller vs Mikroprozessor

Der ESP32 ist dem Raspberry Pi Zero aufgrund von seinen Eigenschaften in Bezug auf dieses Projekt in allen Fällen überlegen. Es sind zwar beide Wifi und Bluetooth fähig. Jedoch besitzt der Raspberry Pi Zero keine analogen Eingänge, was ihm einen klaren Nachteil verschafft. Zudem ist der ESP32 mit 12.95.- kostengünstiger. Einen Vergleich bietet der Raspberry Pi Zero mit einem Kostenpunkt von 26.90.- mit zusätzlichen 13.- für das zusätzlich benötigte Modul.

Somit wird der ESP32 der Controller sein, welcher für dieses Projekt verwendet wird.

2.1.4 Bodenfeuchtigkeitssensor

2.1.4.1 VH 400

Abbildung 10 VH400

Der VH 400 ist ein analoger Bodenfeuchtigkeitssensor von Vegetronix. Es handelt sich um einen kapazitiven Sensor. Dies hat den Vorteil, dass dieser nicht rostet. Er benötigt weniger als 13 mA Strom, kann mit 3.5 bis 20 Volt Gleichstrom gespiesen werden und ist bei 25 Grad Celsius auf 2% genau. Er verfügt über einen analogen Output von 0 bis 3 Volt und wird mit 3 Kabeln angeschlossen. Zudem hat er ein Wasserdichtes Gehäuse. Er kann in drei verschiedenen Varianten bestellt werden, indem man zwischen einem 2, 5 oder 10 Meter Kabel wählen kann. Er kostet, je nachdem welche Kabellänge man wählt, zwischen 40.80.- und 56.20.-. Dieser Sensor ist somit teuer, jedoch von guter Qualität. (Vegetronix VH400, 2023)

2.1.4.2 Kapazitiver Bodenfeuchtigkeitssensor

Abbildung 11 Bodenfeuchtigkeitssensor

Hierbei handelt es sich um einen kapazitiven Bodenfeuchtigkeitssensor von DFRobot. Dieser rostet ebenfalls nicht wie der VH 400. Er benötigt gleichermassen wenig Strom und kann mit 3.3 bis 5.5 Volt Gleichstrom gespiesen werden. Da es keine genauen Angaben zur Genauigkeit gibt, ist unklar, ob dieser Sensor genauer oder ungenauer als der VH 400 messen würde. Er besitzt ebenfalls einen analogen Output von 0 bis 3 Volt und wird mit 3 Kabeln angeschlossen. Dieser Sensor verfügt leider über kein Wasserdichtes Gehäuse und läuft somit Gefahr einen Kurzschluss durch Wasserkontakt zu erleiden. Ein weiterer Nachteil ist, dass dieser Sensor nur mit einem 30 Zentimeter Kabel geliefert wird. Mit einem Preis von 17.90.- ist er jedoch eine kostengünstige Variante zum DH 400. Zudem ist dieser Sensor auf Digitec schnell erhältlich. (Digitec Kapazitiver Bodenfeuchtigkeitssensor, 2023)

2.1.4.3 Arduino Kit Bodenfeuchtigkeitssensor

Im Arduino Kit ist ein Bodenfeuchtigkeitssensor vorhanden. Er hat auch eine Betriebsspannung von 3.3 bis 5.5 Volt und braucht sehr wenig Strom. Ebenfalls hat er einen analogen Output von 0 bis 3 Volt und wird mit 3 Kabeln angeschlossen. Dieser Sensor wäre kostenlos, da er sich bereits im Besitz des projektdurchführenden Studenten befindet. Es handelt sich hier ebenfalls um einen kapazitiven Sensor. Jedoch ist dieser nicht vor Rost geschützt und würde nach einigen Betriebsstunden rosten. Dies wäre sehr schädlich für die Pflanze. Zudem ist dieser ebenfalls nicht wasserdicht und könnte somit auch einen Kurzschluss erleiden.

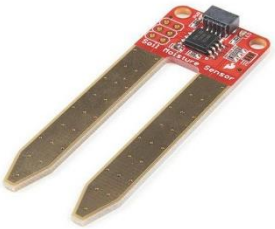


Abbildung 12 Arduino-Kit Sensor

2.1.4.4 Fazit Bodenfeuchtigkeitssensoren

Der Sensor vom Arduino Kit kommt nicht in Frage, da dieser rosten und der Pflanze schaden könnte. Der kapazitive Sensor von DFRobot ist zwar preiswerter als der VH 400, jedoch ist er nicht Wasserfest. Zudem verfügt nur über ein sehr kurzes Anschlusskabel. Der VH 400 kostet mehr, ist jedoch den anderen Sensoren in allen Punkten überlegen und man verfügt über Angaben zur Genauigkeit.

Somit ist der VH 400 mit einem 2 Meter Kabel der Bodenfeuchtigkeitssensor, welcher für das Bewässerungssystem verwendet werden soll. (Distrelec Arduino Bodenfeuchtigkeitssensor, 2023)

2.1.5 Füllstandsensor

2.1.5.1 Kontaktloser Kapazitiver Füllstandsensor

Abbildung 13 Füllstandsensor

Dieser Sensor ist von DFRobot. Es handelt sich hier um einen kapazitiven, kontaktlosen Füllstandsensor. Eine Stromversorgung von 5 Volt Gleichstrom und nur 25 Mikroampere werden von diesem Bauteil benötigt. Er wird mit 3 Kabeln angeschlossen. Dabei handelt es sich um ein digitales Signal. Er kann in einer Umgebung von -25 bis 125 Grad Celsius eingesetzt werden. Er misst durch eine Wand von 1 bis 3 Millimeter dicke. Diese darf lediglich nicht aus Metall sein. Da dieser Sensor jedoch nur an einem Punkt resp. einer Höhe im Tank die Verfügbarkeit des Wassers messen kann, werden für das Projekt mindestens 5 Stück benötigt, um den Füllstand präzise messen zu können. Zudem hat er ein Kabel, welches 30 Zentimeter lang ist und kann direkt bei DFRobot für 12.60.- pro Stück bestellt werden. (DFRobot Füllstandsensor, 2023)

2.1.5.2 Arduino Kit Füllstandsensor

Abbildung 14 Arduino Kit Füllstand

Im Arduino Kit ist ein Füllstandsensor enthalten. Er hat ebenfalls eine Betriebsspannung von 3 bis 5 Volt Gleichstrom und wird mit 3 Kabeln angeschlossen. Dabei handelt es sich um einen Digitales Signal. Dieser Sensor währe kostenlos, da er bereits vom Studenten erstanden wurde. Es handelt sich hier um einen kapazitiven Füllstandsensor. Jedoch muss dieser Sensor im Wassertank eingebaut werden und wird nach gewisser Zeit rosten. Da die Pflanzen nach einiger mit rostigem Wasser gegossen werden würden und sich die Pumpe durch dir Partikel im Wasser verstopfen oder gar beschädigt werden könnte, kommt dieses Bauteil weniger in Betracht. (Digikey Arduino Füllstandsensor, 2023)

2.1.5.3 Schwimmsensor

Abbildung 15 Schwimmsensor

Dieser Sensor ist ebenfalls von DFRobot. Es handelt sich ebenfalls um einen Schwimmsensor, welcher im Tank eingebaut werden muss. Er Schwimmt an der Wasseroberfläche und misst somit den aktuellen Füllstand. Er hat eine Betriebsspannung von 2.5 bis 5.5 Volt Gleichstrom und eine Betriebstemperatur von -20 bis 100 Grad Celsius und ist ein digitaler Sensor. Dieser kann lediglich messen, ob der Füllstand des Tankes hoch oder niedrig ist. Daher ist er eher ungenau und zudem aufwendig einzubauen. Er kostet 9.10.- bei Digikey und ist somit jedoch preiswerter. (Digikey Schwimmsensor, 2023)

2.1.5.4 Fazit Füllstandsensoren

Der Füllstandsensor vom Arduino Kit kommt nicht in Frage, da dieser mit der Zeit rostet und die Pumpe kaputt machen könnte, sowie der Pflanze schaden oder gar töten könnte. Der Schwimmsensor von DFRobot ist eine kostengünstigere Alternative zu kontaktlosem kapazitivem Sensor. Jedoch ist dieser sehr ungenau. Der kontaktlose kapazitive Füllstandsensor hat den Vorteil, dass er nicht im Tank installiert werden muss. Er kann einfach and die Wand des Wasserbehälters geklebt werden. Somit braucht es kein Loch und dadurch besteht keine Gefahr, dass Wasser auslaufen könnte. Jedoch werden von diesem Sensor 5 Stück benötigt, was ihn im Vergleich teuer macht. Dafür kann aber der Wasserstand um einiges genauer gemessen werden.

Somit ist der kapazitive kontaktlose Füllstandsensor von DFRobot der Sensor, welcher für das automatische Bewässerungssystem verwendet wird.

2.1.6 Temperatur- und Luftfeuchtigkeitssensor

2.1.6.1 DHT 11

Der DHT 11 ist ein Temperatur- und Luftfeuchtigkeitssensor. Er benötigt eine Betriebsspannung von 3.3 bis 5.5 Volt Gleichstrom. Der Sensor benötigt 3 Kabel, die Speisung, der Ground und ein Datenkabel. Er misst die Luftfeuchtigkeit von 20 bis 80% mit einer Abweichung von 5%. Die Temperatur misst er von 0 bis 50 Grad Celsius mit einer Abweichung von 2 Grad. Dies ist eher ungenau. Dafür ist der Sensor preiswert und kostet bei Digitec nur 7.90.-. (Digitec DHT11, 2023)

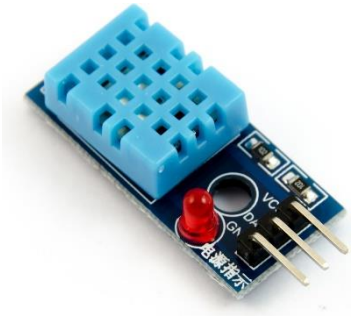


Abbildung 16 DHT11

2.1.6.2 DHT 22

Abbildung 17 DHT22

Dieser Sensor, der DHT 22, ist der grosse Bruder vom DHT11. Er misst ebenfalls die Temperatur und Luftfeuchtigkeit, ist aber um einiges genauer. Er benötigt eine Betriebsspannung von 3.3 bis 5.5 Volt Gleichstrom. Der Sensor benötigt 3 Kabel, die Speisung, der Ground und ein Datenkabel. Er misst die Luftfeuchtigkeit von 0 bis 99.9% mit einer Abweichung von 2 bis 5%. Die Temperatur misst er von -40 bis 80 Grad Celsius mit einer Abweichung von 0.5 Grad. Dies ist um einiges genauer als der DHT11. Jedoch kostet der DHT22 deutlich mehr als der DHT11 und liegt auf Digitec bei einem Kostenpunkt von 16.95.-. (Digitec DHT22, 2023)

2.1.6.3 AM2315C

Abbildung 18 AM2315C

Dieser Sensor AM2315C ist ein robuster Temperatur- und Feuchtigkeitssensor, welcher für Ausseneinsätze geeignet ist. Er hat eine Betriebsspannung von 3.5 bis 5.5 Volt Gleichstrom. Die Luftfeuchtigkeit misst er bei 25 Grad Celsius mit einer Abweichung von 2 Grad. Die Temperatur misst der von -40 bis 80 Grad Celsius mit einer Abweichung von 0.1 Grad. Dieser Sensor kann an einer Wand befestigt werden und sieht zudem ästhetisch aus. Da dieser Sensor ausschliesslich für den Ausseneinsatz ausgelegt ist, eignet er sich nicht für dieses Projekt. Er ist zudem teurer als die DHT-Sensoren und kostet 48.90.- bei Galaxus. (Galaxus AM2315, 2023)

2.1.6.4 Fazit Temperatur- und Luftfeuchtigkeitssensoren

Der DHT 11 ist sehr ungenau und kommt deshalb für dieses Projekt nicht in Frage. Der AM2315C ist ebenfalls ungeeignet, da der für den Ausseneinsatz gebaut und zudem auch zu teuer ist. Im Preis-Leistungs-Vergleich schneidet der DHT 22 um einiges besser ab. Zudem ist er genauer als der DHT 11 und günstiger als der AM2315.

Somit wird der DHT 22 für dieses Projekt verwendet, da er sich durch ein gutes Preis-Leistungs-Verhältnis auszeichnet und sehr genau ist.

2.1.7 Wasserpumpe

Abbildung 19 Pumpe

Bei der Wasserpumpe gab es keine all zu grosse Auswahl, da es für 3 bis 6 Volt Gleichstrom hauptsächlich die gleichen Pumpen gibt, welche lediglich unterschiedliche Bezeichnungen haben. Sie werden unter dem Namen Taucherpumpe oder Micro Wasserpumpe verkauft. In diesem Projekt wird eine Pumpe von der Bastelgarage bestellt, welche 8.90.- pro Stück kostet. Sie hat einen Arbeitsstrom von 130 bis 220 mA und fördert 80 bis zu 100 Liter Wasser pro Stunde. Zudem wird sie direkt mit dem passenden Schlauch geliefert, was einen deutlichen Vorteil darstellt. Sie entspricht dem Schutzgrad IP68, ist somit gegen Staub geschützt und zudem wasserdicht bis zu 1.5 Meter Tiefe und beugt so Kurzschlüssen aufgrund von Wasserkontakt vor. (Bastelgarage Wasserpumpe, 2023)

2.1.8 Relais

Abbildung 20 Relais

Für das Projekt wird lediglich ein Relais benötigt, welches die Pumpe ansteuert. Hierfür wird das Relais vom Arduino Kit verwendet. Im Online-Vergleich gab es keine weiteren nennenswerte Modelle zur Auswahl, da es sich nur um Module, welche 2 oder mehr Relais haben, handelte. Da für diese Projekt nur 1 Relais im Rahmen dieses Projekts gebraucht wird, ist dieses völlig ausreichend. Es hat eine Betriebsspannung von 5 Volt DC. (Digitec Relais, 2023)

2.1.9 LCD-Display

2.1.9.1 Arduino Kit LCD-Display

Im Arduino Kit ist ein LCD-Display vorhanden. Dieses ist somit kostenlos und muss nicht zusätzlich bestellt werden. Zudem wurde mit diesem Display bereits gearbeitet und es besteht das Grundverständnis zur Programmierung. Bei der Programmierung kann eine Bibliothek herunterladen werden, welche den Prozess vereinfacht. Zudem reicht dieses Display aus, um die Temperatur- und Luftfeuchtigkeit, den Füllstand und die Bodenfeuchtigkeit der Pflanze anzuzeigen. Dieses Display hat zudem eine Betriebsspannung von 5 Volt DC und benötigt kaum Strom. (Berrybase LCD Display, 2023)

Abbildung 21 LCD-Display

2.1.9.2 TFT-Display

Abbildung 22 TFT-Display

Eine Alternative zum LCD-Display wäre ein TFT (Thin Film Transistor) Display. Es hat ebenfalls eine Betriebsspannung von 5 Volt DC. Dieses ist in seiner Handhabung im Vergleich zum LCD jedoch sehr komplex. Hinzu kommt, dass kein Vorwissen vorhanden ist und noch nie mit so einem Display gearbeitet worden ist. Es besteht zudem noch die Unsicherheit, ob dieses überhaupt mit einem Mikrocontroller kompatibel ist. (Digikey TFT Display, 2023)

2.1.9.3 Entscheidung

Das TFT-Display ist sehr komplex und es besteht die Möglichkeit das es nicht mit einem Mikrocontroller kompatibel sein könnte. Das LCD-Display ist mit einem Mikrocontroller kompatibel, da es bereits für andere Projekte verwendet wurde.

Somit wird in diesem Projekt das LCD-Display verwendet, da es aufgrund seiner Simplizität besticht und garantiert werden kann, dass die Anforderungen an die Funktionen erfüllt werden. Zudem ist dieses bereits im Arduino Kit vorhanden und muss nicht zusätzlich beschaffen werden.

2.1.10 Netzteil und Einspeisung

Als Netzteil wird ein normales Netzteil von einem Samsung Handy verwendet. Dies liefert 5 Volt DC und 2 Ampere, welches für das gesamte Projekt ohne Probleme reicht.

Abbildung 23 Netzteil

Dazu wird ein USB zu USB-Kabel verwendet, welches ebenfalls bereits vorhanden ist.

Abbildung 24 USB zu USB

Beim Gerät wird eine USB-Buchse eingebaut, die auch schon verfügbar ist.

Abbildung 25 USB-Buchse

2.1.11 Stückliste Einkauf

Bauteil	**Menge**	**Lieferant**	**Preis + Lieferkosten in CHF**
ESP32	2	Digitec	25.90.-
VH 400	1	Vegetronix	40.80-. + 6.90.-
Kontaktloser Kapaziti-ver Füllstandsensor	5	DFRobot	63.- + 11.70.-
DHT 22	2	Digitec	33.90.-
Wasserpumpe	2	Bastelgarage	17.80.- + 3.90.-
			Total: 203.90.-

Tabelle 1: Stückliste Einkauf

Alle Bauteile, welche eingekauft werden müssen, kosten zusammengerechnet 203.90 Schweizer Franken. Da der VH 400 Bodenfeuchtigkeitssensor und die kontaktlosen Kapazitiven Füllstandsensoren von DFRobot aus den USA kommen, werden diese später eintreffen als die anderen Bauteile. Die Teile von Digitec und der Bastelgarage werden innerhalb von 2 bis 3 Werkstagen eintreffen. Zudem werden die Teile von Digitec ohne Lieferkosten geliefert, da der Mindestbestellwert erreicht wurde. Die restlichen Bauteile aus den USA könnten 2 bis 3 Wochen benötigen, bis sie ankommen werden. Bis spätestens Ende August sollte jedoch alles geliefert worden sein. Zudem wurden vom VH 400 nur ein Sensor bestellt, da dieser im Vergleich mit den anderen Teilen eher teuer ist. Bei den kapazitiven Füllstandsensoren wurde ebenfalls nur die benötigte Menge bestellt, da der Preis pro Stück 12.60 Franken beträgt. Sollten diese Bauteile defekt sein könnte im Notfall auf schlechtere und günstigere Varianten ausgewichen werden.

2.1.11.1 Stückliste Bereits vorhandene Teile

Bauteil	Menge
Relais	1
LCD-Display	1
Samsung Netzteil	1
USB zu USB-Kabel	1
USB-Buchse	1
M12 Kabelverschraubung	1
4/6mm Silikonschlauch	1 x 2 Meter
Loch Platine	1

Tabelle 2: Stückliste Vorhanden

Diese Bauteile sind bereits auf Lager und müssen nicht bestellt werden. Zudem kann bereits mit der Programmierung begonnen werden, sobald der ESP32 von Digitec geliefert wurde. Dies sollte nur wenige Tage dauern, da Digitec laut Angabe sehr schnell liefern wird.

2.2 Schema

Die Folgenden Schemas wurden mit der Software von Fritzing gezeichnet. Fritzing ist eine open-source Software, welche zum Zeichnen von Schemas für Mikrocontroller verwendet werden kann. (Fritzing, 2023)

Damit klar ersichtlich ist, welche Kontakte vom ESP-WROOM-32 vorhanden sind und für was sie gebraucht werden können, wurde folgendes Pinout von Electronicshub verwendet:

(Electronicshub, 2023)

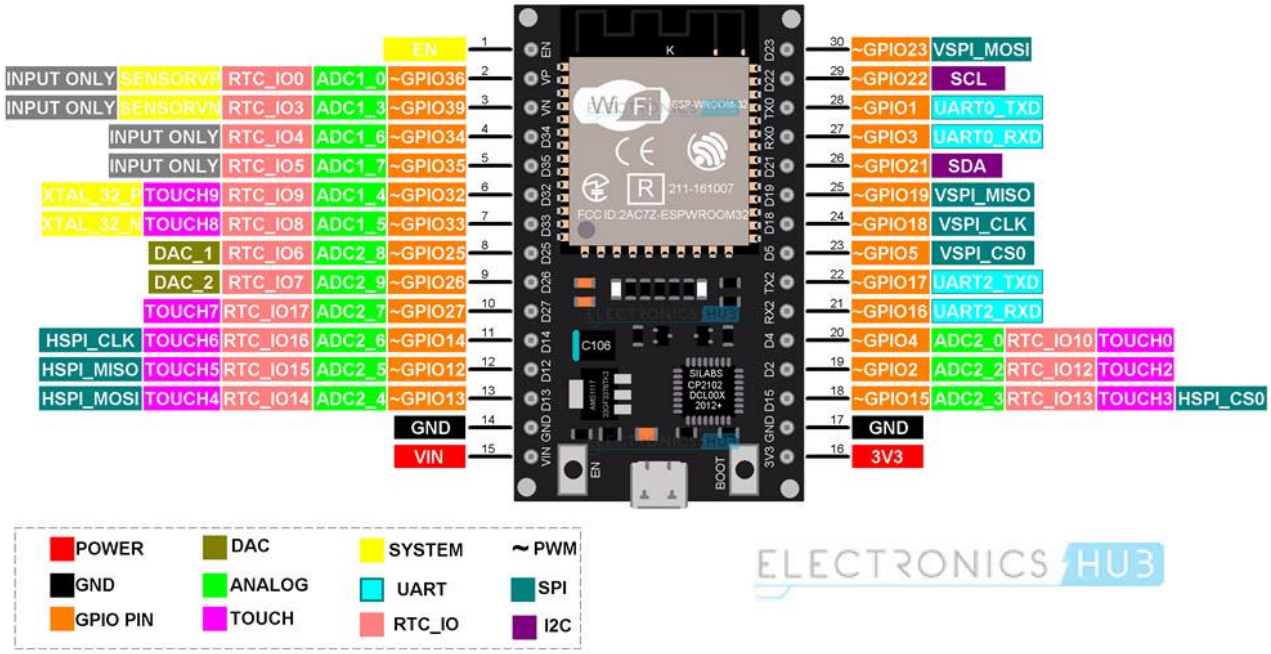


Abbildung 26 Pinout ESP32

2.2.1 Schema Relais und Pumpe

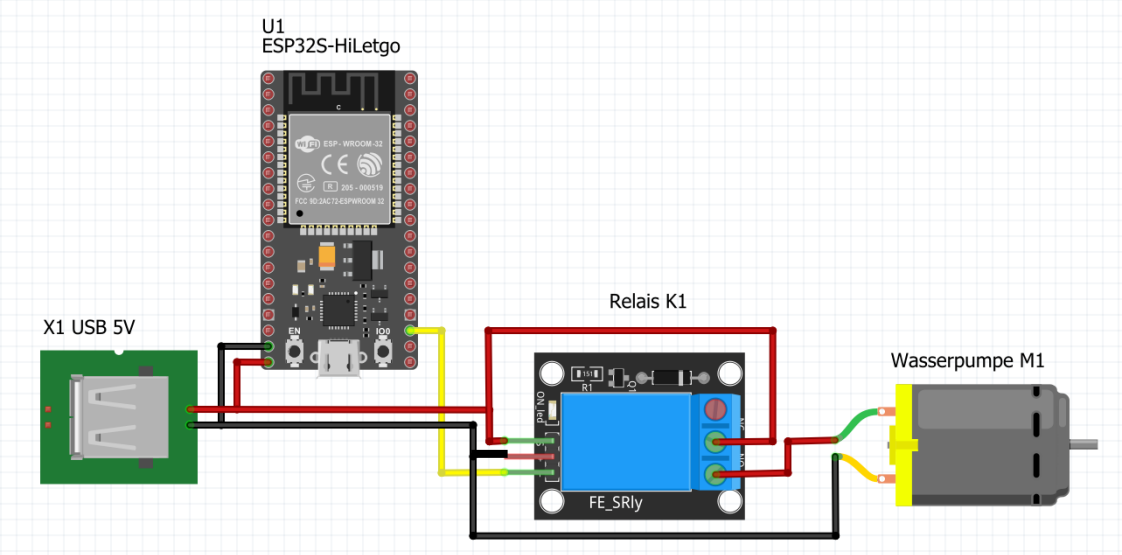


Abbildung 27 Schema Relais

In diesem Teilschema wurde der ESP32 auf den Kontakten «VIN» und «GND» mit einer Spannunng von 5 Volt Gleichstrom eingespeist. Das «Relais K1» wurde ebenfalls auf den Kontakten «VCC» und dem «COM» vom Wechselkontakt mit 5 Volt versorgt. Zudem wurde der «GND» auf den Ground der Einspeisung gehängt. Das Relais wird über den «IN» Kontakt vom EPS32 über den Digitalen GPIO15 Pin angesteuert (gelbes Kabel) und ein- oder ausgeschaltet. Der positive Kontakt der Pumpe wurde auf den «NO» (normaly open) Kontakt vom Relais und der «GND» auf den Ground der Einspeisung gehängt

2.2.2 Schema DHT22

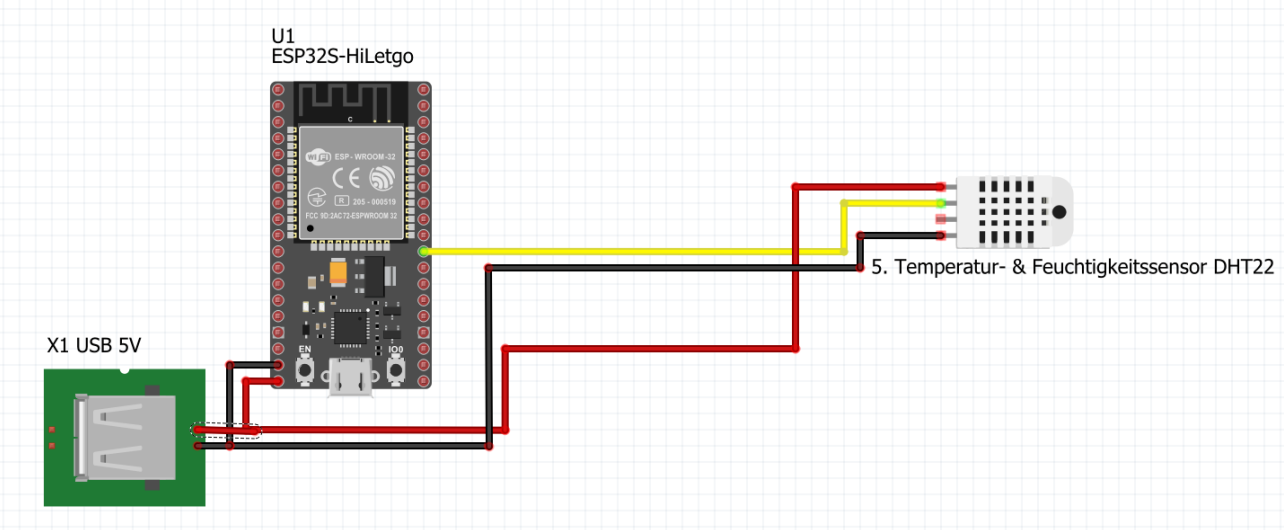


Abbildung 28 Schema DHT22

Hier wurde der ESP32 ebenfalls mit 5 Volt versorgt. Der Temperatur- und Luftfeuchtigkeitssensor «DHT 22» wurde ebenfalls auf den Kontakten «VCC» und «GND» an der Einspeisung angeschlossen. Um die Werte vom DHT 22 auszulesen, muss zum Schluss noch der Kontakt «DAT» am ESP32 auf dem Digitalen GPIO18 Pin angeschlossen werden.

