# Teil: Programmierung der Software für den Arduino

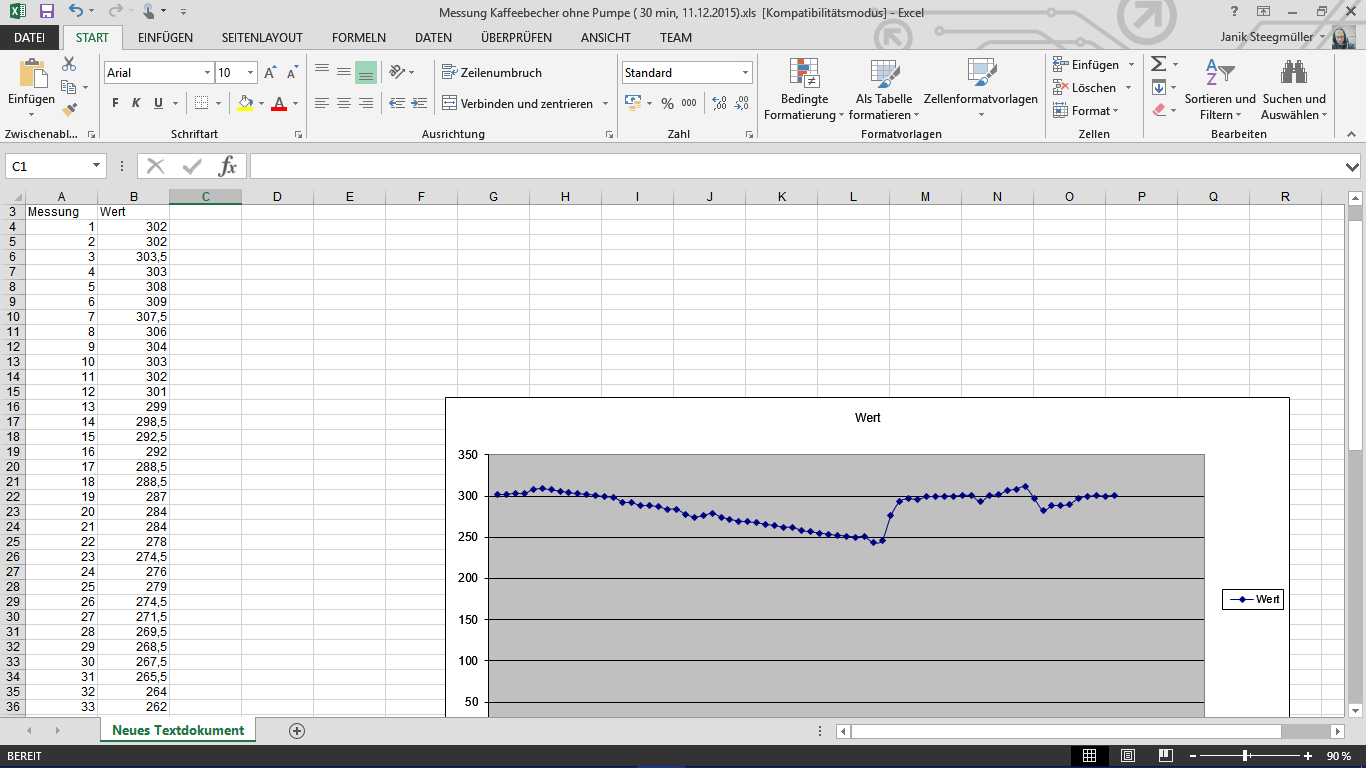
Ein großer Teil unseres Projektes war die Programmierung des Arduinos, der die Aufgabe hat, verschiedene Ereignisse zum richtigen Zeitpunkt auszuführen und je nach Stand der Messung z.B. das Gießen der Pflanze zu veranlassen. Auch wandelte er das analoge Signal des Feuchtigkeitssensors mit Hilfe des 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.

Nachdem die ersten Messungen erfolgreich waren, machten wir uns Gedanken über die Ausgabe der Messwerte. Zu diesem Zeitpunkt war der 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. Genauere Erläuterung dazu finden sie ein paar Seiten weiter.



Ausgabe der Messwerte

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.