Arbeitsjournal

Tag 1

An Tag 1 habe ich mir zuerst mit den unbekannten Bauteilen beschäftigt. In meinem Fall waren das das Elektretmikrofon MAX4466 von Adafruit(Mic-Doc), das Wemos Buzzer Shield (Buzzer-Doc) und das Wemos Battery Shield (Battery-Doc) sowie ein Vibrationsmotor. Die entsprechenden Datasheets sind jeweils dahinter angegeben, wobei der Vibrationsmotor keine Bezeichnung hat und sich die Recherche so etwas schwieriger gestaltet hat.

ElektretmikrofonMAX4466

Das MAX4466 ist ein Mikrofon, dass Frequenzbereiche zwischen 20Hz und 20kHz wahrnehmen kann. Dazu ist ein Amplifier fest verlötet. Über ein Trimmpoti kann der Gain zwischen x25 und x125 flexibel eingestellt werden. Dazu muss einfach nur die Schraube auf der Rückseite entsprechend gedreht

werden. Die Schraube ist im Bild rot markiert. Leider hat das Poti keinen Indikator und kann auch nicht ausgelesen werden. Daher muss die Einstellungsrichtung durch ausprobieren herausgefunden werden. Zusätzlich ist der Widerstand an den Anschlägen relativ empfindlich. Man muss also aufpassen, dass Poti beim Einstellen nicht zu beschädigen.
Die Benutzung an sich ist ziemlich einfach. Der Vcc-Pin wird mit einer Stromquelle zwischen 2,5V und 5,5VDC und der GND-Pin mit Masse verbunden werden. Über den OUT-Pin können die Audio-Signale ausgegeben werden, wobei hier Gleichstrom anliegt. Die Verbindung bei einem Mikrocontroller muss also entweder über einen Analog-Pin stattfinden oder mit einem Kondensator auf AC umgewandelt werden, sodass ein Digital-Pin benutzt werden kann. Die gelesenen Signal-Werte können zwischen 0 und 1024 liegen. Die Zahlen korrespondieren dabei mit der Frequenzhöhe der Töne. Da ich mich aber nur für die Lautstärke interessiere, muss hier später noch eine Umrechnung stattfinden. Da am Out-Pin eine Vorspannung von Vcc/2 anliegt, entspricht das dem Wert bei absoluter Stille. Es ist also kein „0“-Wert.

Wemos Buzzer Shield & Wemos Battery Shield

Da beides Shields sind, ist die Nutzung ziemlich einfach. Sie können ganz einfach aufgesteckt werden. Das Buzzer Shield als Art Lautsprecher kann dann über den Pin D5 angesprochen werden. Das Battery Shield dient nur zum Anschließen der Batterie, die über den Mini-USB-Port geladen werden kann.