2.2.3 Schema Füllstandsensoren

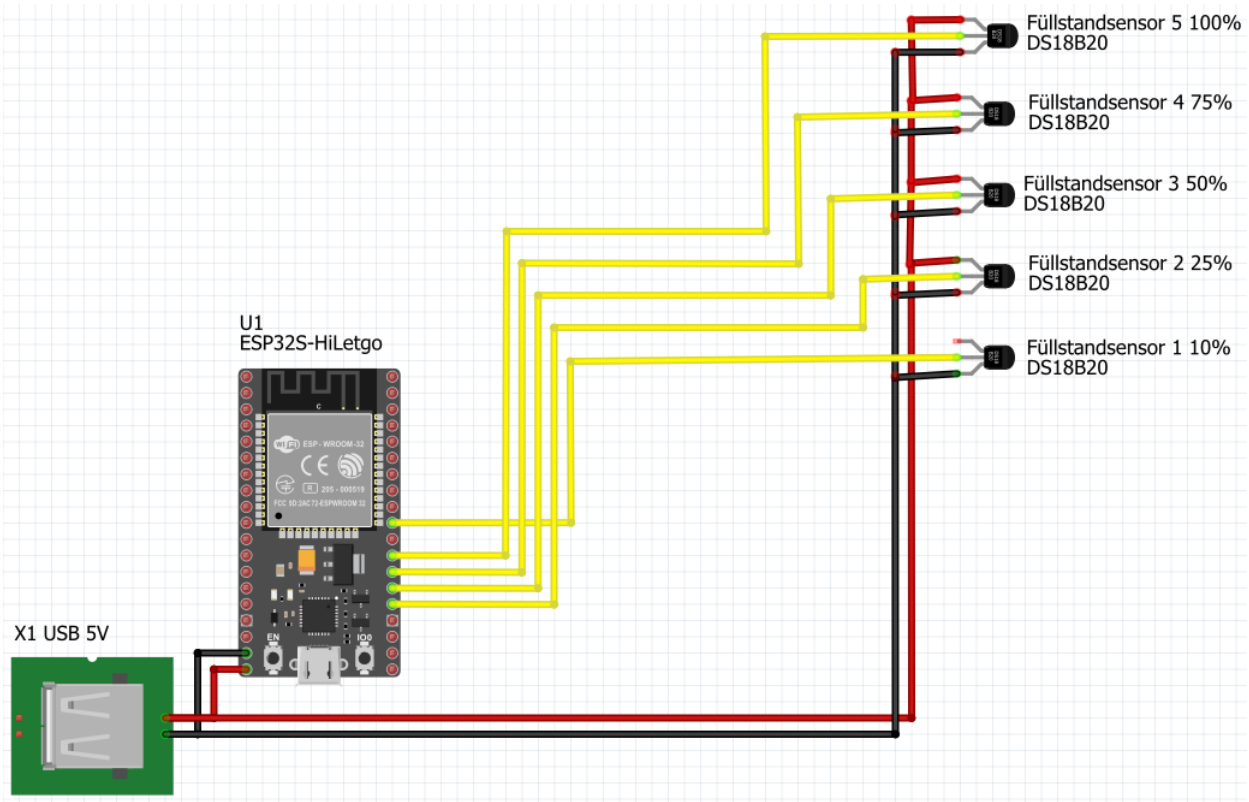


Abbildung 29 Schema Füllstandsensoren

Da für den in diesem Projekt gebrauchten Füllstandsensor leider kein Bild auf Fritzing verfügbar war, wurde ein anderer Digitaler Sensor für dieses Schema verwendet. Die Fünf Füllstandsensoren wurden alle mit 5 Volt DC gespiesen. Zudem musste der Digitale kontakt von allen Sensoren jeweils einzeln auf den ESP32 geführt werden. Für den Füllstandsensor 1, welcher den Wasserstand von 10% anzeigt wurde der digitale Pin GPIO19 verwendet. Der Füllstandsensor 2 25% wurde auf den GPIO4 Pin, Füllstandsensor 3 50% wurde auf den GPIO16 Pin, Füllstandsensor 4 75% wurde auf den GPIO17 Pin und Füllstandsensor 5 100% wurde auf den GPIO5 Pin angeschlossen. Somit kann bei jedem Sensor das digitale Signal ausgelesen werden und angezeigt werden, wie hoch der Füllstand des Wassertanks ist.

2.2.4 Schema Bodenfeuchtigkeitssensor

Für den Bodenfeuchtigkeitssensor VH 400, welcher für dieses Projekt verwendet wird, gibt es ebenfalls kein Bild für die Fritzing Software. Deshalb wurde hierfür ein anderer kapazitiver Bodenfeuchtigkeitssensor für das Schema verwendet. Der Sensor braucht auch eine Speisung von 5 Volt Gleichstrom auf den Kontakten «VCC» und «GND». Zudem muss der analoge Kontakt (gelbes Kabel) auf einen analogen Kontakt vom EPS32 angeschlossen werden. Hierfür wurde der Kontakt ADC4 vom ESP32 verwendet.

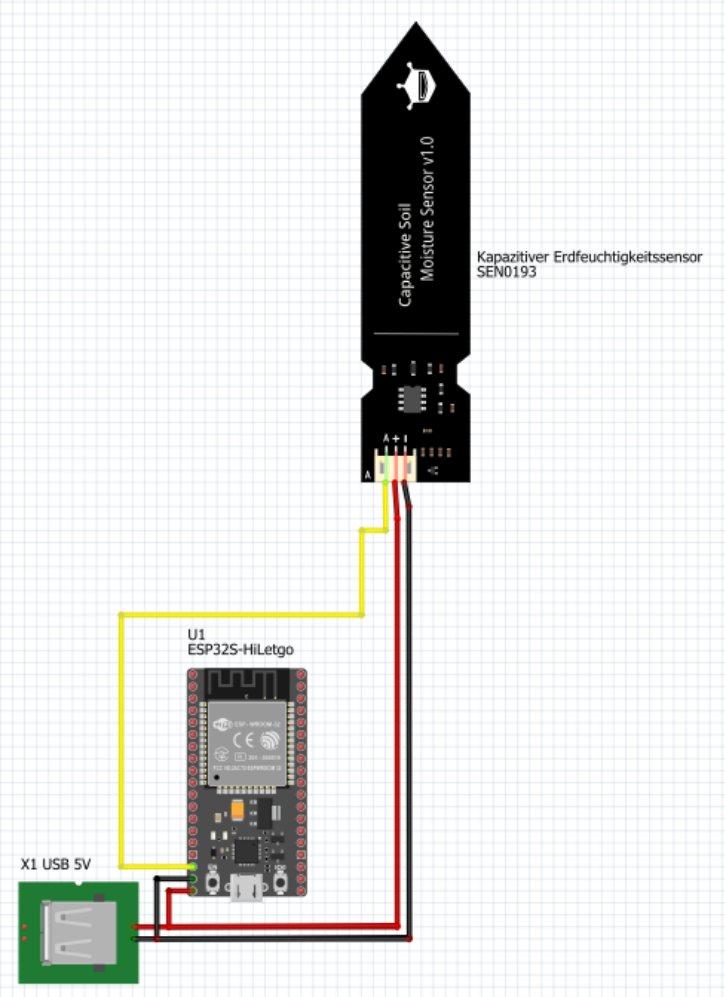


Abbildung 30 Schema Bodenfeuchtigkeitssensor

2.2.5 Schema LCD-Display

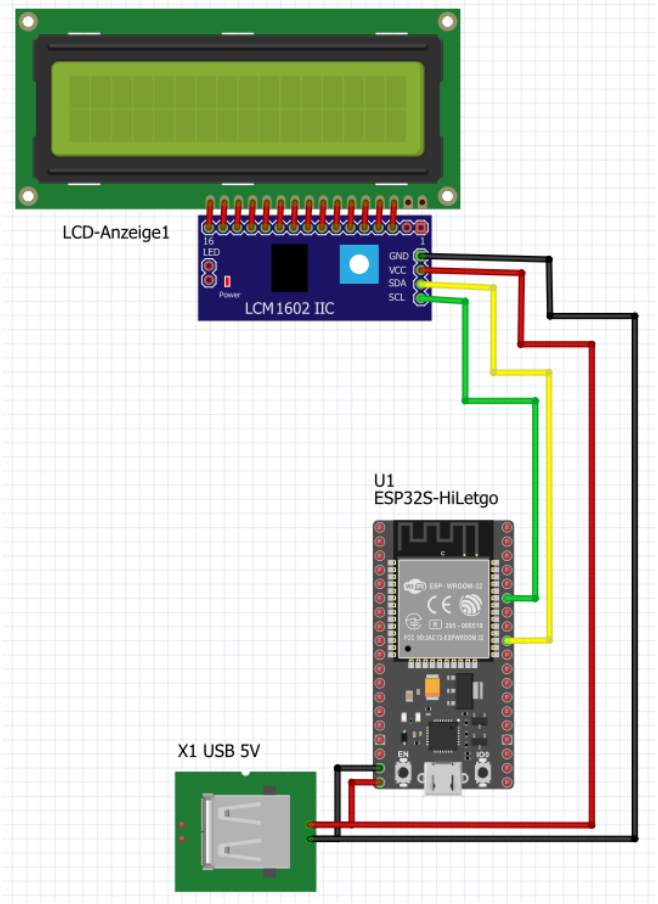


Abbildung 31 Schema LCD-Display

Anschliessend wurde noch das Teilschema vom LCD-Display gezeichnet. Hierfür musste das Display wieder mit eine 5 Volt Gleichstrom Speisung auf den Kontakten «VCC» und «GND» versorgt werden. Zudem muss der «SDA» Kontakt vom Display auf den «SDA» vom ESP32 angeschlossen werden, welcher sich den Platz mit dem GPI21 Pin teilt. Der «SCL» Kontakt vom Display muss ebenfalls auf den «SCL» Kontakt vom ESP32 angehängt werden, welcher sich den Platz mit dem GPIO22 Pin teilt. Der SDA-Kontakt ist der Serial Data Kontakt. Dies ist die Datenleitung, über welche alle Daten übermittelt werden. Der SCL-Kontakt ist der Serial-Clock Kontakt, auch Taktleitung genannt. Dieser gibt dem Display die Taktfrequenz vor.

2.2.6 Komplettes Schema

Zum Schluss wurden alle einzelnen Teile der Schemas zu einem einzigen kompletten Schema zusammengefügt.

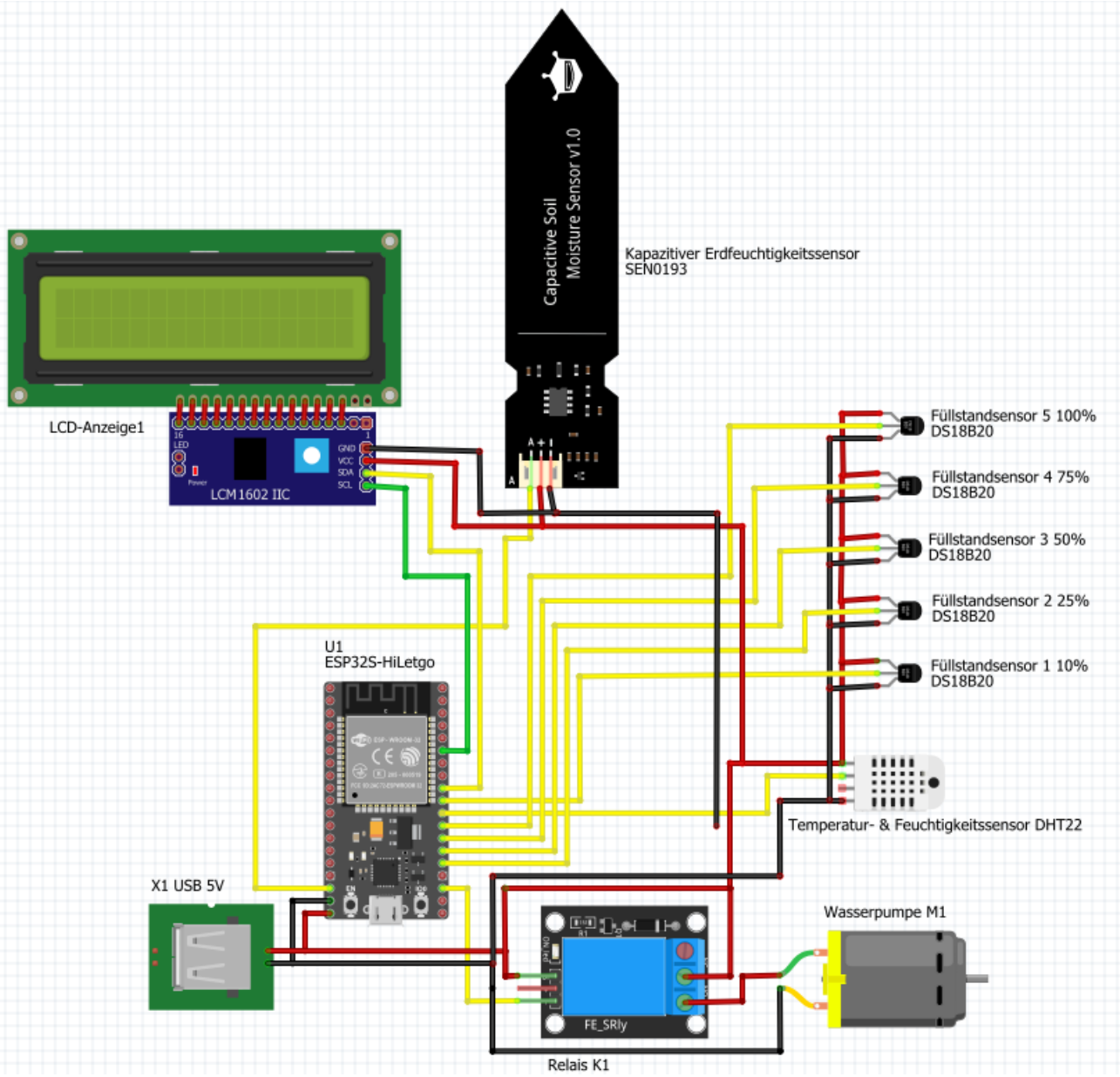


Abbildung 32 Schema komplett

2.3 Bauteile Testen

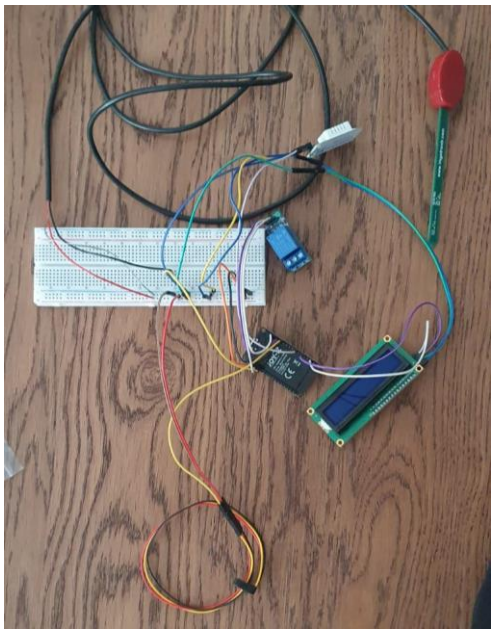


Abbildung 34 Versuchsaufbau

Nachdem das Schema fertig gezeichnet wurde, konnten alle Bauteile auf ihre Funktion getestet werden. Hierfür wurde für jedes Bauteil ein Test-Code geschrieben, welche in diesem Abschnitt erklärt werden. Der Code wurde im Programm «Visual Studio Cod» geschrieben. Zusätzlich musste noch eine Extension namens «PlatformIO» installiert werden. PlatformIO wird benötigt, damit die Programmierumgebung auf Mikrocontroller ausgelegt ist. Beim Starten der Extension kann der Mikrocontroller, welchen man benötigt, gewählt werden. Zudem sind alle Bibliotheken, welche im Programm «ArduinoIDE» vorhanden sind, ebenfalls integriert. Dies erleichtert auch das Hochladen der Software auf den Controller. Die verwendete Programmiersprache ist eine Variante von C++, welche von Arduino entwickelt wird. (Visual Studio Code, 2023)

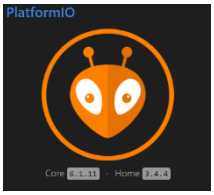


Abbildung 33 PlatformIO

2.3.1 Bibliotheken

Als Erstes wurden die Bibliotheken, welche für die Programmierung der verschiedenen Bauteile gebraucht werden, importiert. Dies wird mit dem Befehl «#include» und dem Namen der jeweiligen Bibliothek gemacht. Für dieses Projekt werden Bibliotheken für den DHT 22 und das LCD-Display gebraucht. (Visual Studio Code, 2023)

```
#include <Arduino.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

Abbildung 35 Bibliotheken

2.3.2 Relais Testscript

```
//relais
int relaisPin = 15;
```

Abbildung 36 Relais Pin

Als erstes muss der Datentyp festgelegt werden. Dieser wird als «int» (integer) definiert. Mit einem Integer wird festgelegt, dass dieser Wert als ganze Zahl ausgegeben wird. Anschliessend muss die Variable initialisiert werden, da Variablen immer sprechend benannt werden, wird sie «relaisPin» genannt. Dies nennt sich «Upper Camel Case» bzw «PascalCase». Es wird ebenfalls festgelegt, welcher Kontakt vom

ESP32 angesteuert werden muss. In diesem Fall muss der Pin 15 angesteuert werden, da an diesem das Relais angeschlossen wurde.

```
void setup() {
  Serial.begin(115200);   //setzt serial monitor geschwindigkeit auf 115200 Wichtig:muss auch in Platform.ini angepasst werden

  pinMode(relaisPin, OUTPUT); //Inizialisiert Pin das er schreiben kann

  digitalWrite(relaisPin, LOW); //setzt Pin am anfang auf low
}
```

Abbildung 37 Relais Setup

Als nächstes muss im Setup die Übertragungsgeschwindigkeit vom ESP32 definiert werden, welche 115200 bit/s sind. Wichtig ist, dass dies auch in Platform.ini angepasst wird, damit PlatformIO ebenfalls weiss, welche Geschwindigkeit benötigt wird. Danach wird der «pinMode()» vom «relaisPin» als Ausgang (OUTPUT) definiert. Zudem initialisiert es den Pin, damit dieser schreiben kann. Danach wird der Pin mit «digitalWrite()» auf «0» (LOW) gesetzt. Dies muss vorgenommen werden, damit der Zustand vom Ausgang beim Starten der Software auf «0»gesetzt wird.

```
[env:esp32dev]
platform = espressif32
board = esp32dev
framework = arduino
monitor_speed = 115200
```

Abbildung 38 Takt

Zum Schluss wird das Relais im Loop zu Testzwecken ein- und ausgeschaltet. Hierzu wird eine Verzögerung von einer Sekunde festgelegt. Die Einheit von einem «delay()» ist in Milisekunden anzugeben. Das Relais wird mit diesem Code im Ein-Sekunden-Takt Schalten, bis die Stromquelle abgehängt wird. (Arduino Forum, 2023)

```
void loop() {
  // TESTSCRIPT RELAIS
  digitalWrite(relaisPin, LOW); //setzt Pin  auf low

  delay(1000);

  digitalWrite(relaisPin, HIGH); //setzt Pin auf high
  delay(1000);
}
```

Abbildung 39 Relais Loop

2.3.3 DHT 22 Testscript

```
#define DHTPIN 18    //definiert den DHT pin auf pin 18
#define DHTTYPE DHT22 //Definiert den DHT Typ (11 oder 22)
DHT dht(DHTPIN, DHTTYPE);
```

Abbildung 41 DHT22 Pin

Als erstes muss der DHTPIN und TYPE definiert werden, damit die Bibliothek weiss, wo der Sensor angeschlossen ist und um welchen Sensor es sich handelt.

```
void setup() {
  Serial.begin(115200);   //setzt serial monitor geschwindigkeit auf 115200 Wichtig:muss auch in Platform.ini angepasst werden

  dht.begin();    //DHT wird gestartet
}
```

Abbildung 40 DHT22 Setup

Als nächstes muss wieder im Setup die Übertragungsgeschwindigkeit vom ESP32 definiert werden, welche 115200 bit/s sind. Zudem muss dies auch wieder im Plarform.ini angepasst werden. Dann wird der DHT 22 mit «dht.begin()» gestartet.

```
void loop() {
  //TESTSCRIPT DHT22
  float h = dht.readHumidity(); //Liesst die Luftfeuchtigkeit aus
  float t = dht.readTemperature(); //Liesst die Temperatur in grad Celsius aus

  //Zeigt die werte in der konsole an
  Serial.print("Humidite: ");
  Serial.print(h);
  Serial.print("%  Temperature: ");
  Serial.print(t);
  Serial.print("°C, ");
  delay(3000);
}
```

Abbildung 42 DHT22 Loop

Dann wird zum Schluss die Luftfeuchtigkeit und Temperatur vom DHT im Loop ausgelesen. Der Datentyp der beiden Werte ist ein Float, da es sich um Kommazahlen handelt. Die Luftfeuchtigkeit liest man mit «h = dht.readHumidity()» und die Temperatur mit «t = dht.readTemperatur()» aus der Bibliothek aus. Danach werden die Werte mit dem Befehl «Serial.print()» in der Konsole ausgegeben, damit man die aktuell gemessene Temperatur und Luftfeuchtigkeit bekommt. Zusätzlich wurde eine Verzögerung von 3 Sekunden eingebaut, damit man mit der Ausgabe der Werte nicht zugespammt wird. Zuerst wurde dies mit einer Verzögerung von einer Sekunde programmiert. Dabei ist es zu Fehlermeldungen gekommen. Nach zusätzlicher Recherche im Internet wurde klar, dass man eine Verzögerung von mindestens 2 Sekunden programmieren muss, da es bei Abfragen von kleiner als 2 Sekunden häufig zu Fehlermeldungen kommen kann, weil dies zu schnell ist für den ESP. (Upesy, 2023)

2.3.4 Füllstandsensoren Testscript

```
//Fuellstandsensor
int tankSensorPin1 = 19; //Tiefster Wassertanksensor 10%
```

Abbildung 43 Füllstand Pin

Der Füllstandsensor wird als integer definiert. Seine Variabel ist «tankSensorPin1» und er befindet sich auf dem GPIO19 Pin am ESP32. Für den Füllstandsensor 2 bis 5 wurde der GPIO-Pin 4,16,17 und 5 verwendet. Diese wurden ebenfalls mit dem gleichen Code getestet. Zuerst wurde anstelle vom GPIO19 der GPIO2 verwendet, was jedoch zu Problemen führte, da dieser Standardgemäss vom ESP immer auf 1 gesetzt wird.

