hpWeMosD1Mini_PlanzenClient_V3

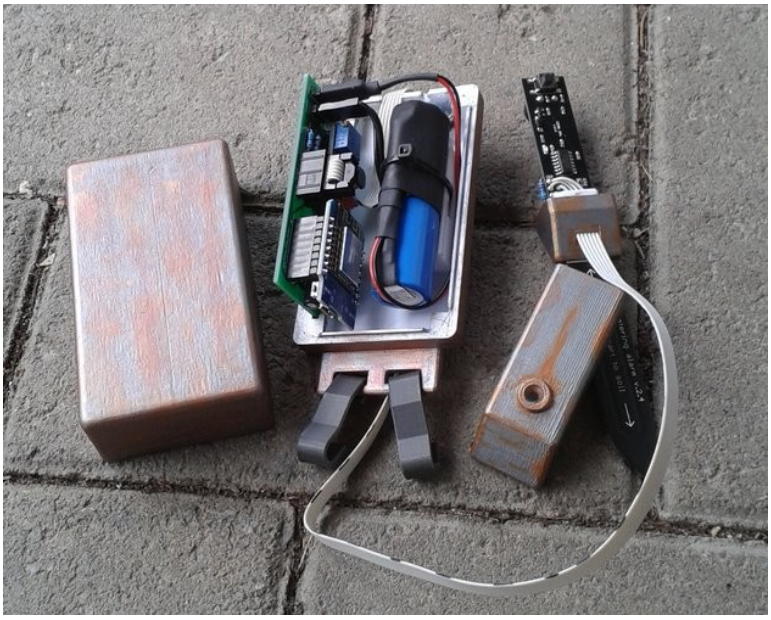
1. General

On my terrace I grow tomatoes, peppers and other greens in pots. Because of the last hot summers I was very unsettled about the amount of water I had to give, I was constantly torn between too much and too little water.

so in 2019 I have to think about an own irrigation system for my purposes. After many experiments I have developed a suitable system, and could already test this system last year for 3 months on my beans.

This year, 2020, it is planned to water all plants completely automatically.

At this point I would like to introduce the plant clients I have developed, and give suggestions for own developments, or to encourage the reproduction. The corresponding plant server can be found at [plant-watering-server-wemos-lolin32](#) .

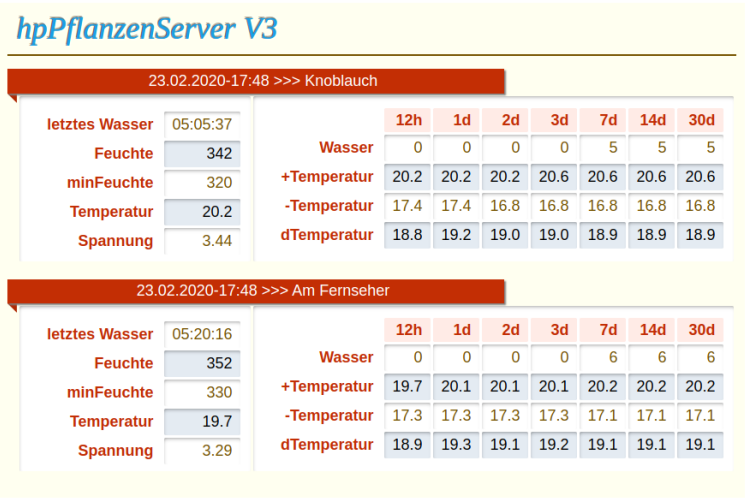


The inner life

Waiting for spring ;-)

2. Operating

Each plant pot gets its own plant client with its own water pump, and soil moisture and water quantity adjusted to this plant. The plant clients exchange information with the plant server via WiFi at regular intervals. This server creates a website in the local network, on which the status of the individual plant clients is displayed and the individual plant clients can be set



Local WebSite of the plant server

In general, however, a plant client works independently of the plant server, all settings are stored locally in the plant client. If a connection to the server is not possible, the client continues its program undisturbed with the saved settings. According to the set time, the client will search again for contact with the plant server. This behaviour ensures a high level of system security, so an error in the data transmission or in the server does not lead to a standstill of the entire irrigation system.

The Plant Client draws its energy from the built-in rechargeable battery. To ensure that the battery functions for as long as possible without recharging, the Plant Client spends most of the time "sleeping". From this deep sleep it is regularly woken up (e.g. every 30 minutes) to read the connected moisture sensor, activate the water pump for the appropriate time if necessary, measure temperature and battery voltage, and send the values to the plant server. On this occasion, the server will also accept the possibly changed settings from the server.

The Plant Client sends to the server:

- unique client number
- set client name
- measured moisture content of the plant soil
- Air temperature at the client
- if necessary, running time of the water pump
- Battery voltage of the client.

By clicking on the title of a plant client, the client settings can be entered. By clicking on "OK" the settings are accepted and sent to the plant client at the next opportunity and saved there.

