

# Pflanzengießanlage

Stefan Schubäck, Matthias Nagl, Christoph Hofbauer, Dmitrii Cetvericov, Markus Fischer-Has,

**Zusammenfassung**—Auch unter den Studenten gibt es den einen oder anderen mit einem grünen Daumen, der seine Wohnung durch ein paar Pflanzen aufgewertet hat. Wenn nicht das ständige Gießen wäre. Vor jedem längeren Urlaub stellt sich die Frage, was machen mit den Pflanzen? Den Nachbarn fragen und hoffen er ist da und verlässlich? Die Pflanzen „Absaufen“ lassen und hoffen, dass die sich das Wasser einteilen? Beides keine zufriedenstellende Lösung. Wieso lässt man sich die Pflanze nicht selber gießen? Aus diesen Überlegungen ist die Idee entstanden, eine Pflanze mit Sensoren auszustatten, die uns über den aktuellen Zustand informieren. Im Folgenden wird der Aufbau zweier Lösungen vorgestellt. Der erste Ansatz basiert auf dem Arduino-System, dass einen einfachen Einstieg in Umgang mit Mikrocontroller ermöglicht. Der zweite Ansatz baut auf ähnliche Hardware ohne die Verwendung der Arduino Software Programmierung umgebung. Dadurch wird die Verwendung von Sleep-Modi und Interrupts möglich was somit zu einer Low-Energy Variante des Gießsystems führt.

## I. EINGRENZUNG

Es lassen sich viele Faktoren finden, die einen Einfluss auf das Wohlbefinden einer Pflanze haben. Hierunter fallen die Bodenfeuchtigkeit im Topf der Pflanze, die durchschnittliche Sonneneinstrahlung, der Luftqualität, der Nährstoffgehalt des Bodens sowie die Temperatur an der Wurzel der Pflanze. Diese nicht abschließende Aufzählung zeigt, dass eine Eingrenzung der zu untersuchenden Faktoren notwendig ist um das Projekt in der gegeben Zeit fertigzustellen. Auch in Hinblick des Einsatzgebietes gibt eine Vielfalt von unterschiedlichen Möglichkeiten wie z.B. Einpflanzenbetrieb oder Mehrpflanzenbetrieb sowie Indoor oder Outdoor. Im Rahmen des Projekts werden demnach folgende Eingrenzung vorgenommen:

- Die Bodenfeuchtigkeit wird abgeleitet aus der Widerstandsänderung einer Messgabel die sich im Erdreich der Pflanze befindet. Die Widerstandsänderung ergibt sich aus der Änderung des Wassergehalts der Erde.
- Das System soll für eine alleinstehende Pflanze aufgebaut werden.
- Das Einsatzgebiet wird im Innenbereich sein. Dadurch fallen Einschränkungen bzgl. Witterungsbeständigkeit weg.
- Die Stromversorgung wird über ein Netzteil bewerkstelligt.
- Das System soll über eine Drahtlose Verbindung einstellbar sein. Diese Einschränkung gilt nur für die Arduino Variante, da in der Sparversion auf die Kommunikation verzichtet wird. Hier wird die Einstellung direkt im Code vorgenommen.

Weitere Ideen wie z.B. eine Autarke Lösung mit Batteriebetrieb, ein Mehr-Pflanzenbetrieb oder ein System für den Garten wurden zwar diskutiert, aber wegen dem erhöhten Zeitaufwand und begrenzten Budgets nicht weiter verfolgt. Im folgenden wird die *Arduino Version* ausführlich vorgestellt.

Die *Sparversion* wird im Anschluss erläutert, da viele der Funktionen grundsätzlich identisch sind wird darauf verzichtet diese erneut auszuformulieren.

## II. ARDUINO VARIANTE

In Abbildung 1 ist der schematische Aufbau des Projektes zu sehen. Das Gießsystem besteht aus der Steuereinheit, dem Wasserbehälter sowie der Pumpe mit dem Schlauchsystem. Aus dem Steuergerät wird der Sensor über einen Chin-Anschluss herausgeführt. Auf der linken Seite wird das Netzteil mit einem Hohlstecker verbunden. Im Folgenden beschreiben wir das Vorgehen für das Gießsystem basierend auf der Arduino Variante. Es wird hierzu auf folgende Bereiche eingegangen:

- Zu Beginn beschreiben wir verschiedenen Möglichkeiten des Wassertransports zur Pflanze.
- Im weiteren stellen wir das Vorgehen zur Erstellung des Gehäuses für die Elektronik dar.
- In Abschnitt Elektronik werden wir auf den Aufbau der Platine und die hiermit verbunden Bereiche *Sensorschaltung*, *Stromversorgung* und *Kommunikation* eingehen.
- Anschließend gehen wir auf das Programm ein, mit dem entschieden wird wann der Zeitpunkt erreicht ist um die Pflanze zu gießen.
- Abschließend wir eine Kostenplan vorgestellt.

