

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 „absaugen“ lassen und hoffen, dass sie sich das Wasser einteilen? Beides keine zufriedenstellende Lösung. Wieso lässt man die Pflanze sich nicht selbst gießen? Aus diesen Überlegungen ist die Idee entstanden eine Pflanze mit Sensoren auszustatten und so deren Zustand zu überwachen und über diesen zu informieren. Im Folgenden wird der Aufbau zweier Lösungen vorgestellt.

Der erste Ansatz basiert auf dem Arduino-System, welches einen einfachen Einstieg in Umgang mit Mikrocontroller ermöglicht. Der zweite Ansatz basiert auf ähnlicher Hardware, jedoch ohne die Verwendung der Arduino-Software. Dadurch wird die Verwendung von Sleep-Modi und Interrupts möglich was eine Low-Energy Variante des Gießsystems ermöglicht.

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, die Luftqualität, der Nährstoffgehalt des Bodens sowie die Temperatur. Diese nicht abschließende Aufzählung zeigt, dass eine Eingrenzung der zu untersuchenden Faktoren notwendig ist, um das Projekt in der gegebenen Zeit fertigzustellen. Auch hinsichtlich des Einsatzgebietes gibt eine Vielfalt von unterschiedlichen Möglichkeiten wie z.B. Einpflanzenbetrieb oder Mehrpflanzenbetrieb sowie In- oder Outdoor-Betrieb. 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 in Abhängigkeit des Wassergehalts bzw. Feuchtigkeitgrades der Erde.
- Das System soll für eine alleinstehende Pflanze aufgebaut werden.
- Das Einsatzgebiet wird im Innenraumbereich sein, dadurch fallen Einschränkungen bzgl. der 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 Grund energetischer Optimierung auf die Kommunikation verzichtet wird. Hier wird die Einstellung direkt im Code vorgenommen.

Weitere Ideen wie z.B. eine autarke, batteriebetriebene Lösung, ein Mehr-Pflanzenbetrieb oder ein System für den Garten wurden zwar diskutiert, aber wegen dem erhöhten Zeitaufwand

hinsichtlich weiterer zu lösender Problemstellungen und begrenzten Budgets nicht weiter verfolgt. Im folgenden wird die *Arduino-Variante* ausführlich vorgestellt. Die *Sparversion* wird im Anschluss erläutert. Da viele der Funktionen grundsätzlich identisch sind und übernommen werden können, 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 Cinch-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 *Sensorik*, *Kommunikation* und *Stromversorgung* eingehen.
- Anschließend wird das Programm für den Mikrocontroller beschrieben.
- Abschließend wird ein Kostenplan vorgestellt.

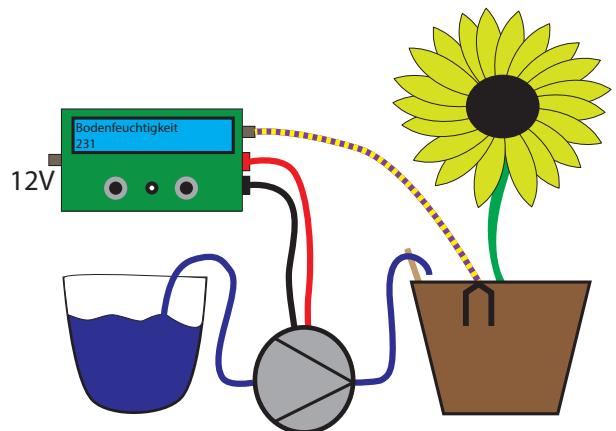


Abbildung 1. Schematischer Aufbau der Gießanlage

A. Wassertransport

Um den Einsatzbereich so flexibel wie möglich zu gestalten, haben wir uns für ein Pumpensystem entschieden. Dadurch ist die Anordnung der Pflanze zum Wassertank nicht relevant. Der

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 €

hiermit einhergehende, erhöhte Strombedarf ist zu verkraften, da das System nicht auf Batterien angewiesen ist.