```
void setup() {
  Serial.begin(115200);   //setzt serial monitor geschwindigkeit auf 115200 Wichtig:muss auch in Platform.ini angepasst werden

  pinMode(tankSensorPin1, INPUT); //Inizialisiert den Pin damit er lesen kann
}
```

Abbildung 44 Füllstand Setup

Dann muss erneut im Setup die Übertragungsgeschwindigkeit vom ESP32 von 115200 bit/s definiert werden. Danach wird der «pinMode()» vom «tankSensorPin1» als Eingang (INPUT) definiert. Zudem initialisiert es den Pin damit er lesen kann.

Zum Schluss wird der Zustand vom Sensor im Loop ausgegeben. Dafür wird der Befehl «Serial.println(digitalRead(tankSensorPin1))» verwendet. Damit wird der Zustand 1 oder 0 in der Konsole ausgegeben, je nachdem, ob der Sensor angibt oder nicht. (DF Robot, 2023)

```
void loop() {
  //TESTSCRIPT Fuellstandsensor
  Serial.println(digitalRead(tankSensorPin1));
  delay(1000);
  }
```

Abbildung 45 Füllstand Loop

2.3.5 Bodenfeuchtigkeitssensor Testsript

```
//Erdfeuchtigkeitssensor
int soilSensorPin = 13;
```

Abbildung 46 Bodensensor Pin

In diesem Abschnitt wird der Erdfeuchtigkeitssensor als integer definiert. Die Variabel vom Sensor lautet «soilSensorPin» und befindet sich auf dem GPIO13 Pin. Dieser Pin ist ebenfalls ein analoger Eingang und muss später noch als solcher definiert werden.

Im Loop wird der analoge Eingang vom Sensor als solcher definiert und anschliessend in der Konsole ausgegeben. Hierfür wurde eine Verzögerung (delay) von einer Sekunde verwendet, damit man kurz Zeit hat zu reagieren und den ausgegebenen Wert zu lesen. (Vegetronix VH400, 2023)

```
void loop() {
  //TESTSCRIPT Erdfeuchtigkeitssensor
  Serial.println(analogRead(soilSensorPin));
  delay(1000);
}
```

Abbildung 47 Bodensensor Loop

2.3.6 LCD-Display Testscript

```
LiquidCrystal_I2C lcd(0x27,16,2); // setzt die LCD addresse auf 0x27 für 16 Zeichen auf 2 linien
```

Abbildung 48 LCD-Display Adresse

Beim LCD-Display muss man zuerst die richtige Adresse setzen. In diesem Fall hat das Display die Adresse 0x27. Zudem kann man mit diesem Display zeitgleich 16 Zeichen auf 2 Linien anzeigen. Damit man dies in der Bibliothek vom Display definieren kann, muss man den Befehl «LiquidCrystal_I2C lcd(0x27,16,2)» verwenden. Um die Adresse von 0x27 herauszufinden, wurde ein I2C Scanner Script verwendet, welches einem die Adresse in der Konsole ausgibt.

```
void setup() {
  Serial.begin(115200);   //setzt serial monitor geschwindigkeit auf 115200 Wichtig:muss auch in Platform.ini angepasst werden

  lcd.init();     //Initialisiert LCD
  lcd.backlight();  //Hintergrundbeleuchtung wird eingeschaltet
}
```

Abbildung 49 LCD-Display Setup

Dann muss wieder im Setup die Übertragungsgeschwindigkeit vom ESP32 von 115200 bit/s definiert werden. Das LCD-Display wird mit dem Befehl «lcd.init()» initialisiert. Zudem muss die Hintergrundbeleuchtung vom Display mit «lcd.backlight()» eingeschalten werden.

```
void loop() {
  //TESTSCRIPT LCD Display
  lcd.setCursor(2,0);   //Setzt den Czrsor auf das 2. Zeichen in Linie 0
  lcd.print("Hello world!"); //Gibt Hello World auf dem Display aus
  delay(3000);
  lcd.clear(); // Leert das display
}
```

Abbildung 50 LCD-Display Loop

Zum Schluss muss im Loop der Cursor auf die richtige Position gesetzt werden. Hieurfür wird er mit dem Befehl «lcd.setCursor(2,0)» auf das 2. Zeichen und die Linie 0 gesetzt. Linie 0 steht für die 1. Zeile. Danach wird mit dem Befehl «lcd.print()» «Hello World!» auf dem Display angezeigt. Dies wiederholt sich alle 3 Sekunden, da das Display am Ende wieder geleert wird mit dem Befehl «lcd.clear()». (Lastminuteengineers, 2023)

2.4 Löten

Abbildung 51 Lötkolben

Nachdem alle Bauteile fehlerfrei getestet worden sind, müssen sie zusammengelötet werden. Dafür wurde ein Lötkolben der Marke Weller verwendet. Zum Löten wurde eine Temperatur von 330 Grad Celsius verwendet. Dieses Wissen und Können wurde von Thomas bereits in der Ausbildung zum Automatiker erarbeiten und erweist sich auch in seinem Beruf als Nützlich. Dabei werden aber nur grössere Bestandteile, wie zum Beispiel Stecker oder zwei Kabel, gelötet. Bei dieser Lötarbeit war die Schwierigkeit, dass bereits eine längere Zeit nicht mehr mit derart kleinen Komponenten gearbeitet wurde. Deshalb wird eine sehr dünne Lötspitze von 1mm verwendet, damit Präzision gewährleistet werden kann. Zudem steht eine Entlötlitze zur Verfügung, um ungewollte Lötverbindungen zu entfernen.

2.4.1 Platine bestücken

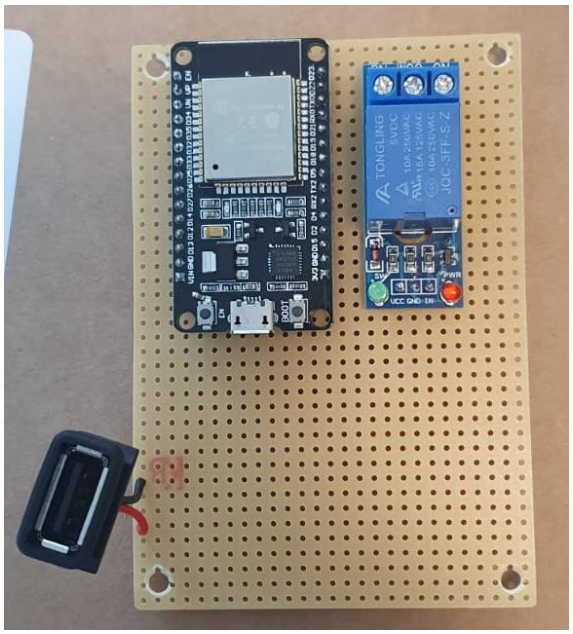


Abbildung 52 Platine bestückt

Als erstes wurde die Lock Platine mit der Trennscheibe auf die benötigte Grösse zugeschnitten. Zudem mussten in den vier Ecken 3.5mm grosse Löcher gebohrt werden, welche später zur Befestigung der Platine benötigt werden. Anschliessend wurde die Platine mit Isopropanol gereinigt, damit sie Staub- und Fettfrei ist. Dies ist nötig, damit der Lötprozess ohne weitere Probleme ausgeführt und ein sauberer Zinnfluss garantiert werden kann. Danach wurde die Platine mit dem ESP32, Relais und USB-Buchse bestückt. Diese wurden sorgfältig angelötet. Die Pins vom ESP32 und Relais waren sehr nah zusammen. Deshalb bestand das Risiko das ungewollte Zinnbrücken zwischen den einzelnen Kontakten entstehen, wenn man nicht vorsichtig genug arbeitet. Leider trat dieses Problem am Anfang auf, da das Löten kleinerer Teile viel Fingerspitzengefühl erfordert und sich daher als Herausforderung erwies. Dieses Problem wurde überwunden, indem die Verbindung mit der Entlötlitze entfernt werden konnte. Vorsichtig wurden die restlichen Kontakte ohne weitere Probleme an der Platine angelötet.

2.4.2 LCD-Display und DHT22

Beim LCD-Display und DHT22 mussten Kabel an die Pins angelötet werden. Die Kabel wurden alle gleichmässig zugeschnitten, damit sie anschliessend mit einem Kabelbinder zusammengebunden werden können und sodass sie sauber aussehen. Hierzu wurde ein rotes Kabel für die 5 Volt Speisung und ein schwarzes Kabel für den 0 Volt verwendet. Für den digitalen Kontakt am DHT22 wurde ein gelbes Kabel angelötet. Beim LCD-Display wurde ein gelbes Kabel für den SDA und ein grünes Kabel für den SCL-Kontakt verwendet.

2.4.3 Komponenten

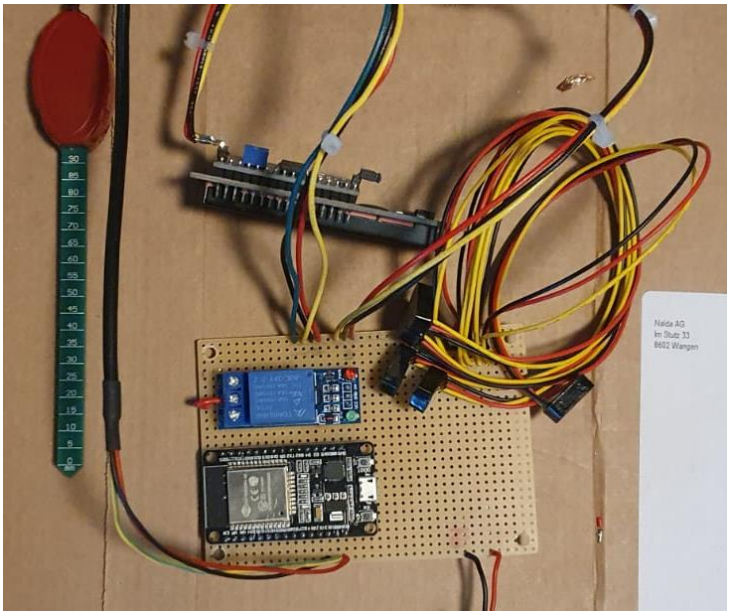


Abbildung 53 Komponenten gelötet

Im nächsten Schritt wurden alle Komponenten an die Platine gelötet. Als erstes wurden die 5 Kontaktlosen Füllstandsensoren (siehe Bild 53 oben rechts) auf die Platine gelötet. Diese mussten jedoch alle nochmals abgelötet werden, da sie in einer sehr unpraktischen Reihenfolge angelötet wurden. Dabei waren alle auf der äussersten Reihe der Platine nebeneinander angelötet. Dies verhinderte das schnelle und einfache verbinden der 5 Volt und 0V Speisung. Beim erneuten Löten der Sensoren wurde darauf geachtet, dass alle 5 Volt und 0 Volt Kontakte in einer Reihe sind. Die Sensoren wurden schlussendlich wie auf dem Schema ersichtlich angelötet. Als nächstes wurde der DHT22 und das LCD-Display am Rand der Platine angelötet. Dabei entstanden keine weiteren Probleme. Zum Schluss musste noch der Bodenfeuchtigkeitssensor an der Platine befestigt werden. Dieser wurde in der Nähe vom ESP32 angelötet, damit dieser einfach mit dem analogen kontakt verbunden werden kann.

2.4.4 Verbindungen

Zum Schluss mussten die ganzen Verbindungen der Speisung als auch der digitalen und analogen Kontakten gemacht werden. Dafür wurde die Platine bei einer Lötvorrichtung in 2 Krokodilklemmen eingeklemmt, damit diese einen festen Stand hat. Es konnte bei Bedarf mit einer Lupe gearbeitet werden.

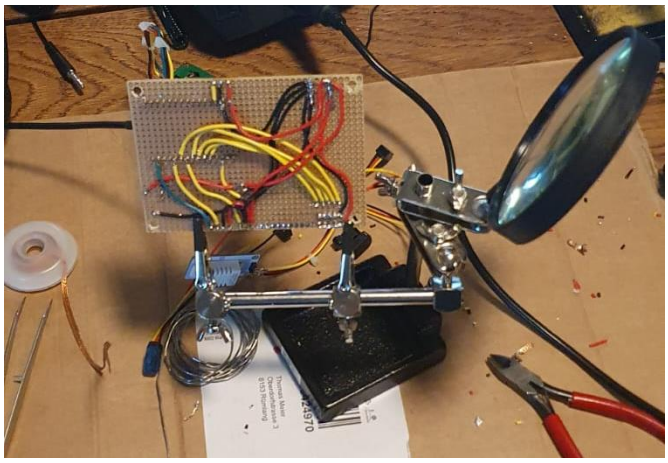


Abbildung 54 Lötvorrichtung

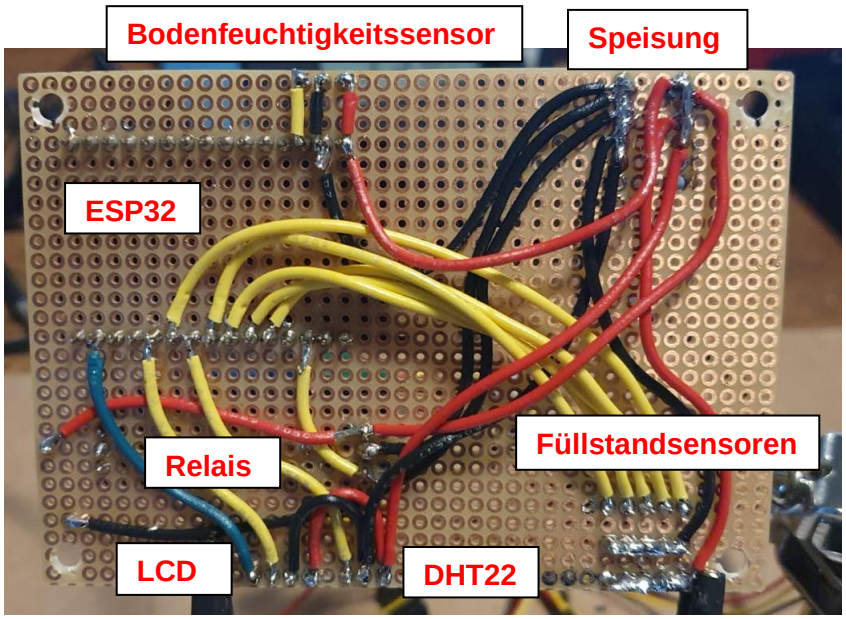


Abbildung 55 Lötverbindungen

Als Erstes wurde die gesamte 5 Volt Speisung miteinander verbunden. Dabei wurden bei der Haupteinspeisung der Platine (Abbildung 55) Brücken gebildet. Danach wurde der ESP32 mit 5 Volt versorgt. Dabei wurde eine Brücke zum Bodenfeuchtigkeitssensor gemacht. Als nächstes wurde das Relais Modul gespiesen. Dabei wurde auch eine Brücke zum COM-Kontakt vom Wechsler des Relais gemacht, damit dieser beim Einschalten die Pumpe mit 5 Volt versorgen wird. Danach wurde der DHT22 und das LCD-Display gespiesen. Es wurde eine Brücke vom DHT22 zum Display gemacht. Zum Schluss wurden die Kapazitiven Füllstandsensoren mit 5 Volt versorgt. Danach wurde mit schwarzen Kabeln das gleiche wie vorher beim 5 Volt mit dem 0 Volt gemacht. In einem weiteren Schritt wurden alle Sensoren mit gelben Kabeln auf den ESP32 verbunden. Bei den Füllstandsensoren war dies eine Herausforderung, da diese beim Anlöten des Verbindungskabels vom ESP32 zum Sensor das gelbe Sensorkabel stets herausfiel. Nach einigen Versuchen konnte dies jedoch verhindert und die Teile erfolgreich verbunden werden. In einem letzten Schritt wurde der SDA (gelbes Kabel) und SDL (grünes Kabel) vom LCD-Display an den ESP32 angeschlossen.

2.4.5 Test der Platine

Nachdem alles erfolgreich zusammengelötet wurde, konnte die Platine auf ihre Funktionen geprüft werden. Dabei wurde der gleiche Code verwendet, welcher bereits gebraucht wurde, um die Bauteile nach dem Erhalten auf ihre Funktionen zu prüfen. Der Test verlief fehlerfrei und alles funktionierte einwandfrei. Da dieser Test nur mit der direkten Einspeisung am ESP32 mittels Mikro USB erfolgte, wurden danach die gleichen Tests mit der Einspeisung an der USB-Buchse durchgeführt. Hier verlief ebenfalls alles fehlerfrei.

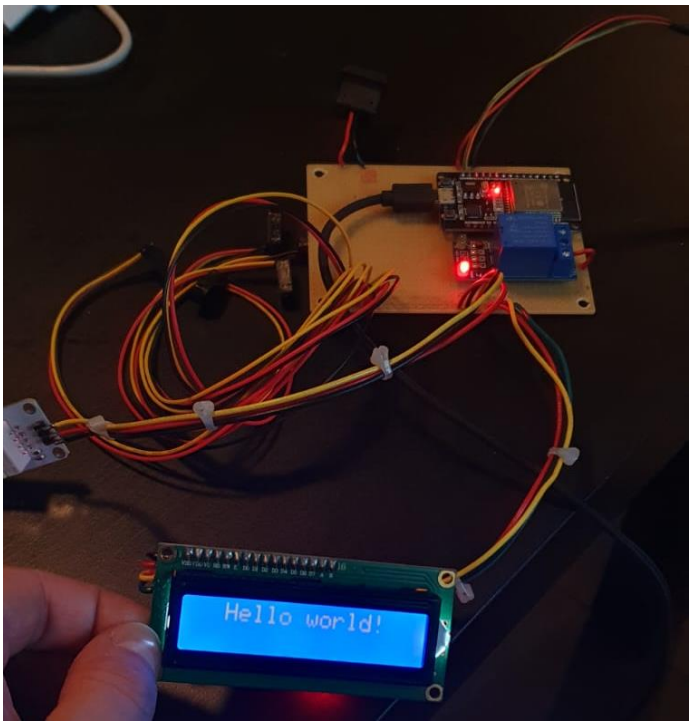


Abbildung 56 Test Platine

2.4.6 Fazit Löten

Das Ergebnis der Lötarbeit war sehr zufriedenstellend. Nach einem etwas holprigen Start am Anfang beim Löten vom ESP32 verlief die Arbeit fehlerfrei. Es entstand lediglich ein kleiner Fehler beim Anlöten der kontaktlosen kapazitiven Füllstandsensoren, wobei diese zuerst am Rand in Reihe angelötet wurden und somit die Speisung nicht gut miteinander verbunden werden konnte. Dies führte zu einem Mehraufwand, wobei diese abgelötet und in besserer Position wieder angelötet werden mussten. Zudem sind die Lötstellen schön glänzend geworden und es sind keine kalten Lötstellen entstanden. Die Inbetriebnahme der Platine nach dem Löten verlief ebenfalls ohne Zwischenfälle und verlief beispielhaft.

2.5 CAD-Software

Für das Design des Gehäuses wird ein CAD-Programm gebraucht. Da noch nie mit einer CAD-Software gearbeitet wurde, musste man sich zuerst einlesen und für ein Programm entscheiden. Hierfür wurden 3 verschiedene Programme miteinander verglichen. Es werden die Programme FreeCAD, SolidWorks und Fusion 360 im nächsten Schritt verglichen.

2.5.1 FreeCAD

Abbildung 57 FreeCAD

Das FreeCAD ist, wie es der Name schon sagt, komplett kostenlos. Dieses Programm gehört zu den gängigsten auf dem nicht-kommerziellen Markt. Zudem ist die Benutzeroberfläche den besten kostenpflichtigen CAD Software Optionen fast ebenbürtig. (sagt wer? Begründung geben) Sofern man sich bereits mit den Grundlagen der CAD-Modellierung auskennt, ist diese Software das Richtige. FreeCAD ist eine gute Option für Anfänger, sowie erfahrene 3D Modell-Designer. Jedoch gab es in den Bewertungen einige Kommentare über diverse Fehler, welche bei diesem CAD-Programm während der Modellierung immer wieder auftreten, den Prozess verlangsamen und somit eine weitere Herausforderung hinzukommt. (FreeCAD, 2023)

2.5.2 SolidWorks

Abbildung 58 SolidWorks

Die CAD-Software Solid Works ist mit Abstand das beste Tool für die 3D Modellierung. Diese wird sehr oft kommerziell genutzt. Zudem werden die Konstrukteure auf dieser Software ausgebildet. Eine Lizenz für diese Software kostet jährlich jedoch über 1000 Franken. Es ist zwar eine Studentenversion verfügbar, welche noch immer einen stolzen Betrag von 300 Franken kostet und somit das Budget des Projekts im Alleingang überschreitet. Deshalb kommt diese Software nicht in Frage. (SolidWorks, 2023)

2.5.3 Fusion 360

Abbildung 59 Fusion 360

Fusion 360 wird als das Schweizer Taschenmesser der kostenlosen CAD-Software angepriesen. Es ist abgesehen von Solid Works einer der besten CAD-Software-Optionen auf dem Markt. Diese Software deckt alle Bedürfnisse ab, wenn es um die 3D Modellierung, Dokumentation oder Fertigungs-Vorbereitung geht. Es ist eine fortschrittliche Software, welche sehr einfach zu bedienen und zudem anfängerfreundlich ist. Da diese Software nur für die kommerzielle Nutzung kostenpflichtig ist, scheint sie das richtige Tool für dieses Projekt zu sein. (Fusion 360, 2023)

Fazit CAD

Da SolidWorks kostenpflichtig ist und das Budget dieses Projektes sprengt, kommt es nicht in Frage. Das FreeCAD scheint eine gute kostenlose CAD-Software zu sein. Jedoch hat diese Software viele Fehler, welche den Prozess der Modellierung verlangsamt und unnötig Probleme generiert. Deshalb wird für dieses Projekt die CAD-Software Fusion 360 verwendet. Diese scheint sehr zuverlässig zu sein, da sie laut (Scoobe3d, 2023) sogar als Schweizer Taschenmesser der kostenlosen CAD-Programme angepriesen wird. Zudem eignet sich diese Software gut für die Dokumentation, welche in diesem Projekt ebenfalls eine wichtige Rolle spielt.

2.6 Gehäuse 3D-Design

Als erstes mussten die Bauteile vermessen werde. Dafür wurde eine Schieblehre verwendet. Dabei musste man Schritt für Schritt alles genau vermessen und darauf achten, dass kein Mass vergessen ging. Die Masse wurden alle auf einen Notizzettel geschrieben. Nachdem alles vermessen wurde, wurde festgelegt, dass das ganze Gehäuse 150x150x150mm gross wird. Dabei wird der Elektrokasten 150x60x150mm und der Wassertank 150x90x150mm gross. Somit umfasst der Wassertank eine Füllmenge von 2 Liter, wobei diese Menge problemlos zum Bewässern für eine Zimmerpflanze für ca. 3 Wochen, abhängig von der Pflanze, reichen wird. Nachdem die Software installiert wurde, musste ein neues Projekt erstellt werden. Dies war oben links einfach zu finden. Danach mussten die verschiedenen Parameter eingegeben werden (siehe Abbildung 60). Anschliessend konnte bereits mit der Modellierung gestartet werden.

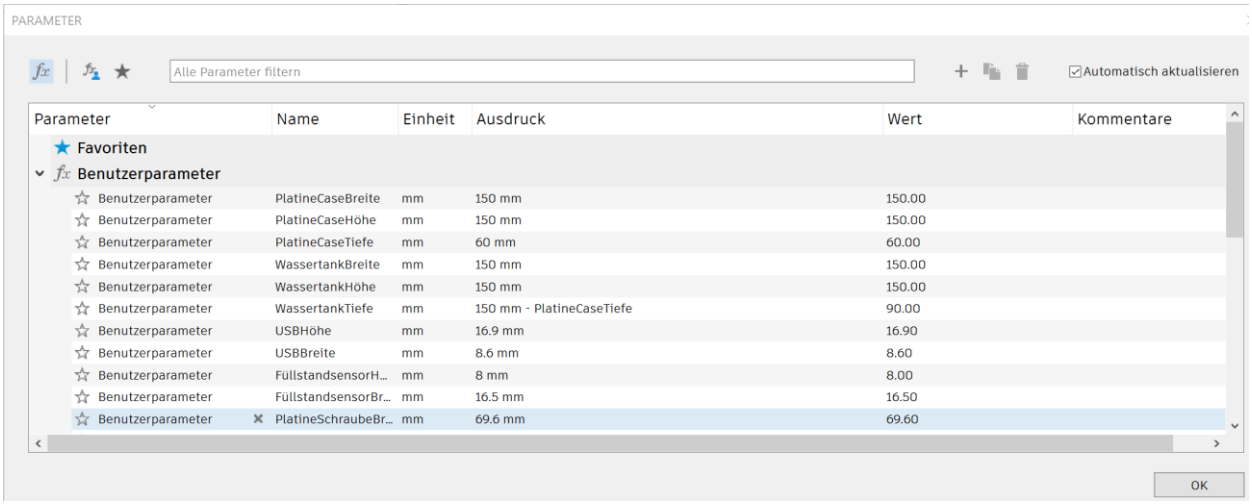


Abbildung 60 Parameter

2.6.1 Grundfunktionen

In der Navigationsleiste befinden sich alle Funktionen, welche zum Konstruieren des Gehäuses benötigt werden. Im Bereich «Erstellen» kann eine neue Skizze, sowie diverse andere Formen erstellt werden. Im Bereich «Ändern» können Ecken abgerundet, sowie ausschnitte und Bohrungen angebracht werden. Der Bereich «Zusammenfügen» dient, wie es der Name vermuten lässt, dazu verschiedene Objekte beziehungsweise Skizzen miteinander zu verbinden.

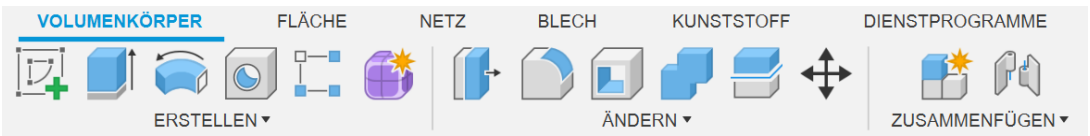


Abbildung 61 Grundfunktionen

2.6.2 Skizze Grundriss

Zuerst wurde eine Skizze vom Grundriss des Gehäuses gezeichnet. Hierzu konnten die Parameter, welche am Anfang definiert wurden, eingegeben werden. Es wurden die Parameter «PlatinenCaseBreite», «PlatinenCaseTiefe», WasserTankBreite» und «WasserTankTiefe» verwendet. Dabei wurden die beiden Ecken vom Wassertank abgerundet, um das Design zu verschönern. Somit sieht das Gehäuse nicht nur wie ein Würfel aus.

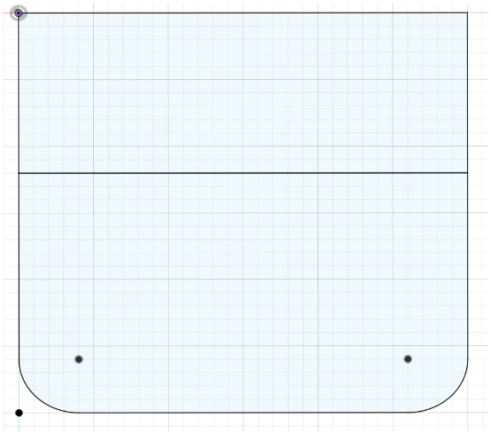


Abbildung 62 Grundriss

2.6.3 Extrusion

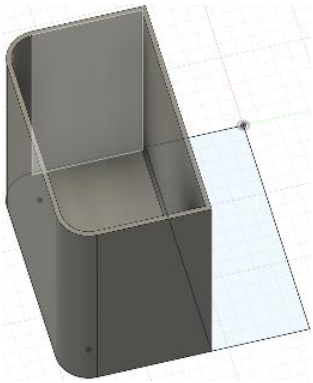


Abbildung 63 Wassertank

Nachdem die Skizze fertig gezeichnet wurde, konnte diese mithilfe der Funktion «Extrusion» von einem 2D in ein 3D Modell umgewandelt werden. Hierzu wurde zuerst der Wassertank in ein 3D Modell umgewandelt und anschliessend mit der Funktion «Schale» ausgehöhlt. Es wurde eine Wanddicke von 3mm verwendet, damit das Gehäuse genug Stabilität bieten kann. Derselbe Schritt wurde mit dem Gehäuse für die Platine durchgeführt. Hierbei wurde dieses jedoch nicht nach oben, sondern nach hinten ausgehöhlt (siehe Abbildung 63).

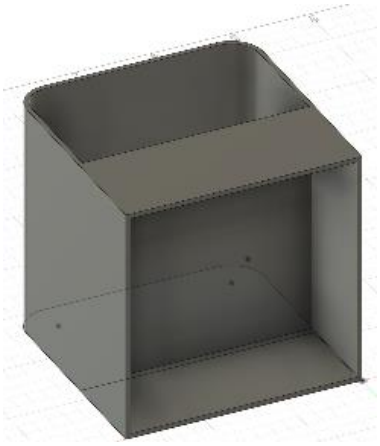


Abbildung 64 Platinen Gehäuse

2.6.4 Platinen Gehäuse

Abbildung 65 Platinen Gehäuse komplett

In einem nächsten Schritt wurden neue Skizzen im Platinen-Gehäuse erstellt. Zuerst wurde die Platine mit den gemessenen Massen gezeichnet. Dies waren äusserst hilfreicher Orientierungspunkte, da jetzt genau gesehen werden konnte, wie viel Platz diese benötigt und sie somit richtig platziert werden konnte. Danach wurden die Befestigungsbolzen gezeichnet und generiert. Diese wurden 6mm lang gestaltet, damit die Lötstellen und Kabel auf der Rückseite der Platine genügend Platz zur Verfügung haben. Zudem wurden diese abgerundet, damit sie ohne «Supports» mit dem 3D-Drucker gedruckt werden können. Danach wurden die Füllstandsensoren eingezeichnet, welche sich rechts neben der Platine befinden (auf der Abbildung rechts?). Diese wurden über die gesamte Höhe des Wassertanks gleichmässig verteilt. Damit die Pumpe niemals ohne Wasser läuft, wurde der unterste Sensor so eingezeichnet, damit sich dieser an der oberen Kante der Pumpe befindet. Die Sensoren Zeigen Somit einen Füllstand von 100%, 75%, 50%, 25%, 10% und 0% an. Es wurde anschliessend eine Versenkung von 1mm generiert, damit diese nach dem Drucken problemlos an der richtigen Stelle positioniert werden kann. Zum Schluss wurde an der linken Seite des Gehäuses ein Ausschnitt für die USB-Buchse und M12 Verschraubung gemacht.

2.6.5 Deckelhalterungen

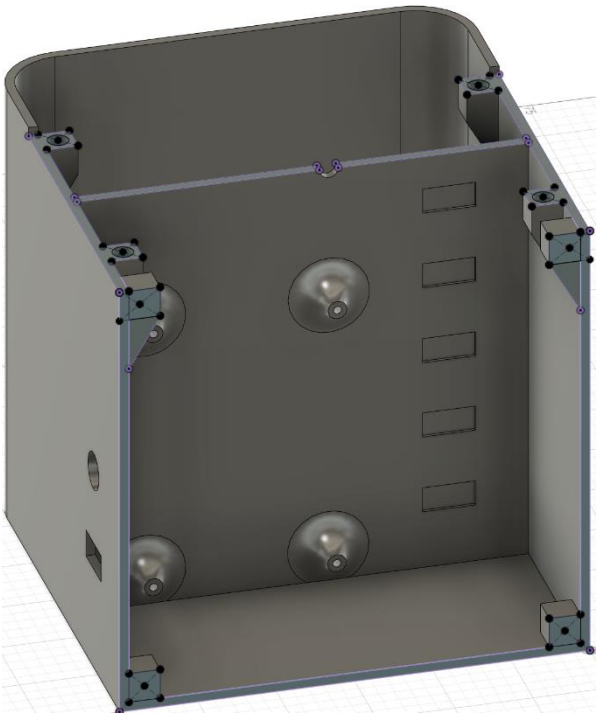


Abbildung 66 Deckelhalterungen

Zuerst wurde der Deckel vom Gehäuse der Platine entfernt. Dann wurde der Rand vom Platinen-Gehäuse bis zur Mitte vom Wassertank um 3mm gesenkt, damit später ein Deckel erstellt werden kann. In einem weiteren Schritt wurde an der Trennwand vom Wassertank zum Platinen-Gehäuse ein Halbkreis ausgeschnitten, damit hier später das Kabel der Pumpe durchgeführt werden kann. Dies wurde ganz oben gemacht, damit die Gefahr eines Lecks minimiert werden kann. Danach wurden die 4 Halterungen für den Deckel eingezeichnet. Bei ihnen wurde ein Loch von 6.5mm gemacht, damit später der Deckel mittels Bolzen ganz einfach darauf gesteckt werden kann. Diese wurden auch wieder abgewinkelt, damit beim 3D-Drucken keine Supports nötig sind. In einem letzten Schritt wurden die vier Halterungen für den Deckel vom Platinen-Gehäuse gezeichnet und generiert. Hierbei musste die zwei oberen auch wieder abgewinkelt werden. Zum Schluss wurde an den Halterungen der Rückwand vier M3 Gewinde gebohrt, damit diese ganz einfach mit Schrauben angeschraubt und in Zukunft bei Bedarf wieder aufgeschraubt werden kann.

2.6.6 Deckel Wassertank

Abbildung 67 Deckel Wassertank

Danach wurde der Deckel vom Wassertank gezeichnet und generiert. Der Deckel ist, wie die Wände ebenfalls, 3mm dick. Dabei ist wichtig, dass dieser vom restlichen Gehäuse getrennt bleibt, damit dieser einzeln exportiert und gedruckt werden kann. Nachdem der Deckel gezeichnet wurde, wurden am unteren Ende vier Bolzen generiert, welche den Deckel mit den vorher gezeichneten Halterungen in Position halten. Danach wurde ein Loch für den Schlauch gemacht, welches einen Durchmesser vom 10mm hat. Anschliessend wurde der Ausschnitt für das LCD-Display und den DHT22 gemacht. Hier war es wichtig, dass die Toleranzen knapp bemessen sind, damit diese hineingesteckt werden können und gut halten.

2.6.7 Deckel Platinen Gehäuse

Zum Schluss wurde noch der Deckel vom Platinen-Gehäuse gezeichnet. Dieser ist wieder 3mm dick und muss von den restlichen beiden Teilen, dem Gehäuse und dem Deckel vom Wassertank, getrennt bleiben. Hier war wichtig, dass die Löcher im Deckel, mit denen der Halterungen übereinstimmen. Damit die Schrauben nicht zu weit überstehen, wurde eine zylindrische Einsenkung von 2mm eingefügt.

Abbildung 68 Deckel Platinen Gehäuse

2.6.8 Export der Dateien

Anschliessend mussten die 3 Teile in eine STL-Datei exportiert werden. Eine STL-Datei ist bei den 3D Druckern üblich und wird benötigt, damit diese gedruckt werden können.

2.7 3D Drucken

Zum Drucken wird der 3D Drucker Bambu Lab X1-Carbon der Marke Bambu Lab verwendet. Dieser Drucker kostet neu 1200 Franken. Jedoch war dieser bereits vorhanden und musste nicht zusätzlich gekauft werden. Zum Drucken der Teile wird das Material «Polylactid Acid» kurz PLA verwendet. Dieses Material ist ein auf nachwachsenden Rohstoffen wie Zuckerrohr oder Mais basierender Polyester. Es ist auch eines der einfachsten Materialien zum 3D Drucken, da es eine niedrige Drucktemperatur hat und sich somit nicht so leicht verzieht. Zudem entstehen beim Drucken keine Toxischen Dämpfe, sowie unangenehme Gerüche.

2.7.1 Gehäuse

Als erstes wurde die STL-Datei vom Gehäuse in der Software von Bambu Lab geöffnet. Anschliessen musste das richtige Material zum Drucken ausgewählt werden. Dabei wurde schwarzes PLA von der Marke eSun verwendet. Da bereits vorhin diverse Teile mit diesem Material gedruckt wurden sind bereits die optimalen Parameter eingestellt und somit alles kalibriert. Danach musste die Datei zum Drucker gesendet werden und wurde automatisch gedruckt. Es dauerte 5 Stunden und 34 Minuten und benötigte 257 Gramm PLA bis das Gehäuse fertig gedruckt war.

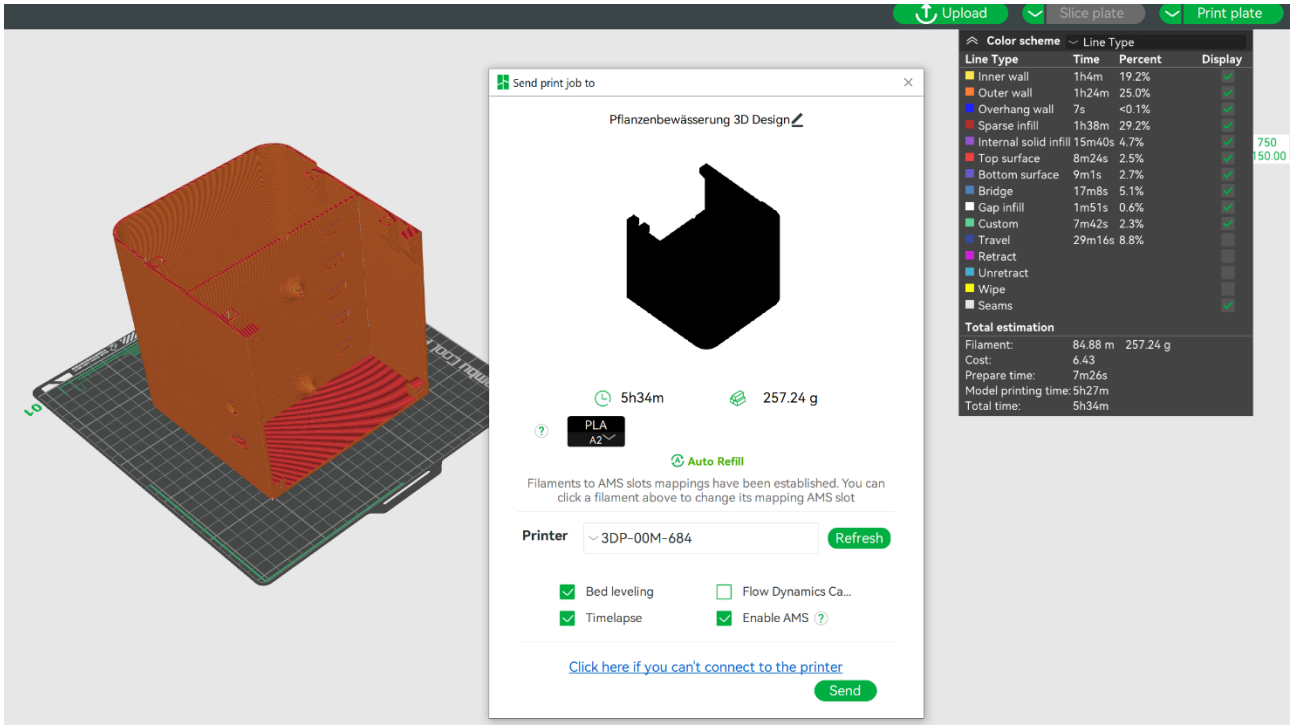


Abbildung 69 Gehäuse 3D

2.7.2 Deckel

Bei den beiden Deckeln wurden genau die gleichen Schritte wie beim Gehäuse gemacht. Hier wurde lediglich eine andere Farbe verwendet. Es wurde weisses PLA der Marke eSun verwendet. Der Deckel vom Wassertank brauchte 57 Minuten und benötigte 28 Gramm PLA zum Drucken. Der Deckel vom Platinen Gehäuse dauerte 1 Stunde und 13 Minuten und benötigte 43 Gramm PLA.

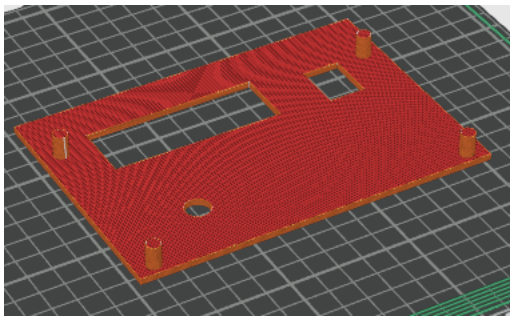


Abbildung 71 Wassertank Deckel 3D

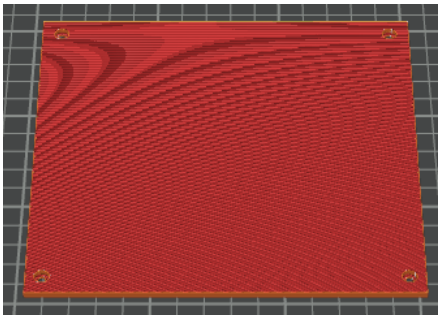


Abbildung 70 Platinen Gehäuse Deckel 3D

2.7.3 Ergebnisse

Mit den Ergebnissen vom Drucken war ich sehr zufrieden, da alles sauber gedruckt wurde und die Deckel perfekt auf das Gehäuse passen.

Abbildung 72 3D Gehäuse gedruckt

2.8 Zusammenbau

Zuerst musste die USB-Buchse und der Bodenfeuchtigkeitssensor wieder von der Platine abgelötet werden, da diese zuerst durch die ausschnitte geführt werden mussten. Anschliessend wurden diese wieder angelötet. Zudem musste der 0V der Pumpe an der Platine angelötet und die 5V Speisung am Relais NO-Kontakt angeschraubt werden. Dann wurde die Platine ins Gehäuse geschraubt. Jedoch konnte diese nur mit den zwei oberen Schrauben befestigt werden, da der Abstand zwischen den oberen zu den unteren Schrauben zu klein war. Dabei wurden die Masse beim Messen der Platine falsch aufgenommen und somit falsch gezeichnet. Dies führte aber zu keinen weiteren Problemen, da die Platine auch mit zwei Schrauben sehr gut haltet.

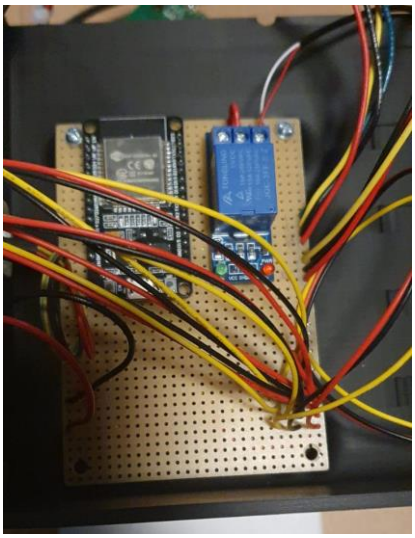


Abbildung 73 Platine befestigt

Abbildung 74 Gehäuse mit Sensoren

In einem weiteren Schritt wurden die Füllstandsensoren in ihre Einbuchtungen und die Pumpe in den Wassertank geklebt. Zudem wurde das Kabel der Pumpe in den dafür vorgesehenen Ausschnitt geklebt. Dazu wurde Sekundenkleber verwendet.

Danach wurde der Schlauch an der Pumpe befestigt und durch die dafür vorgesehene Öffnung im Deckel geführt. Anschliessend wurde das LCD-Display und der DHT22 in die dafür vorgesehene Öffnung gedrückt. Diese passten perfekt. Anschliessend wurde der Deckel mit Sekundenkleber am Gehäuse angeklebt. Somit kann garantiert werden, dass kein Wasser zur Platine kommt, solange der Wassertank nicht überfüllt wird.

Abbildung 75 Gehäuse mit Deckel

2.8.1 Fazit Zusammenbau

Alle Teile passten gut zusammen. Das einzige Probleme war, dass nur zwei der vier Schrauben passten, um die Platine zu befestigen. Dies stellte jedoch kein grosses Problem dar, da diese auch mit zwei Schrauben ausreichend befestigt werden konnten. Zudem entstand ein Fleck vom Sekundenkleber im Wassertank, als die Pumpe befestigt wurde. Dies ist zwar kein technisches Manko, aber sieht optisch nicht ganz sauber aus. Der restliche Zusammenbau verlief problemlos.

2.9 Code

Der folgende Code baut auf den verschiedenen Test-Scripts auf, welche verwendet wurden, um die Bauteile auf ihre Funktion zu prüfen. Die verschiedenen Teile des Codes im folgenden Abschnitt wurden nach dem Schreiben einzeln auf ihre Funktion geprüft.

2.9.1 Bibliotheken

