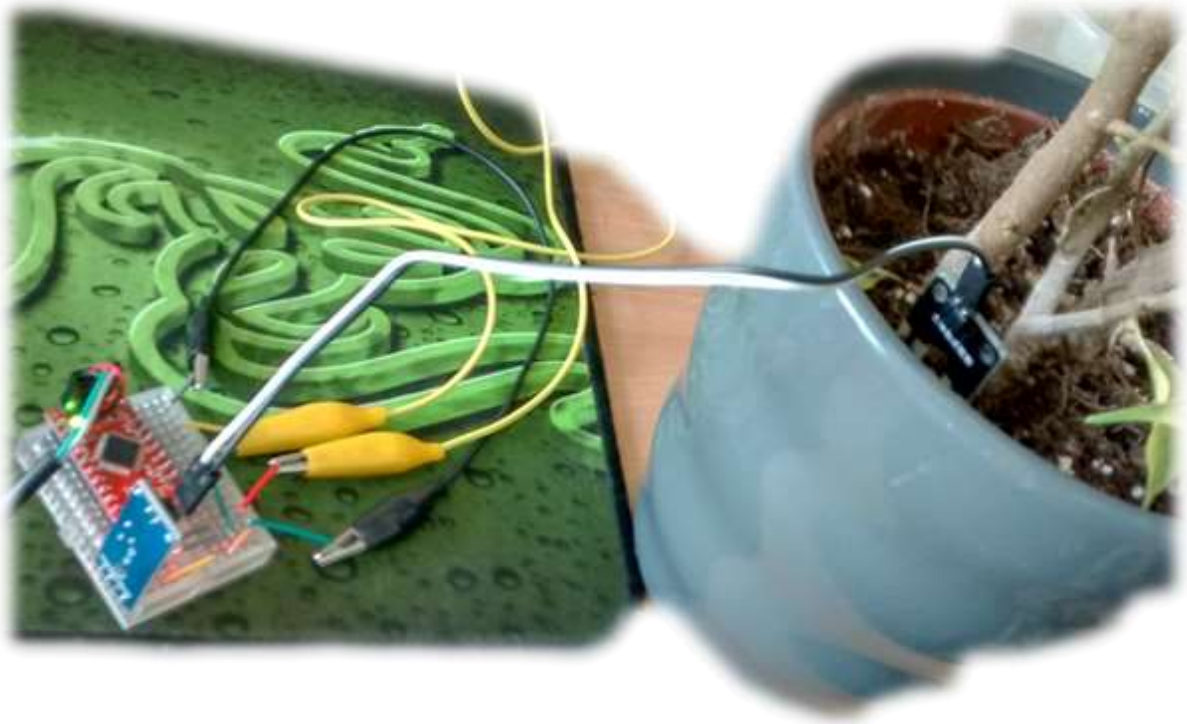
JuFo 2016

Intelligente Pflanzenbewässerung



Sven Hartmann, 17 Jahre

Thanh Huy Duong, 17 Jahre

Janik Steegmüller, 17 Jahre

Gewerbliche Schule Tübingen, TG 11/2

Projektbetreuer: Michael Hallmann

Martin Ruckh

Kurzfassung

Gibt es eine Möglichkeit, eine Pflanze, egal welche, mit der genau für sie richtigen Menge an Wasser zu versorgen, und dies vollautomatisch? Mit dieser Fragestellung haben wir unser diesjähriges Projekt begonnen. Mit Hilfe von verschiedenster Technik wie Arduinos, Feuchtesensoren und Pumpen haben wir versucht einen Mechanismus zu entwickeln, der dies möglich machen könnte. Aufgrund von verschiedenen Erkenntnissen ist uns bewusst geworden, dass dieses Projekt kein einfaches wird. Problematiken ergaben sich beim Timing der verschiedenen Ereignisse wie dem Messen und Gießen, aber auch bei der Hardware wie den Messorganen an sich. Ein Beispiel hierfür sind die Sensoren, die schneller als erwartet korrodierten. Nach langer Überlegung kamen wir zum Entschluss, unser Projekt aufzuteilen. Dieses Jahr beschäftigen wir uns mit der korrekten Bewässerung und Messung der Feuchtigkeit bei Pflanzen.