Local WebSite offers client settings

The following client settings are possible:

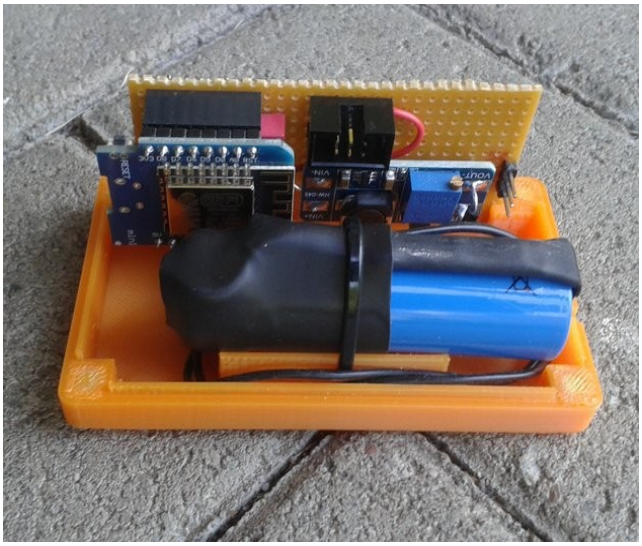
- Client name (max. 30 characters).
- Client sleep time (max. 60 minutes).
- Pump time of the water pump if the moisture falls below the minimum moisture level (max. 60 seconds).
- Water stop after watering, although the moisture level falls below the minimum moisture level (max. 24*60 minutes). This function is necessary to give the water the necessary time to infiltrate the soil.
- If the measured value of the humidity sensor falls below the set value of the minimum humidity, it will waterpunpe the set time. Good values for the minimum moisture are between 250 and 350.

Each plant client is identified by a 12 to 16 digit number supplied by the installed temperature sensor.

The measurement of the soil moisture is carried out with the sensor "Chirp" connected. The Chirp is a development of Albertas Mickėnas (Miceuz). A detailed description can be found on https://github.com/Miceuz/PlantWateringAlarm und https://wemakethings.net/chirp/ . For the application here, the Chirp is connected to the plant client via cable and programmed with a special program.