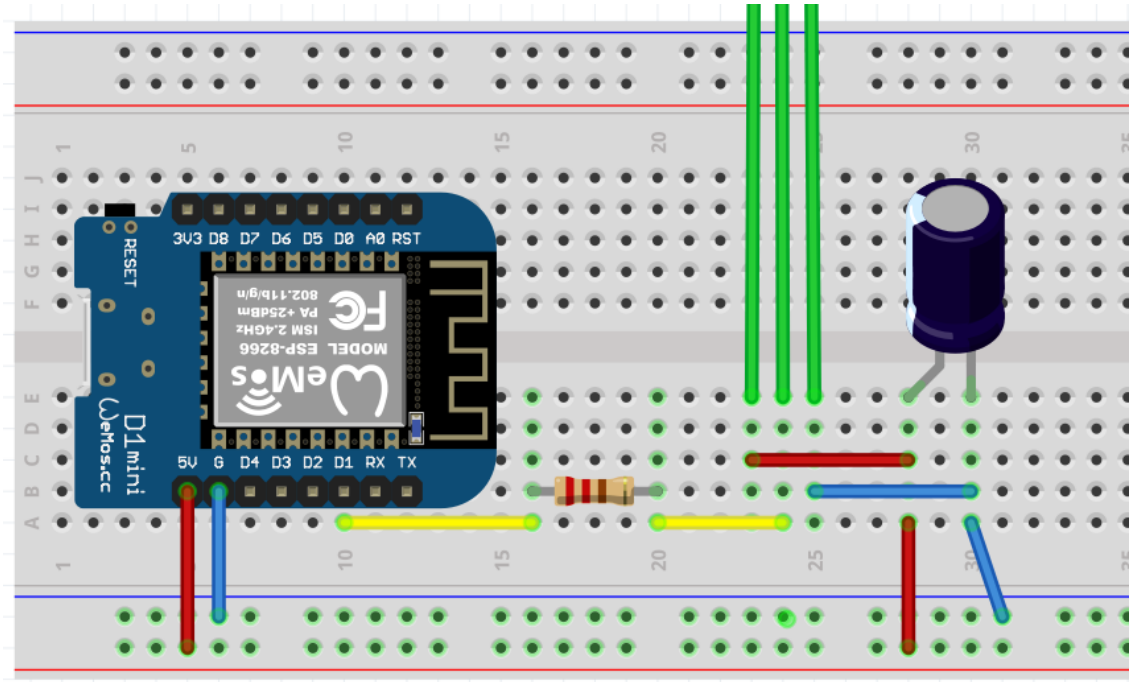
Vibrationsmotor

jidvsiudsviudsiun

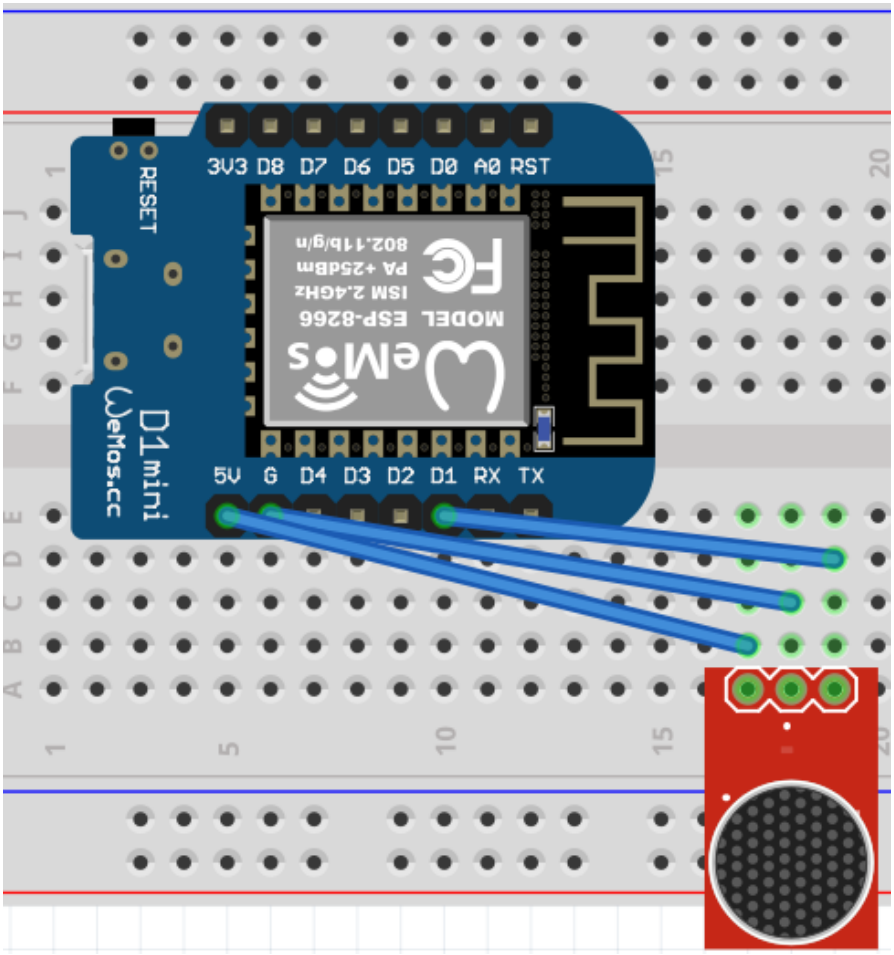
Erste Schaltungen

Nachdem die Verwendung an sich klar war, habe ich die ersten Schaltungen geplant und gebaut. Das Projekt basiert auf Server-Client-Kommunikation. Die „Server“-Schaltung ist ziemlich einfach und hat nur das Mikrofone extern angeschlossen.
Die „Client“-Schaltung kann dann verschiedene externe Module haben. Für den Anfang habe ich hier den bereits bekannten LED-Strip angeschlossen. Den Code für die drahtlose Kommunikation war ebenfalls bereits bekannt. Bei der Client-Schaltung stellen die grünen Kabel den LED-Strip da. Dieser

ist leider nicht in Fritzing verfügbar:



Server-Schaltung:



Tag 2 & 3

An Tag 2 gab es die ersten Herausforderungen, nachdem die Basics funktioniert haben. Als nächstes Auf der Agenda stand der Code.