```
//Hier werden alle Bibliotheken integriert, welche für das Projekt gebraucht werden.
#include <Arduino.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

Abbildung 76 Bibliotheken Final

Als erstes werden die verschiedenen Bibliotheken integriert. In diesem Fall handelt es sich um die Bibliotheken für das LCD-Display, sowie den DHT22.

2.9.2 LCD-Display

```
//Display
bool isMainPage = false;
LiquidCrystal_I2C lcd(0x27,16,2); // setzt die LCD addresse auf 0x27 für 16 Zeichen auf 2 Linien
```

Abbildung 77 LCD-Display

Als erstes wird eine boolesche Variabel mit dem Namen «isMainPage deklariert. Diese wird später verwendet, um auf dem Display zwischen zwei Seiten zu wechseln. Danach wird die Adresse vom Display festgelegt, welche bereits bei den Test-Scripts ermittelt wurde. Zudem wird die Grösse des Displays festgelegt, welches 16 Zeichen auf 2 Zeilen umfasst.

```
void showLcd() {    //Funktion fürs LCD-Display
  lcd.clear();
  if (isMainPage) {   //Definiert dieses if als MainPage
    lcd.setCursor(0,0);   //Setzt den Cursor auf den 0. Buchstaben, Zeile 0
    lcd.print("Soil: "+ String(getSoilMoisture()) + "%");  //Gibt den aktuellen Bodenfeuchtigkeitswert auf dem LCD Display aus
    lcd.setCursor(0,1);   //Setzt den Cursor auf den 0. Buchstaben, Zeile 1
    lcd.print("Water: "+ String(getWaterTankLevel()) + "%");   //Gibt de naktuellen Füllstand vom Wassertank auf dem LCD Display aus
  }
  else {
    lcd.setCursor(0,0);   //Setzt den Cursor auf den 0. Buchstaben, Zeile 0
    lcd.print("Humidity: " + String(dht.readHumidity()) + "%"); //Gibt die aktuelle Luftfeuchtigkeit auf dem  LCD Display aus
    lcd.setCursor(0,1);   //Setzt den Cursor auf den 0. Buchstaben, Zeile 1
    lcd.print("Temp: " + String(dht.readTemperature()));  //Gibt die aktuelle Temperatur auf dem LCD Display aus
    lcd.print((char)223);   //223 ist der Code für das Grad Zeichen in der ASCII Tabelle
    lcd.print("C");   //Schreibt ein C hinter das Gradzeichen
  }
  isMainPage = !isMainPage;   //invertiert die MainPage, damit zwischen Seite 1 und 2 gewechselt wird
```

Abbildung 78 Funktion LCD-Display

Dann wurde die Funktion für das LCD-Display geschrieben, welche «showLcd» heisst. Als erstes wird das Display mit «lcd.clear();» geleert, damit keine Werte überschrieben werden und es zu keinen Fehlerhaften anzeigen kommt. Danach wird mit «if (isMainPage)» die Hauptseite angezeigt,

solange «isMainPage = true» ist. Auf der Hauptseite wird die aktuelle Erdfeuchtigkeit sowie der Füllstand vom Wassertank angezeigt. Dabei wird die Erdfeuchtigkeit auf der Zeile 0 und der Füllstand auf der Zeile 1 angezeigt. Falls die «isMainPage = false» ist, wird die zweite Seite auf dem Display angezeigt. Auf der zweiten Seite sieht man die aktuelle Luftfeuchtigkeit in Prozent und Temperatur in Grad Celsius. Die Luftfeuchtigkeit wird auch hier wieder auf Zeile 0 und die Temperatur auf Zeile 1 angezeigt. Zudem muss das Gradzeichen aus der ASCII Tabelle genommen werden. Dies macht man mit «lcd.print((char)223);». Man benötigt den Datentyp «char» um ein einzelnes Zeichen zu speichern. Anschliessend wird mit «isMainpage = !isMainPage» zwischen den beiden Seiten gewechselt. So werden diese nach jedem Zyklus im Loop gewechselt.

2.9.3 DHT22

Beim DHT22 wird wie beim Test-Script wieder der Pin und der Typ definiert. Danach wird die Temperatur mit «dht.readHumidity()» und Luftfeuchtigkeit mit «dht.readTemperature()» in der Funktion vom LCD Display ausgegeben. Wie diese ausgelesen werden konnte, wurde der DHT-Bibliothek von Adafruit entnommen werden.

```
//DHT22
#define DHTPIN 18   //Definiert den DHT pin auf pin 18
#define DHTTYPE DHT22 //Definiert den DHT Typ (11 oder 22)
DHT dht(DHTPIN, DHTTYPE);
```

Abbildung 79 DHT22 Pin

2.9.4 Relais

```
//Relais
int relaisPin = 15;
```

Abbildung 80 Relais Pin

Beim Relais musste lediglich der Pin am ESP32 definiert werden. Weitere Schritte, welche im Setup und Loop ausgeführt werden, werden in einem späteren abschnitt erklärt.

