hpWeMosD1Mini_PlanzenClient_V3

1. Allgemein

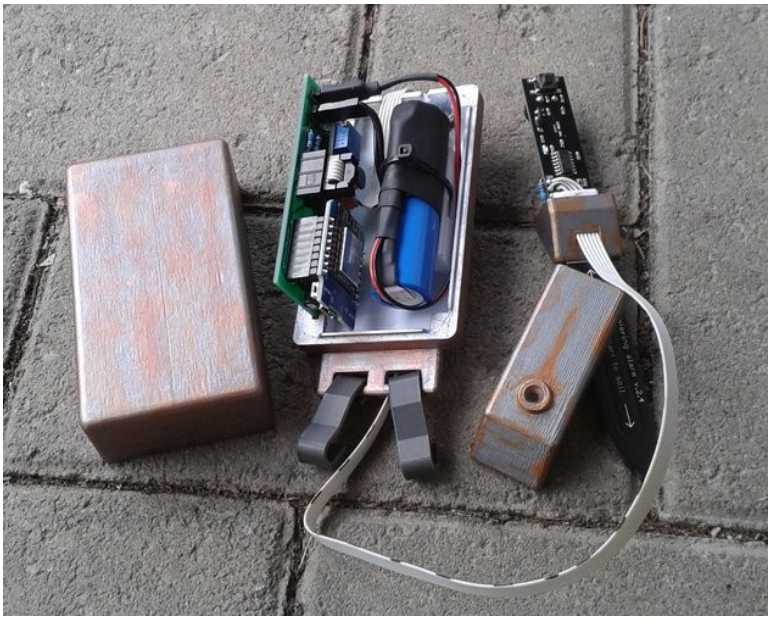
Auf meiner Terrasse züchte ich Tomaten, Paprika und anderes Grünzeug in Töpfen. Durch die letzten heißen Sommer war ich sehr verunsichert über die Menge der Wassergabe, war ständig hin und her gerissen zwischen zu viel und zu wenig Wasser.

Planzen-Client mit Chirp-Sensor

So habe ich mir 2019 Gedanken über ein eigenes Bewässerungssystem für meine Zwecke zu machen. Nach vielen Experimenten habe ich ein passendes System entwickelt, und konnte schon im vergangenen Jahr dieses System 3 Monate lang an meinen Bohnen testen.

Dieses Jahr, 2020, ist geplant alle Pflanzen komplett automatisch zu bewässern.

An dieser Stelle möchte ich den von mir entwickelten Pflanzen-Clienten vorstellen, und Anregungen für eigene Entwicklungen geben, oder zum Nachbau anregen. Der zugehörige Pflanzen-Server ist unter plant-watering-server-wemos-lolin32 zu finden.

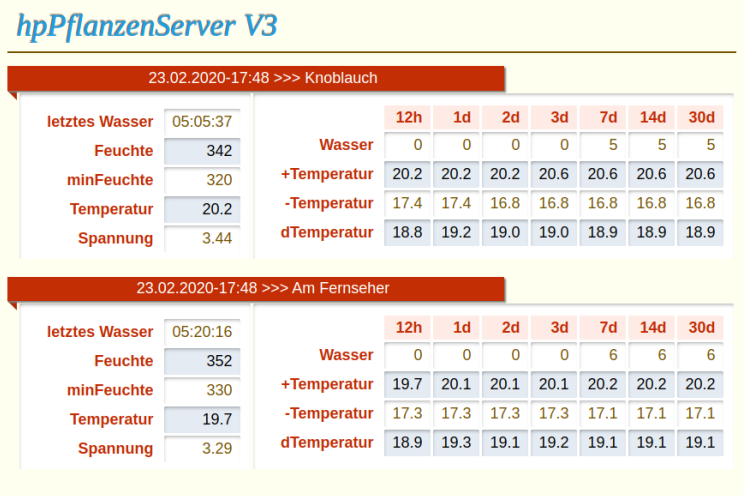


Das Innenleben

Auf das Frühjahr warten ;-)

2. Funktionsweise

Jeder Pflanzentopf bekommt seinen eigenen Pflanzen-Clienten mit eigener Wasserpumpe, und auf diese Pflanze eingestellte Bodenfeuchtigkeit und Wassermenge. Die Pflanzen-Clienten tauschen in regelmäßigen Abständen Informationen mit dem Pflanzen-Server über WiFi aus. Dieser erzeugt im lokalen Netz eine WebSite, auf der der Status der einzelnen Pflanzen-Clienten angezeigt wird, und die einzelnen Pflanzen-Clienten eingestellt werden können.



Lokale WebSite des Pflanzen-Servers

Generell funktioniert ein Pflanzen-Client jedoch unabhängig vom Pflanzen-Server, alle Einstellungen werden im Pflanzen-Clienten lokal gespeichert. Sollte eine Verbindung zum Server nicht möglich sein, setzt der Client ungestört, mit den gespeicherten Einstellungen, sein Programm fort. Entsprechend der eingestellten Zeit wird der Client erneut den Kontakt mit dem Pflanzen-Server suchen. Durch dieses Verhalten wird eine hohe Systemsicherheit erreicht, ein Fehler in der Datenübertragung oder im Server führt also nicht zum Stillstand des gesamten Bewässerungssystems.

Seine Energie bezieht der Planzen-Client aus dem eingebauten Akku. Damit der Akku möglichst lange Zeit ohne Nachladung funktioniert, verbringt der Pflanzen-Client die meiste Zeit „schlafend“. Aus diesem Tiefschlaf wird er regelmäßig aufgeweckt (zB. alle 30 Minuten), um den angeschlossenen Feuchtesenor zu lesen, gegebenenfalls die Wasserpumpe entsprechende Zeit zu aktivieren, Temperatur und Akkuspannung zu messen, und die Werte zum Planzen-Server zu senden. Bei dieser Gelegenheit übernimmt er dann auch die möglicherweise geänderten Einstellungen vom Server.

Der Planzen-Client sendet an den Server:

- unverwechselbare Clienten-Nummer
- eingestellter Clienten-Name
- gemessene Feuchtigkeit der Pflanzenerde
- Lufttemperatur am Clienten
- gegebenenfalls Laufzeit der Wasserpumpe
- Akkuspannung des Clienten.