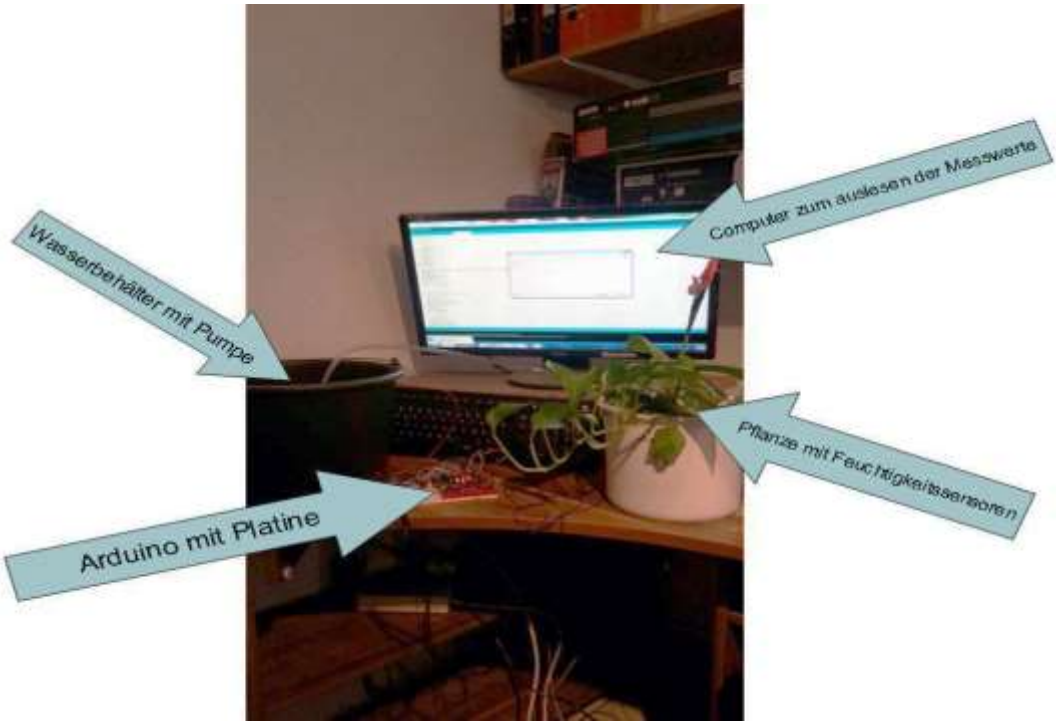
Inhaltsverzeichnis

Inhalt

Einleitung

Wasser ist ein kostbares Gut. Alle Lebewesen benötigen Wasser zum Überleben. Durch Bevölkerungswachstum und extensive Landwirtschaft gibt es in manchen Gebieten auf der Erde Wasserknappheit. Weiterhin führt die Klimaerwärmung zu einer Ausdehnung von Dürrezonen. Dies hat uns motiviert, uns mit dem Thema Minimierung des Wasserverbrauchs auseinanderzusetzen.
Wir wollen mit unserem Projekt untersuchen in wie weit es möglich ist, ein optimales Pflanzenwachstum unter Verwendung von möglichst wenig Wasser zu erreichen.
Um uns diesem Thema anzunähern, wollen wir ein Modellsystem aufbauen, das es ermöglicht einer Pflanze über eine Pumpe mit angeschlossenem Schlauchsystem kontrolliert Wasser zuzuführen. Es gibt bereits Systeme die man an eine Wasserleitung anschließen kann und die zeitgesteuert Wasser über Schläuche auch einer Pflanze zuführen kann. Diese Systeme sind nicht in der Lage sich auf veränderte Bedingungen einzustellen, die eine Anpassung der Wassermenge erfordern.
In unserem System soll unter Verwendung von zusätzlichen Sensoren die Möglichkeit geschaffen werden, eine Pflanze mit der optimalen Menge an Wasser zu versorgen.