2.9.5 Füllstandsensor

Zuerst wurden die fünf verschiedenen Füllstandsensoren als Integer und die jeweiligen Pins am ESP32 definiert. Die Füllstandsensoren messen die Werte 100%, 75%, 50%, 25% und 10% im Wassertank.

```
//Fuellstandsensor
int tankSensorPin1 = 19; //10% Wassertanksensor
int tankSensorPin2 = 4; //25% Wassertanksensor
int tankSensorPin3 = 16; //50% Wassertanksensor
int tankSensorPin4 = 17; //75% Wassertanksensor
int tankSensorPin5 = 5; //100% Wassertanksensor
```

Abbildung 81 Füllstandsensoren

```
int getWaterTankLevel() {   //Funktion zum Ermitteln des aktuellen Füllstands
  if (!digitalRead(tankSensorPin5)) {   //Gibt den Wert 100 zurück wenn der TankSensor 5 angibt
    return 100;
  }
  if (!digitalRead(tankSensorPin4)) {   //Gibt den Wert 75 zurück wenn der TankSensor 4 angibt
    return 75;
  }
  if (!digitalRead(tankSensorPin3)) {   //Gibt den Wert 50 zurück wenn der TankSensor 3 angibt
    return 50;
  }
  if (!digitalRead(tankSensorPin2)) {   //Gibt den Wert 25 zurück wenn der TankSensor 2 angibt
    return 25;
  }
  if (!digitalRead(tankSensorPin1)) {   //Gibt den Wert 10 zurück wenn der TankSensor 1 angibt
    return 10;
  }
  return 0;   //Wenn keiner der Sensoren angibt, wird der Wert 0 ausgegeben
}
```

Abbildung 82 Funktion Füllstandsensoren

In diesem Abschnitt der Software wird die Funktion zum Ermitteln des aktuellen Füllstands gemacht. Diese startet beim 100%-Sensor und geht bis zum 10%-Sensor. Dies fängt beim 100%-Sensor an, da so die anderen Sensoren nicht abgefragt werden müssen wenn dieser einen Füllstand misst. Da die Sensoren, sobald sie angeben einen Wert von 0 ausgeben, müssen diese mit dem «!» invertiert werden. Die Grundfunktion für die Sensoren ist «if (!digitalRead(tankSensorPin5))» wobei lediglich die Nummer des Sensors angepasst werden musste. Mit dem «return ;» wird der aktuelle Füllstand ausgegeben. Wenn keiner der Sensoren angibt, wird mit «return 0;» der Wert 0 ausgegeben. Somit ist der Tank leer. Diese Funktion wird bereits in der Funktion vom Display verwendet, um den aktuellen Füllstand anzuzeigen. Zudem wird sie in einem weiteren Schritt im Loop verwendet, welches in einem späteren Abschnitt erklärt wird.

2.9.6 Erdfeuchtigkeitssensor

Der Erdfeuchtigkeitssensor wird auf den Pin 13 vom ESP32 programmiert.

```
//Erdfeuchtigkeitssensor
int soilSensorPin = 13;
```

Abbildung 83 Erdfeuchtigkeitssensor Pin

In einem weiteren Schritt wird die Funktion zum Messen der aktuellen Erdfeuchtigkeit geschrieben. Diese heisst «getSoilMoisture()». Als erstes wird mit «int sensorValue = analogRead(soilSensorPin);» der aktuelle Wert des Sensors ausgelesen. Danach wird der Wert von trockener Erde definiert, welcher 0 entspricht. Um den maximalen Wert für die Nässe zu ermitteln wurde der Sensor in ein Glas Wasser gesteckt. Dabei kam der Wert «3770» raus, welcher der maximal gemessenen Feuchtigkeit entspricht. In einem weiteren Schritt wird der Feuchtigkeitswert in Prozent umgerechnet. Dies geschieht mit «float soilMoisture = map(sensorValue, dryValue, wetValue, 0, 100);» Anschliessend wird der berechnete Prozentwert mit «return (int)soilMoisture;» ausgelesen. (ChatGPT Spellcheck, 2023)

```
int getSoilMoisture() {   //Funktion zum auslesen der Erdfeuchtigkeit

  int sensorValue = analogRead(soilSensorPin);    //Liesst den Sensorwert aus

  int dryValue = 0;   //Hier wird der Wert für die trockene Erde definiert
  int wetValue = 3770;    ////Hier wird der Wert für die nasse Erde definiert

  float soilMoisture = map(sensorValue, dryValue, wetValue, 0, 100); // Berechnet die Bodenfeuchtigkeit in Prozent

  return (int)soilMoisture;  //Gibt den aktuellen Feuchtigkeitswert in % an
}
```

Abbildung 84 Funktion Erdfeuchtigkeitssensor

2.9.7 Setup

Im Setup wird als erstes die Übertragungsgeschwindigkeit vom ESP32 definiert, welche 115200 bit/s sind. Wichtig ist, dass dies auch in Platform.ini angepasst wird, damit PlatformIO ebenfalls weiss, welche Geschwindigkeit benötigt wird.

```
void setup() {
  Serial.begin(115200);   //setzt serial monitor geschwindigkeit auf 115200 Wichtig:muss auch in Platform.ini angepasst werden

  pinMode(relaisPin, OUTPUT); //Inizialisiert Pin das er schreiben kann
  digitalWrite(relaisPin, LOW); //setzt Pin am anfang auf low

  pinMode(tankSensorPin1, INPUT); //Inizialisiert Pin das er lesen kann
  pinMode(tankSensorPin2, INPUT); //Inizialisiert Pin das er lesen kann
  pinMode(tankSensorPin3, INPUT); //Inizialisiert Pin das er lesen kann
  pinMode(tankSensorPin4, INPUT); //Inizialisiert Pin das er lesen kann
  pinMode(tankSensorPin5, INPUT); //Inizialisiert Pin das er lesen kann

  dht.begin();    //DHT wird gestartet

  lcd.init();     //Initialisiert LCD
  lcd.backlight();  //Hintergrundbeleuchtung ein
}
```

Abbildung 85 Setup

Danach wird der Relais Pin initialisiert, damit er schreiben kann. Zudem wird der Zustand des Pins am Anfang auf LOW gesetzt, damit garantiert werden kann, dass dieser nicht Zustandslos ist. Des Weiteren werden alle Pins der Füllstandsensoren initialisiert. Danach wird der DHT22 gestartet, das LCD-Display initialisier und die dazugehörige Hintergrundbeleuchtung eingeschaltet.

2.9.8 Loop

Im Loop wird als erstes die Funktion vom LCD-Display mit «showLcd();» abgerufen. Somit werden alle Werte auf dem Display angezeigt, welche in dieser Funktion enthalten sind. Danach wird geschaut, ob die Pflanze Wasser benötigt. Wenn die Erdfeuchtigkeit unter 30% und der Wassertank nicht leer ist, wird das Relais eingeschaltet und die Pflanze für 5 Sekunden bewässert. Danach wird das Relais wieder ausgeschaltet. Der Loop wiederholt sich alle 2 Sekunden.

```
void loop() {
  showLcd();

  //Watering
  if ((getSoilMoisture() < 30) and (getWaterTankLevel() > 0)) {
    digitalWrite(relaisPin, HIGH);
    delay(5000);
    digitalWrite(relaisPin, LOW);
  }

  delay(2000);    //Loop wird nach 2 Sekunden erneut ausgeführt
}
```

Abbildung 86 Loop

2.9.9 Versionierung auf GitHub

Damit ein Überblick über die Versionierung der Software besteht, wurde diese auf GitHub hochgeladen. So kann von überall auf den Code zugegriffen werden. Zudem besteht stets Klarheit um welche Version der Software es sich handelt.

Das Projekt findet man unter Folgendem Link auf GitHub:

https://github.com/Thommy321/Diplomarbeit-Automatisches-Pflanzenbew-sserungssystem/tree/main

2.9.10 Fazit Code

Bei der Programmierung gab es grössere Schwierigkeiten als erwartet. Zum einen waren die Programmiere-Skills sehr eingerostet. Zudem war es eine Herausforderung die verschiedenen Funktionen zu schreiben, sowie den Wert den Erdfeuchtigkeitssensors in Prozent auszugeben.

2.10 Inbetriebnahme

Bei der Inbetriebnahme wurden die verschiedenen Funktionen des Bewässerungssystems getestet und in einem Protokoll festgehalten. Es wurde eine Zimmerpflanze verwendet, um die Erdfeuchtigkeit zu messen, wie man auf dem Bild sieht.

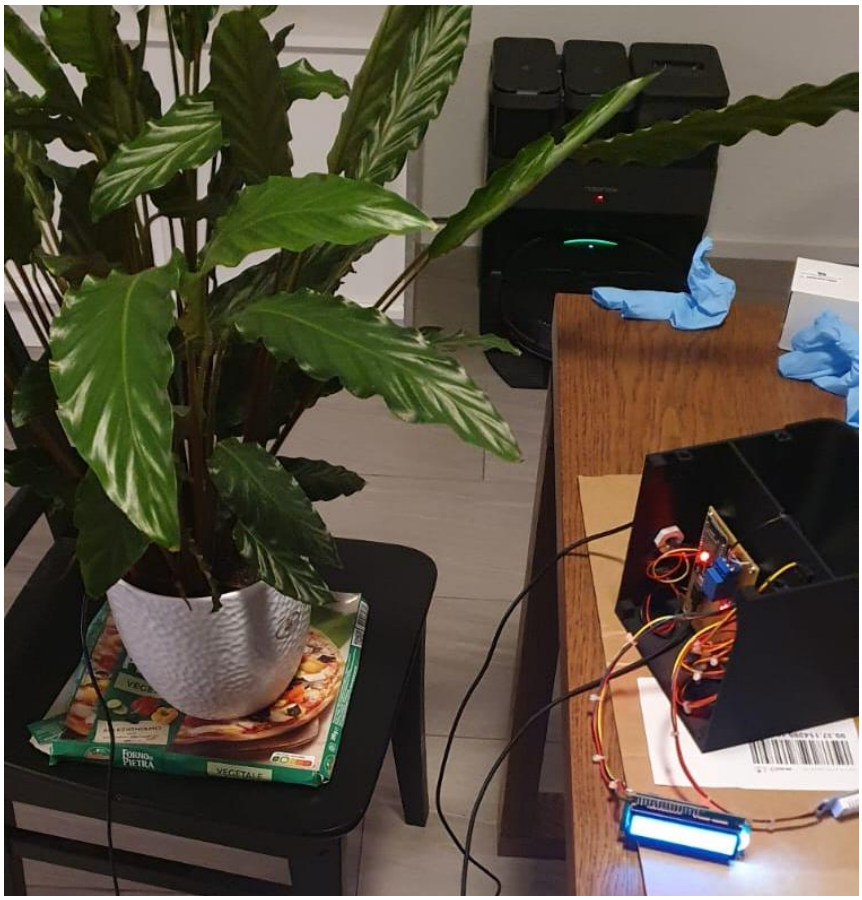


Abbildung 87 Inbetriebnahme

2.10.1 Inbetriebnahme Protokoll

Bauteil	Testart	Funktion (EF/NEF)
LCD-Display	Prüfung der Werte	EF
DHT22	Werteänderung auf dem LCD durch anhauchen und Überprüfung mit einem externen Temperaturmesser	EF
Füllstandsensoren	Werteänderung auf LCD durch Finger am Sensor	EF
Erdfeuchtigkeitssensor	Überprüfung der Prozent Werte auf LCD Kein Kontakt = 0% In Wasser = 100% Erde = aktueller Feuchtigkeitswert	EF
Relais	Zieht an, sobald Sensor weniger als 30% und Wassertank nicht leer ist.	EF

Tabelle 3 Inbetriebnahmeprotokoll

Sämtliche Bauteile konnten auf ihre Funktion geprüft werden. Diese funktionierten alle einwandfrei. Alle Werte wurden richtig auf dem LCD-Display angezeigt, welche mit den in der Tabelle beschriebenen Methoden getestet wurden. Zudem wurde eine Zimmerpflanze 1 Woche lang mit dem Gerät bewässert. Dieser scheint es danach immer noch prächtig zu gehen, wie man auf dem Bild sehen kann.

Abbildung 88 Bewässerung Test

2.11 Soll-Ist-Vergleich

Soll:

Automatische Bewässerung der Zimmerpflanze: Das System muss in der Lage sein, die Bewässerung der Zimmerpflanze basierend auf den Messungen der Sensoren und den vordefinierten Bewässerungsparametern automatisch zu steuern.

Anzeige des Pflanzenzustands auf dem LCD-Display: Das LCD-Display soll den aktuellen Zustand der Pflanze, einschliesslich Bodenfeuchtigkeit, Füllstand des Wassertanks, Temperatur und Luftfeuchtigkeit, in Echtzeit anzeigen.

Zugriff auf den Pflanzenzustand über ein mobiles Gerät: Der Benutzer soll den Zustand der Pflanze von überall aus über sein Handy überwachen und gegebenenfalls Einstellungen anpassen können.

Integration in ein 3D-gedrucktes Gehäuse: Das gesamte System, einschliesslich aller Komponenten und Verkabelungen, wird in einem benutzerfreundlichen und ästhetisch ansprechenden 3D-gedruckten Gehäuse untergebracht, das sowohl funktional, praktisch als auch sicher ist.

Ist:

Automatische Bewässerung der Zimmerpflanze: Das Bewässerungssystem ist in der Lage eine Zimmerpflanze automatisch zu bewässern. Dies wurde für eine Woche fehlerfrei getestet.

Anzeige des Pflanzenzustands auf dem LCD-Display: Es wird der aktuelle Zustand der Pflanze auf einem LCD-Display angezeigt. Der Zustand beinhaltet die Bodenfeuchtigkeit, Füllstand des Wassertanks, Temperatur und Luftfeuchtigkeit. Diese Informationen sind in Echtzeit auf dem Display ersichtlich.

Zugriff auf den Pflanzenzustand über ein mobiles Gerät: Der Zustand der Pflanze ist nicht auf dem Handy ersichtlich. Dieses Feature wurde wegen äusserst aufwendigen Programmierarbeit und aufgrund von zeitlichen Problemen gestrichen.

Integration in ein 3D-gedrucktes Gehäuse: Das gesamte System wurde in ein benutzerfreundliches und ästhetisches 3D gedrucktes Gehäuse integriert. Dieses ist funktional, praktisch und sicher.

2.12 Begründung der Lösung

Die Entwicklung und Umsetzung des automatischen Bewässerungssystems für Zimmerpflanzen war ein herausforderndes Projekt, bei dem es galt, die festgelegten Erfolgskriterien zu erfüllen. Die folgende Begründung erläutert die Erfüllung der festgelegten Erfolgskriterien sowie die Ursachen, warum das dritte Kriterium nicht erreicht werden konnte. Zudem begrenzte sich das Budget für diese Arbeit auf 200 Franken.

Automatische Bewässerung der Zimmerpflanze:

Die erste Erfolgskomponente unseres Projekts, die automatische Bewässerung der Zimmerpflanze, wurde erfolgreich umgesetzt. Das System konnte über einen Zeitraum von einer Woche hinweg zuverlässig und fehlerfrei eine Zimmerpflanze bewässern. Dies zeigt, dass die Grundfunktion des Systems ordnungsgemäss funktioniert und die Pflanze ausreichend mit Wasser versorgt wurde.

Anzeige des Pflanzenzustands auf dem LCD-Display:

Eine weitere wichtige Anforderung meines Projekts war die Echtzeit-Anzeige des Pflanzenzustands auf einem LCD-Display. Dieses Ziel wurde erreicht. Das Display bietet dem Benutzer (zudem suggeriert, dass dies zusätzlich Informationen sind, allerdings sind diese Punkte der «zustand» der Pflanze) wertvolle Informationen über Bodenfeuchtigkeit, den Füllstand des Wassertanks sowie die Temperatur und Luftfeuchtigkeit. Diese Daten ermöglichen es den Benutzern, den Zustand ihrer Zimmerpflanze zu überwachen und entsprechend zu handeln.

Zugriff auf den Pflanzenzustand über ein mobiles Gerät:

Das dritte Erfolgskriterium, der Zugriff auf den Pflanzenzustand über ein mobiles Gerät, musste aufgrund fehlender Ressourcen und des aufwendigen zeitintensiven Charakters gestrichen werden. Es wäre angedacht gewesen hierfür einen Webserver mit dem ESP32 zu machen. Trotz des Wunsches, diese Funktion zu integrieren, stellte sich heraus, dass die Entwicklung und Gestaltung einer Website und Fernsteuerung technische Kenntnisse erfordert, die ausserhalb des Projektumfangs lagen. Dies war eine bewusste Entscheidung, um die verfügbaren Ressourcen auf die Kernziele des Projekts zu konzentrieren und sicherzustellen, dass diese erfolgreich umgesetzt werden.

Integration in ein 3D-gedrucktes Gehäuse:

Abschliessend wurde das gesamte System erfolgreich in ein 3D-gedrucktes Gehäuse integriert. Dieses Gehäuse ist nicht nur funktional, sondern auch ästhetisch ansprechend und benutzerfreundlich. Es schützt das System und passt sich gut in die Umgebung von Zimmerpflanzen ein.

Zusammenfassend wurden die Hauptziele des Projekts, die automatische Bewässerung von Zimmerpflanzen und die Anzeige des Pflanzenzustands auf einem LCD-Display, erfolgreich erreicht. Trotz der Einschränkungen wurde eine effektive Lösung entwickelt, die Pflanzenliebhabern dabei hilft, ihre Pflanzen gesund zu halten.

3 Reflexion

Die Durchführung der Diplomarbeit zum Thema "Automatisches Bewässerungssystem für Zimmerpflanzen" war eine herausfordernde und lehrreiche Erfahrung. In dieser Reflexion wird auf die Höhepunkte, Herausforderungen und Erkenntnisse eingegangen, die während des Projekts aufgetreten sind.

3.1 Höhepunkte

Erfüllte Erfolgskriterien: Es war äusserst befriedigend zu sehen, dass das Hauptziel der Arbeit, die automatische Bewässerung der Zimmerpflanzen, erfolgreich erreicht wurde. Das System funktionierte zuverlässig und trug dazu bei, die Pflanzen gesund zu halten.

Technische Fähigkeiten: Während des Projekts konnte ich meine technischen Fähigkeiten erheblich verbessern, insbesondere im Bereich 3D-Design und Programmierung. Dies wird mir in meiner zukünftigen beruflichen Laufbahn sicherlich von Nutzen sein.

Zeitmanagement: Die Arbeit an der Diplomarbeit half mir, meine Fähigkeiten im Zeitmanagement zu schärfen. Entscheidend und wichtig war, den Zeitplan einzuhalten und die Pufferzeit sinnvoll zu nutzen, um die Dokumentation fertigzustellen.

3.2 Herausforderungen

Budgetüberschreitung: Die Überschreitung des Budgets war eine unangenehme Überraschung und hat mir gezeigt, wie wichtig eine präzise Kostenkalkulation und die Berücksichtigung aller Ausgaben sind. In Zukunft wird sorgfältiger mit finanziellen Aspekten umgegangen.

Nicht erreichte Ziele: Das Streichen des Ziels, den Zugriff auf den Pflanzenzustand über ein mobiles Gerät zu ermöglichen, war zwar frustrierend, aber nötig, um sich auf den Kern des Projekts zu konzentrieren. Diese Entscheidung unterstreicht die Bedeutung der realistischen Zielsetzung, insbesondere in Bezug auf den zeitlichen Aspekt.

Beschaffung von Bauteilen: Die Beschaffung der Bauteile gestaltete sich aufgrund von Lieferverzögerungen als problematisch. Es war notwendig, flexibel zu sein und alternative Pläne zu entwickeln, um Verzögerungen zu minimieren. (ChatGPT Spellcheck, 2023)

3.3 Erkenntnisse

Realismus in der Zielsetzung: Die Diplomarbeit hat aufgezeigt, wie wichtig es ist, realistische Ziele zu setzen, die im Rahmen der verfügbaren Ressourcen und Fähigkeiten erreicht werden können. Weniger Ziele zu haben und diese vollständig zu erreichen, ist besser, als viele zu entwickeln und nur einen Teil dieser zu erreichen oder einen Anteil unvollendet zu lassen.

Budgetkontrolle: Die sorgfältige Budgetkontrolle und die Berücksichtigung aller Kosten, einschliesslich Lieferkosten, sind entscheidend. Eine genaue finanzielle Planung hilft, unerwartete Überschreitungen zu vermeiden.

Selbstreflexion: Diese Arbeit half dabei, die eigenen Fähigkeiten realistisch einzuschätzen. Es ist wichtig, rechtzeitig Unterstützung zu suchen und sich auf die persönlichen Stärken zu fokussieren.

Die Diplomarbeit war eine wertvolle Erfahrung, die einen nicht nur fachlich, sondern auch persönlich bereichert hat. Sie hat einem die Bedeutung von Planung, realistischen Erwartungen und kontinuierlicher Selbstverbesserung nähergebracht. Dies sind wichtige Lektionen, die sich mit Sicherheit in künftigen beruflichen Projekten als hilfreich erweisen werden. (ChatGPT Spellcheck, 2023)

4 Schlusswort

Mit dem Abschluss dieser Diplomarbeit, die sich mit der Entwicklung eines automatischen Bewässerungssystems für Zimmerpflanzen befasst, schliesse ich ein bedeutendes Kapitel meiner akademischen als auch praktischen Reise ab. Diese Arbeit war nicht nur eine Gelegenheit, meine technischen Fähigkeiten zu testen und zu vertiefen, sondern auch eine Lektion in den Bereichen des Projektmanagements, in der Selbstreflexion und zu der Bedeutung realistischer Zielsetzung.

Ich habe erfolgreich ein System geschaffen, das Zimmerpflanzen automatisch bewässert und den Zustand der Pflanzen in Echtzeit überwacht. Dieses Ergebnis war der Höhepunkt monatelanger Forschung, des Designs und der Implementierung. Das automatische Bewässerungssystem hat das Potenzial, Pflanzenliebhabern die Pflege ihrer Zimmerpflanzen zu erleichtern und sicherzustellen, dass sie stets optimal versorgt werden.

Die Erfolgskriterien dieser Arbeit wurden zum grossen Teil erfüllt, wobei die automatische Bewässerung und die Echtzeitanzeige des Pflanzenzustands auf dem LCD-Display im Mittelpunkt standen. Aufgrund einiger Herausforderungen, wie der geringen Budgetüberschreitung und dem Verzicht auf das Ziel, den Pflanzenzustand über ein mobiles Gerät abrufbar zu machen, wurden wertvolle Erkenntnisse gewonnen. Diese Erkenntnisse werden in Zukunft als Leitfaden für realistische Zielsetzungen und eine bessere Budgetkontrolle dienen.

Ich möchte mich bei meinen betreuenden Dozenten, insbesondere bei meinem Diplomcoach Christian Meier und Diplomexperten Philipp Amacker, für ihre Unterstützung und Anleitung während des gesamten Projekts bedanken. Ihre Unterstützung und Einsichten waren von unschätzbarem Wert.

Zum Abschluss möchte ich einen Ausblick auf die Zukunft geben. Dieses Projekt hat gezeigt, dass die Integration von Technologie in die Pflanzenpflege sowohl praktisch als auch lohnenswert sein kann. Die Ergebnisse könnten als Grundlage für weitere Forschung und Entwicklungen im Bereich der Pflanzenautomatisierung dienen. Die Fortsetzung der Arbeit an der Integration von Fernzugriffsfunktionen und die Verbesserung der Benutzerfreundlichkeit sind nur einige der möglichen Entwicklungsrichtungen.

Darüber hinaus besitzt die Anwendung von Sensortechnologie in der Agrarindustrie, insbesondere auf Feldern, ein grosses Potenzial. Die Überwachung von Bodenfeuchtigkeit, Temperatur und anderen Umweltparametern kann dazu beitragen, die Effizienz und Nachhaltigkeit der Landwirtschaft zu steigern. Dies könnte ein vielversprechendes Forschungsfeld für die Zukunft sein.