Mit einem Klick auf den Titel eines Pflanzen-Clienten, wird die Eingabe der Client-Einstellungen ermöglicht. Mit Klick auf „OK“ werden die Einstellungen übernommen, und bei der nächsten Gelegenheit an den Pflanzen-Clienten gesendet, und dort gespeichert.

Lokale WebSite bietet Client-Einstellungen

Folgende Client-Einstellungen sind möglich:

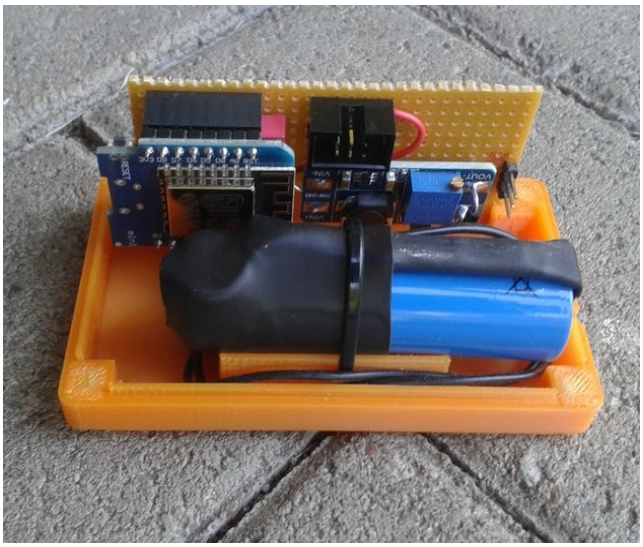
- Client-Name (max. 30 Zeichen).
- Schlafzeit des Clienten (max. 60 Minuten).
- Pumpzeit der Wasserpumpe wenn die Mindestfeuchte unterschritten wird (max. 60 Sekunden).
- Wasserstopp nach einer Wassergabe, obwohl die Mindestfeuchte unterschritten wird (max. 24*60 Minuten). Diese Funktion ist erforderlich um dem Wasser die notwendige Zeit zu geben um in den Boden einzusickern.
- Unterschreitet der Messwert des Feuchtesensors den eingestellten Wert der Mindestfeuchte, wird sie Wasserpunpe die eingestellte Zeit eingeschaltet. Gute Werte für die Mindestfeuchte bewegen sich zwischen 250 und 350.

Jeder Pflanzen-Client identifiziert sich über eine 12 bis 16 stellige Nummer, die von dem verbauten Temperatursensor geliefert wird.

Die Messung der Erdfeuchte erfolgt mit angeschlossenen Sensor „Chirp“ . Der Chirp ist eine Entwicklung von Albertas Mickėnas (Miceuz). Eine ausführliche Beschreibung ist auf https://github.com/Miceuz/PlantWateringAlarm und https://wemakethings.net/chirp/ zu finden. Für die Anwendung hier, ist der Chirp über Kabel mit dem Pflanzen-Clienten verbunden, und mit einem speziellen Programm programmiert.