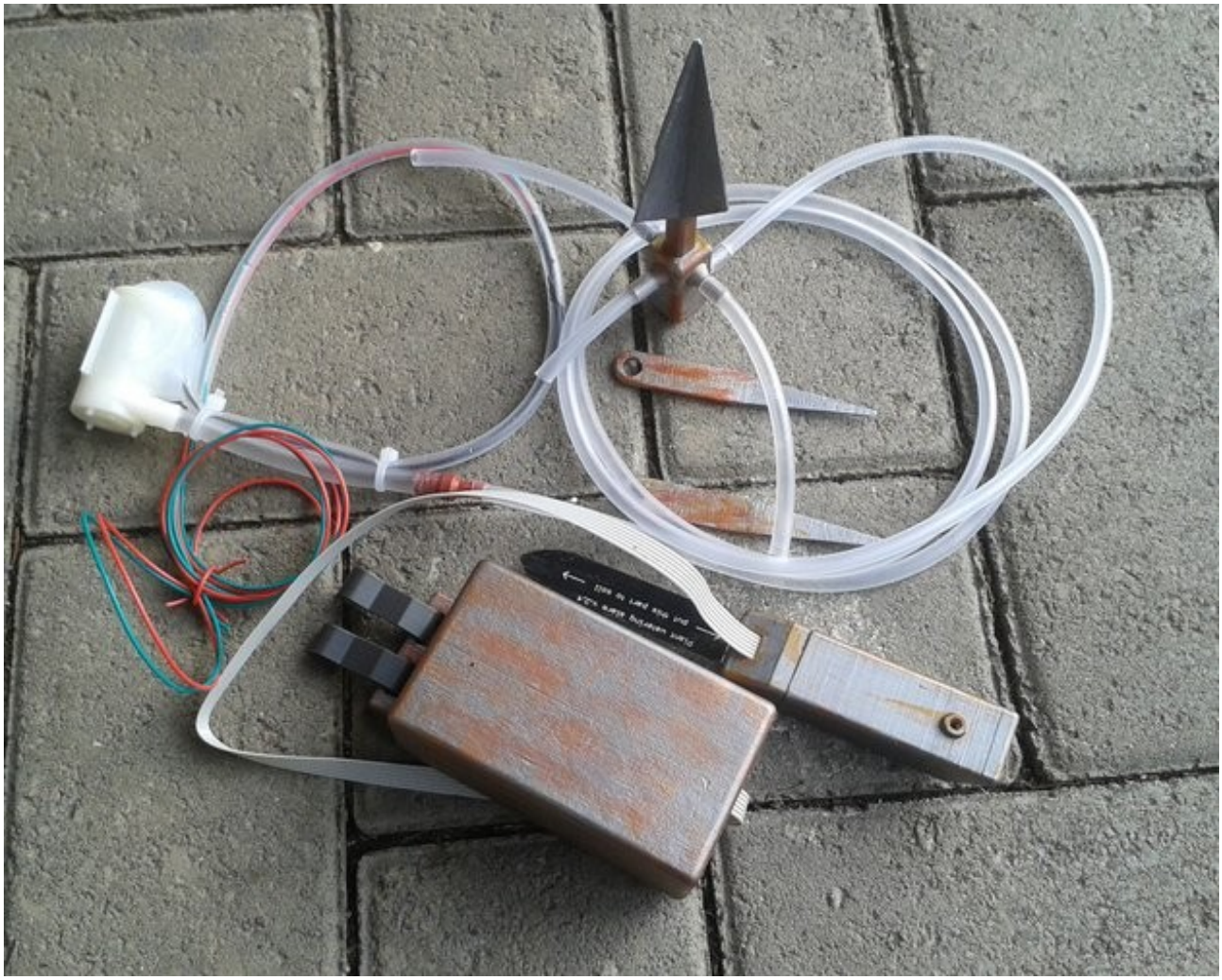
3. Evolution

In June 2019, I started the development of the irrigation system, and then tested it on my runner beans during a test phase and finished developing it.

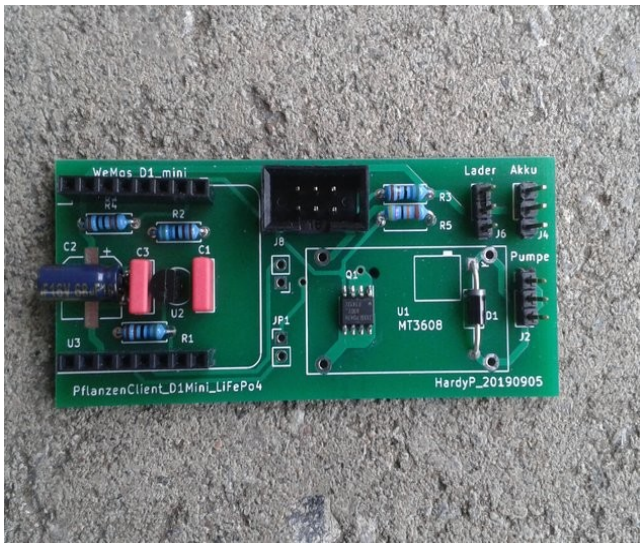


First design from 2019

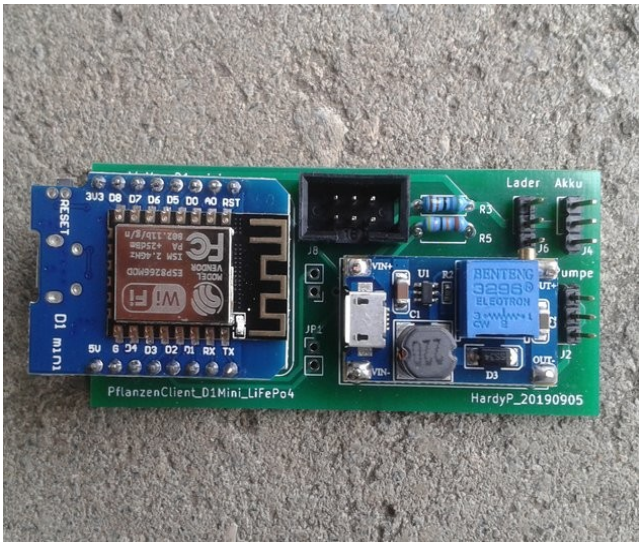
Test operation 2019



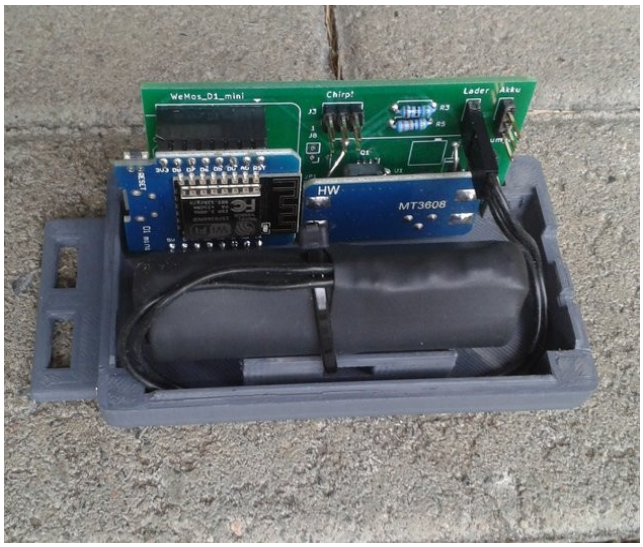
Complete client with sensor, pump and hose