Insgesamt war die Diplomarbeit ein bedeutsames Projekt, das meine Fähigkeiten und mein Wissen erweitert hat. Sie erinnert mich daran, dass die Welt der Technik und Forschung niemals stillsteht, und ich freue mich darauf, meinen Beitrag zu weiteren Entwicklungen zu leisten. Als hoffentlich bald

frischgebackener Diplomierter Techniker in Elektrotechnik freue ich mich auf die neuen Herausforderungen, die meine berufliche Zukunft bereithält. Die erworbenen Fähigkeiten und das Wissen, die ich während meines Studiums und dieser Diplomarbeit erworben habe, werden mir bei der Bewältigung dieser Herausforderungen von unschätzbarem Wert sein. (ChatGPT Spellcheck, 2023)

Literaturverzeichnis

(August 2023). Von Vegetronix VH400: https://vegetronix.com/soil-moisture-sensor abgerufen

(August 2023). Von Distrelec Arduino Bodenfeuchtigkeitssensor: https://www.distrelec.ch/en/qwiic-soil-moisture-sensor-sparkfun-electronics-sen-17731/p/30216207?cq_src=google_ads&cq_cmp=18320642092&cq_con=&cq_term=&cq_med=pla&cq_plac=&cq_net=x&cq_pos=&cq_plt=gp&gclid=Cj0KCQjw1aOpBhCOARIsACXYv-fwHWPChCEWIIaw2No0H8nPgRw abgerufen

(August 2023). Von Digitec Raspberry Pi Zero: https://www.digitec.ch/de/s1/product/raspberry-pi-zero-w-bcm2835-entwicklungsboard-kit-14527596?gclid=CjwKCAjwxaanBhBQEiwA84TVXFE_JRsJUb8BtOwGVQysmPklAwX2uSGYF_5gXhTOqTUQRB4kXeJ-kRoCXNwQAvD_BwE&gclsrc=aw.ds abgerufen

(August 2023). Von Digitec Kapazitiver Bodenfeuchtigkeitssensor: https://www.digitec.ch/de/s1/product/dfrobot-kapazitiver-bodenfeuchtesensor-entwicklungsboard-kit-24881755?gclid=CjwKCAjwxaanBhBQEiwA84TVXKKC1zveNKq0mPubBBtaTXBAxGrZWObimDf27tM48brkSJl1ER47bBoClsEQAvD_BwE&gclsrc=aw.ds abgerufen

(August 2023). Von Digitec Raspberry Pi 4: https://www.digitec.ch/de/s1/product/raspberry-pi-4-2g-model-b-armv8-entwicklungsboard-kit-11267870?gclid=CjwKCAjwxaanBhBQEiwA84TVXGtc6jHIYrmlZWnuo0IFOqDVyMxM4qzMV53_urFJl8rcSGC-g0Ly1RoC93sQAvD_BwE&gclsrc=aw.ds abgerufen

(August 2023). Von Digitec ESP8266: https://www.galaxus.ch/en/s1/product/purecrea-wemos-d1-mini-esp8266-nodemcu-lua-board-development-boards-kits-35827905?gclid=CjwKCAjwxaanBhBQEiwA84TVXGfmyWn7ifCm-z5RDUJDqS64O5JZwWmJuxVd7d1TuiAW4eBFggZ09hoC8-UQAvD_BwE&gclsrc=aw.ds abgerufen

(August 2023). Von Digitec ESP32: https://www.digitec.ch/de/s1/product/joy-it-debo-jt-esp32-entwicklerboards-lx6-entwicklungsboard-kit-14171511 abgerufen

(August 2023). Von Digitec DHT22: https://www.digitec.ch/de/s1/product/waveshare-dht22-temperatur-und-luftfeuchtigkeitssensor-steckbar-entwicklungsboard-kit-24963577?ip=dht2 abgerufen

(August 2023). Von Galaxus AM2315: https://www.galaxus.ch/en/s1/product/purecrea-aosong-am2315-i2c-temperature-humidity-sensor-development-boards-kits-35828788?gclid=CjwKCAjwxaanBhBQEiwA84TVXFrj-eeKt18LZsJ-

DnXOh4A2WiDnKNgWuFMBJ8a6jekcs7FLxMlDuRoCbeMQAvD_BwE&gclsrc=aw.ds abgerufen

(August 2023). Von Digitec Relais: https://www.digitec.ch/de/s1/product/purecrea-1-kanal-relais-modul-elektronikmodul-25475141?utm_source=google&utm_medium=cpc&campaignid=20384870329&adgroupid=&adid=&dgCidg=Cj0KCQjw1aOpBhCOARIsACXYv-ef14BsDAk5enTeoANNsVZgiYjPk0LHoJaWBQjDlfwOthjRCTy5z-8aAkJ abgerufen

(August 2023). Von Arduino: https://docs.arduino.cc/hardware/mega-2560 abgerufen

(August 2023). Von Bastelgarage Wasserpumpe: https://www.bastelgarage.ch/micro-wasserpumpe-horizontal-100l-h-3-6v-mit-schlauch abgerufen

(August 2023). Von Berrybase LCD Display: https://www.berrybase.ch/alphanumerisches-lcd-16x2-blau/weiss-mit-i2c-backpack?sPartner=g_shopping_ch&gclid=Cj0KCQjw1aOpBhCOARIsACXYv-eGCXLtTQ-q3lho1yYLJEXYfeYFKwUsuu-8EVIix6wSj6qZuwIMgdoaAr1ZEALw_wcB abgerufen

(August 2023). Von DFRobot Füllstandsensor: https://www.dfrobot.com/product-690.html abgerufen

(August 2023). Von Digikey Arduino Füllstandsensor: https://www.digikey.ch/de/products/detail/adafruit-industries-llc/4965/14302510?utm_adgroup=&utm_source=google&utm_medium=cpc&utm_campaign=PMax%20Shopping_Product_High%20ROAS&utm_term=&productid=14302510&utm_content=&utm_id=go_cmp-20198980972_adg-_ad-__de abgerufen

(August 2023). Von Digikey Schwimmsensor: https://www.digikey.ch/de/products/detail/dfrobot/SEN0509/15848101?utm_adgroup=&utm_source=google&utm_medium=cpc&utm_campaign=PMax%20Shopping_Product_High%20ROAS&utm_term=&productid=15848101&utm_content=&utm_id=go_cmp-20198980972_adg-_ad-__dev-c_ext-_prd- abgerufen

(August 2023). Von Digikey TFT Display: https://www.digikey.ch/de/products/detail/adafruit-industries-llc/4311/10313914?utm_adgroup=General&utm_source=google&utm_medium=cpc&utm_campaign=Shopping_Product_All%20%28Catch-up%29&utm_term=&productid=10313914&gclid=CjwKCAjwxaanBhBQEiwA84TVXP9-5c0TAUBd abgerufen

(August 2023). Von Digitec Analog/Digital Wandler: https://www.digitec.ch/de/s1/product/mic-mcp3008-8-kanal-10-bit-adc-mit-spi-interface-elektronikmodul-

5998576?gclid=CjwKCAjwxaanBhBQEiwA84TVXGNaqvrm3K9pFHV5pBL49YFYZAv9HchpZXyBgDggPGJX3KKamp9T5xoCoWwQAvD_BwE&gclsrc=aw.ds abgerufen

(August 2023). Von Digitec Banana Pi: https://www.digitec.ch/de/s1/product/banana-pi-m2-berry-cortex-a7-entwicklungsboard-kit-6395120?gclid=CjwKCAjwxaanBhBQEiwA84TVXLrxIcrz0xmcTSyxUTOFxqzJDV7lgXOmJIJ2gV74ZlIm2_5WJauCeRoCebsQAvD_BwE&gclsrc=aw.ds abgerufen

(August 2023). Von Digitec DHT11: https://www.digitec.ch/en/s1/product/purecrea-dht11-temperature-and-humidity-sensor-development-boards-kits-32964684?gclid=CjwKCAjwxaanBhBQEiwA84TVXDtcQ58SflzZZ92-x3Nv7x7glxSWBc5JZPu1THh06wL0SLcbCUl-7xoC1eAQAvD_BwE&gclsrc=aw.ds abgerufen

(September 2023). Von Fritzing: https://fritzing.org/ abgerufen

(September 2023). Von Electronicshub: https://www.electronicshub.org/wp-content/uploads/2021/02/ESP32-Pinout-1.jpg abgerufen

(September 2023). Von Visual Studio Code: https://code.visualstudio.com/download abgerufen

(September 2023). Von Arduino Forum: https://forum.arduino.cc/t/motorsteuerung/699905 abgerufen

(September 2023). Von Upesy: https://www.upesy.com/blogs/tutorials/esp32-dht22-am2302-humidity-temperature-sensor-with-arduino-code# abgerufen

(September 2023). Von DF Robot: https://wiki.dfrobot.com/Non-contact_Liquid_Level_Switch_SKU_FIT0212 abgerufen

(September 2023). Von Lastminuteengineers: https://lastminuteengineers.com/esp32-i2c-lcd-tutorial/ abgerufen

(September 2023). Von FreeCAD: https://www.freecad.org/ abgerufen

(September 2023). Von SolidWorks: https://www.solidworks.com/de/product/all-products abgerufen

(September 2023). Von Fusion 360: https://www.autodesk.ch/de/products/fusion-360/trial-intake abgerufen

(September 2023). Von ChatGPT Spellcheck: https://chat.openai.com/ abgerufen

(September 2023). Von Scoobe3d: https://scoobe3d.com/beste-kostenlose-cad-software/ abgerufen

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

4.1 Code

```
//Hier werden alle Bibliotheken integriert, welche für das Projekt gebraucht werden.
#include <Arduino.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

//Display
bool isMainPage = false;
LiquidCrystal_I2C lcd(0x27,16,2); // setzt die LCD addresse auf 0x27 für 16 Zeichen auf 2 Linien

//DHT22
#define DHTPIN 18   //Definiert den DHT pin auf pin 18
#define DHTTYPE DHT22 //Definiert den DHT Typ (11 oder 22)
DHT dht(DHTPIN, DHTTYPE);

//Relais
int relaisPin = 15;

//Fuellstandsensor
int tankSensorPin1 = 19; //10% Wassertanksensor
int tankSensorPin2 = 4; //25% Wassertanksensor
int tankSensorPin3 = 16; //50% Wassertanksensor
int tankSensorPin4 = 17; //75% Wassertanksensor
int tankSensorPin5 = 5; //100% Wassertanksensor

//Erdfeuchtigkeitssensor
int soilSensorPin = 13;

//Methoden
int getWaterTankLevel() {   //Funktion zum Ermitteln des aktuellen Füllstands
  if (!digitalRead(tankSensorPin5)) {   //Gibt den Wert 100 zurück wenn der TankSensor 5 angibt
    return 100;
  }
  if (!digitalRead(tankSensorPin4)) {   //Gibt den Wert 75 zurück wenn der TankSensor 4 angibt
    return 75;
  }
  if (!digitalRead(tankSensorPin3)) {   //Gibt den Wert 50 zurück wenn der TankSensor 3 angibt
    return 50;
  }
  if (!digitalRead(tankSensorPin2)) {   //Gibt den Wert 25 zurück wenn der TankSensor 2 angibt
    return 25;
  }
  if (!digitalRead(tankSensorPin1)) {   //Gibt den Wert 10 zurück wenn der TankSensor 1 angibt
    return 10;
  }
  return 0;   //Wenn keiner der Sensoren angibt, wird der Wert 0 ausgegeben
}

```

```
51
52  int getSoilMoisture() {   //Funktion zum auslesen der Erdfeuchtigkeit
53
54    int sensorValue = analogRead(soilSensorPin);    //Liesst den Sensorwert aus
55
56    int dryValue = 0;   //Hier wird der Wert für die trockene Erde definiert
57    int wetValue = 3770;    ////Hier wird der Wert für die nasse Erde definiert
58
59    float soilMoisture = map(sensorValue, dryValue, wetValue, 0, 100); // Berechnet die Bodenfeuchtigkeit in Prozent
60
61    return (int)soilMoisture;  //Gibt den aktuellen Feuchtigkeitswert in % an
62  }
63
64
65  void showLcd() {    //Funktion fürs LCD-Display
66    lcd.clear();
67    if (isMainPage) {   //Definiert dieses if als MainPage
68      lcd.setCursor(0,0);   //Setzt den Cursor auf den 0. Buchstaben, Zeile 0
69      lcd.print("Soil: "+ String(getSoilMoisture()) + "%");  //Gibt den aktuellen Bodenfeuchtigkeitswert auf dem LCD Display aus
70      lcd.setCursor(0,1);   //Setzt den Cursor auf den 0. Buchstaben, Zeile 1
71      lcd.print("Water: "+ String(getWaterTankLevel()) + "%");   //Gibt de naktuellen Füllstand vom Wassertank auf dem LCD Display aus
72    }
73    else {
74      lcd.setCursor(0,0);   //Setzt den Cursor auf den 0. Buchstaben, Zeile 0
75      lcd.print("Humidity: " + String(dht.readHumidity()) + "%"); //Gibt die aktuelle Luftfeuchtigkeit auf dem  LCD Display aus
76      lcd.setCursor(0,1);   //Setzt den Cursor auf den 0. Buchstaben, Zeile 1
77      lcd.print("Temp: " + String(dht.readTemperature()));  //Gibt die aktuelle Temperatur auf dem LCD Display aus
78      lcd.print((char)223);   //223 ist der Code für das Grad Zeichen in der ASCII Tabelle
79      lcd.print("C");   //Schreibt ein C hinter das Gradzeichen
80    }
81    isMainPage = !isMainPage;   //invertiert die MainPage, damit zwischen Seite 1 und 2 gewechselt wird
82  }
83
84
85  void setup() {
86    Serial.begin(115200);   //setzt serial monitor geschwindigkeit auf 115200 Wichtig:muss auch in Platform.ini angepasst werden
87
88    pinMode(relaisPin, OUTPUT); //Inizialisiert Pin das er schreiben kann
89    digitalWrite(relaisPin, LOW); //setzt Pin am anfang auf low
90
91    pinMode(tankSensorPin1, INPUT); //Inizialisiert Pin das er lesen kann
92    pinMode(tankSensorPin2, INPUT); //Inizialisiert Pin das er lesen kann
93    pinMode(tankSensorPin3, INPUT); //Inizialisiert Pin das er lesen kann
94    pinMode(tankSensorPin4, INPUT); //Inizialisiert Pin das er lesen kann
95    pinMode(tankSensorPin5, INPUT); //Inizialisiert Pin das er lesen kann
96
97    dht.begin();    //DHT wird gestartet
98
99    lcd.init();     //Initialisiert LCD
100   lcd.backlight();  //Hintergrundbeleuchtung ein
101 }
102
```

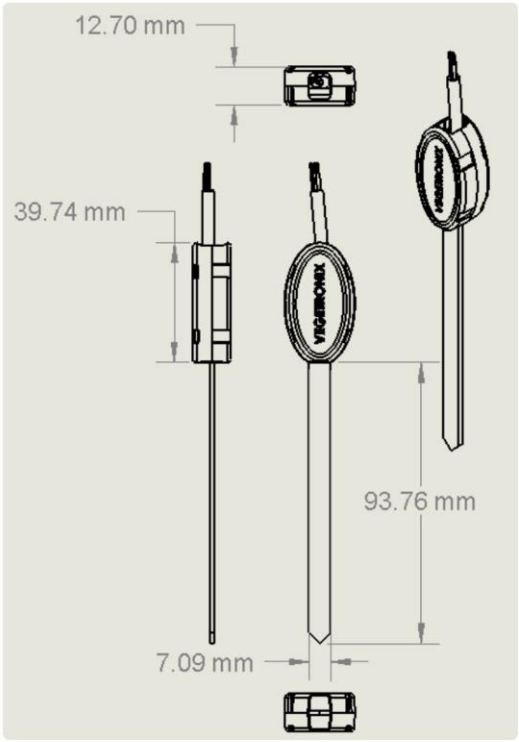
```
103
104 void loop() {
105   showLcd();
106
107   //Watering
108   if ((getSoilMoisture() < 30) and (getWaterTankLevel() > 0)) {   //Gibt der Pflanze für 5 Sekunden Wasser, sobald die Erdfeuchtigkeit unter 30% ist und der Wassertank nicht leer ist
109     digitalWrite(relaisPin, HIGH);
110     delay(5000);
111     digitalWrite(relaisPin, LOW);
112   }
113
114   delay(2000);    //Loop wird nach 2 Sekunden erneut ausgeführt
115 }
```

4.2 VH 400 Datenblatt

Soil Moisture Sensor Probe Specifications

VH400 Sensor	
Power consumption	< 13mA
Supply Voltage	3.5V to 20 VDC.
Dimensions	See drawing below.
Power on to Output stable	400 ms
Output Impedance	10K ohms
Operational Temperature	-40ºC to 85ºC
Accuracy at 25°C	2%
Output	0 to 3V related to moisture content
Shell Color	Red
Voltage Output Curves	Curves, Piecewise linear equations
Certifications	CE Declaration of Conformity

VH400 Soil Moisture Sensor Probe Drawing



VH400 Soil Moisture sensor Drawing

Soil Moisture Sensor Probe Wiring Table

Bare	Ground
Red	POWER: 3.5V to 20 VDC.
Black	OUT: (0 to 3V related to moisture content.)

4.3 DHT22 Datenblatt

Beschreibung

Der DHT22 hat einen kalibrierten AM2302 Sensor verbaut, mit dem du Temperatur und Luftfeuchtigkeit messen kannst. Gegenüber dem preiswerten DHT11 hat der DHT22 eine höhere Genauigkeit und einen grösseren Messbereich. Auf dem Sensor ist ein XH2.54mm Stecker verbaut. Somit kannst du auch selbst gecrimpte Kabel einfach an den Sensor anschliessen. Angesteuert wird der DHT22 über eine Digitale 1-Wire Schnittstelle. Somit ist der Betrieb mit einem Arduino oder ESP8266 einfach zu realisieren. Eine fertige Arduino Library für die Ansteuerung findest du im Library Manager unter „DHT Sensor Library“. Auf dem Board sind bereits die Widerstände verbaut, die zur Ansteuerung benötigt werden. Somit kannst du das Board einfach an dein Arduino anstecken und loslegen.

Anwendungen Beispiele für den DHT22:
Wetterstation
Luftrohr
Feuchteregler
Temperaturregler
Messen der Raumtemperatur und Luftfeuchtigkeit
Test- und Detektionsgerät

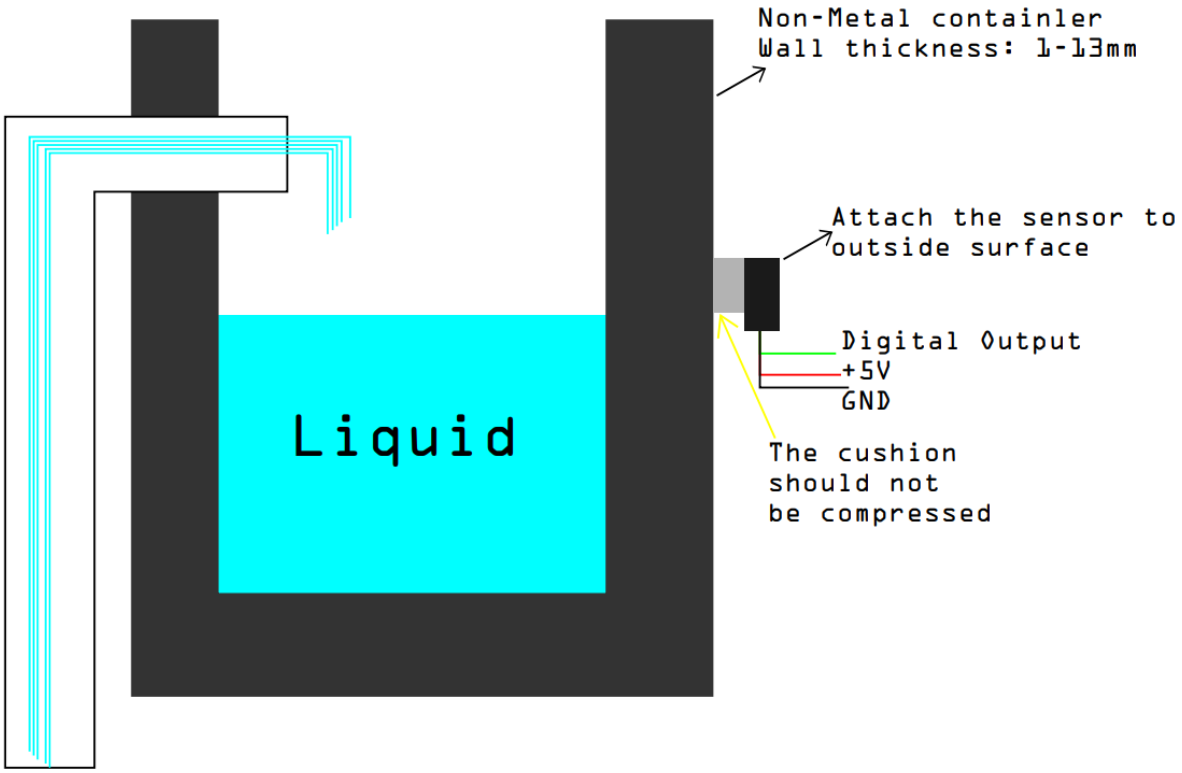
Technische Details:
Modell: DHT22 AM2302
Betriebsspannung: 3.3VDC – 5.5VDC
Benötigt nur ein Pin (1-Wire)
Messbereich Luftfeuchtigkeit: 0% - 99.9% (2 bis 5% Messgenauigkeit
Messbereich Temperatur: -40°C – 80°C (±0.5 °C Messgenauigkeit)
Auflösung: 0.1% rH/0.1 °C
Antwortzeit: ca. 2s bei höchster Auflösung
Abmessung: 40 x 16 x 10mm
Gewicht: 8g

Lieferumfang:
1x DHT22 Temperatur und Luftfeuchtigkeitssensor steckbar
1x XH2.54mm auf 1x3Pin Female Dupont Kabel 20cm.

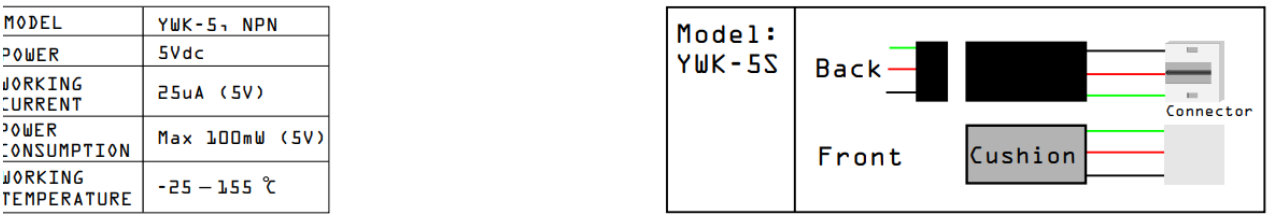
↑ Weniger anzeigen

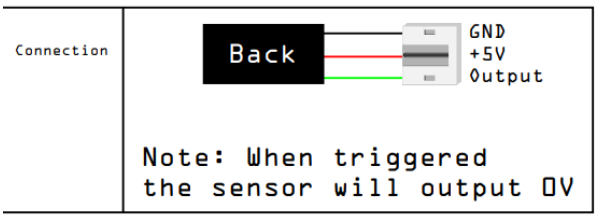
4.4 Füllstandsensor Datenblatt

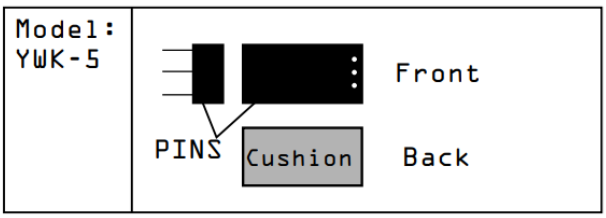
Non-contact Liquid Levels Switch



MODEL	YWK-5, NPN
POWER	5Vdc
WORKING CURRENT	25uA (5V)
POWER CONSUMPTION	Max 100mW (5V)
WORKING TEMPERATURE	-25 – 155 ℃







4.5 PLA-Datenblatt

Produktinformationen & technische Daten:

Art.-Nr.:	ESUN-PLA+175B1	**Produktarten:**	PLA Filament
Hersteller-Nr.:	PLA+175B1	**Farbe:**	Schwarz
Marken (Hersteller):	eSUN	**Nettogewicht:**	1000 g, 3000 g, 5000 g
Inhalt:	1.000 g	**Empf. Drucktemperatur:**	210 - 230 °C
Durchmesser:	1,75 mm	**Empf. Heizbetttemperatur:**	45 - 60 °C

Produktinformationen & technische Daten:

Art.-Nr.:	ESUNCN-PLA+175W5	**Produktarten:**	PLA Filament
Hersteller-Nr.:	PLA+175W5	**Farbe:**	Weiß
Marken (Hersteller):	eSUN	**Nettogewicht:**	1000 g, 3000 g, 5000 g
Inhalt:	5.000 g	**Empf. Drucktemperatur:**	210 - 230 °C
Durchmesser:	1,75 mm	**Empf. Heizbetttemperatur:**	45 - 60 °C

4.6 ESP32-WROOM-32 Datenblatt

About This Document

This document provides the specifications for the ESP32-WROOM-32 module.

Revision History

For revision history of this document, please refer to the last page.

Documentation Change Notification

Espressif provides email notifications to keep customers updated on changes to technical documentation. Please subscribe at www.espressif.com/en/subscribe.

Certification

Download certificates for Espressif products from www.espressif.com/en/certificates.

Disclaimer and Copyright Notice

Information in this document, including URL references, is subject to change without notice. THIS DOCUMENT IS PROVIDED AS IS WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

All liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein. The Wi-Fi Alliance Member logo is a trademark of the Wi-Fi Alliance. The Bluetooth logo is a registered trademark of Bluetooth SIG.

All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners, and are hereby acknowledged.

Contents

List of Tables

List of Figures

1. Overview

ESP32-WROOM-32 is a powerful, generic Wi-Fi+BT+BLE MCU module that targets a wide variety of applications, ranging from low-power sensor networks to the most demanding tasks, such as voice encoding, music streaming and MP3 decoding.