Für den Wassertransport haben wir uns für eine Zahnradpumpe entschieden. Die Zahnradpumpe gehört zur Gruppe der rotierenden Verdränger Pumpen¹. 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 sich trotz dem geringeren Stromverbrauch und geringerem Geräuschpegel nicht durchsetzen. Das Ventil benötigt den geringsten Strom und ist sehr leise, jedoch ist es 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 der Wassertank frei wählbar. Für das Testsystem haben wir eine 1,5 Liter Flasche verwendet. Im Dauerbetrieb wird ein fünf-Liter-Weinballon verwendet. An der Pflanze wird der Schlauch mit einem durchbohrten Kantholz in der Pflanzenerde befestigt.

B. Gehäuse

Das Gehäuse, wie es in Abbildung 2 zu sehen ist, wurde so konzipiert, dass es möglichst klein ist aber dennoch 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



Abbildung 2. Gehäuse Frontansicht

BoxMaker² verwendet. Das Gehäuse besteht aus grünen 3 mm dicken Acrylglass.

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 Probleme aufgetreten, die dazu geführt haben, dass eine neue Version 1.1 erstellt werden musste. Diese neue Version ist jedoch aus zeitlichen Gründen noch nicht komplett aufgebaut. Ein neues Layout wurde 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.

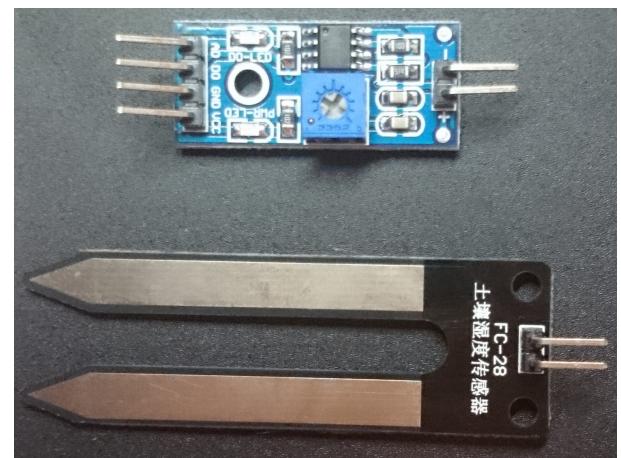


Abbildung 3. Feuchtigkeitssensor mit Vorschaltung

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

¹www.ksb.com/Kreiselpumpenlexikon_de/Pumpenlexikon/1563382/verdraengerpumpe.html

²<http://boxmaker.connectionlab.org/>

demnach einen Integerwert von 0 - 1023 zurück. Der Bodenfeuchtigkeitssensor bestimmt den Wassergehalt des Bodens über eine Widerstandsmessung zwischen den zwei Zinken der Messgabel. Je mehr Wasser im Erdreich vorhanden ist, desto kleiner ist der gemessene Widerstand.

Der Feuchtigkeitssensor benötigt keine weiteren Schaltelemente, da er über eine Vorschaltung verfügt in der bereits ein Spannungsteiler integriert ist. Abbildung 3 zeigt den verwendeten Sensor und die Vorschaltung.

Es ist möglich sowohl den analogen Wert als auch 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 Schaltwert 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 durchgehendem Messens die Gabel erhebliche Korrosionsschäden aufweist, Abbildung 4 zeigt dies deutlich. Die Vorschaltung sieht weder eine Abschaltung des Messprozesses, noch eine Umpolung der Gabel-Polarität vor. Folglich muss die gesamte Vorschaltung stromlos geschaltet werden, um das Auflösen des Sensors möglichst zu verzögern. 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 nur für eine Messung mit Strom versorgt. Eine

Umpolung der Messgabel ist hiermit nicht möglich. Dies kann jedoch relativ einfach gelöst werden, indem der Sensor alle paar Wochen *manuell* umgepolzt wird. Hierfür muss lediglich die Messgabel vom Verbindungsleitung abgesteckt, um 180° gedreht und wieder verbunden werden.