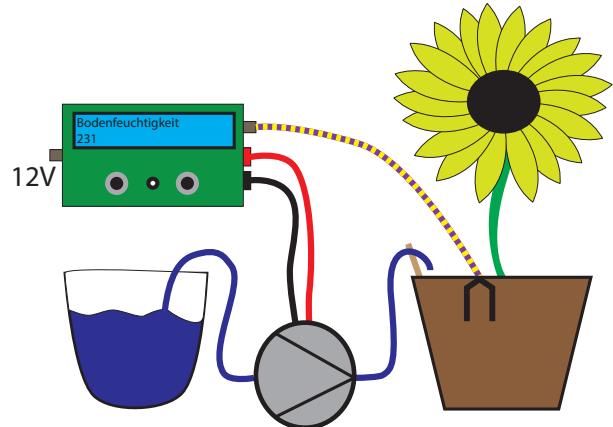


Abbildung 1. Schematischer Aufbau der Gießanlage

### A. Wassertransport

Um den Einsatzbereich so flexibel wie möglich zu halten haben wir uns für ein Pumpensystem entschieden. Dadurch ist die Anordnung der Pflanze zum Wassertank nicht relevant. Der hiermit einhergehenden erhöhte Strombedarf ist zu verkraften, da unser System nicht auf Batterien angewiesen ist.

Tabelle I  
VERGLEICH WASSERPUMPEN UND VENTIL

Eigenschaft	Zahnradpumpe	Kreiselpumpe	Ventil
Selbstsaugend	ja	nein	nein
Lautstärke	sehr laut	mittel laut	leises Klacken
Stromverbrauch	@12V 2,8A	@12V 0,6A	@12V 80mA
Förderleistung	gering	groß	keine eigene
Preis	2,95 €	2,95 €	4,95 €

Für den Wassertransport haben wir uns für eine Zahnradpumpe entschieden. Die Zahnradpumpe gehört zu der Gruppe der rotierenden Verdränger Pumpen<sup>1</sup>. Das Fördermedium wird hierbei zwischen zwei Zahnrädern in einem in sich geschlossenen Volumenbereich gefördert. Die Bauweise dieser Pumpe ermöglicht zudem einen selbstsaugenden Betrieb. Dies bedeutet, dass diese Pumpe in der Lage ist Gase zu transportieren und somit einen Unterdruck in der Zuleitung zu erzeugen, der ausreicht, um das Fördermedium (in unserem Fall Wasser) anzusaugen. Diese Eigenschaft war schlussendlich ausschlaggebend, dass sich die Zahnradpumpe gegenüber den anderen Lösungen durchgesetzt hat. Als Alternativen wurden Ventile und Kreiselpumpen angedacht. Die Kreiselpumpe konnte trotz dem geringeren Stromverbrauch und geringerem Geräuschpegel nicht durchsetzen. Noch weniger Strom und Lärm verursachen Ventile, die aber auf gespeicherte Energie angewiesen sind. Entweder durch Druck im Wassertank oder durch Erhöhung des Tankes über den Ausfluss. Auf Grund der Wahl der Zahnradpumpe kann ein 4 mm Schlauch zur Förderung des Wassers genutzt werden, der nahezu beliebig verlegt werden kann. Ebenso ist das Gefäß frei wählbar, für das Testsystem haben wir eine 1,5 Liter Flasche verwendet. Im Dauerbetrieb wird ein fünf-Liter-Weinballon verwendet. In der Pflanze wird der Schlauch mit einem durchbohrten Kantholz in der Pflanzenerde befestigt.

### B. Gehäuse

Das Gehäuse wurde so konzipiert, dass es möglichst klein ist aber genügend Platz für die Elektronik bietet. Im Gehäuse verbaut sind:

- ein LCD-Display zur Anzeige der Gießparameter (aktuelle Feuchtigkeit, Gießintervall ...)
- zwei Taster zur Steuerung der Displayanzeige und zum manuellen Gießen
- der Photowiderstand zur Messung der Helligkeit (platziert zwischen den Tastern)
- Stromschluss auf der linken Seite
- Ausgänge für die Pumpe und den Feuchtigkeitssensor auf der rechten Seite

Das Gehäuse wurde in FabLab Erlangen mit einem Lasercutter gefertigt. Für das Design des Gehäuses wurde der BoxMaker<sup>2</sup> verwendet. Das Gehäuse besteht aus grünen 3 mm dicken Acrylglass.