At the core of this module is the ESP32-D0WDQ6 chip*. The chip embedded is designed to be scalable and adaptive. There are two CPU cores that can be individually controlled, and the CPU clock frequency is adjustable from 80 MHz to 240 MHz. The user may also power off the CPU and make use of the low-power co-processor to constantly monitor the peripherals for changes or crossing of thresholds. ESP32 integrates a rich set of peripherals, ranging from capacitive touch sensors, Hall sensors, SD card interface, Ethernet, high-speed SPI, UART, I²S and I²C.

Note:
* For details on the part numbers of the ESP32 family of chips, please refer to the document *ESP32 Datasheet*.

The integration of Bluetooth, Bluetooth LE and Wi-Fi ensures that a wide range of applications can be targeted, and that the module is all-around: using Wi-Fi allows a large physical range and direct connection to the Internet through a Wi-Fi router, while using Bluetooth allows the user to conveniently connect to the phone or broadcast low energy beacons for its detection. The sleep current of the ESP32 chip is less than 5 μA, making it suitable for battery powered and wearable electronics applications. The module supports a data rate of up to 150 Mbps, and 20 dBm output power at the antenna to ensure the widest physical range. As such the module does offer industry-leading specifications and the best performance for electronic integration, range, power consumption, and connectivity.

The operating system chosen for ESP32 is freeRTOS with LwIP; TLS 1.2 with hardware acceleration is built in as well. Secure (encrypted) over the air (OTA) upgrade is also supported, so that users can upgrade their products even after their release, at minimum cost and effort.

Table 1 provides the specifications of ESP32-WROOM-32.

Table 1: ESP32-WROOM-32 Specifications

Categories	Items	Specifications
Certification	RF certification	FCC/CE-RED/IC/TELEC/KCC/SRRC/NCC
	Wi-Fi certification	Wi-Fi Alliance
	Bluetooth certification	BQB
	Green certification	RoHS/REACH
Test	Reliablity	HTOL/HTSL/uHAST/TCT/ESD
Wi-Fi	Protocols	802.11 b/g/n (802.11n up to 150 Mbps)
		A-MPDU and A-MSDU aggregation and 0.4 μs guard interval support
	Frequency range	2.4 GHz ~ 2.5 GHz
Bluetooth	Protocols	Bluetooth v4.2 BR/EDR and BLE specification
	Radio	NZIF receiver with –97 dBm sensitivity
		Class-1, class-2 and class-3 transmitter
		AFH

Categories	Items	Specifications
	Audio	CVSD and SBC
Hardware	Module interfaces	SD card, UART, SPI, SDIO, I^2C, LED PWM, Motor PWM, I^2S, IR, pulse counter, GPIO, capacitive touch sensor, ADC, DAC
	On-chip sensor	Hall sensor
	Integrated crystal	40 MHz crystal
	Integrated SPI flash	4 MB
	Operating voltage/Power supply	3.0 V ~ 3.6 V
	Operating current	Average: 80 mA
	Minimum current delivered by power supply	500 mA
	Recommended operating temperature range	–40 °C ~ +85 °C
	Package size	(18.00±0.10) mm × (25.50±0.10) mm × (3.10±0.10) mm
	Moisture sensitivity level (MSL)	Level 3

2. Pin Definitions

2.1 Pin Layout

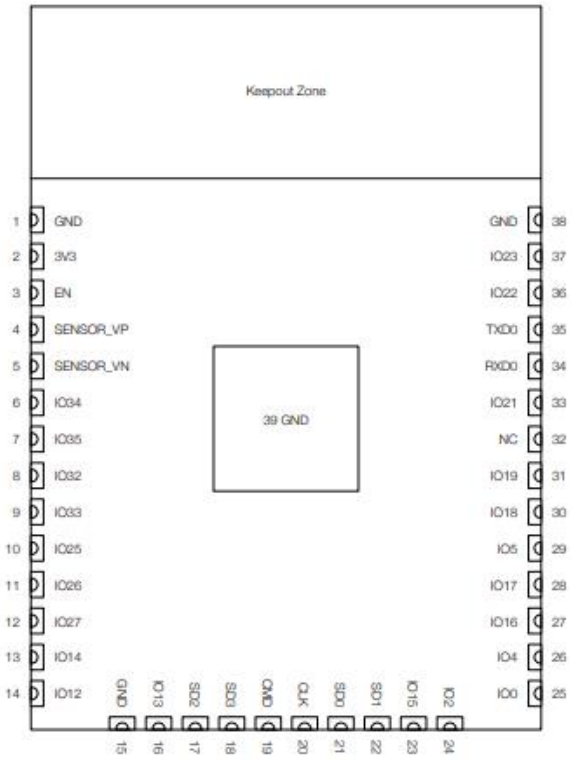


Figure 1: ESP32-WROOM-32 Pin Layout (Top View)

2.2 Pin Description

ESP32-WROOM-32 has 38 pins. See pin definitions in Table 2.

Table 2: Pin Definitions

Name	No.	Type	Function
GND	1	P	Ground
3V3	2	P	Power supply
EN	3	I	Module-enable signal. Active high.
SENSOR_VP	4	I	GPIO36, ADC1_CH0, RTC_GPIO0
SENSOR_VN	5	I	GPIO39, ADC1_CH3, RTC_GPIO3
IO34	6	I	GPIO34, ADC1_CH6, RTC_GPIO4
IO35	7	I	GPIO35, ADC1_CH7, RTC_GPIO5
IO32	8	I/O	GPIO32, XTAL_32K_P (32.768 kHz crystal oscillator input), ADC1_CH4, TOUCH9, RTC_GPIO9
IO33	9	I/O	GPIO33, XTAL_32K_N (32.768 kHz crystal oscillator output), ADC1_CH5, TOUCH8, RTC_GPIO8

Name	No.	Type	Function
IO25	10	I/O	GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6, EMAC_RXD0
IO26	11	I/O	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1
IO27	12	I/O	GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17, EMAC_RX_DV
IO14	13	I/O	GPIO14, ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2
IO12	14	I/O	GPIO12, ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI, HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3
GND	15	P	Ground
IO13	16	I/O	GPIO13, ADC2_CH4, TOUCH4, RTC_GPIO14, MTCK, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER
SHD/SD2*	17	I/O	GPIO9, SD_DATA2, SPIHD, HS1_DATA2, U1RXD
SWP/SD3*	18	I/O	GPIO10, SD_DATA3, SPIWP, HS1_DATA3, U1TXD
SCS/CMD*	19	I/O	GPIO11, SD_CMD, SPICS0, HS1_CMD, U1RTS
SCK/CLK*	20	I/O	GPIO6, SD_CLK, SPICLK, HS1_CLK, U1CTS
SDO/SD0*	21	I/O	GPIO7, SD_DATA0, SPIQ, HS1_DATA0, U2RTS
SDI/SD1*	22	I/O	GPIO8, SD_DATA1, SPID, HS1_DATA1, U2CTS
IO15	23	I/O	GPIO15, ADC2_CH3, TOUCH3, MTDO, HSPICS0, RTC_GPIO13, HS2_CMD, SD_CMD, EMAC_RXD3
IO2	24	I/O	GPIO2, ADC2_CH2, TOUCH2, RTC_GPIO12, HSPIWP, HS2_DATA0, SD_DATA0
IO0	25	I/O	GPIO0, ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1, EMAC_TX_CLK
IO4	26	I/O	GPIO4, ADC2_CH0, TOUCH0, RTC_GPIO10, HSPIHD, HS2_DATA1, SD_DATA1, EMAC_TX_ER
IO16	27	I/O	GPIO16, HS1_DATA4, U2RXD, EMAC_CLK_OUT
IO17	28	I/O	GPIO17, HS1_DATA5, U2TXD, EMAC_CLK_OUT_180
IO5	29	I/O	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
IO18	30	I/O	GPIO18, VSPICLK, HS1_DATA7
IO19	31	I/O	GPIO19, VSPIQ, U0CTS, EMAC_TXD0
NC	32	-	-
IO21	33	I/O	GPIO21, VSPIHD, EMAC_TX_EN
RXD0	34	I/O	GPIO3, U0RXD, CLK_OUT2
TXD0	35	I/O	GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2
IO22	36	I/O	GPIO22, VSPIWP, U0RTS, EMAC_TXD1
IO23	37	I/O	GPIO23, VSPID, HS1_STROBE
GND	38	P	Ground

Notice:

* Pins SCK/CLK, SDO/SD0, SDI/SD1, SHD/SD2, SWP/SD3 and SCS/CMD, namely, GPIO6 to GPIO11 are connected to the integrated SPI flash integrated on the module and are not recommended for other uses.

2.3 Strapping Pins

ESP32 has five strapping pins, which can be seen in Chapter 6 Schematics:

- MTDI
- GPIO0
- GPIO2
- MTDO
- GPIO5

Software can read the values of these five bits from register "GPIO_STRAPPING".

During the chip's system reset release (power-on-reset, RTC watchdog reset and brownout reset), the latches of the strapping pins sample the voltage level as strapping bits of "0" or "1", and hold these bits until the chip is powered down or shut down. The strapping bits configure the device's boot mode, the operating voltage of VDD_SDIO and other initial system settings.

Each strapping pin is connected to its internal pull-up/pull-down during the chip reset. Consequently, if a strapping pin is unconnected or the connected external circuit is high-impedance, the internal weak pull-up/pull-down will determine the default input level of the strapping pins.

To change the strapping bit values, users can apply the external pull-down/pull-up resistances, or use the host MCU's GPIOs to control the voltage level of these pins when powering on ESP32.

After reset release, the strapping pins work as normal-function pins.

Refer to Table 3 for a detailed boot-mode configuration by strapping pins.

Table 3: Strapping Pins

Voltage of Internal LDO (VDD_SDIO)			
Pin	Default	3.3 V	1.8 V
MTDI	Pull-down	0	1
Booting Mode			
Pin	Default	SPI Boot	Download Boot
GPIO0	Pull-up	1	0
GPIO2	Pull-down	Don't-care	0
Enabling/Disabling Debugging Log Print over U0TXD During Booting			
Pin	Default	U0TXD Active	U0TXD Silent
MTDO	Pull-up	1	0

Timing of SDIO Slave					
Pin	Default	Falling-edge Sampling Falling-edge Output	Falling-edge Sampling Rising-edge Output	Rising-edge Sampling Falling-edge Output	Rising-edge Sampling Rising-edge Output
MTDO	Pull-up	0	0	1	1
GPIO5	Pull-up	0	1	0	1

Note:

- Firmware can configure register bits to change the settings of "Voltage of Internal LDO (VDD_SDIO)" and "Timing of SDIO Slave" after booting.
- The module integrates a 3.3 V SPI flash, so the pin MTDI cannot be set to 1 when the module is powered up.

The strapping pins need a setup and hold time before and after the EN signal goes high. For details please refer to Section Strapping Pins in *ESP32 Datasheet*.

3. Functional Description

This chapter describes the modules and functions integrated in ESP32-WROOM-32.

3.1 CPU and Internal Memory

ESP32-D0WDQ6 contains two low-power Xtensa® 32-bit LX6 microprocessors. The internal memory includes:

- 448 KB of ROM for booting and core functions.
- 520 KB of on-chip SRAM for data and instructions.
- 8 KB of SRAM in RTC, which is called RTC FAST Memory and can be used for data storage; it is accessed by the main CPU during RTC Boot from the Deep-sleep mode.
- 8 KB of SRAM in RTC, which is called RTC SLOW Memory and can be accessed by the co-processor during the Deep-sleep mode.
- 1 Kbit of eFuse: 256 bits are used for the system (MAC address and chip configuration) and the remaining 768 bits are reserved for customer applications, including flash-encryption and chip-ID.

3.2 External Flash and SRAM

ESP32 supports multiple external QSPI flash and SRAM chips. More details can be found in Chapter SPI in the *ESP32 Technical Reference Manual*. ESP32 also supports hardware encryption/decryption based on AES to protect developers' programs and data in flash.

ESP32 can access the external QSPI flash and SRAM through high-speed caches.

- The external flash can be mapped into CPU instruction memory space and read-only memory space simultaneously.
  - When external flash is mapped into CPU instruction memory space, up to 11 MB + 248 KB can be mapped at a time. Note that if more than 3 MB + 248 KB are mapped, cache performance will be reduced due to speculative reads by the CPU.
  - When external flash is mapped into read-only data memory space, up to 4 MB can be mapped at a time. 8-bit, 16-bit and 32-bit reads are supported.
- External SRAM can be mapped into CPU data memory space. Up to 4 MB can be mapped at a time. 8-bit, 16-bit and 32-bit reads and writes are supported.

ESP32-WROOM-32 integrates a 4 MB SPI flash, which is connected to GPIO6, GPIO7, GPIO8, GPIO9, GPIO10 and GPIO11. These six pins cannot be used as regular GPIOs.

3.3 Crystal Oscillators

The module uses a 40-MHz crystal oscillator.

3.4 RTC and Low-Power Management

With the use of advanced power-management technologies, ESP32 can switch between different power modes.

For details on ESP32's power consumption in different power modes, please refer to section "RTC and Low-Power Management" in *ESP32 Datasheet*.

4. Peripherals and Sensors

Please refer to Section Peripherals and Sensors in *ESP32 Datasheet*.

Note:
External connections can be made to any GPIO except for GPIOs in the range 6-11. These six GPIOs are connected to the module's integrated SPI flash. For details, please see Section 6 Schematics.

5. Electrical Characteristics

5.1 Absolute Maximum Ratings

Stresses beyond the absolute maximum ratings listed in Table 4 below may cause permanent damage to the device. These are stress ratings only, and do not refer to the functional operation of the device that should follow the recommended operating conditions.

Table 4: Absolute Maximum Ratings

Symbol	Parameter	Min	Max	Unit
VDD33	Power supply voltage	–0.3	3.6	V
I_{output} [1]	Cumulative IO output current	-	1,100	mA
T_{store}	Storage temperature	–40	150	°C

1. The module worked properly after a 24-hour test in ambient temperature at 25 °C, and the IOs in three domains (VDD3P3_RTC, VDD3P3_CPU, VDD_SDIO) output high logic level to ground. Please note that pins occupied by flash and/or PSRAM in the VDD_SDIO power domain were excluded from the test.
2. Please see Appendix IO_MUX of *ESP32 Datasheet* for IO's power domain.

5.2 Recommended Operating Conditions

Table 5: Recommended Operating Conditions

Symbol	Parameter	Min	Typical	Max	Unit
VDD33	Power supply voltage	3.0	3.3	3.6	V
I_{VDD}	Current delivered by external power supply	0.5	-	-	A
T	Operating temperature	–40	-	85	°C

5.3 DC Characteristics (3.3 V, 25 °C)

Table 6: DC Characteristics (3.3 V, 25 °C)

Symbol	Parameter		Min	Typ	Max	Unit
C_{IN}	Pin capacitance		-	2	-	pF
V_{IH}	High-level input voltage		0.75×VDD[1]	-	VDD[1]+0.3	V
V_{IL}	Low-level input voltage		–0.3	-	0.25×VDD[1]	V
I_{IH}	High-level input current		-	-	50	nA
I_{IL}	Low-level input current		-	-	50	nA
V_{OH}	High-level output voltage		0.8×VDD[1]	-	-	V
V_{OL}	Low-level output voltage		-	-	0.1×VDD[1]	V
I_{OH}	High-level source current (VDD[1] = 3.3 V, V_{OH} >= 2.64 V, output drive strength set to the maximum)	VDD3P3_CPU power domain [1, 2]	-	40	-	mA
		VDD3P3_RTC power domain [1, 2]	-	40	-	mA
		VDD_SDIO power domain [1, 3]	-	20	-	mA

Symbol	Parameter	Min	Typ	Max	Unit
I_{OL}	Low-level sink current (VDD[1] = 3.3 V, V_{OL} = 0.495 V, output drive strength set to the maximum)	-	28	-	mA
R_{PU}	Resistance of internal pull-up resistor	-	45	-	kΩ
R_{PD}	Resistance of internal pull-down resistor	-	45	-	kΩ
V_{IL_nRST}	Low-level input voltage of CHIP_PU to power off the chip	-	-	0.6	V

Notes:

1. Please see Appendix IO_MUX of ESP32 Datasheet for IO's power domain. VDD is the I/O voltage for a particular power domain of pins.
2. For VDD3P3_CPU and VDD3P3_RTC power domain, per-pin current sourced in the same domain is gradually reduced from around 40 mA to around 29 mA, V_{OH}>=2.64 V, as the number of current-source pins increases.
3. Pins occupied by flash and/or PSRAM in the VDD_SDIO power domain were excluded from the test.

5.4 Wi-Fi Radio

Table 7: Wi-Fi Radio Characteristics

Parameter	Condition	Min	Typical	Max	Unit
Operating frequency range [note1]	-	2412	-	2484	MHz
Output impedance [note2]	-	-	*note 2*	-	Ω
TX power [note3]	11n, MCS7	12	13	14	dBm
	11b mode	17.5	18.5	20	dBm
Sensitivity	11b, 1 Mbps	-	–98	-	dBm
	11b, 11 Mbps	-	–89	-	dBm
	11g, 6 Mbps	-	–92	-	dBm
	11g, 54 Mbps	-	–74	-	dBm
	11n, HT20, MCS0	-	–91	-	dBm
	11n, HT20, MCS7	-	–71	-	dBm
	11n, HT40, MCS0	-	–89	-	dBm
	11n, HT40, MCS7	-	–69	-	dBm
Adjacent channel rejection	11g, 6 Mbps	-	31	-	dB
	11g, 54 Mbps	-	14	-	dB
	11n, HT20, MCS0	-	31	-	dB
	11n, HT20, MCS7	-	13	-	dB

1. Device should operate in the frequency range allocated by regional regulatory authorities. Target operating frequency range is configurable by software.
2. For the modules that use IPEX antennas, the output impedance is 50 Ω. For other modules without IPEX antennas, users do not need to concern about the output impedance.
3. Target TX power is configurable based on device or certification requirements.

5.5 BLE Radio

5.5.1 Receiver

Table 8: Receiver Characteristics – BLE

Parameter	Conditions	Min	Typ	Max	Unit
Sensitivity @30.8% PER	-	-	–97	-	dBm
Maximum received signal @30.8% PER	-	0	-	-	dBm
Co-channel C/I	-	-	+10	-	dB
Adjacent channel selectivity C/I	F = F0 + 1 MHz	-	–5	-	dB
	F = F0 – 1 MHz	-	–5	-	dB
	F = F0 + 2 MHz	-	–25	-	dB
	F = F0 – 2 MHz	-	–35	-	dB
	F = F0 + 3 MHz	-	–25	-	dB
	F = F0 – 3 MHz	-	–45	-	dB
Out-of-band blocking performance	30 MHz ~ 2000 MHz	–10	-	-	dBm
	2000 MHz ~ 2400 MHz	–27	-	-	dBm
	2500 MHz ~ 3000 MHz	–27	-	-	dBm
	3000 MHz ~ 12.5 GHz	–10	-	-	dBm
Intermodulation	-	–36	-	-	dBm

5.5.2 Transmitter

Table 9: Transmitter Characteristics – BLE

Parameter	Conditions	Min	Typ	Max	Unit
RF transmit power	-	-	0	-	dBm
Gain control step	-	-	3	-	dBm
RF power control range	-	–12	-	+9	dBm
Adjacent channel transmit power	F = F0 ± 2 MHz	-	–52	-	dBm
	F = F0 ± 3 MHz	-	–58	-	dBm
	F = F0 ± > 3 MHz	-	–60	-	dBm
$\Delta f1_{avg}$	-	-	-	265	kHz
$\Delta f2_{max}$	-	247	-	-	kHz
$\Delta f2_{avg}/\Delta f1_{avg}$	-	-	–0.92	-	-
ICFT	-	-	–10	-	kHz
Drift rate	-	-	0.7	-	kHz/50 μs
Drift	-	-	2	-	kHz