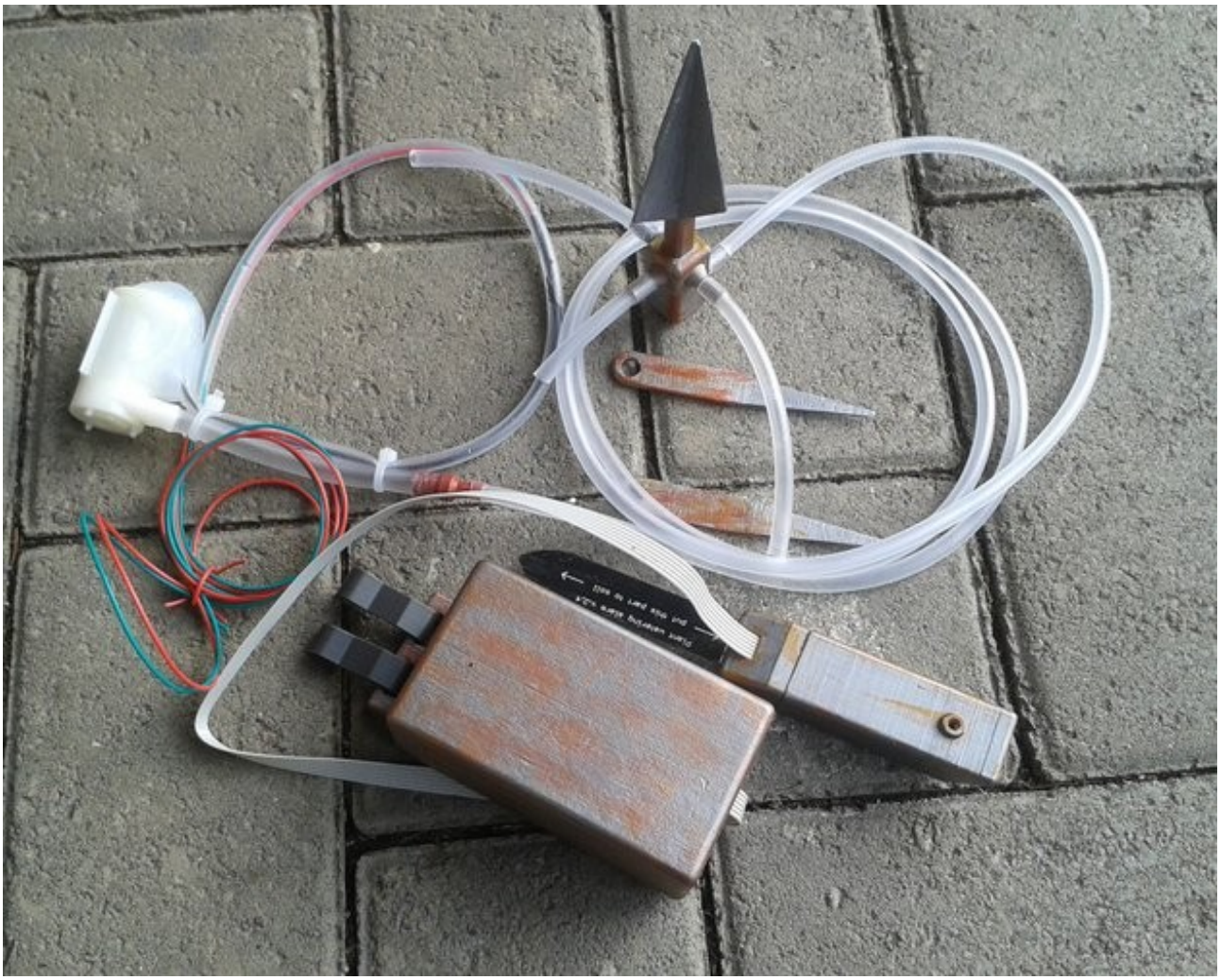
3. Entwicklung

Im Juni 2019 habe ich die Entwicklung des Bewässerungssystems gestartet, und anschließend während einer Testphase an meinen Stangenbohnen getestet und fertig entwickelt.

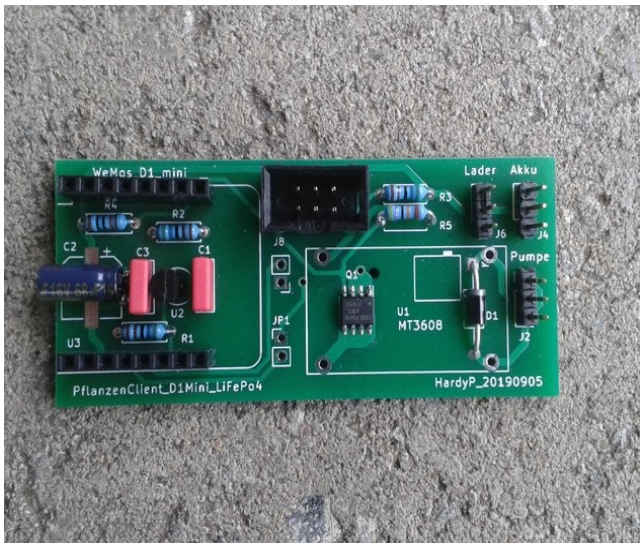


Erster Entwurf von 2019

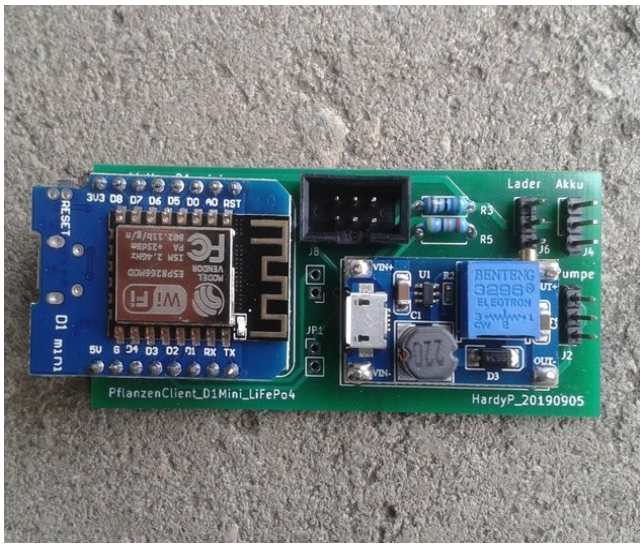
Testbetrieb 2019



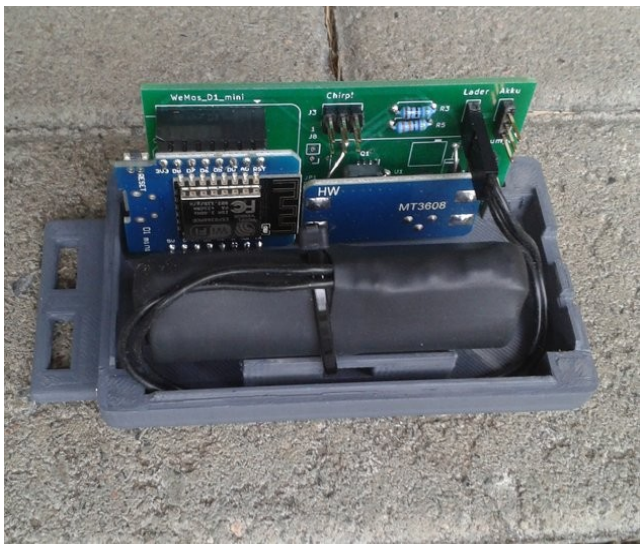
Kompletter Client mit Fühler, Pumpe und Schlauch