<sup>1</sup>[www.ksb.com/Kreiselpumpenlexikon\\_de/Pumpenlexikon/1563382/verdraengerpumpe.html](http://www.ksb.com/Kreiselpumpenlexikon_de/Pumpenlexikon/1563382/verdraengerpumpe.html)

<sup>2</sup><http://boxmaker.connectionlab.org/>



Abbildung 2. Gehäuse Frontansicht

### C. Elektronik

Im folgenden wird der Aufbau der Version 1.0 der Arduino Variante vorgestellt. Während des Aufbaus, vor allem aber während der Testphase sind ein paar Probleme aufgetreten die dazu geführt habe, dass eine neue Version 1.1 erstellt werden musste. Diese neue Version ist jedoch aus zeitlichen Gründen noch nicht komplett aufgebaut. Ein neues Layout wurden bereits erstellt und die Software ist so angepasst worden, dass diese ohne größere Änderungen übernommen werden kann. Daher wird im folgenden die Funktionsweise auf Basis der Version 1.0 erläutert, an gegebener Stelle wird auf die Anpassungen eingegangen, die bereits umgesetzt wurden bzw. noch in Planung sind.

1) *Sensorik:* Für den Helligkeitssensor wird ein einfacher Photowiderstand verwendet, der über einen Spannungsteiler an einem der Analogen Pins des Arduino Bords angeschlossen ist. Der Pin verfügt über einen 10 Bit AD Konverter und gibt demnach einen Integerwert von 0 - 1023 zurück. Der Bodenfeuchtigkeitssensor bestimmt den Wassergehalt des Bodens über eine Widerstandsmessung zwischen den zwei Zinken einer Messgabel. Je mehr Wasser im Erdreich vorhanden ist, desto kleiner ist der gemessene Widerstand im Boden.

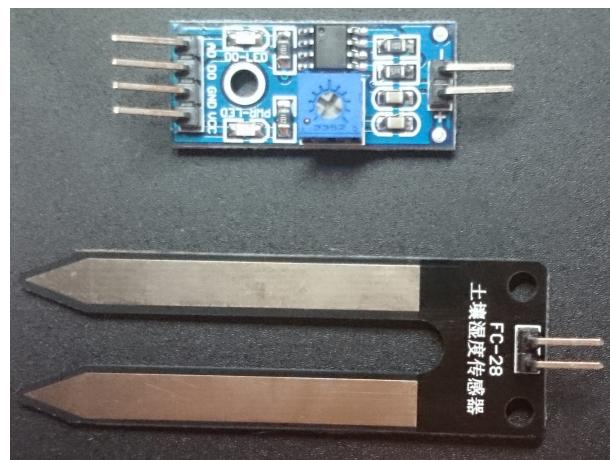


Abbildung 3. Feuchtigkeitssensor mit Vorschaltung

Der Feuchtigkeitssensor benötigt keine weiteren Schaltelemente, da er über eine Vorschaltung verfügt in der ein Spannungsteiler bereits verbaut ist. Abbildung 3 zeigt den verwendeten Sensor und die Vorschaltung. Es ist möglich sowohl den Analogen Wert, oder ein Digitales Signal auszuwerten. Das Digitale Signal liefert einen Nullwert solange ein Grenzwiderstand nicht überschritten wird. Über ein Potentiometer (siehe Abbildung 3) lässt sich diese Grenze einstellen. Wegen des schlechten Zugangs zum Potentiometer im eingebautem Zustand wird der Digitale Output nicht verwendet, sondern der Analoge Messwert selbst ausgewertet und mit einer Variablen im Mikrocontroller abgeglichen.

*Anpassung zu Version 1.1:* Leider zeigte sich, dass nach nur 48 Stunden Dauermessung die Gabel erhebliche Korrosion erlitten hat, Abbildung 4 zeigt dies deutlich. Die Vorschaltung sieht keine Abschaltung des Messprozesses vor noch eine Umpolung der Gabel. Deswegen muss die gesamte Vorschaltung stromlos geschaltet werden, um das Auflösen des Sensors zu verlangsamen. Hierfür haben wir für die Version 1.1, eine Transistorschaltung für den Sensor eingefügt. Über diese Schaltung wird die Vorschaltung des Feuchtigkeitssensor stromlos geschaltet und der Sensor ist nur für eine Messung unter Strom. Eine Umpolung der Messgabel ist auch nicht möglich, dies kann jedoch relativ einfach dadurch gelöst werden, indem der Sensor alle paar Wochen *manuell* umgedreht wird. Hierfür muss lediglich die Messgabel vom Verbindungskabel abgesteckt werden und um 180° gedreht wieder verbunden werden.