5.6 Reflow Profile

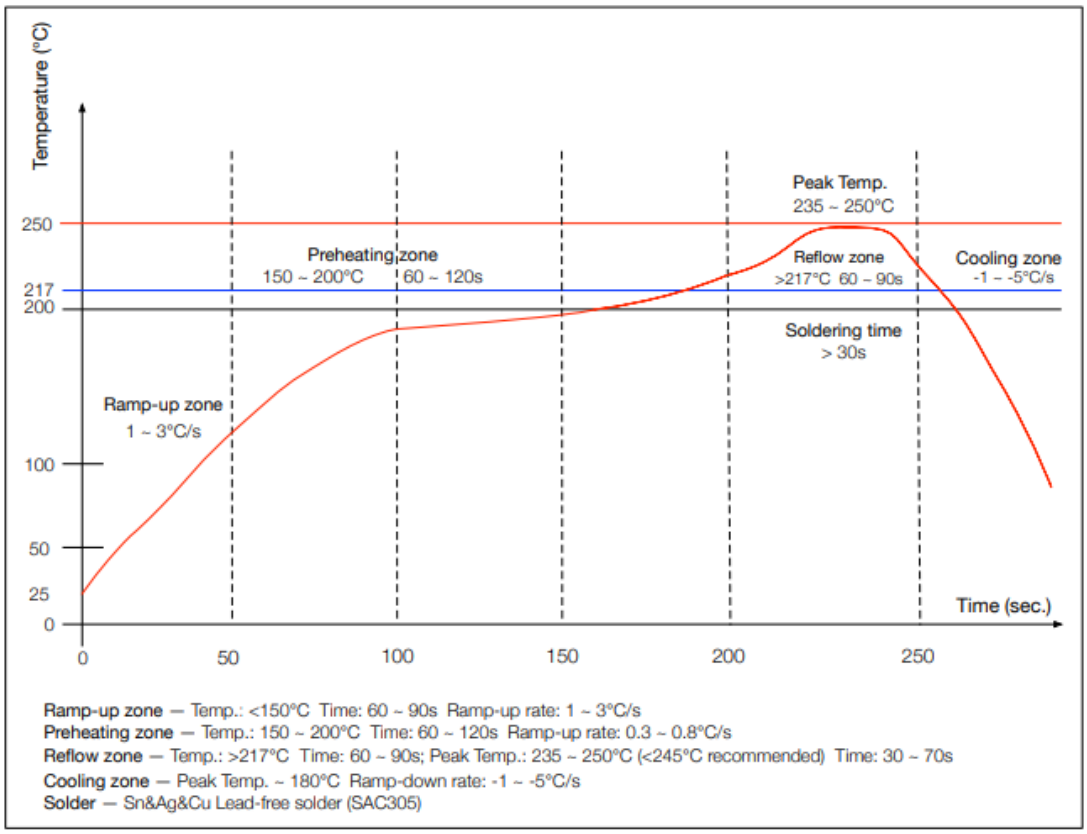


Figure 2: Reflow Profile

6. Schematics

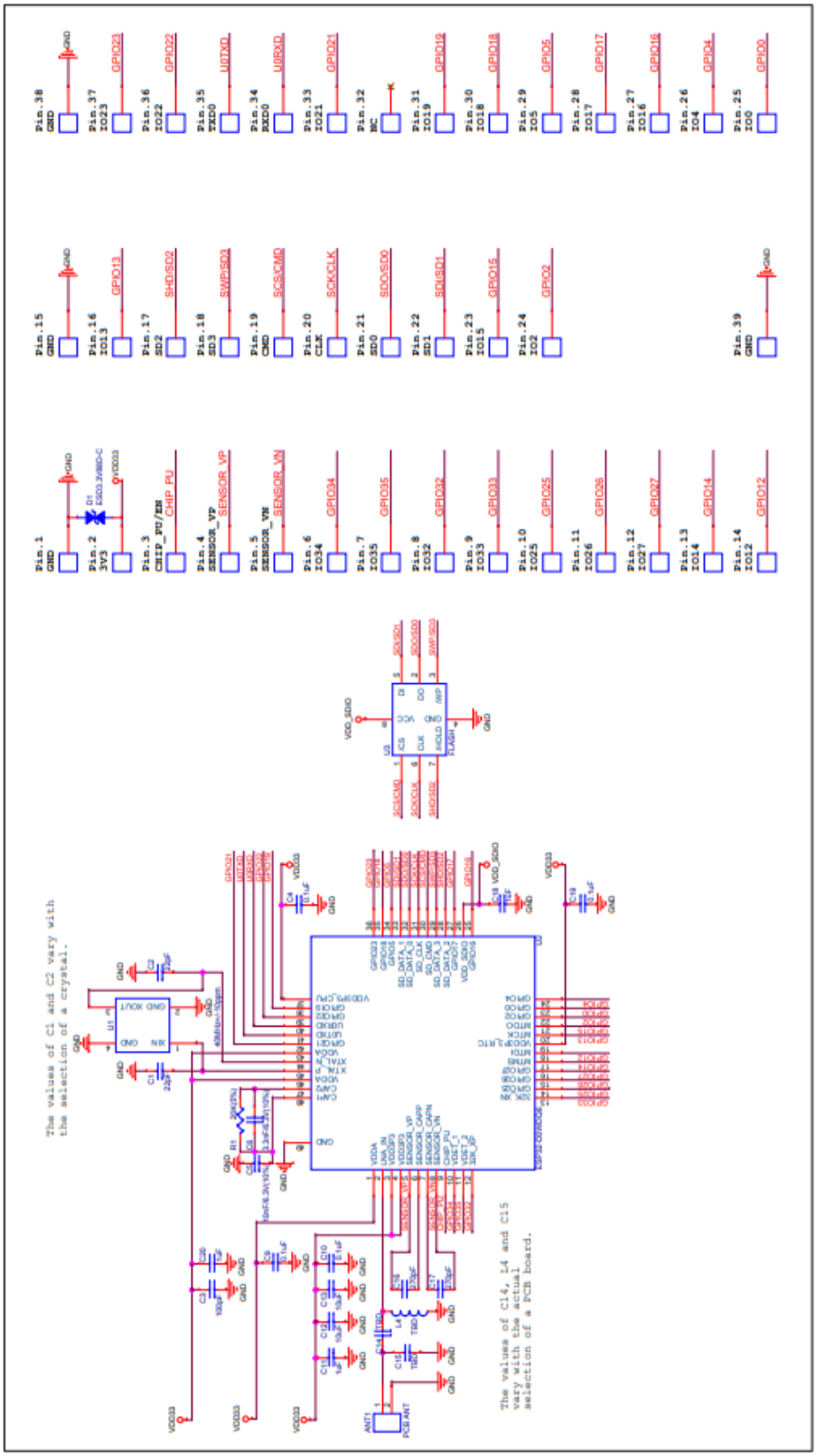


Figure 3: ESP32-WROOM-32 Schematics

7. Peripheral Schematics

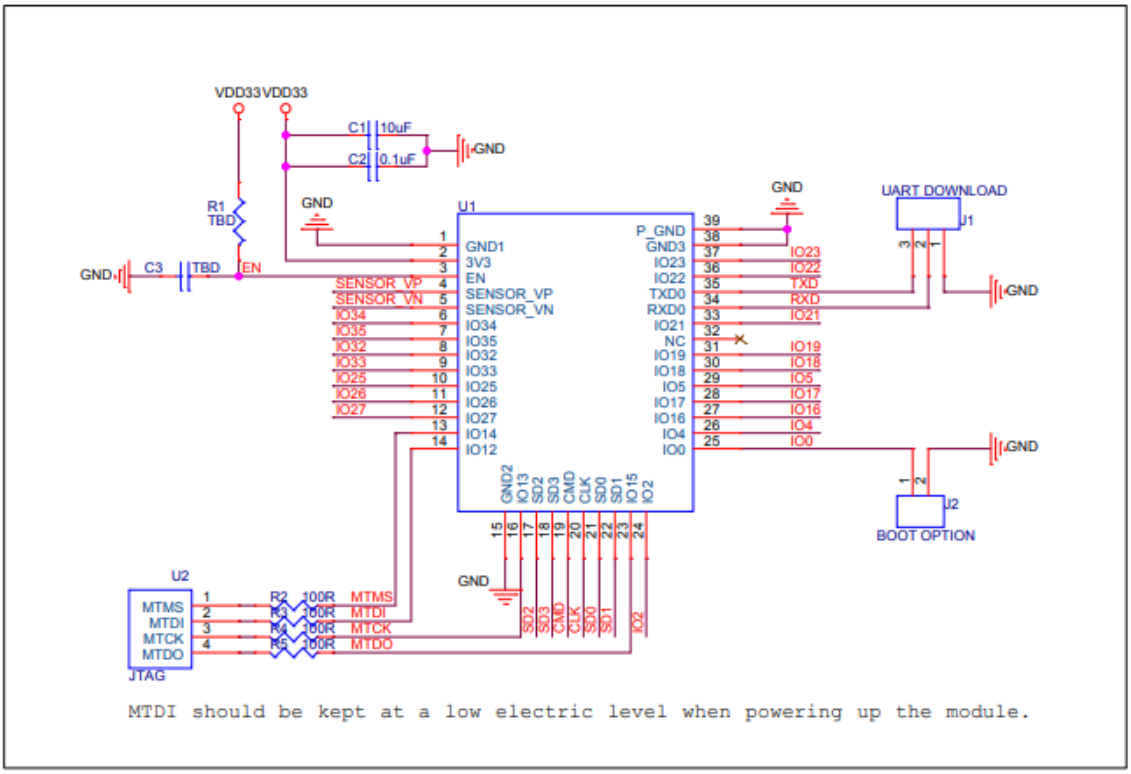


Figure 4: ESP32-WROOM-32 Peripheral Schematics

Note:

- Soldering Pad 39 to the Ground of the base board is not necessary for a satisfactory thermal performance. If users do want to solder it, they need to ensure that the correct quantity of soldering paste is applied.
- To ensure the power supply to the ESP32 chip during power-up, it is advised to add an RC delay circuit at the EN pin. The recommended setting for the RC delay circuit is usually R = 10 kΩ and C = 0.1 μF. However, specific parameters should be adjusted based on the power-up timing of the module and the power-up and reset sequence timing of the chip. For ESP32's power-up and reset sequence timing diagram, please refer to Section *Power Scheme* in *ESP32 Datasheet*.

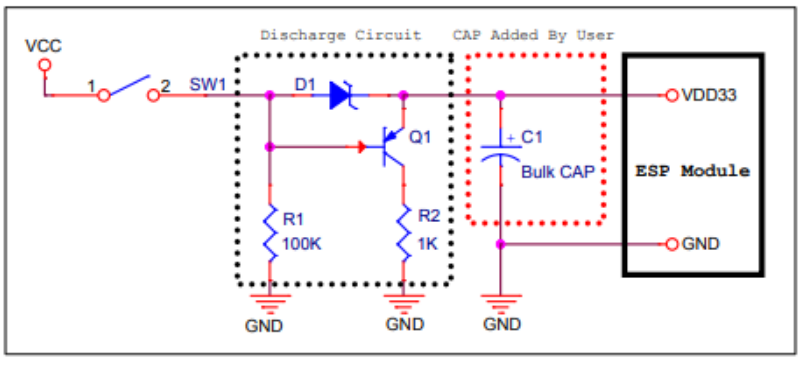


Figure 5: Discharge Circuit for VDD33 Rail

Note:
The discharge circuit can be applied in scenarios where ESP32 is powered on and off repeatedly by switching the power rails, and there is a large capacitor on the VDD33 rail. For details, please refer to Section *Power Scheme* in *ESP32 Datasheet*.

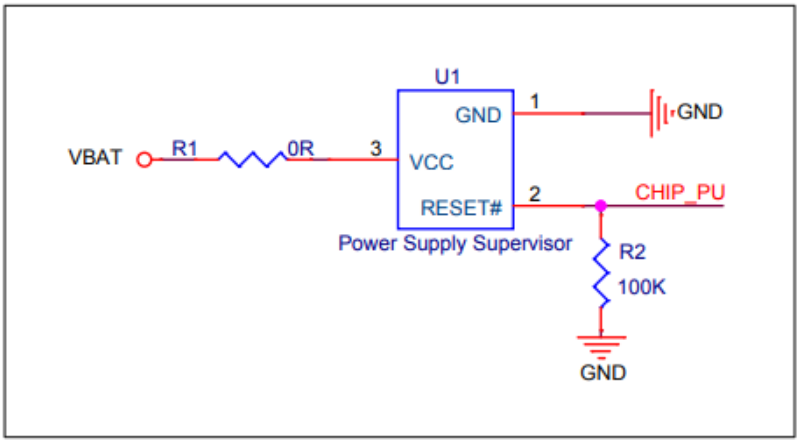


Figure 6: Reset Circuit

Note:
When battery is used as the power supply for ESP32 series of chips and modules, a supply voltage supervisor is recommended to avoid boot failure due to low voltage. Users are recommended to pull CHIP_PU low if the power supply for ESP32 is below 2.3 V.

8. Physical Dimensions

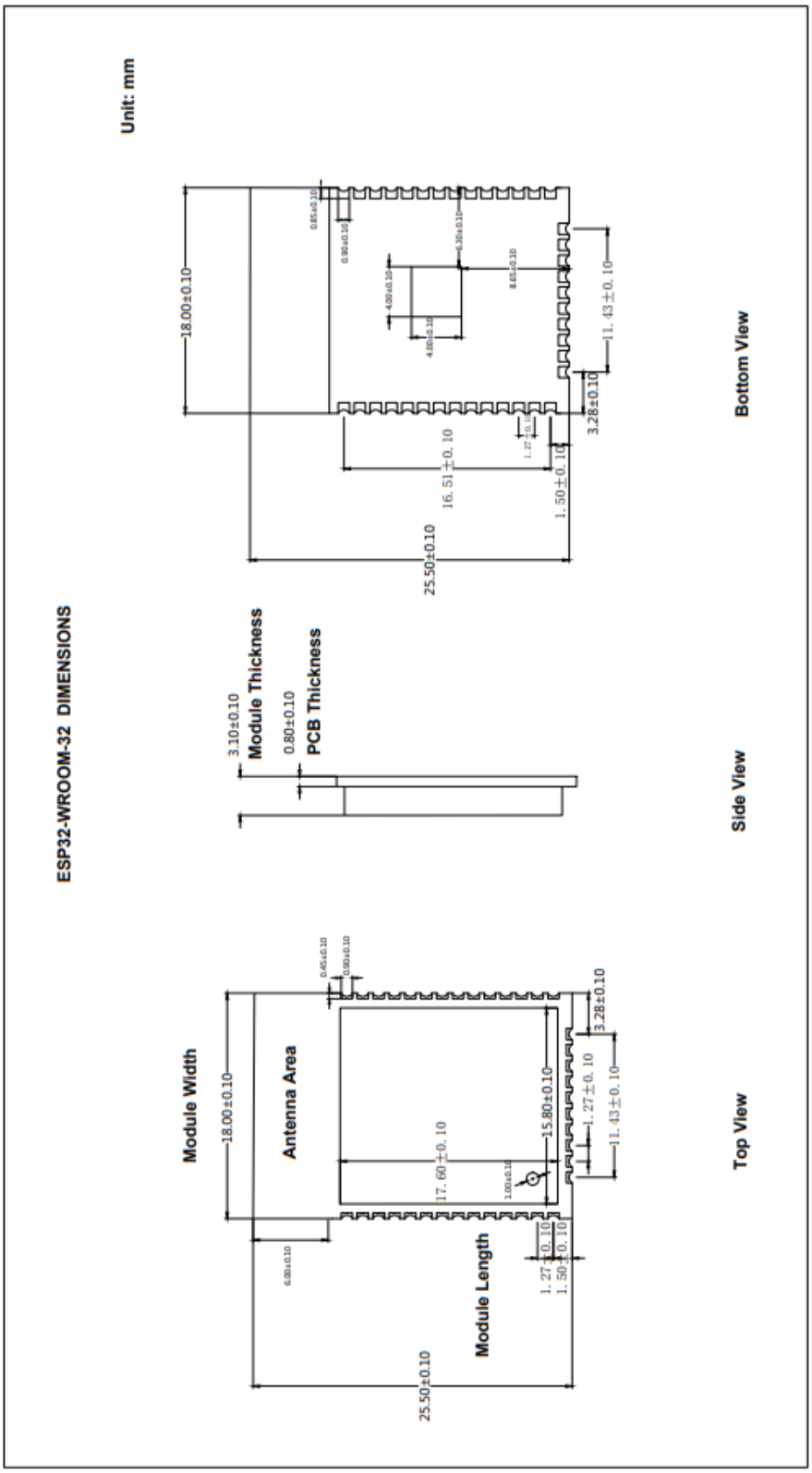


Figure 7: Physical Dimensions of ESP32-WROOM-32

9. Recommended PCB Land Pattern

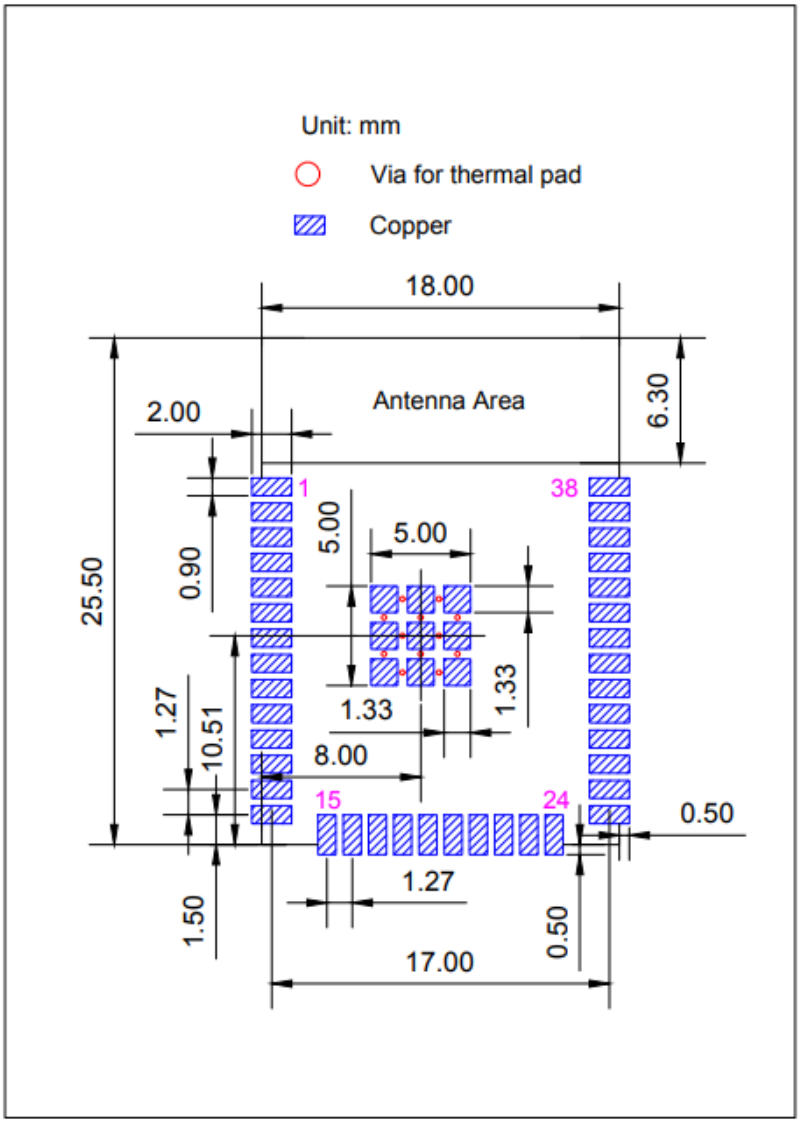


Figure 8: Recommended PCB Land Pattern

10. Learning Resources

10.1 Must-Read Documents

The following link provides documents related to ESP32.

- *ESP32 Datasheet*
  This document provides an introduction to the specifications of the ESP32 hardware, including overview, pin definitions, functional description, peripheral interface, electrical characteristics, etc.
- *ESP-IDF Programming Guide*
  It hosts extensive documentation for ESP-IDF ranging from hardware guides to API reference.
- *ESP32 Technical Reference Manual*
  The manual provides detailed information on how to use the ESP32 memory and peripherals.
- ESP32 Hardware Resources
  The zip files include the schematics, PCB layout, Gerber and BOM list of ESP32 modules and development boards.
- *ESP32 Hardware Design Guidelines*
  The guidelines outline recommended design practices when developing standalone or add-on systems based on the ESP32 series of products, including the ESP32 chip, the ESP32 modules and development boards.
- *ESP32 AT Instruction Set and Examples*
  This document introduces the ESP32 AT commands, explains how to use them, and provides examples of several common AT commands.
- *Espressif Products Ordering Information*

10.2 Must-Have Resources

Here are the ESP32-related must-have resources.

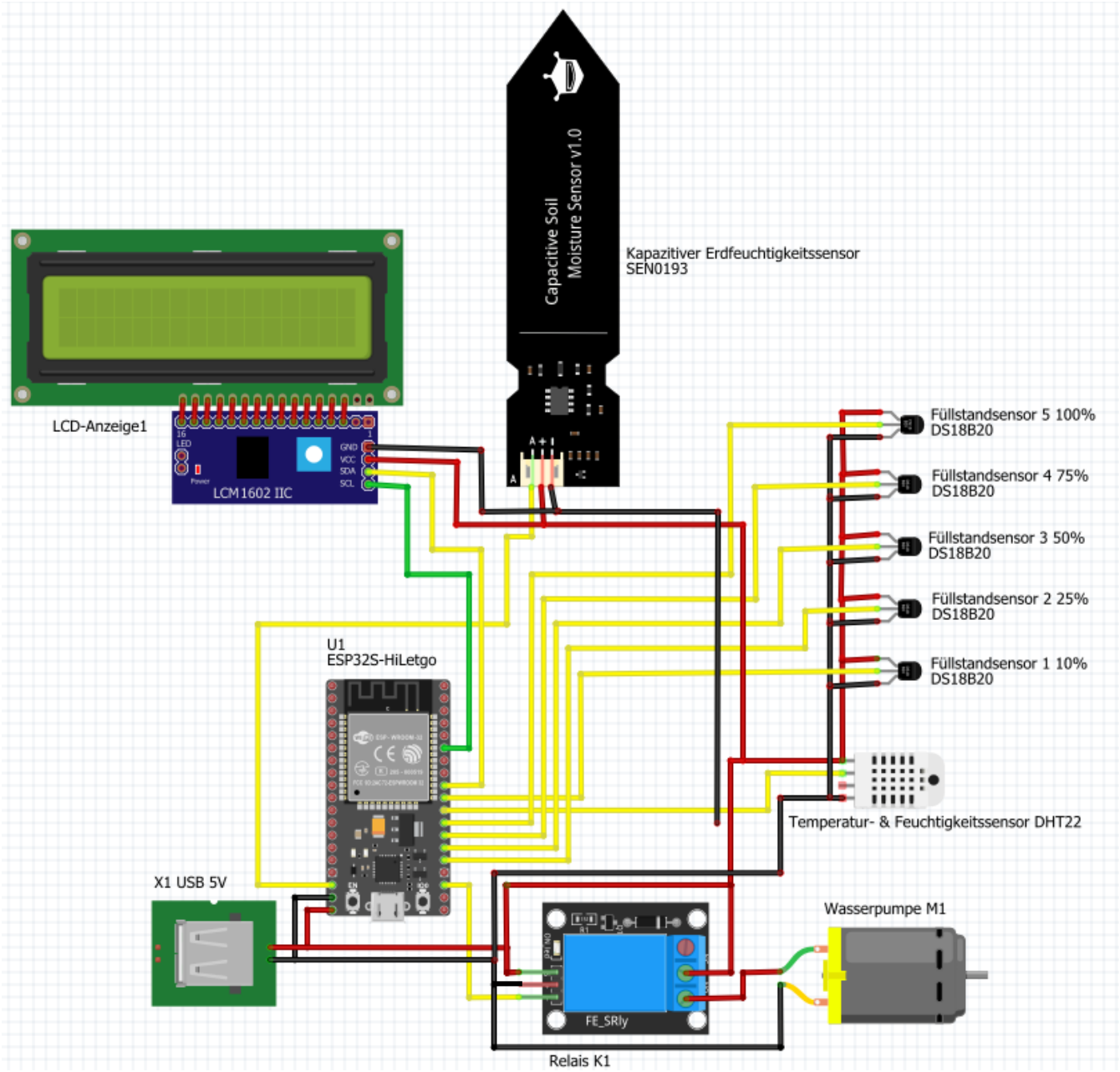
- ESP32 BBS
  This is an Engineer-to-Engineer (E2E) Community for ESP32 where you can post questions, share knowledge, explore ideas, and help solve problems with fellow engineers.
- ESP32 GitHub
  ESP32 development projects are freely distributed under Espressif's MIT license on GitHub. It is established to help developers get started with ESP32 and foster innovation and the growth of general knowledge about the hardware and software surrounding ESP32 devices.
- ESP32 Tools
  This is a webpage where users can download ESP32 Flash Download Tools and the zip file "ESP32 Certification and Test".
- ESP-IDF
  This webpage links users to the official IoT development framework for ESP32.
- ESP32 Resources
  This webpage provides the links to all available ESP32 documents, SDK and tools.

Revision History

Date	Version	Release notes
2019.09	V2.9	• Changed the supply voltage range from 2.7 V ~ 3.6 V to 3.0 V ~ 3.6 V; • Added Moisture sensitivity level (MSL) 3 in Table 1 *ESP32-WROOM-32 Specifications*; • Added notes about "Operating frequency range" and "TX power" under Table 7 *Wi-Fi Radio Characteristics*; • Updated Section 7 *Peripheral Schematics* and added a note about RC delay circuit under it; • Updated Figure 8 *Recommended PCB Land Pattern*.
2019.01	V2.8	Changed the RF power control range in Table 9 from –12 ~ +12 to –12 ~ +9 dBm.
2018.10	V2.7	Added "Cumulative IO output current" entry to Table 4: Absolute Maximum Ratings; Added more parameters to Table 6: DC Characteristics.
2018.08	V2.6	• Added reliability test items the module has passed in Table 1: ESP32-WROOM-32 Specifications, and removed software-specific information; • Updated section 3.4: RTC and Low-Power Management; • Changed the module's dimensions from (18±0.2) mm x (25.5 ±0.2) mm x (3.1±0.15) mm to (18.00±0.10) mm x (25.50±0.10) mm x (3.10±0.10) mm; • Updated Figure 8: Physical Dimensions; • Updated Table 7: Wi-Fi Radio.
2018.06	V2.5	• Changed the module name to ESP32-WROOM-32; • Deleted Temperature Sensor in Table 1: ESP32-WROOM-32 Specifications; • Updated Chapter 3: Functional Description; • Added Chapter 8: Recommended PCB Land Pattern; Changes to electrical characteristics: • Updated Table 4: Absolute Maximum Ratings; • Added Table 5: Recommended Operating Conditions; • Added Table 6: DC Characteristics; • Updated the values of "Gain control step", "Adjacent channel transmit power" in Table 9: Transmitter Characteristics - BLE.
2018.03	V2.4	Updated Table 1 in Chapter 1.
2018.01	V2.3	Deleted information on LNA pre-amplifier; Updated section 3.4 RTC and Low-Power Management; Added reset circuit in Chapter 7 and a note to it.
2017.10	V2.2	Updated the description of the chip's system reset in Section 2.3 Strapping Pins; Deleted "Association sleep pattern" in Table "Power Consumption by Power Modes" and added notes to Active sleep and Modem-sleep; Updated the note to Figure 4 Peripheral Schematics; Added discharge circuit for VDD33 rail in Chapter 7 and a note to it.
2017.09	V2.1	Updated operating voltage/power supply range updated to 2.7 ~ 3.6V; Updated Chapter 7.
2017.08	V2.0	Changed the sensitivity of NZIF receiver to -97 dBm in Table 1; Updated the dimensions of the module; Updated Table "Power Consumption by Power Modes" Power Consumption by Power Modes, and added two notes to it;

Date	Version	Release notes
		Updated Table 4, 7, 8, 9; Added Chapter 8; Added the link to certification download.
2017.06	V1.9	Added a note to Section 2.1 Pin Layout; Updated Section 3.3 Crystal Oscillators; Updated Figure 3 ESP-WROOM-32 Schematics; Added Documentation Change Notification.
2017.05	V1.8	Updated Figure 1 Top and Side View of ESP32-WROOM-32 (ESP-WROOM-32).
2017.04	V1.7	Added the module's dimensional tolerance; Changed the input impedance value of 50Ω in Table 7 Wi-Fi Radio Characteristics to output impedance value of 30+j10 Ω.
2017.04	V1.6	Added Figure 2 Reflow Profile.
2017.03	V1.5	Updated Section 2.2 Pin Description; Updated Section 3.2 External Flash and SRAM; Updated Section 4 Peripherals and Sensors Description.
2017.03	V1.4	Updated Chapter 1 Preface; Updated Chapter 2 Pin Definitions; Updated Chapter 3 Functional Description; Updated Table Recommended Operating Conditions; Updated Table 7 Wi-Fi Radio Characteristics; Updated Section 5.6 Reflow Profile; Added Chapter 10 Learning Resources.
2016.12	V1.3	Updated Section 2.1 Pin Layout.
2016.11	V1.2	Added Figure 7 Peripheral Schematics.
2016.11	V1.1	Updated Chapter 6 Schematics.
2016.08	V1.0	First release.

4.7 Schema


Capacitive Soil
Moisture Sensor v1.0
Kapazitiver Erdfeuchtigkeitssensor
SEN0193
LCD-Anzeige1
LCM1602 IIC
GND
VCC
SDA
SCL
Füllstandsensor 5 100%
DS18B20
Füllstandsensor 4 75%
DS18B20
Füllstandsensor 3 50%
DS18B20
Füllstandsensor 2 25%
DS18B20
Füllstandsensor 1 10%
DS18B20
U1
ESP32S-HiLetgo
Temperatur- & Feuchtigkeitssensor DHT22
X1 USB 5V
Wasserpumpe M1
FE_SRly
Relais K1