Bestückte Platine des Clienten

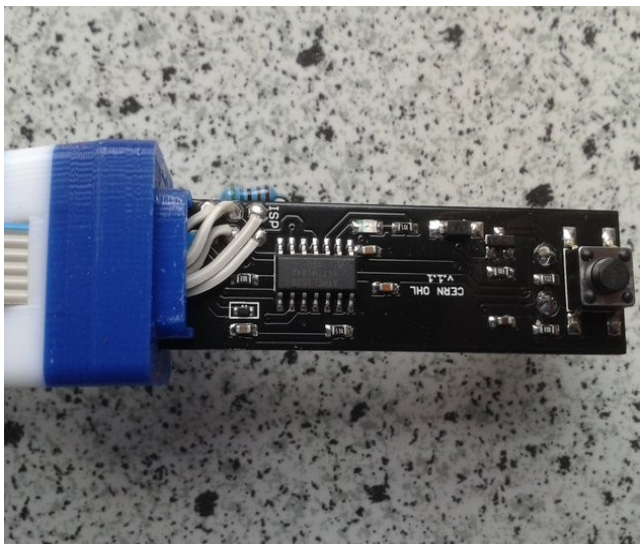


Platine mit esp8266 und StepUp-Converter



Platine mit Akku im Gehäuse

Client komplett



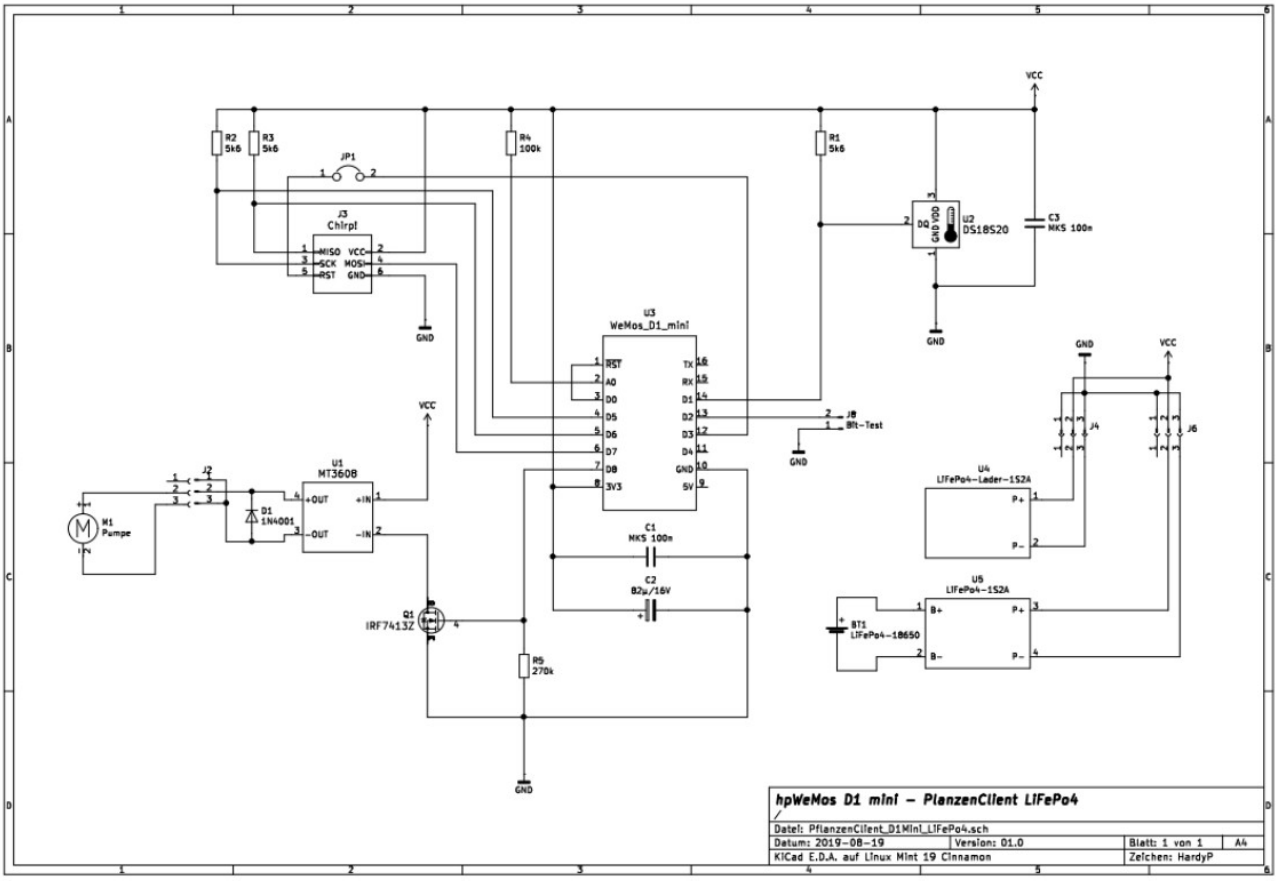
Chirp Verkabelung

Chirp im Gehäuse

4. Hardware

4.1. Client

Zentrales Bauteil des Clienten ist der 32Bit-Controller ESP8266 der auf dem Board WeMos-D1-Mini verbaut ist.



Über Port-D1 wird eine 1-Wire-Verbindung zum Temperaturfühler DS18B20 (U2) hergestellt. Mit dem DS18B20 wird nicht nur die Temperatur am Clienten gemessen, er liefert auch eine einmalige 12- bis 14stellige Clienten-Nummer, mit der sich die Clienten beim Pflanzen-Server anmeldet.

Über Port-D8 wird der NMOS-Transistor IRF7413Z (Q1) geschaltet. Der Widerstand 270k zieht das Gate des NMOS sicher auf GND-Potenzial wenn der esp8266 sich im Tiefschlaf befindet. Eigentlich ist dieser Widerstand nicht notwendig, da der WeMos-D1-mini so einen Widerstand an dem Port-D8 schon enthält, ich habe aber auch schon D1-Mini (nicht original) gehabt die diesen Widerstand nicht hatten. Schaden tut er zumindest nicht. ;-) Der NMOS-Transistor Q1 schaltet den StepUp-Converter

MT3608 (U1), an dessen Ausgang die Pumpe angeschlossen werden kann. Da ich eine 5V-Mini-Tauchpumpe zur Wassergabe benutze, erzeugt der StepUp-Converter aus der Akkuspannung ca. 5,5V für die Tauchpumpe.

Die D5,D6 und D7 sind zu dem 6 poligen Wannenstecker geschaltet, ebenso GND und VCC liegen dort auf. Über diesen Wannenstecker wird der Feuchtefühler (Chirp) an den Clienten angeschlossen. D5 und D6 sind Eingänge für den esp8266, sie sind als OpenDrain ausgeführt. R2 und R3 sind die zugehörigen PullUp-Widerstände. D5 ist der Takteingang, und D6 ist die Datenleitung für die Datenübertragung durch den Chirp. D7 ist ein Datenausgang des esp8266 und ebenfalls als OpenDrain ausgeführt. Der PullUp-Widerstand wird auf dem Chirp eingelötet. Der esp8266 zieht D7 nach unten, wenn er Daten vom Chirp anfordert.

Über den an A0 angeschlossenen 100k-Widerstand (R4) misst der esp8266 seine Betriebsspannung. Da der WeMos-D1-mini intern bereits einen Spannungsteiler aus 220k und 100k enthält, kann er so max. 4,2V messen.