Systembeschreibung Hardware



Die Hardware für die automatische Pflanzenbewässerungsanlage besteht aus folgenden Komponenten:

- PC/Laptop
- Arduino Microrechner
- Feuchtigkeitssensor
- Relais
- Wasserpumpe
- Schlauchsystem
- Wasserreservoir

Feuchtigkeitssensor

Der Feuchtigkeitssensor besteht aus zwei Elektroden die an eine Verstärkerschaltung angeschlossen sind. Im trockenen Zustand hat die Blumenerde eine geringere elektrische Leitfähigkeit als im feuchten Zustand. Dadurch fließt im feuchten Zustand ein größerer Strom, wenn man an die Elektroden eine Spannung anlegt. Der Verstärker misst diesen Strom und liefert an den Arduino eine Spannung zwischen 0 und 5 V je nach Feuchte der Erde.

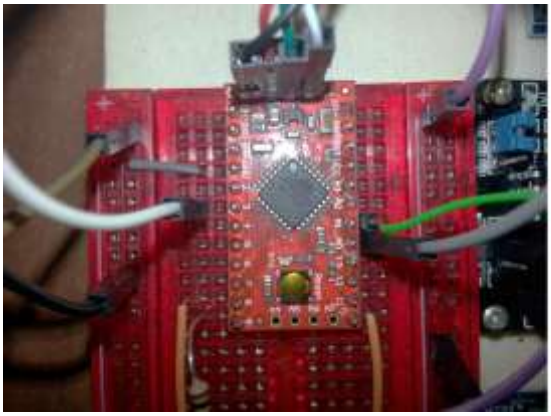
Relais

Das Relais hat die Aufgabe, die Pumpe ein- und auszuschalten. DA die Pumpe einen hohen Strombedarf hat und mit 12 V Spannung betrieben wird, kann dieser nicht direkt an den Arduino angeschlossen werden.

Wasserpumpe

Die Wasserpumpe ist in der Lage aus einem Reservoir (z.B. Eimer oder Tonne) Wasser über ein Schlauchsystem zu der Pflanze zu befördern.

Arduino Microrechner



Der Arduino ist die zentrale Kontrolleinheit, die über ein entsprechendes Softwareprogramm folgende Funktionen erfüllt

- Ansteuerung der Pumpe durch Ein/Ausschalten des Relais
- Erfassung der Feuchtigkeit der Erde über den Feuchtesensor
- Zeitsteuerung
- Kommunikation mit PC bzw. Server /Client

Der Arduino ist in der Lage den Feuchtegehalt von der Erde z.B. in einem Blumentopf zu messen. Dazu wird ein analoges Signal, das der Feuchtigkeitssensor liefert, über einen Analog/Digitalwandler an einem Analogeingang erfasst. Das Relais ist an einem Digitalausgang angeschlossen und kann über entsprechende Programmierung ein- bzw. ausgeschaltet werden.
Der Arduino kann über eine Serielle Schnittstelle mit einem PC/Laptop verbunden werden. Dadurch ist es möglich Softwareprogramme nach Entwicklung auf dem PC auf den Arduino zu laden und vom Arduino gemessene Daten über einen Seriellen Monitor auf dem PC verfügbar zu machen. Diese Daten können dann bei geeigneter Formatierung in eine Excelanwendung importiert werden. Die Messergebnisse können danach als Diagramme dargestellt werden.

Alternativ zur Anbindung des Arduino über Serielle Schnittstelle lässt sich der Arduino auch über eine Netzwerkverbindung mit einem PC verbinden.

Erläuterung zum Netzwerkanschluss

Dazu wurde seitens der Schule ein Softwarecode zur Verfügung gestellt, der es erlaubt den Arduino als Server zu betreiben. Auf einem PC läuft parallel dazu eine Clientanwendung. Der Arduino kann damit die Messwerte zum PC schicken, der dann diese Messwerte zu einer Excel Datei weiterleiten kann, um eine Tabelle aus Messreihen zu erstellen.