Herausforderung 1

Die per Wifi gesendeten Werte waren doch sehr komisch. Problem: Ich habe im ersten Code-Entwurf die Werte als Integer gespeichert und dann auch so gesendet. Die Methode client.write() sendet aber einen Byte-Wert. Die gesendeten Werte sind also die Überträge des Byte-Casts.
Mir sind zwei Lösungen eingefallen: Entweder mehrere Bytes senden und entsprechend verarbeiten oder die gelesenen Werte zwischen 0 und 1024 auf die Byte-Werte zwischen 0 und 255 mappen. Da das Mappen deutlich weniger Code produziert als mehrere Bytes zu senden und die Verarbeitung deutlich einfacher macht, habe ich mich für diese Lösung entschieden. Die Werte werden wie folgt gemappt: (Wert*255)/1024.

Herausforderung 2

Das Mikro erkennt nicht alle Geräusche. Bei ein paar weiteren Tests ist mir aufgefallen, dass die wahrgenommene Lautstärke nicht stimmen können.
Lösung: Nach einigem Probieren und ein wenig Recherche ist mir aufgefallen, dass mein Loop-Delay von bei 100ms gelegen hat. Das hat dafür gesorgt, dass lange nicht alle Frequenzen wahrgenommenen werden konnten. Einen optimalen Output habe ich zwischen 5-10ms gefunden.

Herausforderung 2.5

Obwohl die Werte jetzt ungefähr gestimmt haben, hatte ich trotzdem starke Abweichungen bei gleichem Geräusch an gleicher Stelle.
Lösung: Nach einiger Zeit ist mir das Trimmerpoti wieder eingefallen, dass ich bisher noch nicht benutzt habe. Vorsichtig habe ich versucht das Poti einzustellen und immer wieder getestet. Nach einiger Zeit habe ich eine recht gute Einstellung gefunden, die etwa mittig sein sollte. Ich schätze also den Gain auf ca. x75 ein. Jetzt waren die Schwankungen minimal im einstelligen Bereich und ansonsten ebenso konstant.

Herausforderung 3

Transformation von Frequenz auf Lautstärke. Die gelesenen Werte sind immer noch die Frequenzwerte. Bei der Transformation von habe ich am Beispiel von Adafruit orientiert. In diesem Beispiel wird auch erklärt, welches delay genutzt werden sollte, um Optimale Werten zu bekommen. Das hätte mir das vorherige Problem um Einiges vereinfacht. Im Beispiel werden die Amplitudenhöhen gemessen und gegeneinander gerechnet. Anschließend wird es dann in die anliegende Spannung in Volt umgerechnet. Ich habe hier meine bereits vorher erstellte Formel angewendet, sodass es auch direkt per client.write() verarbeitet werden kann.

Herausforderung 4

Nachdem ich den Beispielcode auf meine Anwendung angepasst habe und mir die ersten Werte korrekt ausgegeben wurden, hat der Wemos D1 Mini plötzlich und unvermittelt einen „Soft WDT reset“ gemacht. Die Fehlersuche hat hier doch etwas länger gedauert, da mir der Fehlercode noch nicht begegnet ist
Lösung: WDT steht für Watchdog. Watchdog ist ein Timer, der Ausfälle des Systems erkennen soll und im Falle eines Time-Outs den Controller automatisch zurücksetzt. Im Beispiel wird eine blockierende While-Schleife benutzt, die für eine bestimmte Zeit Daten sammelt. Da ich den Zeitraum auf 500ms erhöht habe, hatte hier der WDT ein timeout. Um den Timer zu resetten, muss

regelmäßig entweder yield() oder delay() aufgerufen werden. Da delay() auch andere Operationen blockiert, habe ich hier yield() gewählt.

Herausforderung 4.5

Bei längerem Ausführen des Codes, wurden irgendwann keine Daten mehr gesendet. Der Code hat „blockiert“, es hat aber auch kein WDT-Reset stattgefunden.
Lösung: Ich bin mir nicht so ganz sicher, warum mein Code mit yield() nicht funktioniert hat. Da ich aber als zweite Alternative noch delay() zur Verfügung hatte, habe ich es mit delay() ausprobiert und es hat problemlos funktioniert. Es wird wahrscheinlich nichts mit dem Aufruf zu tun haben, sondern an anderer Stelle im Code Probleme gegeben haben. Aber da es funktioniert hat, habe ich mich nicht weiter damit beschäftigt.