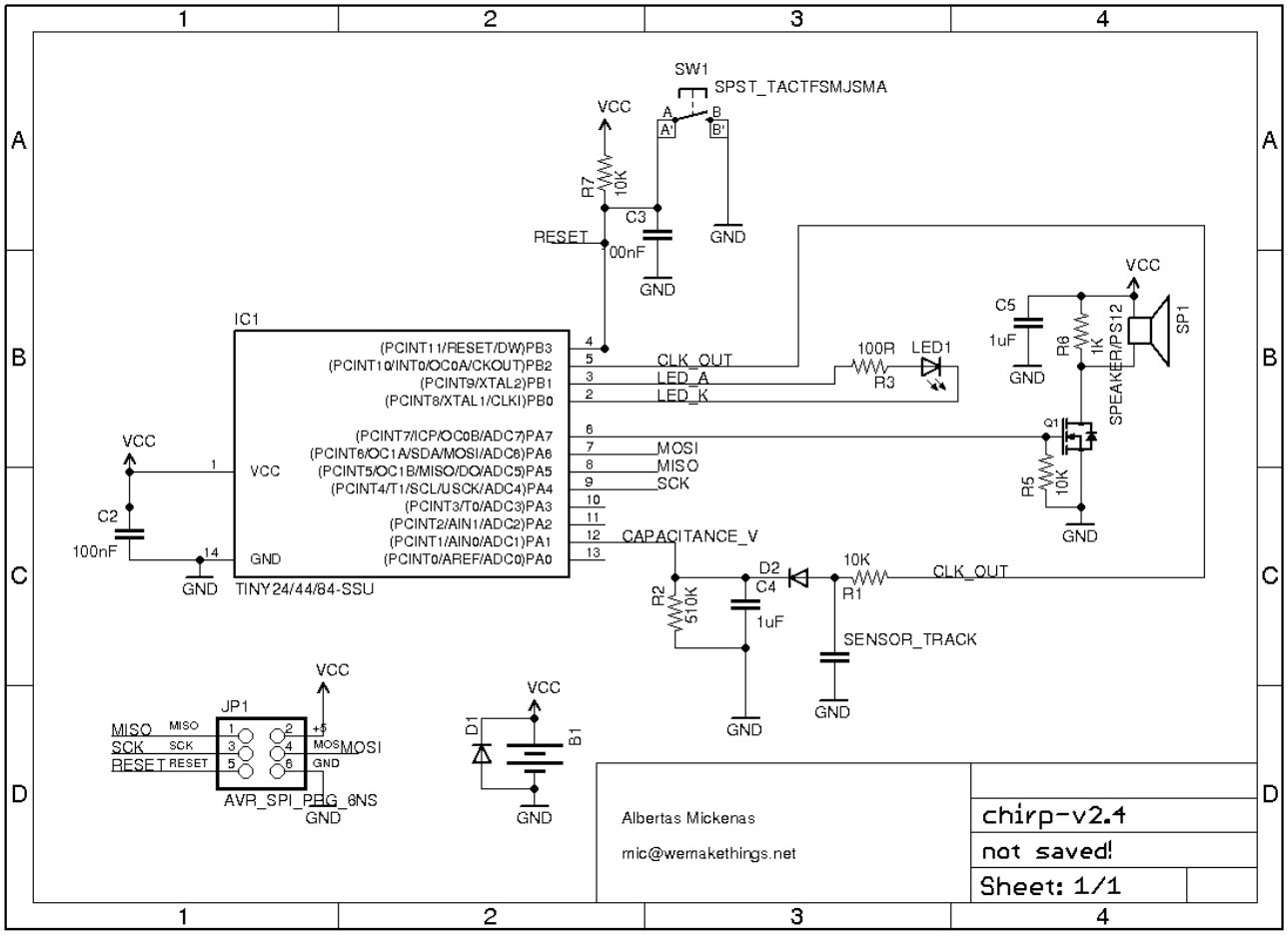
Für den esp8266 wird ein Spannungsbereich von 1,7V bis 3,6V angegeben, ideal für eine Versorgung durch einen LiFePo4-Akku. Der LiFePo4-Akku hat einen recht flachen Spannungsverlauf, und eine maximale Ladespannung von ca. 3,6V. Aus diesem Grund ist der LiFePo4-Akku direkt an den 3,3V Anschluss des WeMos-D1-mini geführt. Auf jeden Fall sollte man den Akku gegen Tiefentladung und Überladung schützen. Dafür wird das LiFePo4 Protection Board (U5) direkt an die Lötfahnen des Akkus angelötet.

Folgende Bauteile werden benötigt:

1 WeMos-D1-Mini esp8266 Controller
1 SetUp-Converter MT3608
1 Temperatur-Fühler DS18B20
1 Akkuzelle 18650 3,2V LiFePo4 mit U-Lötfahne
1 LiFePo4 Protection Board 1S2A oder 1S10A
1 NMOS-Transistor IRF7413Z
1 Diode 1N4001
3 Widerstände 5,6k
1 Widerstand 100k (am besten 0,1%)
1 Widerstand 270k
2 MKS-Kondensatoren 0,1µ (5,12mm Raster)
1 Elko 82µF/16V
1 Wannensteckleiste 6polig (2,54mm Raster)
Kleinteile (kürzbare Stiftleisten, Schrumpfschlauch etc)

4.2. Chirp

Der von Albertas Mickėnas (Miceuz) entwickelte Chirp verwendet zur Erdfeuchtemessung ein kapazitives Messverfahren. Dazu wird ein Rechtecksignal über einen Widerstand auf ein großes Pad gegeben. Dieses in die Erde gesteckte Pad bildet zusammen mit der umgebenen Erde einen parasitären Kondensator, dessen Kapazität sich in Abhängigkeit von der Feuchtigkeit verändert. Der Widerstand und der Kondensator bilden ein veränderbares Tiefpassfilter. Die positiven Signalanteile des gefilterten Rechtecksignals gelangen über eine Diode auf einen Kondensator und werden dort gespeichert. Der so gespeicherte Spitzenwert kann dann gemessen werden.