Erfahrung bei der Anpassung der Software für einen funktionalen Serverbetrieb

Für den Server haben wir als Referenz einen Beispielcode von der Arduino Webseite für einen Ethernet Server übernommen und daraus unseren eigenen Code geschrieben.[1]
Unser erster Schritt war es erst mal zu Testen ob der Code funktioniert. Dafür haben wir den Code auf den Arduino geladen und ihn dann mit einem Client verbunden. Wenn es funktioniert, sollte „Neuer Client“ und die IP-Adresse vom Arduino, ein „Hallo Client“ als Antwort beim Client und dann die zu verschickende Nachricht vom Client zu dem Server angezeigt werden. Dies funktionierte auch.
Unser zweiter Schritt war es, dauerhaft eine Zahl vom Server an den Client zu verschicken, die nach einer gewissen Zeit hochgezählt werden sollte. Dafür erstellten wir einen Timer. Dies lief aber nicht. Es wurden zwar die vom ersten Schritt beschrieben Daten angezeigt, auf dem Client erschien aber keine Zahl.
Als ersten Lösungsansatz wollten wir erst einmal wissen, ob der Timer überhaupt arbeitet. Dafür haben wir dem Server den Befehl geben, dass er jede zwei Sekunden ein Wort auf dem Bildschirm ausgibt, was wiederum erfolgreich war. Das Problem lag also nicht am Timer, sondern am Code selbst. Der Code war so geschrieben, dass er für mehre Clients gedacht war, die sich mit dem Server verbinden könnten. Die Schleife war so programmiert, dass nach jedem neuen Durchlauf geprüft wurde, ob ein neuer Client zur Verfügung steht. Unser Client, der bereits verbunden war, wurde gar nicht mehr beachtet. Da unser System nur einen Client enthält, mussten wir den Code dementsprechend anpassen.
Nachdem dies alles gelöst war, funktionierte alles einwandfrei.
Unser nächster Schritt war es, den Client so zu umschreiben, dass er die Messwerte zu einer Excel Datei schicken konnte.

Programmierung des Clients

Der Code für den Client war leichter zu programmieren. Wir haben zuerst die maximale Anzahl der Zeichen angegeben, die vom Server verschickt werden würden (1026). Da wir nicht wussten, wie viele Zeichen die gesendeten Messwerte haben würden, wird der Pufferspeicher, in dem die Daten enthalten sind, solange durchgegangen, bis das Programm eine „0“ findet, die das Ende der Zeichenkette anzeigen soll. Die Messwerte werden zusammen mit dem Dateinamen, der Anzahl der Messung und dem Dateiformat „.csv“ verschickt. Danach wird die Anzahl der Messung hochgezählt. Abschließend musst noch gesagt werden, wann der Client stoppen soll, Daten an Excel zu versenden.

Abschließend wurde noch eine eigene Klasse für den Server erstellt und diese in den Code der Gießanlage eingebunden.

[1]Webseite Arduino

Programmierung der Software für den Arduino

Ein großer Teil unseres Projektes war die Programmierung des Arduinos, der die Aufgabe hat, verschiedene Ereignisse zum richtigen Zeitpunkt auszuführen und je nach Stand der Messung z.B. das Gießen der Pflanze zu veranlassen. Auch wandelte er das analoge Signal des Feuchtigkeitssensors mit Hilfe des integrierten A/D-Wandlers in ein Digitales um, mit dem unser Arduino schlussendlich rechnen konnte. Anfangs beschränkte sich unser Code auf wenige Zeilen, in denen wir mit einfachen Delays die Messungen durchführen konnten. Hierbei gaben uns unsere Sensoren eine Wert zwischen 1023 und 0, den es erstmal zu interpretieren galt. Dieser Wert kam aufgrund des 10-Bit A/D-Wandlers zustande (2^10 = 2024, abzüglich der 0).

Mit der Zeit wurde uns allerdings bewusst, dass einfache Delays unser Projekt erschweren würden. In der Zeit in denen die Delays einsetzten stoppte unser Arduino andere Aufgaben, die hätten weiterlaufen sollen. Während z.B. der Arduino der Pumpe das Signal gab Wasser in den Blumentopf zu pumpen, stoppte er für einen gewissen Zeitraum die Messungen. Dies würde im späteren Verlauf unseres Projektes für eventuelle Problematiken sorgen, weshalb wir uns eine Lösung überlegen mussten. Unsere Idee war ein Timersystem zu programmieren, das sowohl funktional als auch anpassbar war.