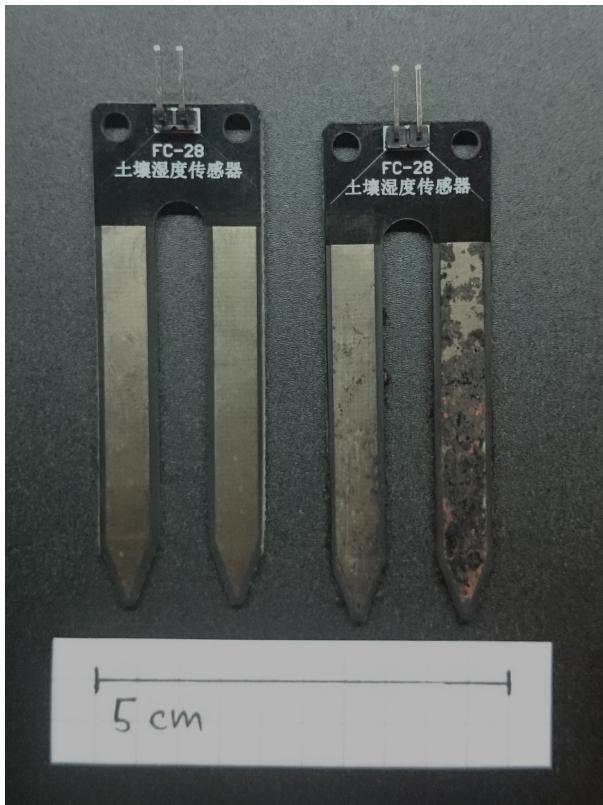


Abbildung 4. Vergleich neuer Sensor und Sensor mit 48h Dauerbetrieb

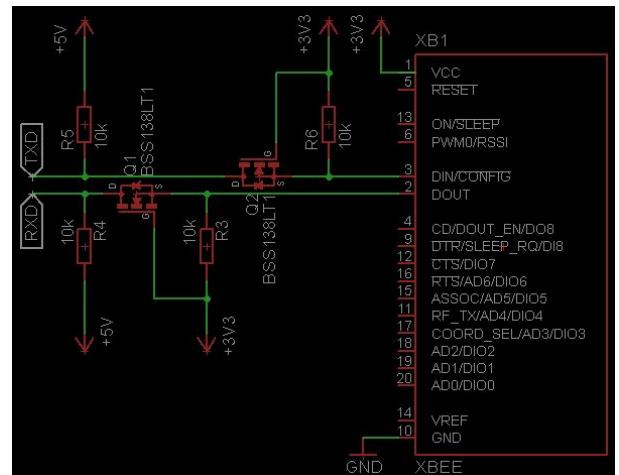


Abbildung 5. Pegelwandler mit Small-Signal-Transistor BSS138W

*2) Kommunikation:* Jede Pflanze braucht unterschiedlich viel Wasser. Genauso hat die Zusammensetzung der Erde einen Einfluss auf den gemessenen Widerstand des Feuchtigkeitssensors. Ziel der Kommunikation soll es deswegen sein, dass Gießsystem im laufenden Betrieb zu kalibrieren. Wir haben uns für eine drahtlose Kommunikation, auf Basis des XBee Standard, entschieden. Das XBee Modul wurde direkt, d.h. ohne Verwendung eines *Arduino XBEE-Shields*, mit dem Arduino verbunden. Hierfür ist es notwendig einen Pegelwandlung von 5 V auf 3,3 V vorzunehmen, da die Logik-Pins des XBee Moduls nicht mit 5 V betrieben werden können. Abbildung 5 zeigt die Umsetzung der Pegelschaltung. Die Labels TXD und RXD repräsentieren die jeweiligen Anschlüsse am Arduino Micro. In dieser Schaltung lassen sich drei Zustände erkennen<sup>3</sup>

- Wird keiner der beiden Seiten (Mikrocontroller bzw. XBee) auf GND gezogen, so werden die Eingänge durch ihre Pull-Up Widerstände auf 5 V bzw. 3,3 V gezogen. Dadurch ist das Spannungsgefälle *Gate-zu-Source* gleich Null, da an beiden Anschlüssen 3,3 V anliegen. Der Transistor ist nicht leitend.
- Wird am XBee (DOUT) ein LOW Signal angelegt, so steigt das Spannungsgefälle *Gate-zu-Source* an und der Transistor wird leitfähig, was dazu führt das auch die Pins am Mikrocontroller (RXD) auf LOW gezogen werden.
- Wird hingen auf der Seite des Mikrocontroller (TXD) ein LOW Signal angelegt, so wird über die Diode im Transistor das Spannungspotenzial der Source solange reduziert, bis das Spannungsgefälle *Gate-zu-Source* groß genug wird, damit der Transistor leitfähig wird. Sobald dieser Grenzwert überschritten wird, wird auch DOUT am Xbee auf LOW gezogen.