Auf dem Chirp arbeitet ein AVR-Tiny44 – Mikroprozesser, mit einer Taktfrequenz von 1MHz. Zur Messung der Erdfeuchte gibt er den Takt des Timer0 über PB2 (CLK OUT) auf die Messeinrichtung. Der Spitzenwert am Kondensator wird von seinem Analog-Digital-Converter über PA1 (ADC1) gemessen.

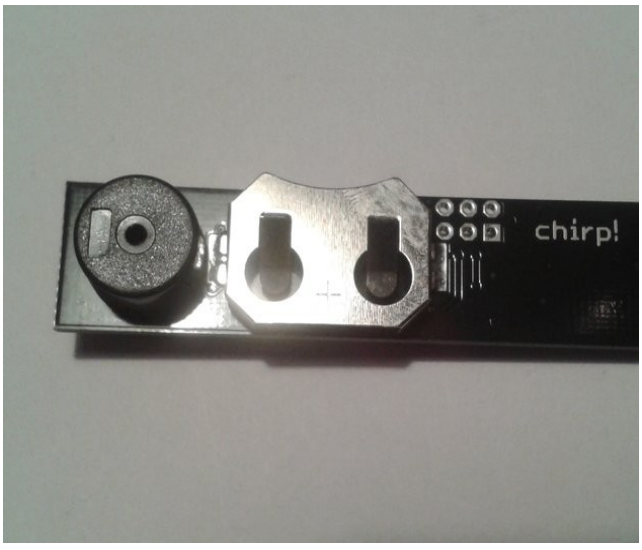
Die Spannungsversorgung und der Datenaustausch zwischen Client und Chirp erfolgt über die AVR-SPI-Schnittstelle.

Der Chirp hat eine maximal Schlafzeit (PowerDown) von max. 8 Sekunden. Wenn der Client von seinem Tiefschlaf aufwacht und Messdaten vom Chirp benötigt, zieht er die MOSI-Leitung auf GND. Die Wartezeit auf die Antwort des Chirp kann bis zu 8 Sekunden dauern. Ist der Chirp aufgewacht, taktet er die SCK-Leitung und gibt dabei die 16Bit auf der MISO-Leitung aus.

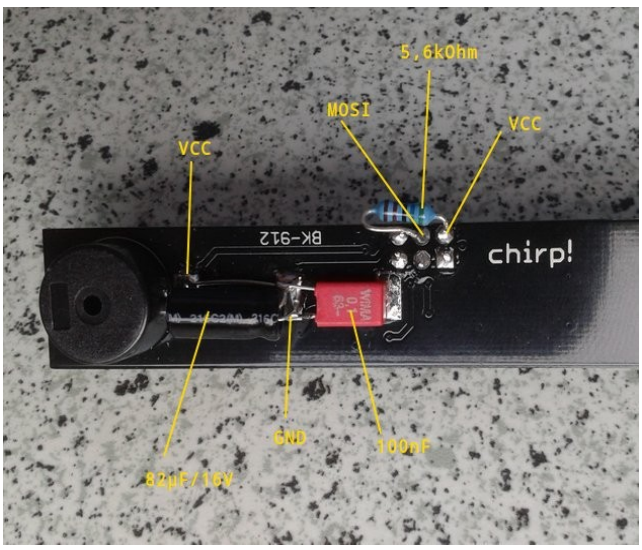
Datenprotokoll zwischen Client und Chirp

<u>Modifikationen des Chirp:</u>

Der auf dem Original-Cirp verbaute wird entfernt, an dessen Stelle wird ein Elko (82µF/16V) und ein MKS-Kondensator (0,1µF) eingelötet. Der PullUp-Widerstand (5,6kOhm) der MOSI-Leitung wird an die AVR-ISP des Chirp angelötet, sowie ein 6poliges Flachkabel.



Original Chirp



Modifizierter Chirp

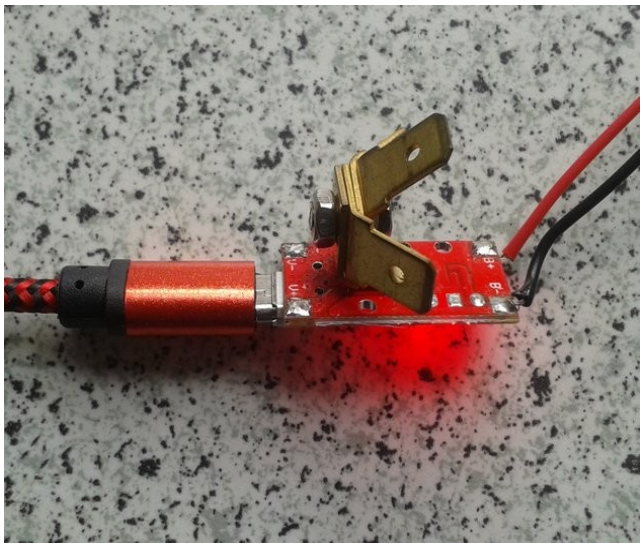
4.3. LiFoPo4-Lader

Die Dauer einer Akkuladung ist sehr stark davon abhängig, wie oft die Wasserpumpe eingeschaltet wird. Bei mir hat es immer ca. 2 Monate gedauert bis ich dn LiFePo4-Akku nachladen musste.

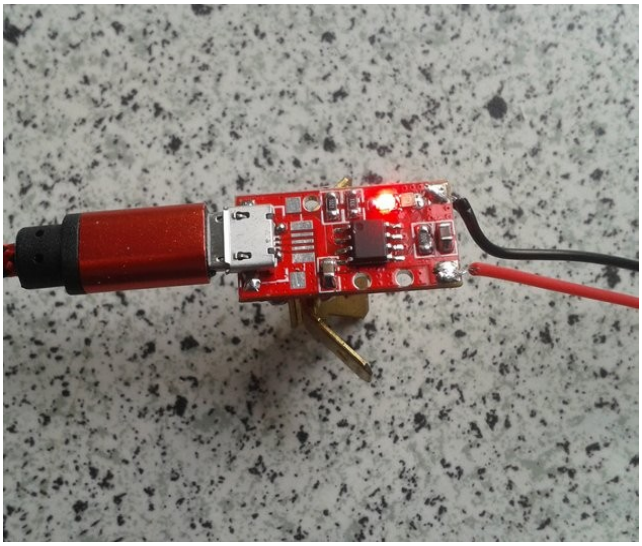
Das mache ich mit einem kleinen USB-Lader der ähnlich aufgebaut ist wie die normalen TP4056, allerdings sind hier die Spannungsverhältnisse an einen LiFePo4-Akku angepasst.