Hierbei gab uns der Arduino einen Timer vor, der jede zehn Sekunden ein Signal ausgab. Dieses benutzten wir um weitere Timer in Form von einfachen Zählvariablen hochzuzählen. Durch diese Taktik wurde unser Programm einerseits anpassbarer, aber auch sicherer was Bugs betraf. Das Problem hätte sich auch durch das Erstellen von vielen „richtigen“ Timern lösen lassen können, allerdings war hier die Gefahr von Bugs zu groß, die durch das gleichzeitige laufen von zu vielen Timern entstehen könnten. Durch diese Anpassbarkeit konnten wir auch sicherstellen, dass unser Programm für verschiedene Szenarien funktionsfähig war, wie z.B. einem reinen Mess-Modus, indem das Überleben der Pflanze im Hintergrund stand und wir uns auf die Verteilung des Wassers und der Optimalen Wassermenge beschäftigten. Ziel war es uns hierbei, ein anpassbares Programm zu erstellen, das je nach Bedarf handelte.

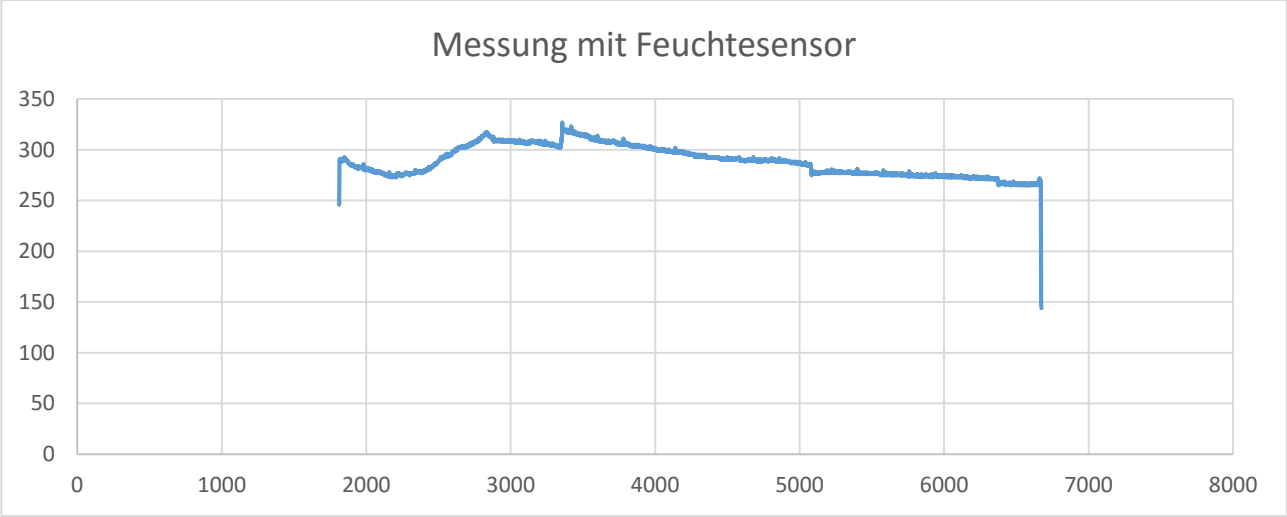
Als uns klar wurde, dass wir unser Projekt in zwei Stadien aufteilen würden, machten wir uns mehr Gedanken über den Modus „Messen“, da wir, um eine vernünftige Intelligente Pflanzenbewässerung zu erreichen, viele Messdaten über die Verteilung des Wassers bei verschiedenen Gegebenheiten wie z.B. verschiedenen Böden oder Töpfen sammeln mussten. Da wir trotzdem beide Modi („Messen“ und „Überleben (der Pflanze)“) beibehalten wollten, entschieden wir uns eine Art Schalter zu programmieren, mit Hilfen dessen wir zwischen den Modus wechseln konnten. Diesen Schalter realisierten wir, indem wir dem Arduino beibrachten, einen Pin zu lesen und, je nach Spannung die anlag, zwischen den Modi zu schalten.

	A	B
3	Messung	Wert
4	1	302
5	2	302
6	3	303,5
7	4	303
8	5	308
9	6	309
10	7	307,5
11	8	306
12	9	304
13	10	303
14	11	302
15	12	301
16	13	299
17	14	298,5
18	15	292,5
19	16	292
20	17	288,5
21	18	288,5
22	19	287
23	20	284
24	21	284
25	22	278
26	23	274,5
27	24	276
28	25	279
29	26	274,5
30	27	271,5
31	28	269,5
32	29	268,5
33	30	267,5
34	31	265,5
35	32	264
36	33	262