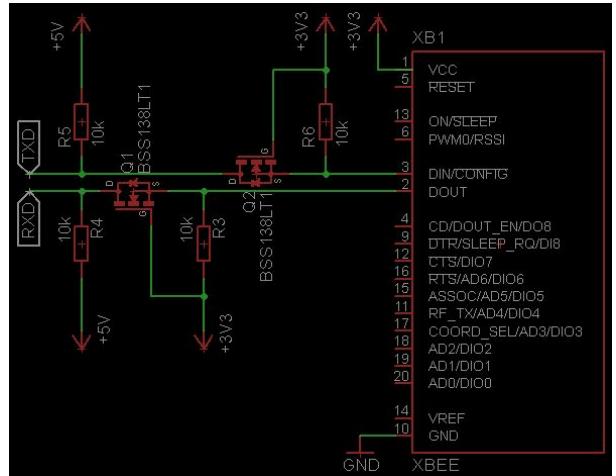


Abbildung 5. Pegelwandler mit Small-Signal-Transistor BSS138W

2) Kommunikation: Ziel der Kommunikation ist es, das Gießsystem im laufenden Betrieb kalibrieren zu können. 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 war es notwendig eine Pegelschaltung von 5 V auf 3,3 V vorzunehmen, da die Logik-Pins des XBee-Moduls nicht mit 5 V des Arduinos betrieben werden können. Abbildung 5 zeigt die Umsetzung der Pegelschaltung. Die Lables *TXD* und *RXD* repräsentieren die jeweiligen Anschlüsse am Arduino Micro. In dieser Schaltung lassen sich drei Zustände erkennen³:

- 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 Anschläßen 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 DIN 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.