Die Ladeschlussspannung ist 3,6V, der Ladestrom beträgt max. 1A. Damit der Temperaturschutz nicht zu schnell den Ladestrom begrenzt, habe ich auf den Kühl-Pad des Reglers einen Bastel-Kühlkörper drauf gelötet.

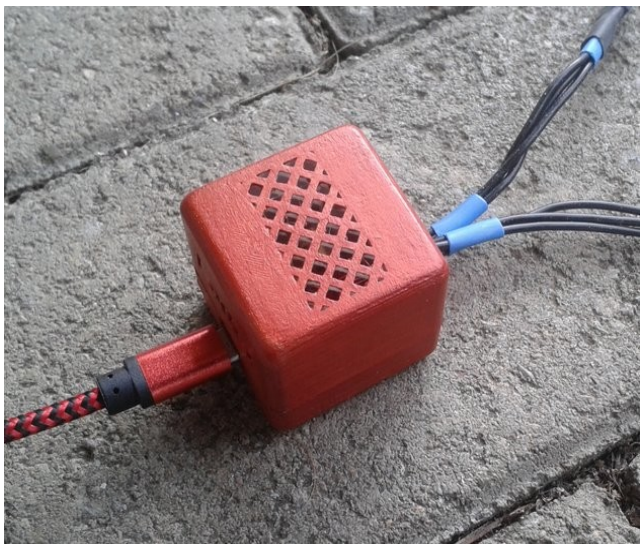
Der Ladevorgang wird durch eine rote LED, und der Ladeschluss durch eine grüne LED gemeldet.



LiFePo4-USB-Lader mit Kühlkörper



LiFePo4-USB-Lader von unten

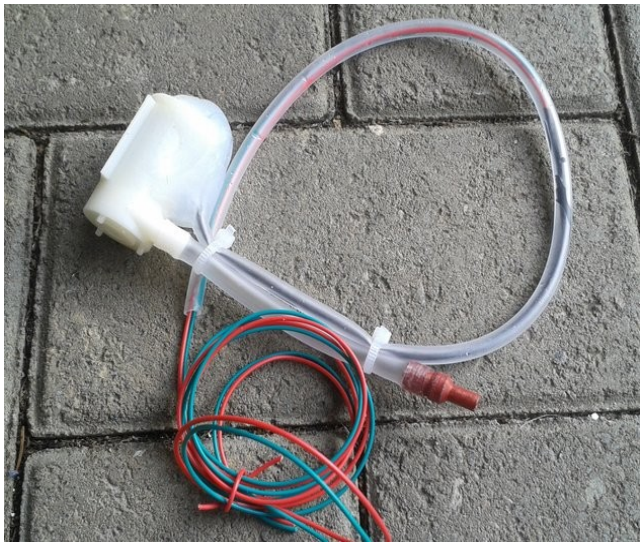


LiFePo4-USB-Lader im Gehäuse

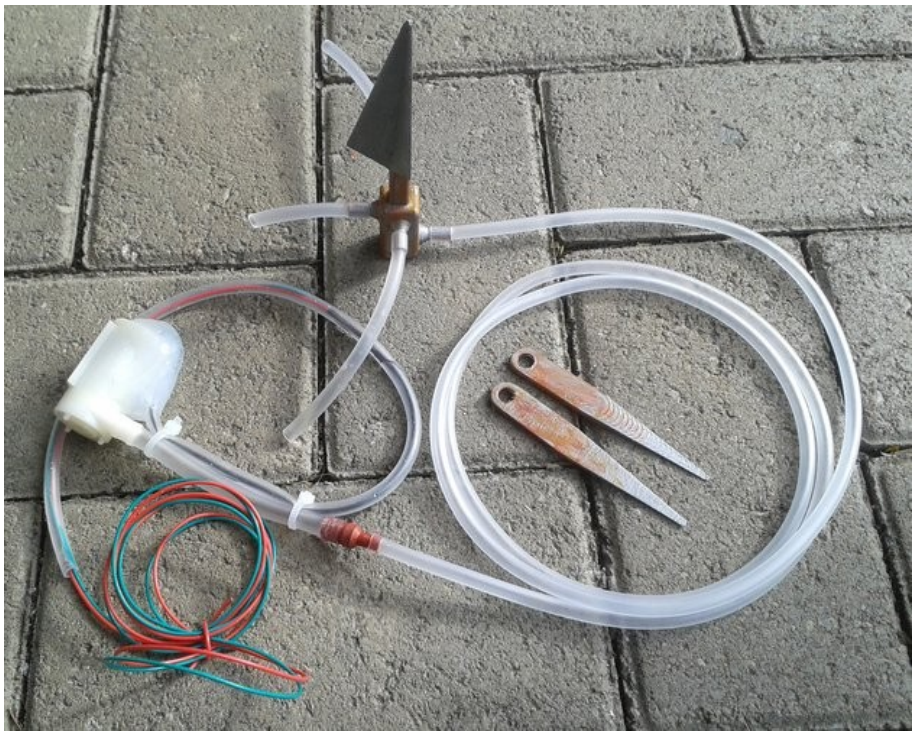
4.4. Wasser

Da bei mir der Wasserbehälter sehr nahe bei der Pflanze steht, wird nur eine sehr kleine Pumpe benötigt, was sich auch sehr vorteilhaft auf die Akkulaufzeit auswirkt. Deshalb verwende ich eine kleine Tauchpumpe mit 5V Betriebsspannung. Diese Pumpe ist sehr preiswert bei eBay erhältlich. Leider ist das Anschlusskabel sehr kurz und muss verlängert werden. Damit die Lötstelle nicht mit Wasser in Kontakt kommt, wird die ein Silikonschlauch über das verlängerte Anschlusskabel geschoben, und an der Tauchpumpe mit Aquarium -Silikon abgedichtet. Als Silkonschlauch verwende ich Luftschlauch vom Aquarium.