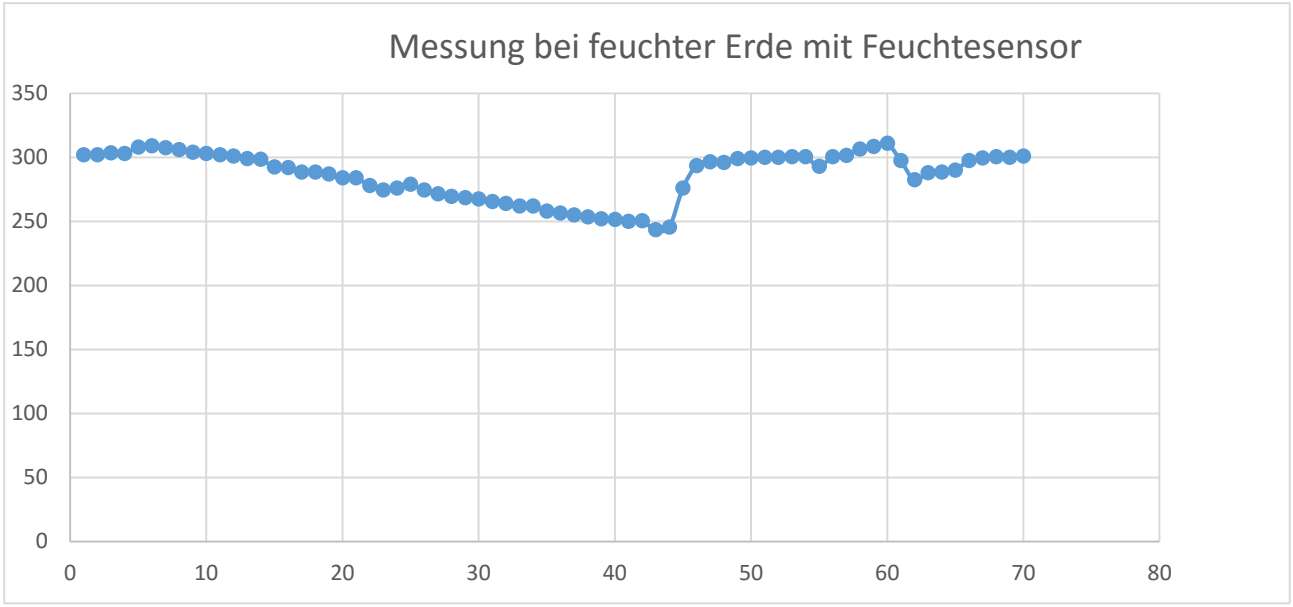
Nachdem die ersten Messungen erfolgreich waren, machten wir uns Gedanken über die Ausgabe der Messwerte. Zu diesem Zeitpunkt war die Serielle Schnittstelle die einzige Möglichkeit, Messwerte auszugeben. Allerdings ergaben sich daraus auch Probleme: Was passiert wenn kein Computer vorhanden ist? Aktuell geben wir die Messergebnisse in einer Form aus, die wir einfach in eine Exceltabelle verwandeln konnten, um schnell Tabellen und Diagramme zu erhalten. Die Form sah dabei „ (Zahl der Messung);(Messwert);“ aus. Dieses Format konnte direkt von Excel übernommen werden. Der Wert der Messung, der wie schon beschrieben eine Zahl zwischen 0 und 1023 zugeordnet wird, lässt sich hierbei wie folgt Interpretieren: Je höher die Zahl, desto höher misst der Sensor für Feuchtigkeit den Stromwiederstand im Topf. Dies bedeutet eine geringe Feuchtigkeit. Ist die Zahl niedriger, bedeutet dies einen geringen Stromwiederstand, bedeutet, der Topf ist feucht. Das Bild zur rechten zeigt die Ausgabe der Messergebnisse kopiert in eine Exceltabelle.

Voraussetzung hierfür war allerdings ein Computer, der die ganze Zeit über die Daten aufnahm. Eine Lösung für dieses Problem wäre schnell gefunden, Arduinos mit SD-Karten gab es ja schon. Allerdings wollten wir mehr. Wir wollten die Messergebnisse verarbeiten, unabhängig davon, wo sich unser Projekt momentan befand. Aus diesem Grund haben wir uns dazu entschieden, eine Arduino mit Netzwerkkarte zu verwenden, der unsere Messdaten dort hinschickt, wo wir sie brauchen. So ist es uns möglich, die Messergebnisse bzw. den Status unseres Systems von überall abzurufen.