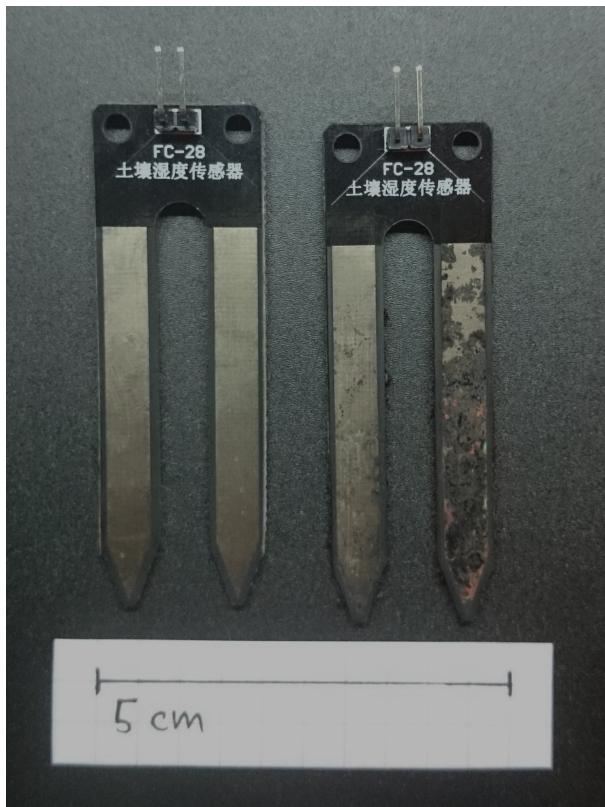


Abbildung 4. Vergleich neuer Sensor und Sernsor mit 48h Dauerbetrieb

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

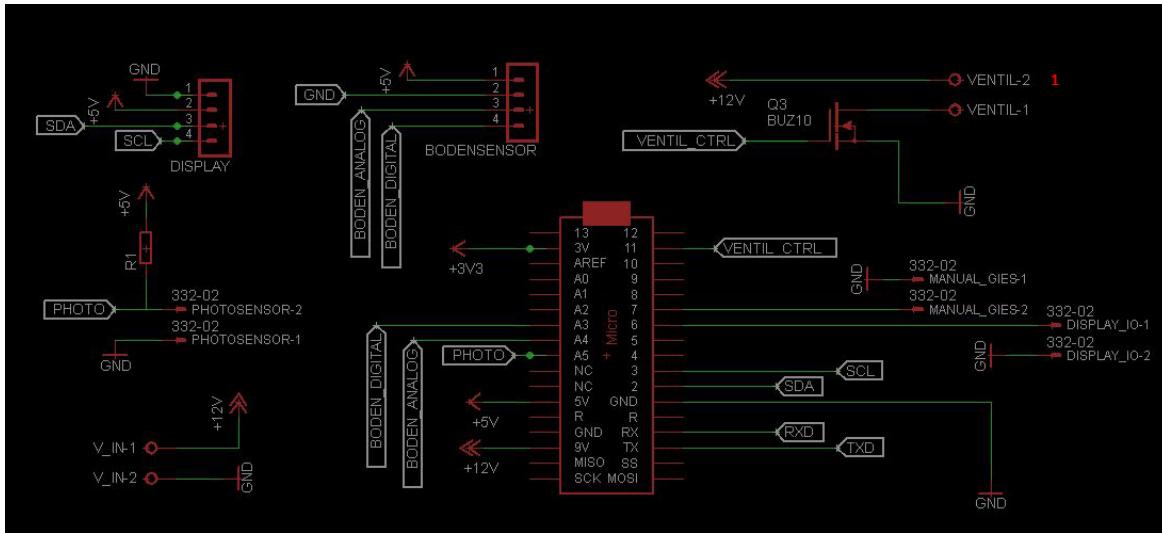


Abbildung 6. Schaltplan V1.0 mit Arduino Micro

Die zweite Schnittstelle zum Anwender ist das Display. Hier lässt sich über einen Taster 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.⁴