Miniatur-Tauchpumpe



Tauchpumpe mit Anschluss und Schlauch



Tauchpumpe komplett mit Erdspieß

5. Software

5.1. Client

Die komplette Software des Clienten für ProgramIO befindet sich im Ordner „hpWeMosD1Mini_PlanzenClient_V3“, und kann mit Paste/Copy in die eigene Entwicklungsumgebung integriert werden.

Aufgabe der Clientsoftware ist

a) den Sensor für die Erdfeuchte zu auszulesen

b) die Umgebungstemperatur zu lesen

c) ggf. die Tauchpumpe für eine definierte Zeit einzuschalten

d) die ermittelten Messwerte zum Pflanzen-Server zu senden.

--- Setup ---

Da der esp8266 sich die meiste Zeit im Tiefschlaf befindet, muss er zwei unterschiedliche Setup-Routinen durchlaufen

a) *SetupColdStart()* wenn die Reset-Taste am Controller betätigt, bzw. die Betriebsspannung angelegt wurde.

b) *SetupDeepSleep()* wenn der Controller aus dem Tiefschlaf erwacht.

c) *SetupErrorStart()* wenn der Controller aufgrund eines Fehlers neu gestartet wurde.

Da der esp8266 im Tiefschlaf seine gesamten Informationen verliert werden vorher wichtige Daten die sich in einer strukturierten Variable befinden, im EEPROM abgelegt.

Diese sind

SerialDEBUG – ggf. Debug-Ausgabe auf der seriellen Konsole

hTrocken – unterhalb dieser Schwelle soll Wasser gegeben werden

hWartenBisWasser – ist dieser Wert>0 wird kein Wasser gegeben, obwohl die Schwelle unterschritten wird. Damit wird genügend Zeit (Minuten) gegeben, damit das Wasser in die Pflanzenerde einsickern kann. Dieser Wert wird bei jedem Aufwachen um den Wert der Schlafdauer reduziert.

hSchlafZeit – Zeit (Minuten) die der esp8266 im Tiefschlaf verbringt

hPumpZeitSoll – Einschaltzeit (Sekunden) der Wasserpumpe

hWartenBisWasserSoll – Wartezeit (Minuten) die gewartet wird, bis wieder neues Wasser gegeben werden darf.

hAlias – Alternative Bezeichnung des Clienten, zB. Name der Pflanze etc.

Nach dem Aufwachen aus dem Tiefschlaf werden diese Daten aus dem EEPROM gelesen, beim Kaltstart und Fehlerstart werden neue Standartwerte gesetzt.

Als nächstes wird die Betriebsspannung gemessen *MesseVCC()* , wird der Wert von 2,9V unterschritten, wird der esp8266 in Dauerschlaf versetzt.

--- Hauptschleife ---

In der Hauptschleife werden die Daten vom Chirp gelesen *DatenVomChirp()* . Sollte dabei ein Fehler auftreten, wird ein neuer Leseversuch gestartet. Ansonsten wird der Wert der Erdfeuchte in *hFeuchte* übergeben.

Der eigentlich Lesevorgang des Chirp geschieht in *LeseChirp()* .

Da der Chirp sich, um Energie zu sparen, auch regelmäßig schlafen legt (8 Sekunden) , kann er nicht direkt auf eine Anfrage des Clienten reagieren. Deshalb muss sich der Client anmelden, in dem er das Signal *ChirpSelect* auf GND zieht, und muss nun eventuell 8 Sekunden auf ein Signal von Chirp warten. Dieser antwortet, in dem er im Takt von *ChirpClock* die Daten auf *ChirpDaten* ausgibt. Erfolgt in dieser Zeit keine Reaktion von Chirp, wird die Leseroutine mit einem Lesefehler abgebrochen.

Nach der Auslesung des Chirp wird die Temperatur vom DS18B20 gelesen. Die unverwechselbare Seriennummer des DS18B20 wird als Sensorkennung verwendet. Damit sind die Daten des Clienten unverwechselbar vom Server zuzuordnen.

In Vergleichen mit *hTrocken* und *hWartenBisWasser* wird ggf. die Wasserpumpe für die eingestellte Zeit eingeschaltet.

Danach wird *GuteNacht()* aufgerufen.

In *DatenZuVomServer()* wird die Kommunikation mit dem Pflanzen-Server abgewickelt. Das erfolgt mit dem UDP-Protokoll. Auf MQTT wurde ganz bewusst verzichtet, da als Pflanzen-Server ebenfalls ein preisgünstiger ESP verwendet werden soll. Die Clienten arbeiten autonom und werden bei gestörter Kommunikation mit dem Server nicht behindert. Kommen sich die Anfragen der einzelnen Clienten an den Server zu nah, werden die Zeitabstände durch Veränderung der Schlafzeiten automatisch verlängert. Konnte der Pflanzen-Server das UDP-Paket empfangen, sendet er ein entsprechendes Echo an den Clienten.

Dabei werden ggf. geänderte Einstellungen des übertragen:

a) *uAlias* - alternative Bezeichnung des Clienten

b) *uTrocken* - Feuchteschwelle unterhalb der Wasser gegeben werden soll
c) *uSchlafZeit* - Schlafzeit des Clienten
d) *uExtraSchlafZeit* - zusätzliche Schlafzeit um die Kontakte zum Server zu entflechten
e) *uWartenBisWasserSoll* - Wartezeit (Minuten) die gewartet wird, bis wieder neues Wasser gegeben werden darf.
f) *uPumpZeitSoll* - Einschaltzeit (Sekunden) der Wasserpumpe

Sollten Daten vom Pflanzen-Server geändert worden sein, erfolgt die Speicherung im EEPROM und er Client legt sich die eingestellte Zeit schlafen.

Kommt kein Echo vom Pflanzen-Server werden die EEPROM-Daten weiter verwendet und er Client legt sich ebenfalls schlafen.

5.2. Chirp

Text folgt noch