Assembled board of the client



Board with esp8266 and StepUp converter



PCB with battery in housing

Client complete

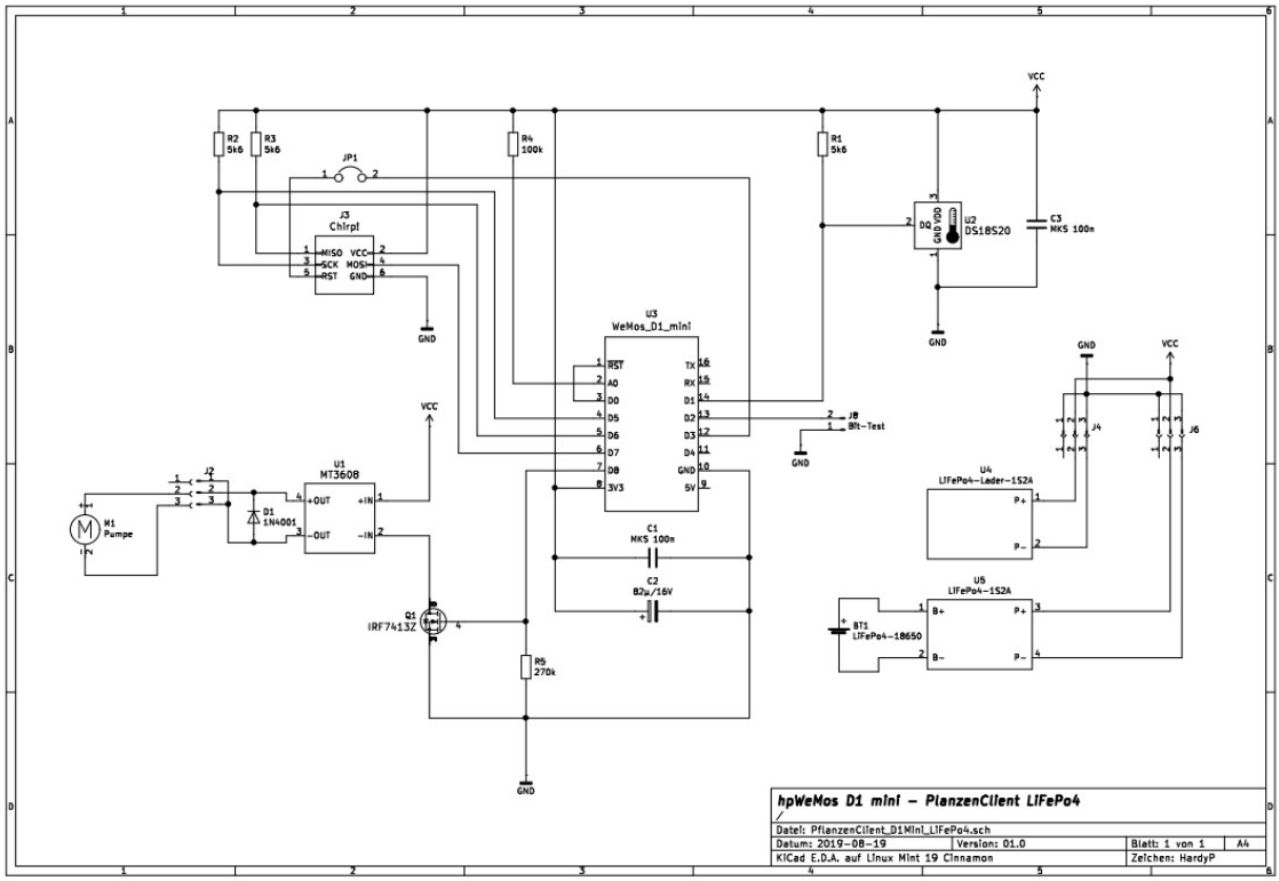
Chirp Wiring

Chirp in housing

4. Hardware

4.1. Client

Central component of the client is the 32Bit controller ESP8266 which is installed on the board WeMos-D1-Mini.



A 1-wire connection to the temperature sensor DS18B20 (U2) is established via port D1. The DS18B20 not only measures the temperature at the client, it also provides a unique 12 to 14-digit client number with which the client logs on to the plant server.

The IRF7413Z NMOS transistor (Q1) is switched via port D8. The 270k resistor safely pulls the NMOS gate to GND potential when the esp8266 is in deep sleep. Actually this resistor is not necessary, because the WeMos-D1-mini already contains such a resistor at the port-D8, but I have also had D1-mini (not original) which did not have this resistor. At least it does not do any damage ;-) The NMOS transistor Q1 switches the step-up converter MT3608 (U1), to whose output the pump can be connected. Since I use a 5V mini submersible pump for watering, the StepUp-Converter generates from the battery voltage about 5.5V for the submersible pump.

The D5, D6 and D7 are connected to the 6 pin connector, also GND and VCC are connected to this connector. The humidity sensor (chirp) is connected to the client via this connector. D5 and D6 are inputs for the esp8266, they are designed as OpenDrain. R2 and R3 are the associated pull-up resistors. D5 is the clock input, and D6 is the data line for data transmission through the chirp. D7 is a data output of the esp8266, and is also designed as an OpenDrain. The PullUp resistor is soldered onto the chirp. The esp8266 pulls D7 down when it requests data from the chirp.