3) Stromversorgung: Eine Stromversorgung über USB ist nicht möglich, da die Pumpe unter Volllast 12 V und 2,8 A benötigt. Deswegen muss auf eine leistungsfähigere Energiequelle gesetzt werden. Um den Strom der Pumpe zu begrenzen, wurde ein $5\ \Omega$ Lastwiderstand in Reihe geschaltet. Dies führt zu geringerer Leistungsaufnahme und einer 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 Helligkeit
- Vergleich der Messwerte mit den festgelegten Grenzen
- Ansteuerung des Pumpensystems
- 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 7 ist der Ablauf des Arudino-Programms 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 Intervall-Abfrage in der Methode *messung()* wird die Funktion *millis()* verwendet. Diese Funktion gibt einen unsigned Long Wert zurück, welcher 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();
}
```

Beim ersten Durchlauf, also direkt nach dem Anschalten des Mikrocontrollers, wird die If-Bedingung immer als *wahr* evaluiert und eine Messung wird durchgeführt. In die Variable *prevMessung* wird 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 die Messung lediglich alle 4 Stunden durchgeführt. Für die Version 1.1 muss in diesem Programmabschnitt noch das An- und Abschalten des Feuchtigkeitssensors durchgeführt werden. Außerdem ist es, um Messfehler zu vermeiden, sinnvoll, nach dem Anschalten des Sensors kurz zu warten bevor eine Messung durchgeführt wird.

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

Unabhängig davon, ob eine Messung ausgeführt wurde oder nicht, wird im Anschluss überprüft, ob der Mode-Taster betätigt wurde. Wie in Abbildung 6 zu sehen ist, wurden die Taster nicht über einen Kondensator entprellt, sondern per Software. Hierfür haben wir die Bounce2 Libary verwendet⁵. Wurde ein Knopfdruck registriert, so wird die Variable *Mode* inkrementell erhöht. Über die Modulberechnung wird sichergestellt, dass die Variable immer das Intervall von *0 bis ANZAHL_MODE* durchläuft. Der Zweite Taster dient zum manuellen Gießen der Pflanze. Solange der Taster gedrückt ist, wird die Pumpe angesteuert und es wird gegossen.

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

Als nächstes wird überprüft, ob ein neuer Messwert vorhanden ist. Dies wird über die Variable *neuerMesswert* bewerkstelligt. Diese Variable wird gesetzt, sobald eine Messung durchgeführt wurde (siehe oben). Ist das nicht der Fall, so wird wie in Abbildung 7 zu sehen ist, die Methode *Gießen()* mit dem Parameter *false* aufgerufen und die Variable *neuerMesswert* auf *false* gesetzt. Damit wird verhindert, dass auf Grundlage des selben Messwertes erneut eine Entscheidung getroffen wird. 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 wird in der Logikkette eine Methode zum Anzeigen der Messwerte auf dem LCD-Display aufgerufen. Diese zeigt die Einstellung an, welche mit der Mode-Variable korrespondiert (1 = akt. Bodenfeuchtigkeit, 2 = akt. Helligkeit, ...). Die 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 Mittel um mehrere Iterationsstufen zu durchlaufen. Dementsprechend benötigten wir für die gesamte Entwicklung etwas über einhundert 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 die Befestigung in der Pflanze. Bei diesen handelt es sich um Reste oder Lagerfunde im Wert von unter 1 €.

III. SPARVERSION

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

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

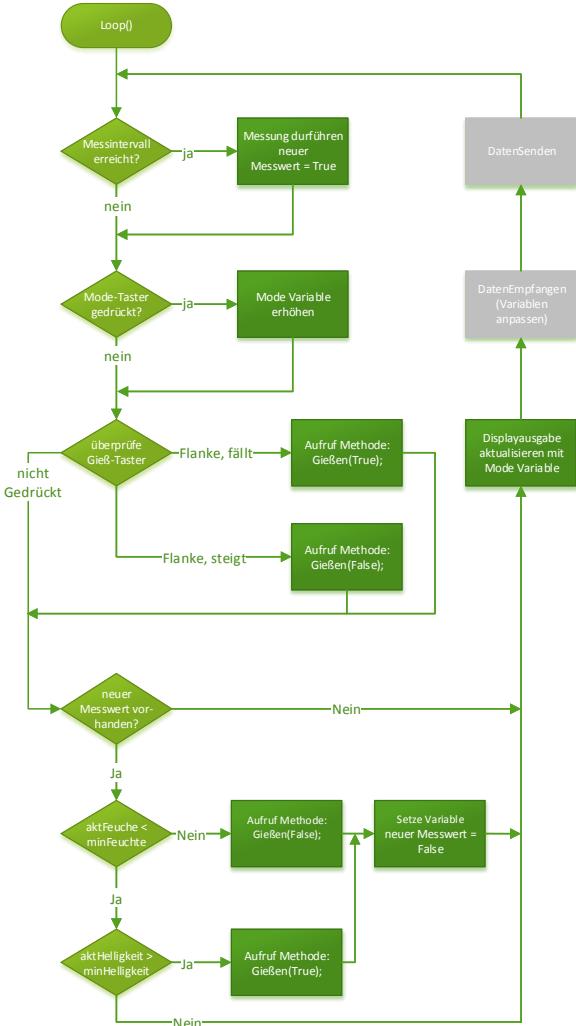


Abbildung 7. Programmablauf der Arduino Version