Bei der Pegelschaltung ist uns in Version 1.0 ein Fehler unterlaufen, weswegen wir keine Verbindung mit einem Computer aufbauen konnten. Dieser Fehler ist für die Version 1.1 behoben.

Die zweite Schnittstelle mit dem Menschen wird über ein Display ermöglicht. Hier lässt sich über einen Taster

<sup>3</sup><http://www.adafruit.com/datasheets/an97055.pdf>; Seite 10f

die einzelnen Variablen mit ihren aktuellen Werten Überprüfen. Bei dem Display handelt es sich um ein 2 Zeiliges LCD-Display mit jeweils 16 Zeichen. Die Kommunikation zwischen Display und Mikrocontroller wird über eine I2C-Schnittstelle bewerkstelligt. Hierfür haben wir die frei verfügbare LiquidCrystal\_I2C Libary von fdebrabander verwendet.<sup>4</sup>

3) Stromversorgung: Eine Stromversorgung über USB ist nicht möglich, da die Pumpe in Volllast 12 V und außerdem 2,8 A benötigt. Deswegen muss auf eine leistungsfähigere Energiequelle gesetzt werden. Wir entschieden uns für Netzteile mit 12 V Ausgangsspannung um die Pumpe direkt anschließen zu können. Um den Strom der Pumpe zu begrenzen haben wir einen  $5\ \Omega$  Lastwiderstand in Reihe geschaltet. Dies führt zu geringerer Leistungsaufnahme und deutlichen Geräuschminderung. Die Ansteuerung der Pumpe über dem Mikrocontroller ist über eine Transistorschaltung gelöst (Ziffer 1 in Abbildung 6). Es ist zu überlegen, den Transistor und den Mikrocontroller über eine Schutzdiode über die Anschlüsse VENTIL-1 und VENTIL-2 vor Überspannung zu schützen, die beim Abschalten der Pumpe auftreten können.

#### D. Programm

Die Aufgaben des Mikrocontrollers sind die folgenden:

- Messung der Feuchtigkeit
- Messung der Helligkeitssensor
- Vergleich der Messwerte mit den festgelegten Grenzen
- Ansteuerung der Pumpensystem
- Ausgabe der wichtigen Variablen auf dem Display über Taster
- Manuelles Gießen über Taster
- Display Hintergrundbeleuchtung abschalten nach 60 Sekunden
- Übertragen/Empfangen von Einstellungen über xBee Verbindung

In Abbildung 8 ist der Ablauf des Arduinoprogramms in der Loop-Methode zu sehen. Wie in Abschnitt II-C1 beschrieben, führt eine dauerhafte Messung der Bodenfeuchtigkeit zu einer schnellen Zersetzung des Sensors. Aus diesem Grund wird die Messung nur einmal in einem Intervall von ca. 4 Stunden durchgeführt. Für die Intervallabfrage in der methode messung() wird die Funktion millis() verwendet. Diese Funktion gibt einen unsigned Long Wert zurück, der der Laufzeit des Arduino in Millisekunden entspricht. Nach ungefähr 49 Tagen wird der Counter einen Überlauf erzeugen und beginnt von vorne zu zählen. Damit dieser Überlauf keine Auswirkungen auf das Gießsystem hat, wird die Abfrage des Messintervalls wie folgt ausgeführt.

```
if ((millis() - prevMessung) >= messInterval)
{
    aktFeuchtigkeit = analogRead(MOISTURE_A);
    aktHelligkeit = analogRead(PHOTO_PIN);
    neueMesswerte = true;
    prevMessung = millis();
}
```