The esp8266 measures its operating voltage via the 100k resistor (R4) connected to A0. Since the WeMos-D1-mini already contains an internal voltage divider consisting of 220k and 100k, it can measure max. 4.2V.

A voltage range of 1.7V to 3.6V is specified for the esp8266, ideal for supply by a LiFePo4 battery. The LiFePo4 battery has a fairly flat voltage curve, and a maximum charging voltage of approx. 3.6V. For this reason the LiFePo4-battery is connected directly to the 3.3V connector of the WeMos-D1-mini. In any case you should protect the battery against deep discharge and overcharge. For this purpose the LiFePo4 Protection Board (U5) is soldered directly to the soldering tags of the battery.

The following components are required:

1 WeMos-D1-Mini esp8266 controller

1 SetUp converter MT3608

1 temperature sensor DS18B20

1 rechargeable cell 18650 3.2V LiFePo4 with U solder tag

1 LiFePo4 Protection Board 1S2A or 1S10A

1 NMOS transistor IRF7413Z

1 diode 1N4001

3 resistors 5,6k

1 resistor 100k (best 0.1%)

1 resistor 270k

2 MKS capacitors 0.1µ (5.12mm pitch)

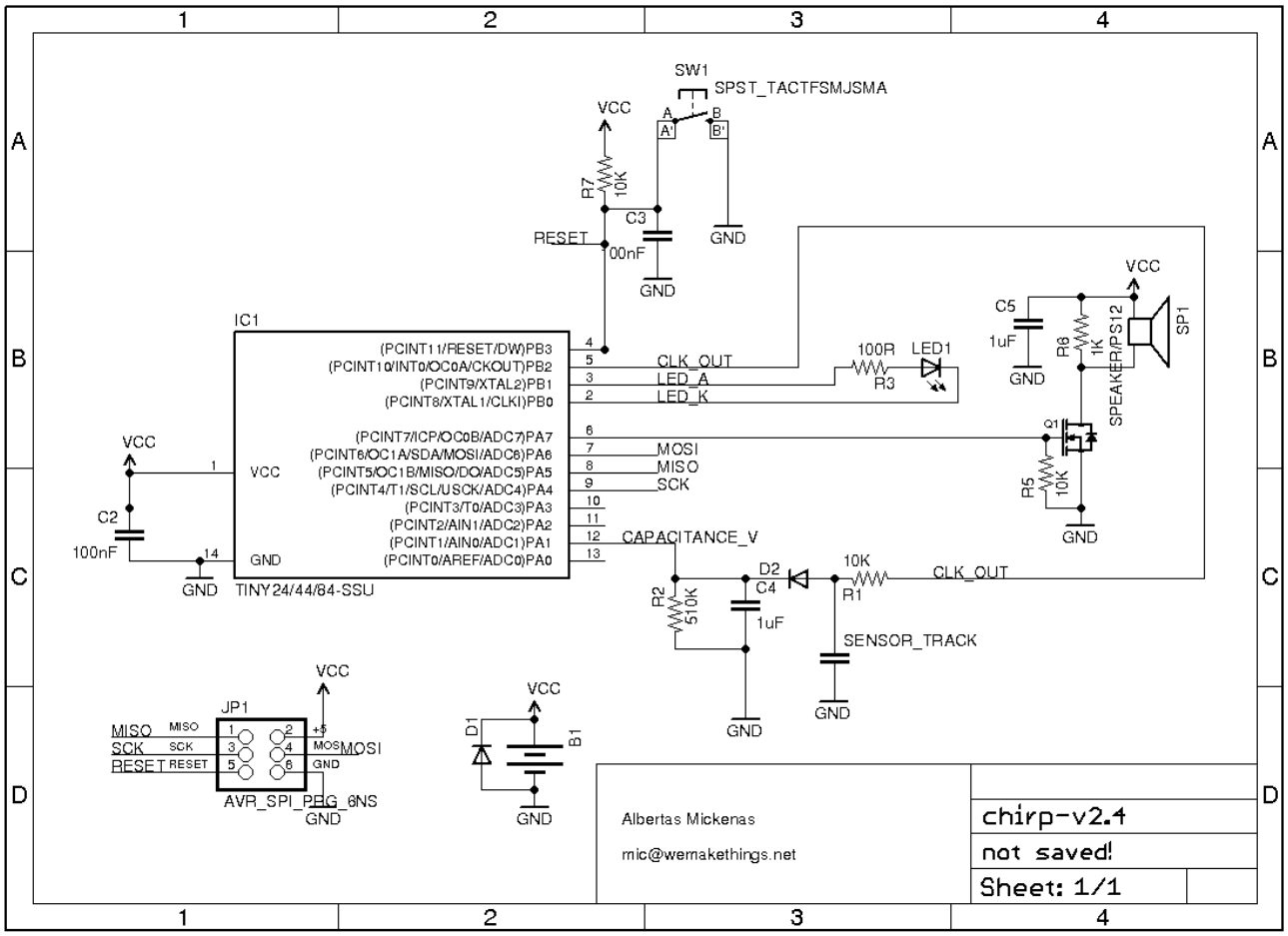
1 capacitor 82µF/16V

1 6-pole socket strip (2.54mm grid)