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 ein Großteil der Verkabelung eingespart, außerdem lässt sich die Hauptplatine deutlich kleiner gestalten. Dies führt zu geringerem Platzbedarf des Systems und folglich zu geringeren Gehäuseabmessungen.

B. Elektronik

Durch das Wegfallen der XBee Platine und deren Beschaltung wird das 3,3 V Netz nicht mehr benötigt. Die Größe der Hauptplatine reduziert sich auf 55mm × 32mm. Dies entspricht weniger als der Hälfte der Fläche der Version 1.1 mit 56mm × 65mm.

C. Programmierung

Um dem Stromverbrauch weiter zu senken wurde diese Version nicht mit Arduino, sondern mit C geschriebenen Programm programmiert. Dies ermöglicht die Nutzung der Sleep-Modi und der 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 bei der Programmierung festgelegt werden.

1) Programmierwerkzeuge: Durch das Wegfallen der Arduino-Bootloaders, kann die Hardware nicht mehr per USB programmiert werden. Mit Hilfe des AVRISP mkII⁶ kann der Mikrocontroller über die ISP-Schnittstelle mit den Binärkode beschrieben werden.

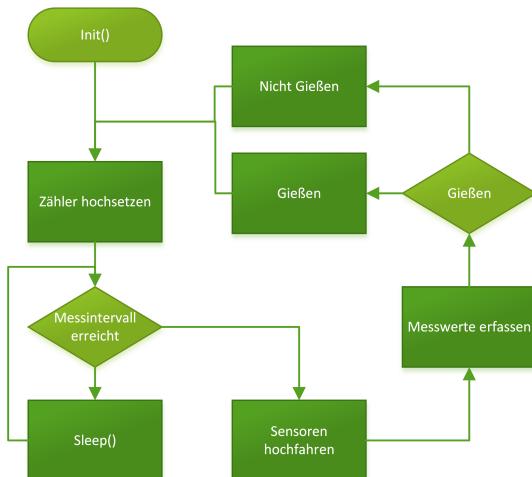


Abbildung 8. Ablaufplan des Programms der Sparversion

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 drei 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 der Bedarf besteht zu Gießen. Nachdem Gießen wird der Zähler wieder hochgestellt und das Programm beginnt von vorne.

3) Kalibrierung der Sensoren: Es wurde auch berücksichtigt, dass der Bodensensor für jede Pflanze bzw. jeden Boden und teilweise jede Einstichstelle im Boden neu kalibriert werden muss. Es wurde entschieden dies manuell durchzuführen.

⁶www.atmel.com/tools/avrispmkii.aspx

Dazu wird der Sensor in den trockenen Boden gesteckt und über die Vorschaltung mit Strom versorgt. Die Spannungsquelle an der Vorschaltung wird auf 5 V eingestellt und die Spannung zwischen analogen Ausgang und Grund mit einem Multimeter gemessen. Nun wird solange gegossen bis die Erde feucht genug ist. Die gemessene 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 `#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 nicht vorhandene Display, das kleinere Gehäuse und die günstigere Hauptplatine. So betragen die Kosten der Sparversion ca. 15 €. Ein Überblick über die respektiven Kosten gibt Tabelle III.

IV. WEITERENTWICKLUNG

A. Konfigurationstool

Um die Daten der Gießanlage auslesen und die Konfiguration auf eine Pflanze vornehmen zu können, fehlt noch das PC-Programm. Dieses Programm soll über eine Oberfläche die Einstellungen der Gießanlage(-n) anzeigen und diese auf die jeweilige Pflanze, Boden- und LichtVerhältnisse anpassen können. Es gibt erste Ansätze, die jedoch noch am Anfang ihrer Entwicklung stehen.

B. Kapazitive Bodenfeuchtigkeitsmessung

Das größte Problem vor dem wir stehen, ist der sich durch Elektrolyse zersetzende Bodenfeuchtigkeitssensor. Durch das selteneren Messen und das Umpolen lässt sich die Lebensdauer des Sensors deutlich verlängern, aber die Selbstzerstörung nicht aufhalten. So nimmt die Leitfähigkeit des Sensors mit der Zeit ab und die Messwerte verringern sich über die Zeit. Außerdem besteht der Sensor aus einer verzinkten Kupferplatine und durch die Korrosion werden Kupferionen frei gesetzt. Diese werden von den Pflanzen aufgenommen. Falls es sich um Nutzpflanzen zum Verzehr handelt, gelangen die giftigen Kupferionen in den menschlichen Körper. Deswegen wurde hier vorgeschlagen einen kapazitiven Sensor zu verwenden. Bei einem kapazitiven Sensor berührt kein Metall die Pflanzerde, so kommt es zu keiner Korrosion. Dafür muss eine Kapazität gemessen werden, was schaltungstechnisch schwer zu realisieren ist⁷.

⁷<http://www.dl8nci.de/lc-meter-001.html>