Tag 4

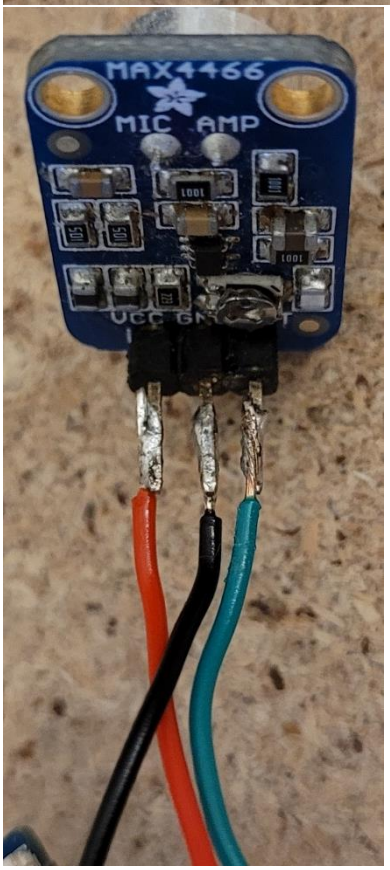
Nachdem ich funktionierende Hardware und funktionierenden Code hatte, stand als nächstes Field-Testing an.

Klingelautstärke

Zuerst musste ich herausfinden, welche Lautstärke-Werte meine Klingel produziert. Meine Idee war, das Mikro möglichst nah an die Klingel zu packen und einen möglichst hohen Schwellwert zur Aktivierung zu wählen. Als Energieversorgung sollte eine Powerbank mit 9000mAh dienen. Die ersten Tests waren sehr erfolgreich und haben zuverlässig getriggert. Als dann zufällig mein Nachbar seine Wohnung verlassen und seine Tür doch mit viel Schwung geschlossen hat, hatte ich einen falschen Alarm. Meine Klingel ist direkt neben der Haustür und die Vibrationen aus dem Treppenhaus sorgen für starke Resonanz im Mikrofon. Da bei mindestens zwei Nachbarn ein falscher Alarm ausgelöst wird und beide Wohnungen WG’s sind, kommt ein Platz in Türnähe in Frage. Nach sehr, sehr viel Testen habe ich eine relativ gute Stelle im offenen Schuhteil der Garderobe gefunden. Vibrationen aus dem Hausflur waren kein Problem mehr. Allerdings hat die Position in Bodenähe dafür gesorgt, dass „nahes Vorbeigehen“ auf meinem Altbau-Hartholzboden zu laut war. Allerdings waren hier sowohl Vibration als auch Lautstärke ein Problem. Lösung: Ich habe den Cube auf einem Gummi-Flip-Flop als Dämpfung gegen Vibration platziert und das Mikro an sich zur Tür ausgerichtet. Danach konnte ich keinerlei Probleme festellen, außer wenn ich Sachen von der Garderobe nehme. In diesem Fall erwarte ich aber kein Klingeln mehr. Das Ganze hat das Setup auch sehr vereinfacht vereinfacht, da ich keine Powerbank mehr brauchte, sondern direkt die Steckdose neben der Garderobe nehmen konnte. Hier die Bilder dazu:

Tag 4

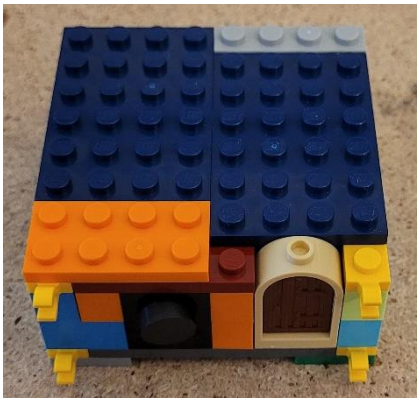
Nach dem sich die Stromversorgung nun vereinfacht hatte und alles Weitere funktioniert hat, habe ich die Schaltung zusammengelötet und in ein passendes Gehäuse (siehe oben) getan. Da dieser Cube stationär ist, war Größe hier kein Problem.