Small parts (shortenable pin strips, heat shrinkable tubing etc)

4.2. Chirp

The Chirp developed by Albertas Mickėnas (Miceuz) uses a capacitive measuring method for measuring the earth's moisture. For this purpose, a square wave signal is applied to a large pad via a resistor. This pad, which is inserted into the earth, forms a parasitic capacitor together with the surrounding earth, whose capacity changes depending on the humidity. The resistor and the capacitor form a variable low pass filter. The positive signal components of the filtered square wave signal reach a capacitor via a diode and are stored there. The peak value thus stored can then be measure



An AVR-Tiny44 microprocessor with a clock frequency of 1MHz works on the Chirp. To measure the earth's humidity it sends the clock of the Timer0 via PB2 (CLK OUT) to the measuring device. The peak value at the capacitor is measured by its analog-digital converter via PA1 (ADC1).

The power supply and the data exchange between client and Chirp is done via the AVR-SPI interface.

he Chirp has a maximum sleep time (PowerDown) of max. 8 seconds. When the client wakes up from its deep sleep and needs measurement data from the chirp, it pulls the MOSI line to GND. The waiting time for the response of the chirp can take

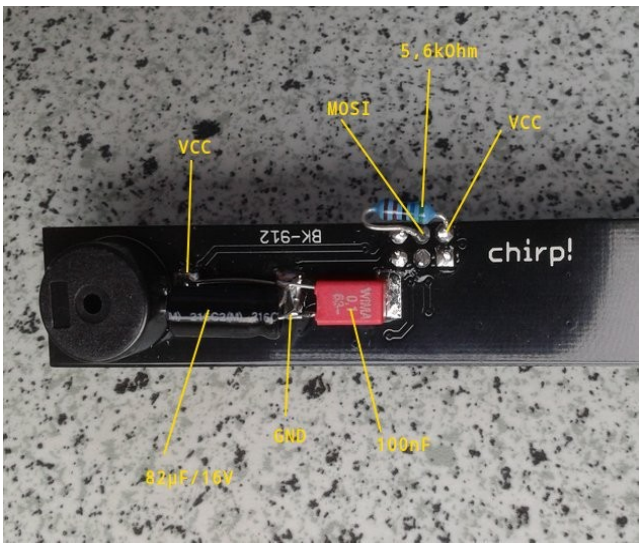
up to 8 seconds. When the chirp wakes up, it clocks the SCK line and outputs the 16Bit on the MISO line.

Data protocol between client and Chirp

Modifikationen des Chirp:

The one installed on the original circuit is removed, in its place an electrolytic capacitor (82µF/16V) and a MKS capacitor (0.1µF) is soldered in. The pull-up resistor (5.6kOhm) of the MOSI line is soldered to the AVR-ISP of the chirp, as well as a 6-pole flat cable.

Original Chirp



Modifiedr Chirp

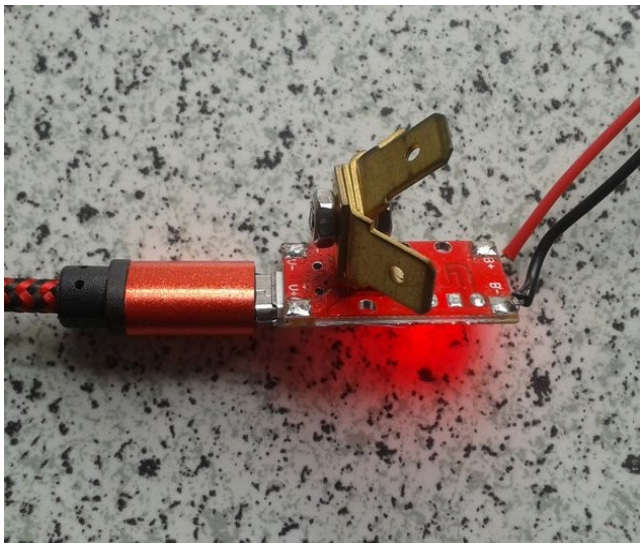
4.3. LiFoPo4 charger

The duration of a battery charge depends very much on how often the water pump is switched on. For me it always took about 2 months to recharge the LiFePo4 battery.