<sup>4</sup><https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library>

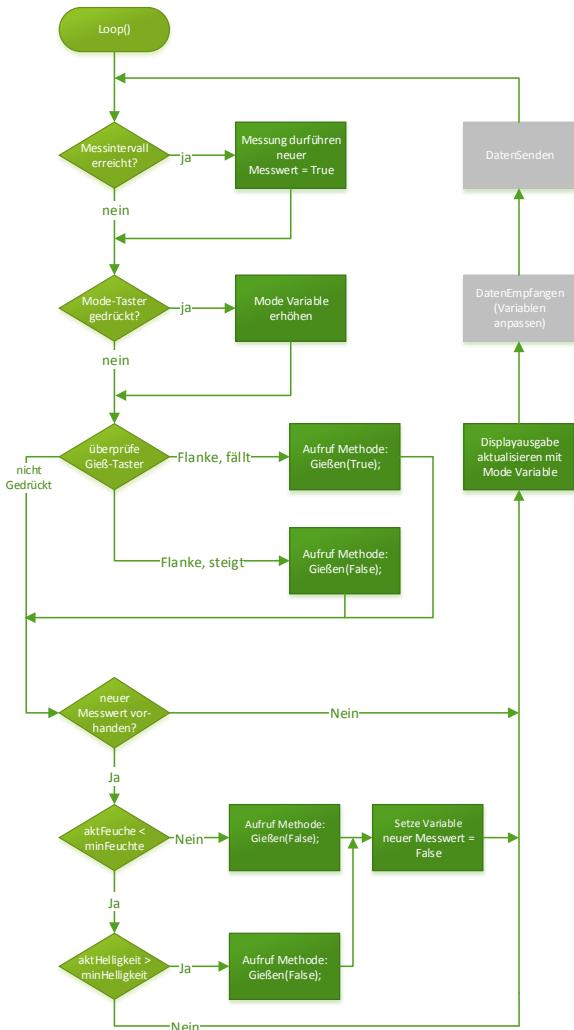


Abbildung 7. Programmablauf der Arduino Version

Beim ersten Durchlauf, also direkt nach dem Anschalten des Mikrocontrollers, wird das If-Bedingung immer als Wahr evaluiert und eine Messung wird durchgeführt und die Variable *prevMessung* wird auf die aktuelle Zeit eingestellt. Dadurch wird die If-Abfrage erst dann wieder Wahr, wenn die Zeit *messInterval* abgelaufen ist. Auch bei einem Überlauf der Funktion millis() wird diese Abfrage ein korrektes Ergebnis liefern, da die Funktion millis() einen unsigned Long zurück gibt. Bei allen Intervall bezogenen Abfragen im Programmcode wird dieses vorgehen verwendet. Anpassung zu Version 1.1: In dieser Abfrage wird nur die Messung alle 4 Stunden durchgeführt. Für die Version 1.1 muss in diesem Programmabschnitt noch die An- und Abschalten des Feuchtigkeitssensors durchgeführt werden. Außerdem ist es sinnvoll nach dem Anschalten des Sensors kurz zu warten, bevor eine Messung durchgeführt wird um Messfehler zu vermeiden.

Unabhängig davon, ob eine Messung ausgeführt wurde oder nicht, wir im Anschluss überprüft ob der Mode-Taster betätigt wurde. Wie in Abbildung 6 zu sehen ist, wurden die

Taster nicht über Kondensator entprellt, sondern per Software. Hierfür haben wir die Bounce2 Libary verwendet<sup>5</sup>. Wurde ein Knopfdruck registriert, so wird die Variable *Mode* um ein erhöht. Über die Modulberechnung wird sichergestellt, dass die Variable immer das Intervall von 0 bis ANZAHK\_MODE durchläuft. Der Zweite Taster dient zum manuellen Gießen der Pflanze. Solange der Taster gedrückt ist, wird die Pumpe angesteuert und gegossen.

```
mode = (mode + 1) % ANZAHL_MODE;
```

Als nächstes wird überprüft, ob ein neuer Messwert vorhanden ist. Dies wird über die Variabel *neueMesswerte* bewerkstelligt. Diese Variable wird gesetzt, sobald eine Messung durchgeführt wurde (siehe oben). Ist das nicht der Fall, so wird wie in Abbildung ?? zu sehen ist die Methode Gießen() mit dem Parameter False aufgerufen und die Variabel *neuerMesswert* auf false gesetzt. Damit wird verhindert, dass auf Grundlage des selben Messwertes erneut eine Entscheidung getroffen wurde. Sagt der Messwert hingegen aus, dass gegossen werden muss aber der Helligkeitssensor verhindert ein Gießen, dann wird die Variable *neuerMesswert* nicht zurückgesetzt. Der Aktuelle Messwert wird demnach solange abgeglichen, bis es hell genug ist, damit gegossen werden kann.

Abschließend in der Kette wird eine Methode zum anzeigen der Messwerte auf dem LCD-Display aufgerufen. Diese Zeigt die Einstellung an, die mit der Mode Variable korrespondiert (1 = akt. Bodenfeuchtigkeit, 2 = akt. Helligkeit, usw.). Die Ausgegraute Bausteine *DatenEmpfang* und *DatenSenden* sind in Version 1.0 noch nicht implementiert, da ein Fehler im Schaltkreis der Kommunikationsschnittstelle eine Kommunikation nicht möglich macht. Diese werden jedoch in Version 1.1 abgebildet.