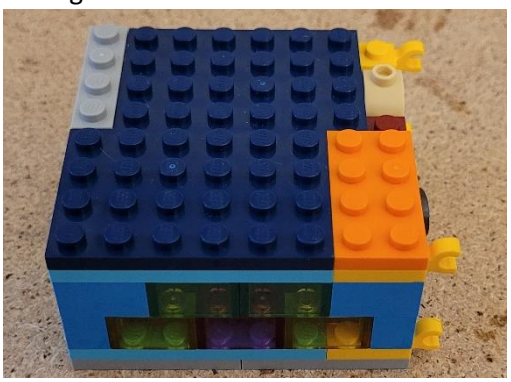
Nachdem der Server soweit fertiggestellt habe, war der nächste Schritt der mobile Cube. Nach einigem Probieren mit den verschiedenen Benachrichtigungs-Modulen, musste ich leider feststellen, dass ich aus größentechnischen Gründen nicht alle benutzen werden kann, sondern mich eher auf die Größe fokussiere. Daher bin ich letztendlich nur beim visuellen Indikator in Form des LED-Streifens geblieben. Die nächste Herausforderung war dann, ein passendes Gehäuse zu finden. Angedacht war hier eine kleine PVC-Box mit den Maßen 7cm x 4,5cm x 2,5cm. Diese ist allerdings komplett schwarz und wird am Ende festverschaubt. Da ich sowohl irgendeine Art von Leuchten oder Blinken brauche als auch eine Möglichkeit, die Batterie zu laden, war das keine Option mehr. Nach einigem Überlegen, insbesondere in Anbetracht der wenigen, verbleibenden Zeit, bin ich auf die Idee gekommen, mir ein selber etwas zu bauen. Da ich leider keinerlei Werkzeug oder Material zur Verwendung von Holz und keinerlei Gerät zur Verwendung von Plastik habe, bin ich auf die einzig

verbliebene Option umgestiegen: Lego-Steine! Ich hatte glücklicherweise noch einige Lego-Steine und ein paar Bodenplatten parat, mit denen ich mich dann Stück für Stück an ein optimales Gehäuse herangetastet habe. Das Endergebnis ist 6cm x 6cm x 3cm groß und sieht so aus:

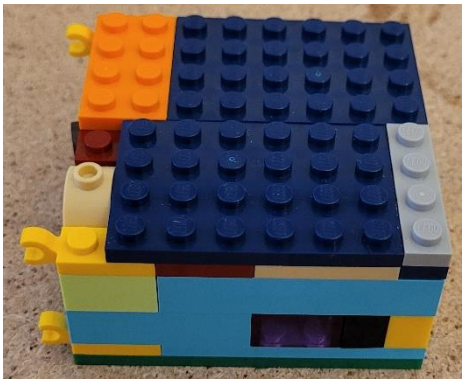
Schräg von vorne



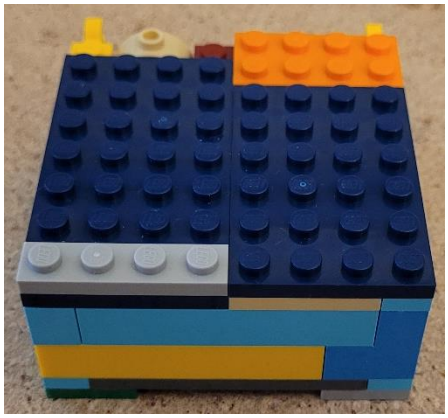
Schräg von der Seite. Die Seite ist aus transparenten Steinen, dahinter sitzt der LED-Streifen:



Schräg von der anderen Seite. Auch hier ist ein transparenter Stein verbaut, damit man die Status-LED des Battery Shields sehen kann:



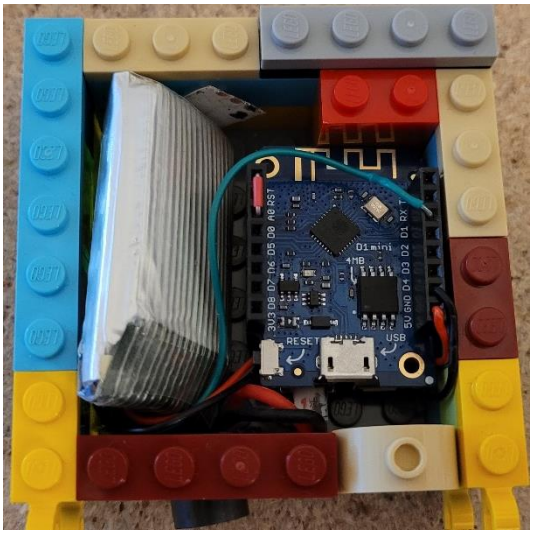
Ansicht schräg von hinten:

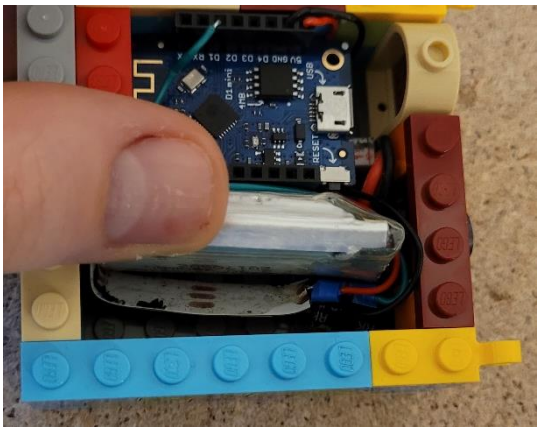


Frontalansicht. Hier ist ein Magnet verbaut, um es ganz einfach an meinem Mikrofon-Arm befestigen zu können und es aber ggf. auch einfach mitnehmen zu können. Die Tür ist der Zugang zum Ladeport.

Ansicht von oben, ohne Technik:

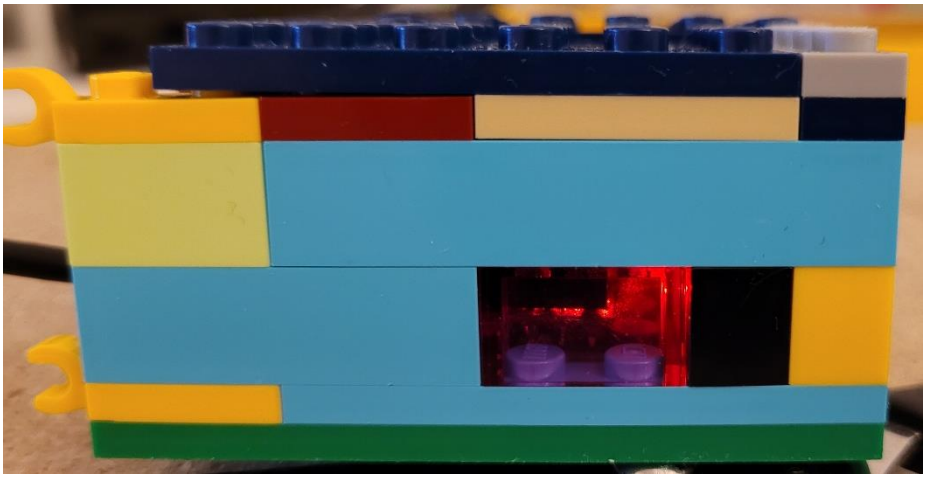
Ansicht von oben mit Technik und den“versteckten“ LED‘s





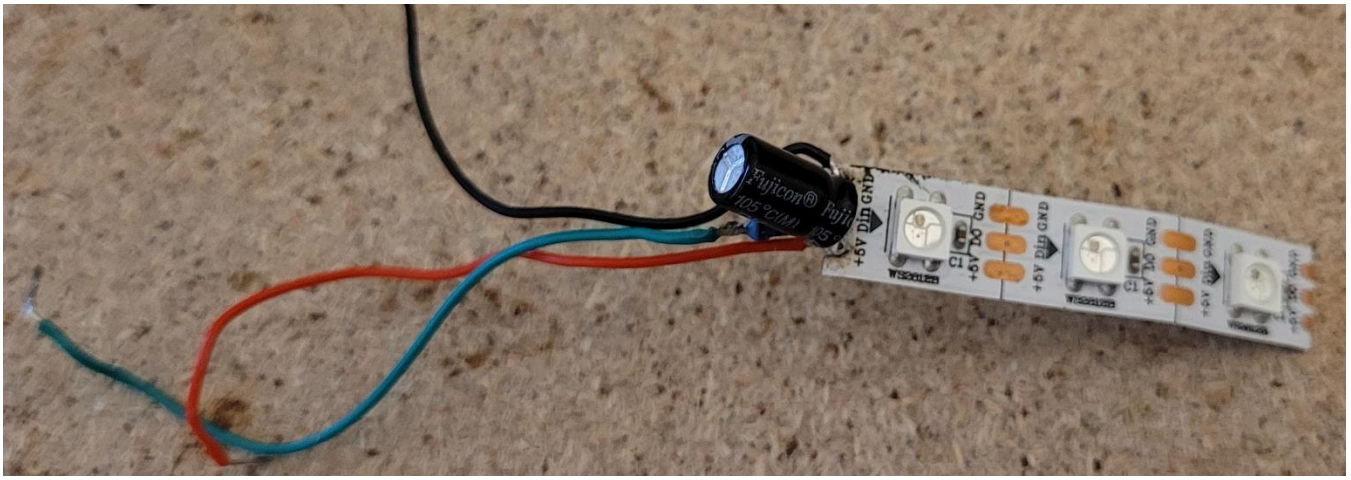
Frontansicht mit erkennbarem Ladeport:

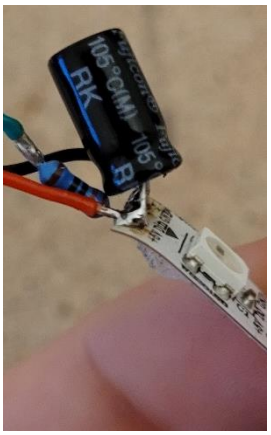
Seitenansicht mit leuchtender Status-LED. Status hier ladend:

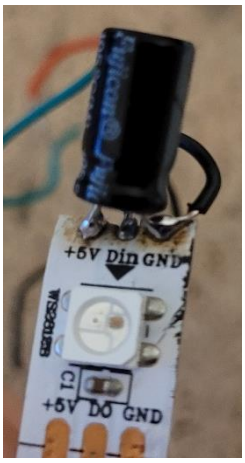


Tag 5

Nachdem der Code funktioniert hat und die Box vorbereitet war, stand als letzter großer Schritt das Verlöten der LED-Strip-Schaltung an, sodass am Ende alles in das Gehäuse passt. Vor dem Verlöten war ich doch sehr aufgeregt, da ich für den LED-Strip mit Widerstand und Kondensator nicht viel Platz hatte. Beim Testen vorher hat alles gepasst, da waren die Teile aber noch unabhängig voneinander. Ziel war es also, alles möglichst klein und kurz anzulöten. Nach etwas Übung und sehr viel Geduld bin ich dem Ergebnis aber doch einigermaßen zufrieden:



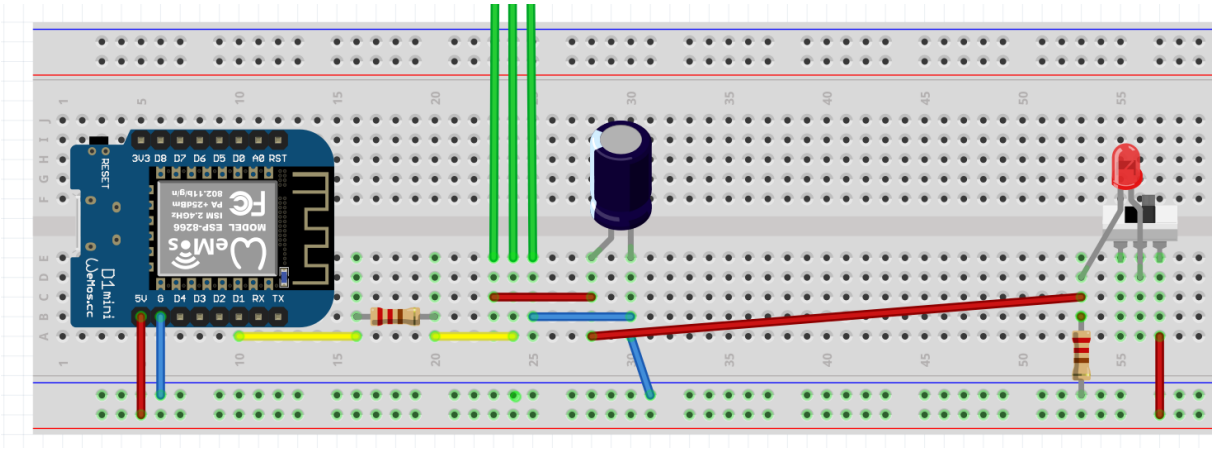




Hier die Lötstellen noch einmal im Detail. Anschließend hat das Fitting im Gehäuse angefangen. Es hat einige Zeit gebraucht, bis ich alles ordentlich verstaut habe und kein Druck auf irgendwelchen Teilen lastete. Die gefittete Schaltung kann in den Bildern oben begutachtet werden.