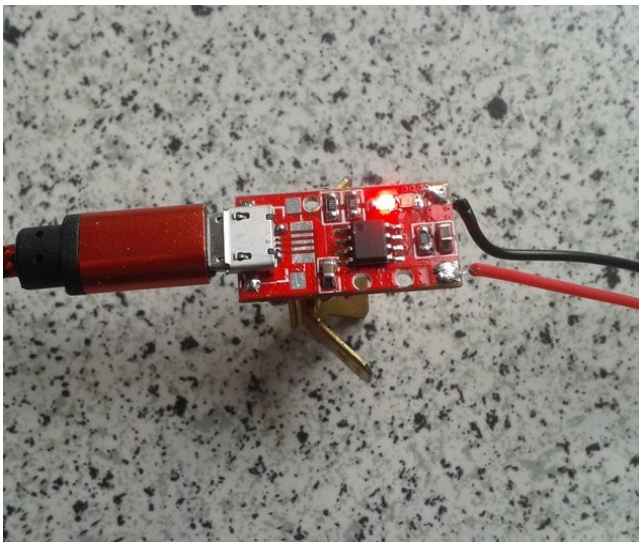
I do this with a small USB charger which is similar to the normal TP4056, but the voltage is adapted to a LiFePo4 battery.

The final charge voltage is 3.6V, the charge current is max. 1A. To prevent the temperature protection from limiting the charge current too fast, I soldered a tinkering heatsink on the cooling pad of the controller.

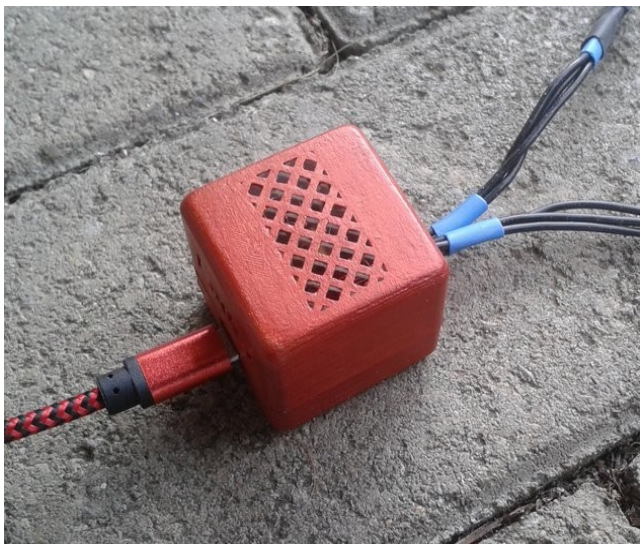
The charging process is indicated by a red LED and the end of charging by a green LED.