### E. Kostenplan

Durch die geringen Kosten pro Entwicklungsstufe hatten wir genügend Geld um mehrere Iterationsstufen zu durchlaufen. Deswegen benötigten wir für die gesamte Entwicklung etwas über hundert Euro. Ein einzelnes Modul kann für etwa 45€ nachgebaut werden. Die Kosten setzen sich nach Tabelle II zusammen.

Nicht berücksichtigt sind das Wassergefäß, die zwei 4mm-Schlauchstücke und Befestigung in der Pflanze. Bei diesen handelt es sich um Reste oder Lagerfunde die von Wert von unter 1€ sind.

## III. SPARVERSION

Um Strom zu sparen, die Größe zu schrumpfen und Kosten günstiger zu werden, haben wir die „Sparversion“ entwickelt. In dieser Variante der Gießanlage wurde die Anlage auf das wesentlichste beschränkt, das Gießen. Sie wurde so konzipiert, dass sie genau auf eine Pflanze zugeschnitten ist. Sie kann nur durch erneutes Programmieren auf andere Pflanzen und Böden angelernt werden.

Tabelle II  
KOSTEN FÜR EINE GIESSANLAGE

Bauteil	Kosten	Bezugsquelle
Arduino Nachbau	ca. 3 €	ebay
LCD-Display	ca. 5 €	ebay
Bodensensoren	ca. 2 €	ebay
Zahnradpumpe	2,95 €	Pollin
XBee	23,55 €	Reichelt
Hauptplatine	ca. 6 €	FabLAB
Gehäuse	ca. 4 €	FabLAB
Gesamt:	ca 45 €	

### A. Aufbau

In dieser Version wird auf das Display und die Kommunikation verzichtet. Dadurch wird viel der Verkabelung gespart, außerdem lässt sich die Hauptplatine deutlich kleiner gestalten. Der Wegfall zu weniger Platzbedarf des Systems führt und damit in ein kleineres Gehäuse passt.

### B. Elektronik

Durch das wegfallen der XBee Platine und deren Beleuchtung wird das 3,3 V Netz nicht mehr benötigt. Dadurch reduziert sich die Größe der Hauptplatine auf 55mm × 32mm. Dies entspricht nicht nicht mal der Hälfte der Fläche der Version 1.1 mit 56mm × 65mm.

### C. Programmierung

Um weiter Stromsparen zu können wurde diese Version nicht mit Arduino, sondern mit C geschriebenen Programm programmiert. Dies ermöglicht die Ausnutzung der Sleep Modi und die Interrupts des ATMega328. Dadurch befindet sich der „Arduino Nano“ hauptsächlich im Schlafmodus und verbraucht deutlich weniger Energie. Die Gießeinstellungen müssen auf Grund der fehlenden Kommunikation und Eingabemöglichkeiten über die Programmierung festgelegt werden.

1) *Programmierwerkzeuge*: Durch das Wegfallen der Arduino Bootloaders, kann die Hardware nicht mehr per USB programmiert werden. Mit Hilfe des AVRISP mkII<sup>6</sup> wird der Microcontroller direkt mit den Binärkode beschrieben.

2) *Logik*: Der Ablaufplan (Abbildung 8) der Sparversion wurde vereinfacht. Das Programm läuft linear immer wieder durch. Die meiste Zeit verbringt das System in der „Sleep-Schleife“ in dem nach jedem Timer-Überlauf der Zähler um eins dekrementiert wird. Nach ungefähr 3 Stunden ist der Zähler klein genug und das Programm fährt die Sensoren hoch, indem er sie über den Transistor mit Strom versorgt. Dann erfolgt die Messung der Sensorwerte. Anhand der Sensorwerte wird nun entschieden ob gegossen wird. Nachdem Gießen wird der Zähler wieder hochgestellt und das Programm beginnt von vorne.

<sup>5</sup><http://playground.arduino.cc/Code/Bounce>

<sup>6</sup>[www.atmel.com/tools/avrispmkii.aspx](http://www.atmel.com/tools/avrispmkii.aspx)