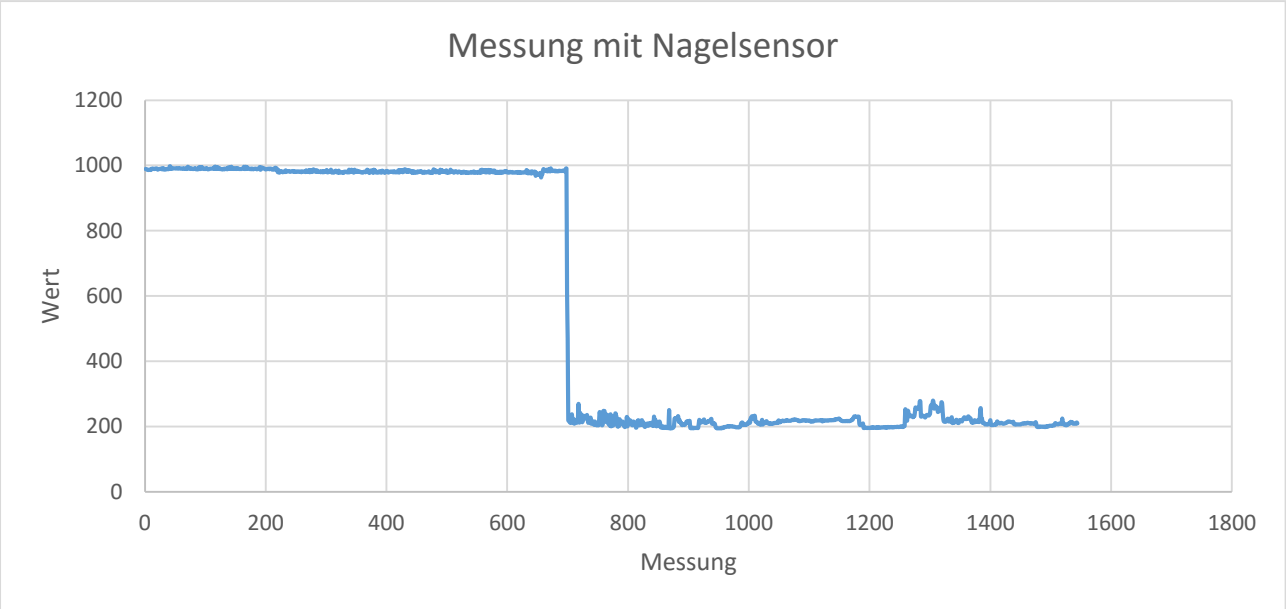
Ergebnisse



Hier maßen wir mit Hilfe des anfangs genutzten Sensors die Feuchtigkeit der Erde über einen längeren Zeitraum (ca. 8h). Die Messergebnisse erschienen uns ungenau und uninterpretierbar, wir gingen von einem defekten Sensor aus. Dies bestätigte sich, er war korrodiert.



Hier maßen wir schon angefeuchtete Erde, wir erwarteten konstante Messergebnisse, allerdings schwankten diese, obwohl wir kein Wasser hinzufügten. Diese Kurzzeit Messung ging 30 Minuten. Wir zweifelten an der Qualität und Genauigkeit der Sensoren, und machten uns auf die Suche nach einer Alternative.



Auf der Suche nach einem geeigneten Sensor hatten wir die Idee, zwei Nägel, verbunden mit dem ehemaligen Empfänger des anfänglich verwendeten Sensors zu verbinden, um eventuell genauere Messergebnisse zu erzielen. Anhand der Grafik sieht man sehr schön, wie anfangs die Erde trocken ist, ab Messung 700 wird Wasser hinzugegeben, der Messwert sinkt, es wird Feuchte gemessen. Die Zeit der Messung betrug 56h.

Diskussion

Zur Zeit der Ausarbeitung haben wir noch nicht genügend Ergebnisse gesammelt, um endgültige Schlussfolgerungen zu ziehen. Wir sind immer noch mit Messungen aller Art beschäftigt um unser Projekt nächstes Jahr erfolgreich weiterführend zu können.