LiFePo4-USB-Charger with heat sink



LiFePo4-USB-Loader from below

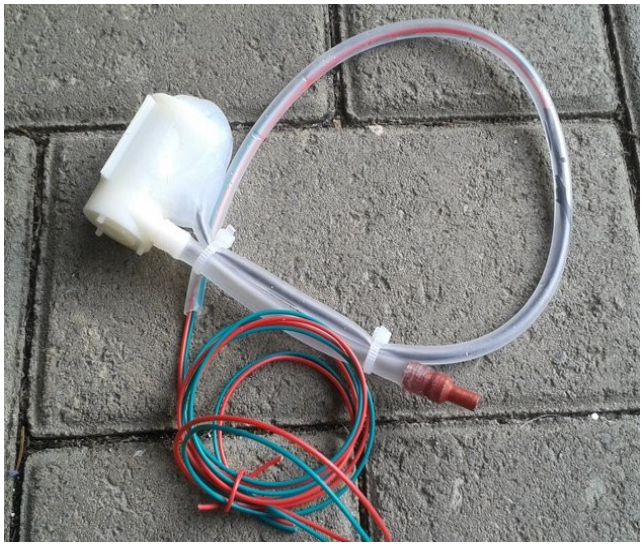


LiFePo4-USB-Charger in housing

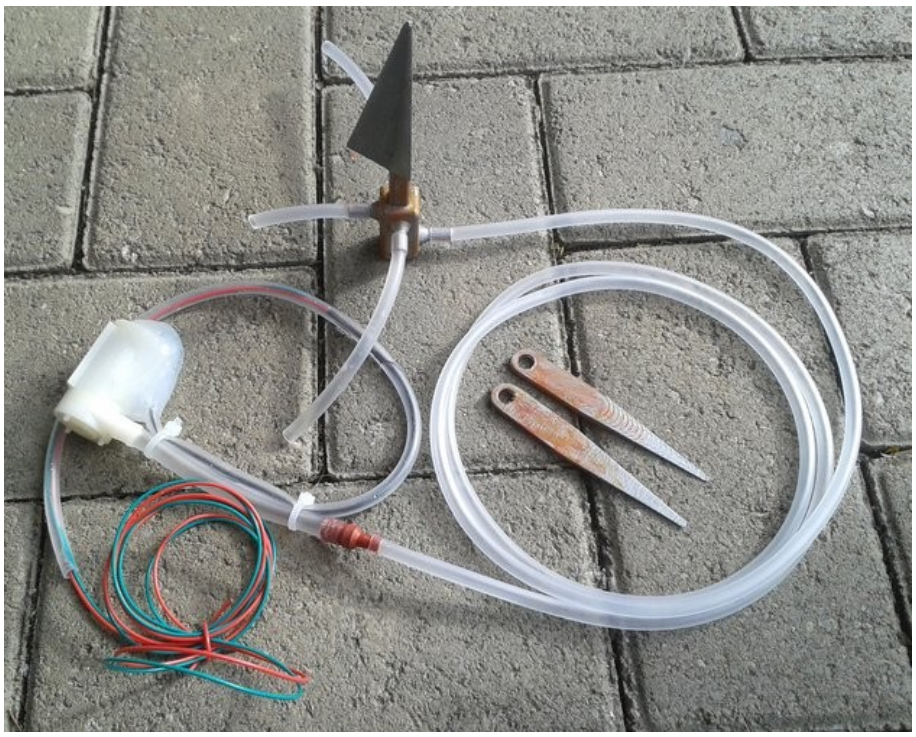
4.4. Watering

Since the water tank is very close to the plant, only a very small pump is needed, which also has a very positive effect on the battery life. Therefore I use a small submersible pump with 5V operating voltage. This pump is available at eBay for a very reasonable price. Unfortunately the connection cable is very short and must be extended. To prevent the soldered joint from coming into contact with water, a silicone hose is pushed over the extended connection cable and sealed with aquarium silicone at the submersible pump. As silicone hose I use air hose from the aquarium..

Miniature submersible pump



Submersible pump with connection and hose



Submersible pump complete with ground spike

5. Software

5.1. Client

The complete client software for ProgramIO is located in the folder "hpWeMosD1Mini_PlanzenClient_V3", and can be integrated into your own development environment with Paste/Copy.

Task of the client software is

a) read the soil moisture sensor

b) ead the ambient temperature

c) if necessary, switch on the submersible pump for a defined time

d) send the measured values to the plant server.

--- Setup ---

As the esp8266 is in deep sleep most of the time, it has to go through different setup routines

a) *SetupColdStart()* when the reset button on the controller is pressed or the operating voltage is applied.

b) *SetupDeepSleep()* when the controller wakes up from deep sleep.

c) *SetupErrorStart()* if the controller was restarted due to an error.

As the esp8266 loses all its information in deep sleep, important data contained in a structured variable is first stored in the EEPROM.

These are

SerialDEBUG – debug output on the serial console if necessary

hTrocken – below this threshold water should be added

hWartenBisWasser – iif this value is>0, no water is given, although the threshold is undershot. This gives enough time (minutes) for the water to infiltrate the plant soil. This value is reduced by the value of the sleep duration each time the plant wakes up.

hSchlafZeit – Time (minutes) that the esp8266 spends in deep sleep

hPumpZeitSoll – switch-on time (seconds) of the water pump

hWartenBisWasserSoll – Wait time (minutes) to wait until new water can be added.

hAlias – Alternative name of the client, e.g. name of the plant etc..

After waking up from deep sleep, this data is read from the EEPROM, new default values are set during cold start and error start.

The next step is to measure the operating voltage *MesseVCC()* , if the value falls below 2.9V, the esp8266 is put into continuous sleep.

--- Main Loop ---

In the main loop the data is read from the chirp *DatenVomChirp()* . If an error occurs, a new read attempt is started. Otherwise the value of the earth humidity is passed in *hFeuchte* .

The actual reading process of the chirp is done in *LeseChirp()* .

Since the Chirp also goes to sleep regularly (8 seconds) to save energy, it cannot respond directly to a request from the client. Therefore the client must log on by pulling the *ChirpSelect* signal to GND and may have to wait 8 seconds for a signal from Chirp. The client responds by outputting the data to *ChirpDaten* in time with *ChirpClock* . If Chirp does not respond within this time, the read routine is aborted with a read error.

After Chirp has been read out, the temperature is read by DS18B20. The unique serial number of the DS18B20 is used as sensor identification. This enables the server to assign the client's data unmistakably.

In comparisons with *hTrocken* and *hWartenBisWasser* , the water pump is switched on for the set time if necessary.

GuteNacht() is then called.

In *DatenZuVomServer()* the communication with the plant server is handled. This is done with the UDP protocol. MQTT was deliberately omitted, since a low-priced ESP should also be used as plant server. The clients work autonomously and are not hindered in case of disturbed communication with the server. If the requests of the individual clients come too close to the server, the time intervals are automatically extended by changing the sleep times. If the plant server was able to receive the UDP packet, it sends a corresponding echo to the client.

If necessary, changed settings of the transferred data are transferred:

a) *uAlias* - alternative name of the client
b) *uTrocken* - moisture threshold below which water should be added
c) *uSchlafZeit* - sleep time of the client
d) *uExtraSchlafZeit* - additional sleep time to unbundle contacts to the server

e) *uWartenBisWasserSoll* - Wait (minutes) to wait until new water can be added.

f) *uPumpZeitSoll* - Switch-on time (seconds) of the water pump

If data from the plant server has been changed, it is stored in the EEPROM and the client goes to sleep for the set time.

If there is no echo from the plant server, the EEPROM data is used further and the client goes to sleep.

5.2. Chirp

text to follow