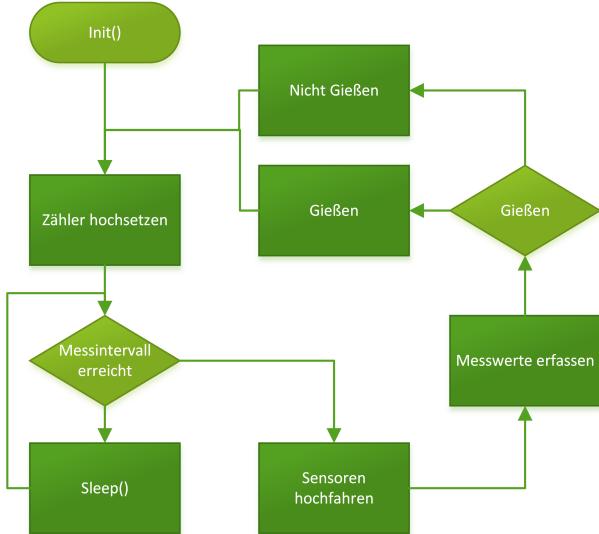


Abbildung 8. Ablaufplan des Programms der Sparversion

3) *Kalibrierung der Sensoren:* Da der Bodensensoren für jede Pflanze, jeden Boden und teilweise jede Einstichstelle im Boden neu kalibriert wird, wurde über diesen Prozess Gedanken gemacht. Es wurde entschieden dies manuell zu machen. Dazu wird der Sensor in den trockenen Boden gesteckt und über die Vorschaltung mit Strom versorgt. Die Spannungsquelle an der Vorschaltung auf 5 V gestellt und mit einem Multimeter die Spannung zwischen analog Ausgang und Grund gemessen. Nun wird solange gegossen bis die Erde feucht genug erscheint, die dazu passende Spannung wird notiert. Mit Hilfe der Formel  $\frac{\text{Messwert}}{5,0\text{V}} \times 1023$  wird der Wert errechnet, der in das Programm in die Definiton von

```
#define FEUCHTE
eingetragen wird.
```

#### D. Kostenplan

Tabelle III  
KOSTEN FÜR EINE SPARVERSION GIESSANLAGE

Bauteil	Kosten	Bezugsquelle
Arduino Nachbau	ca. 3 €	ebay
Bodensensoren	ca. 2 €	ebay
Zahnradpumpe	2,95 €	Pollin
Gehäuse	1,50 €	Pollin
Hauptplatine	ca. 5,5 €	FabLAB
Gesamt:	ca 15 €	

Allein auf Grund des fehlenden XBee-Moduls halbiert sich der Preis der Anlage. Dazu kommt das fehlende Display, das kleinere Gehäuse und günstigere Hauptplatine. So ist der Nachbau der Sparversion ca. 15 € teuer. In Tabelle III sind die Kosten nochmal zusammen getragen.

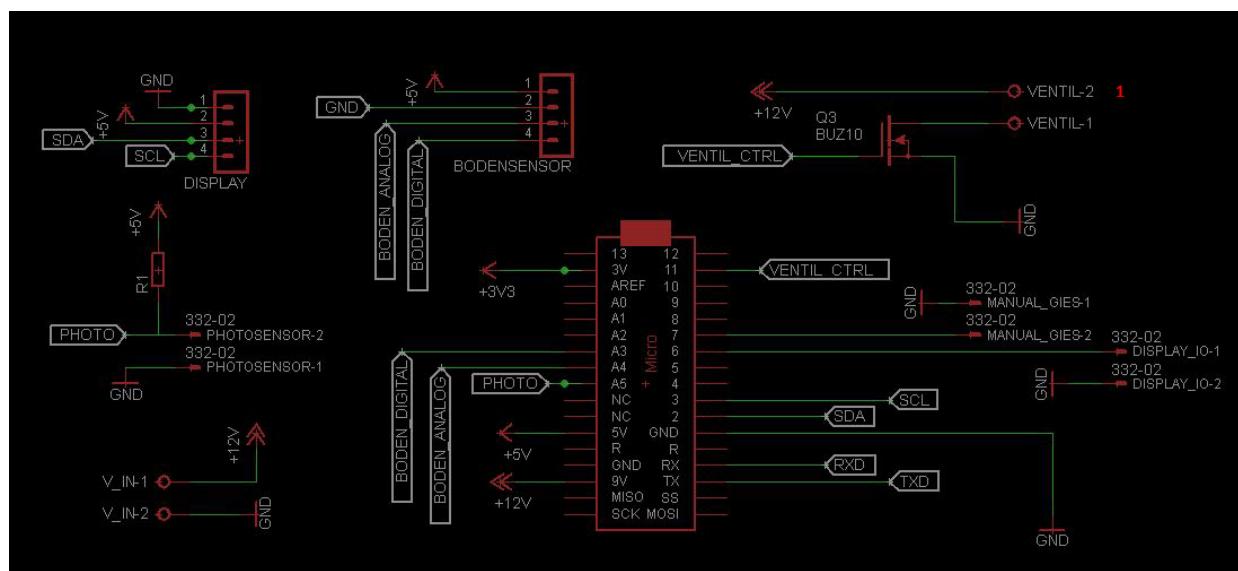


Abbildung 6. Schaltpaln V1.0 mit Arduino Micro