Tag 6

Nachdem alles soweit geklappt hat und ich die fertige Schaltung das erste Mal in Action erlebt habe, ist mir eingefallen, dass ich ursprünglich noch einen An-/Aus-Schalter mit einbauen wollte, damit das tragbare Modul einfach ausgeschaltet werden kann. Nach ein bisschen Planung und einiger Recherche ist dabei folgende Schaltung entstanden:



Problematisch dabei ist, dass die „Umleitung der Stromversorgung“ über den Button hier die Batterie sein müsste. Diese Umsetzung gestaltete sich dann als doch eher schwierig. Daher hat mir Andi zwei Tutorials geschickt, die sich mit dem Energieverbrauch des Wemos D1 Mini beschäftigen. Sie können HIER und HIER gefunden werden. Anhand dieser Beispiele habe ich mich dann für die Variante Deep-Sleep enschieden, die die mit Abstand längste Batterielaufzeit ermöglicht. Der einfache Teil war die Pins D0 und RST direkt miteinander zu verbinden. Wenn der Wemos wieder aufwacht, passiert das über den Pin D0 und damit alle notwendigen Operationen auch ausgeführt werden, wird hier direkt auf den RST-Pin geshortet.
Die deutlich größere Herausforderung war es, den Code noch einmal komplett anzupassen. Insbesondere musste hier darauf geachtet werden, dass auch alle gemessenen Werte verarbeitet werden. Dazu sammelt der Server solange Werte, bis sich ein Client verbindet. Erst dann wird der berechnete Lautstärkewert weitergegeben. Der Client holt sich dann exakt diesen Wert bzw. diese Werte, vergleicht sie wie gehabt mit dem Schwellwert und geht danach wieder in den Deep-Sleep-Mode. Ich habe mich hier für 5 Sekunden Deep-Sleep entschieden. Der „optimale“ Wert würde hier von der Öffnungszeit der Tür abhängen und „wie lange man brauchen würde“. Mit Neustart, Latenz und erneuter Netzwerkverbindung ist die Zeit zwischen zwei Vergleichen bereits bei ca. 10-15 Sekunden und dazu kommt noch die Zeit die ich brauche, um zur Tür zu gehen.
Nachdem der Server seine Werte gesendet hat, stoppt er aktiv die Verbindung mit dem Client und verwirft alle potentiell noch verbleibenden Werte. Hier können ein paar Signale verloren gehen. Da das Klingelzeichen aber deutlich längere Dauer hat, sollte das nicht zu Anomalien führen. Das aktive Stoppen des Clients ist notwendig, damit beim erneuten Verbinden des aufgewachten Wemos auch die Zuordnung eindeutig ist. Man hätte hier auch eine Art von Identifizierung als Problemlösung wählen. Das hätte aber deutlich mehr Code zur Folge. Und da die potentiell verworfenen Werte keinen Einfluss auf das Ergebnis haben, habe ich hier die erste Option gewählt.

Im beiliegenden Ordner können erste Code-Entwürfe und der fertige Code gefunden werden! Außerdem ist ein Video der fertigen Cubes enthalten. Da ich alleine Lebe und Kontakte derzeit schwierig sind, konnte ich leider kein umfassendes „Klingel“-Video machen. Eine vereinfachte Variante liegt aber bei!

Abschlusstest

Nachdem ich dann endgültig alles fertig hatte, habe ich den Cube dann noch direkt ein paar Stunden im Einsatz getestet. So sieht er dann an seinem vorgesehenen Platz aus! Durch den integrierten Magneten kann ich den Cube ganz einfach aufhängen und auch wieder abnehmen, falls ich mal nicht am PC sitze. Durch die Kabelführung am Mikrofonarm habe ich ein Mini-USB Kabel, damit ich die Batterie jederzeit laden kann. Die Position eignet sich ideal, damit der Cube nicht direkt auffällt, aber zentral genug ist, um beim Leuchten sofort ins Auge zu springen. Und sollte ich die Position des Arms ändern, z.B. bei Videokonferenzen, muss ich den Cube nur etwas rotieren. Da der Testzeitraum sehr kurz war, wurde nur einmal „richtig“ geklingelt: Also unerwartet und als ich mein Headset aufhatte. Dort hat alles funktioniert. Bei einem zweiten Mal hatte ich mein Headset nicht auf, es hat aber trotzdem funktioniert. Und da habe ich den Postboten gebeten nochmal zu klingeln, wenn vor der Wohnungstür steht, sodass ich folgendes Bild machen konnte: