

# Fachbericht

## Wetterstation mit Solar Energie

Windisch, 11. Januar 2019

<b>Hochschule</b>	Hochschule für Technik - FHNW
<b>Studiengang</b>	Elektro- und Informationstechnik
<b>Autor/-en</b>	Mischa Knupfer, Andres Minder
<b>Betreuer</b>	Prof. Dr. Taoufik Nouri
<b>Auftraggeber</b>	Prof. Dr. Taoufik Nouri
<b>Version</b>	1.0

## **Abstract**

## Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>1</b>
<b>2 Auftragsbeschreibung</b>	<b>2</b>
<b>3 Ziele</b>	<b>3</b>
<b>4 MCU</b>	<b>5</b>
<b>5 RTC</b>	<b>6</b>
<b>6 Sensoren</b>	<b>7</b>
6.1 Messen der Lufttemperatur . . . . .	7
6.2 Ermittlung der Niederschlagsmenge . . . . .	7
6.3 Anemometer . . . . .	11
6.3.1 Implementation in die Firmware . . . . .	12
6.3.2 Validierung . . . . .	12
6.4 Zählung der Sonnenstunden . . . . .	13
<b>7 Datenspeicherung</b>	<b>14</b>
7.1 Breakoutboard . . . . .	14
7.1.1 Verdrahtung . . . . .	14
7.1.2 SPI (Serial Peripheral Interface) . . . . .	15
7.2 µSD-Karte . . . . .	15
7.3 Implementation in die Firmware . . . . .	16
<b>8 Kommunikationsmodul</b>	<b>17</b>
<b>9 Energieversorgung</b>	<b>18</b>
<b>10 Konzeptvalidierung</b>	<b>19</b>
<b>A Lastenheft</b>	<b>20</b>
<b>B ATmega2560-Arduino Pin Mapping</b>	<b>21</b>

## 1 Einleitung

Pflanzen benötigen eine ihnen entsprechende Umwelt. Durch meteorologische Messdaten kann diese ermittelt und durch Agronome optimal bewirtschaftet werden. Außerdem können bei genügend Messdaten aus einem Erfassungsnetz Wetterprognosen erstellt werden. Die Erhebung solcher Messdaten trägt somit erheblich zum wirtschaftlichen Erfolg in der Agronomie bei und kann bei einem geeigneten grossen Erfassungsnetz Bewohner vor Unwetter warnen. In den ärmeren Teilen Afrikas und Südamerikas sind solche Systeme jedoch nicht verbreitet.

Aus diesem Grund soll eine kostengünstige, erweiterbare und mobile Wetterstation gebaut werden, welche die örtlichen Agronomen unterstützt. Diese Wetterstation soll die Niederschlagsmenge, die Windstärke, die Lufttemperatur und die Sonnenstunden messen können. Außerdem soll die Wetterstation mittels Photovoltaik unterstützt werden, und erhobene Daten via SMS abrufbar sein. Mit einem GPS-Modul soll der Standort der mobilen Wetterstation erfasst werden. Im Projekt 5 wird ein erster Prototyp erstellt, welcher die Datenermittlung mit der Sensorik und Datenspeicherung beinhaltet. Die Stromversorgung (Akku, Solarpanels), das GPS und das Senden der Daten über SMS werden im Projekt 6 implementiert.

Die Lufttemperatur wird über den Temperatursensor XYZ gemessen. Mittels Kipplöffelprinzip wird die Niederschlagsmenge über den Sensor XYZ ermittelt, wobei die Funktionsweise des Kipplöffels zuerst mit einem Selbstbau überprüft wird. Die Windstärke wird über das Anemometer XYZ eruiert. Um die Sonnenstunden zu ermitteln werden zwei Varianten (zum einen via Ladestrom der Photovoltaikanlage, zum anderen mit einem Sensor XYZ) verglichen und die Leistungsärmere implementiert, was jedoch erst im Projekt 6 stattfindet. Es wird für das Prototyping ein ArduinoMega2560 verwendet, welcher die Sensoren über die vorhandenen Peripherieanschlüsse mit dem Mikrocontroller verbindet. Die gesammelten Daten werden auf einer microSD Karte abgespeichert, wobei mit Hilfe eines RTC (Real Time Clock) ein Zeitstempel hinzugefügt wird.

Das Projekt 5 liefert die erwünschten Messdaten von Lufttemperatur, Windgeschwindigkeit und Niederschlagsmenge. Außerdem werden die Daten wunschgemäß auf einer microSD Karte, mit zugehörigem Zeitstempel versehen, gespeichert.

## 2 Auftragsbeschreibung

Das Wetter spielt eine wichtige Rolle in der Agronomie. Regnet es nicht genug, müssen Pflanzen bewässert werden. Trifft auf ein Ort nur wenig Sonnenlicht, so sollten dort nicht die Pflanzen, welche viel Sonnenlicht brauchen, angebaut werden. Windet es zu stark, können Pflanzen beschädigt oder gar zerstört werden. Ist es Tagsüber heiß, so benötigen die Pflanzen mehr Wasser. Hiesige Bauern besitzen den Luxus von guten Wettervorhersagen dank dem Bundesamt für Meteorologie und Klimatologie (MeteoSchweiz). Dieser Luxus ist in anderen Ländern noch nicht gegeben. Prof. Dr. Nouri Taoufik ist aufgefallen, dass in subtropischen Gegenden wie Teile Südamerikas oder Afrikas dieser Luxus ebenso fehlt.

Aus diesem Grund soll eine kostengünstige, erweiterbare und mobile Wetterstation entwickelt werden, welche diese Bauern unterstützt. Diese Wetterstation soll die Regenmenge, die Windstärke, die Lufttemperatur und die Sonnenstunden messen können. Außerdem soll die Wetterstation mittels Photovoltaik unterstützt werden, und erhobene Daten via SMS abrufbar sein.

### 3 Ziele

Die Ziele sind strikt aufgeteilt in die zwei Projekte 5 und 6. Darin enthalten sind die jeweiligen zu erreichenden Muss- und Wunschziele mit ihren quantifizierten Spezifikationen. Diese sind wichtig, da Ortsabhängig unterschiedliche Normwerte gelten und sich dieses Projekt grundsätzlich auf die Schweiz fokussiert.

**Tabelle 3.1:** Ziele P5

	Ziel	Messbereiche	Genauigkeiten	Einheiten
<b>Mussziele P5</b>				
Sensoren	Lufttemperaturmessung	[ -20; 60 ]	± 1	°C
	Windgeschwindigkeitsmessung	[ 10; 25 ]	± 1	m/s
	Niederschlagsmenge	Wasser	± 100	ml/m <sup>2</sup>
Datenspeicherung	Datenabfrage via PuTTY	≥ 9600		Bd/s
RTC	Implementation	Echtzeit	± 1	s/Jahr
<b>Wunschziele P5</b>				
Sensoren	Sonnenstunden Prototyp	Echtzeit		s

Tabelle 3.1 zeigt diverse Ziele im P5, unterteilt in Muss- und Wunschziele. Zu den Musszielen gehören die Lufttemperaturmessung, die Windgeschwindigkeitsmessung, die Niederschlagsmessung, die Implementation des RTC und die mögliche Datenabfrage via Putty vom Datenspeicher. Die Lufttemperatur soll zwischen -20 bis 60 °C ermittelbar sein, mit einer Genauigkeit von ±1 °C. Die Windgeschwindigkeitsmessung soll vor allem stärkere Windgeschwindigkeiten erfassen, um vor Sturm warnen zu können, weshalb niedrigere Windgeschwindigkeiten vernachlässigt werden können. Die Windgeschwindigkeit soll zwischen 10 und 25 m/s auf ±1 m/s genau gemessen werden. Die Niederschlagsmenge soll nur für Regenwasser bestimmt werden mit einer Genauigkeit von ±100 ml/m<sup>2</sup>. Als Wunschziel soll eine Möglichkeit getestet werden um Sonnenstunden zu detektieren, welche dann im P6 umgesetzt wird.

**Tabelle 3.2:** Ziele P6

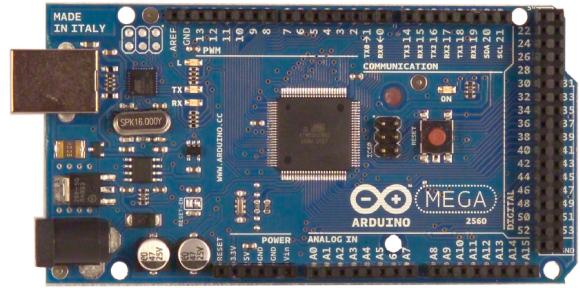
	Ziel	Messbereiche	Genauigkeiten	Einheiten
<b>Mussziele P6</b>				
Speisung	Akkukapazität			
	Ladeschaltung Akku			
	Ladeschaltung Photovoltaik			
Kommunikationsmodul	GPS			
	Mobilfunk (SMS)			
Sensoren	Sonnenstunden			
<b>Wunschziele P6</b>				
Kommunikationsmodul	Mobilfunk (Website)			
Speisung	Akku austauschbar			

Tabelle 3.2 zeigt diverse Ziele im P6, unterteilt in Muss- und Wunschziele. Diese Tabelle ist unvollständig und wird im P6 nachgeführt. Generell kann gesagt werden, dass die Speisung, das Kommunikationsmodul mit GPS und Mobilfunk, sowie die Sonnenstunden-Sensorik implementiert werden sollen. Als Wunschziele sind ein austauschbarer Akku und eine Website zur Datensicherung und ggf. grafischen Darstellung aufgeführt.

## 4 MCU

Die Micro Controller Unit (MCU) ist der zentrale Bestandteil für die Kommunikation, resp. für den Datenaustausch zwischen den unterschiedlichen Modulen. Sie interpretiert die Signale der Sensoren und rechnet sie in die interessierenden Messwerte um. Dann weist die MCU jedem Messwert einen Zeitstempel über das RTC zu und übergibt diesen der Datenspeicherung. Wenn die Daten vom Kommunikationsmodul angefordert werden, liest die MCU die Datenspeicherung aus und übergibt sie dem Kommunikationsmodul.

Für die Entwicklung der MCU wird ein Arduino Mega Board verwendet. Der Vorteil besteht darin, dass elementare Bauteile (Hardware) bereits implementiert sind, wie z.B. Oszillator, der USB-Anschluss und die PCB-Connectors für ein schnelles Prototyping. Die wichtigsten technischen Daten sind in der Tabelle 4.1 aufgelistet.



**Abbildung 4.1:** Arduino Mega Board [1, S.1]

**Tabelle 4.1:** Technische Daten [1, S.3]

Microcontroller	ATmega2560
Operating Voltage	5V
Digital I/O Pins	54
Analog Input Pins	16
Flash Memory	256 KB, 8 KB werden vom bootloader benötigt
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

**5 RTC**

## 6 Sensoren

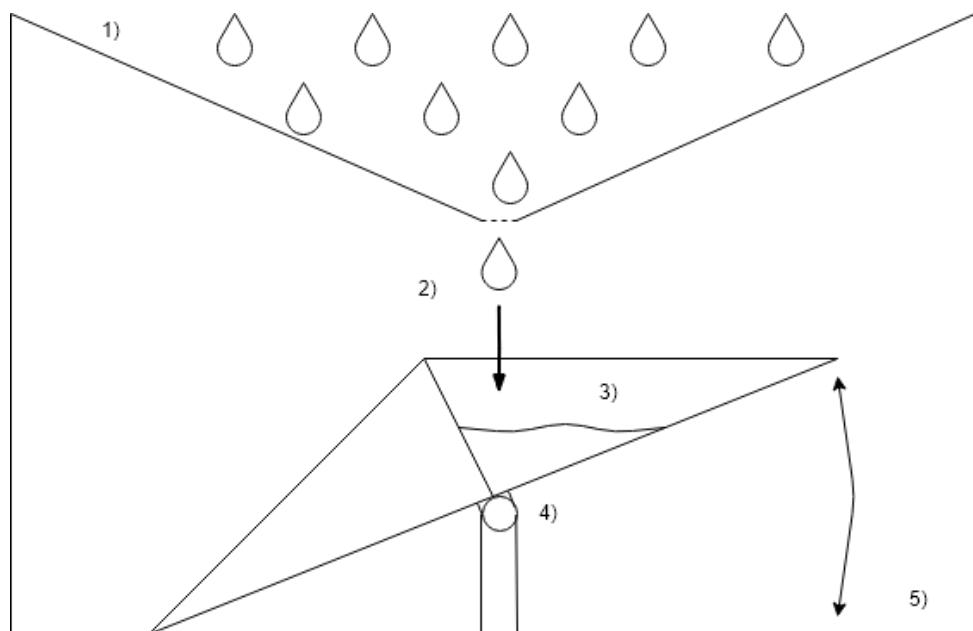
### 6.1 Messen der Lufttemperatur

### 6.2 Ermittlung der Niederschlagsmenge

Dieses Unterkapitel befasst sich mit der Realisierung der Niederschlagsmessung. Diese soll nach einem Kipplöffelprinzip funktionieren und gemäss definierten Zielen eine Genauigkeit von  $\pm 100 \text{ ml/m}^2$  aufweisen. Ausserdem soll als alternative zusätzlich ein Messbecher an der Wetterstation installiert werden, damit der Bauer die Niederschlagsmenge anhand einer Skala ableSEN kann. In einem ersten Schritt soll das Kipplöffelprinzip näher erläutert und mit einem Selbstbau die Funktionsweise getestet werden. Anschliessend soll ein gekaufter Sensor die Wetterstation erweitern und die Implementation in der Firmware thematisiert werden. Zu guter Letzt soll die Validierung des Teilsystems folgen.

#### Das Kipplöffelprinzip

Das Prinzip des Kipplöffels wird in Abbildung 6.1 graphisch dargestellt.



**Abbildung 6.1:** Darstellung des Kipplöffelprinzips

Abbildung 6.1 zeigt das Kipplöffelprinzip. Der Kipplöffel („3“) besteht im Grunde aus zwei Löffeln und ist in der Mitte drehbar mit dem Gehäuse befestigt („4“). Regenwasser wird über eine Öffnung im Gehäusedeckel (Trichter, „1“) zum Kipplöffel befördert („2“). Ist der Löffel mit Regenwasser gefüllt, so kippt dieser aufgrund des Gewichts und leert das Wasser über eine Öffnung im Gehäuseboden („5“) aus. Durch die Kippung wird der andere Löffel in die Ausgangsposition bewegt und kann sich nun mit Wasser füllen. Mit der Hilfe von ReedkontakteN und Magneten wird die Anzahl der Kippbewegungen gezählt. Die Niederschlagsmenge ergibt sich aus der Anzahl Kippbewegungen, multipliziert mit dem Volumen des Kipplöffels.

## Die Realisierung des Niederschlagsmengensensors

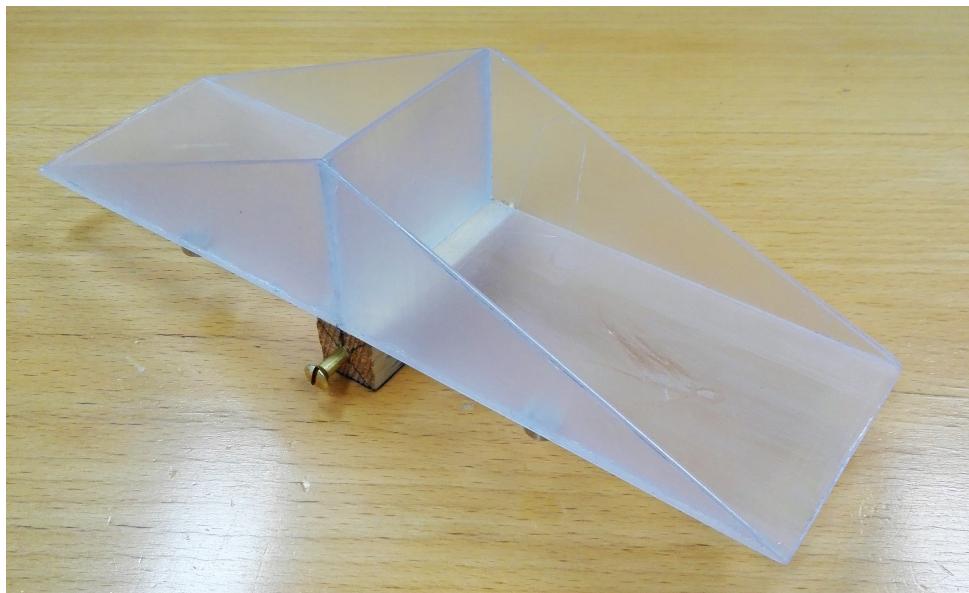
Um die Funktionsweise des Niederschlagsmengensensors zu testen, wird, wie im Pflichtenheft festgehalten, dieser in einem ersten Schritt selbst erstellt. Die Erstellung kann in vier Etappen unterteilt werden. Die erste Etappe ist die Erstellung des Kipplöffels. Die zweite Etappe folgt mit der Erstellung der drehbaren Lagerung. Als dritte Etappe folgt der Trichter und die vierte und letzte Etappe widmet sich dem Gehäuse, wobei der Trichter ein Teil des Gehäuses darstellt. Das Gehäuse wird, bei Verwendung der Eigenproduktion, erst im Projekt 6 mit dem Gehäuse der gesamten Wetterstation erstellt.

### Etappe 1: Realisierung des Kipplöffels

Wichtig für die Erstellung des Kipplöffels sind die Dimensionierung und die Materialwahl.

Das Material soll wetterbeständig, einfach bearbeitbar und günstig sein und eine möglichst glatte Oberfläche haben. Die möglichst glatte Oberfläche ist notwendig, damit das Wasser im Kipplöffel sich nicht an der Oberfläche festhält und somit gut abfließt. Acrylglas erfüllt diese Bedingungen und ist in jedem Baumarkt erhältlich, weshalb es als Material gewählt wird.

Die Dimensionierung ist Abhängig von der gewählten Genauigkeit im Pflichtenheft. Damit eine Genauigkeit von  $\pm 100 \text{ ml}/\text{m}^2$  erreicht werden kann, müssen beide Löffel des Kipplöffels bei genau 100 ml Fassungsvermögen kippen. Damit dies erreicht wird, kann man physikalisch die statische Gleichgewichtsbedingung aufstellen und daraus die Dimensionierung folgern. Dies ist jedoch ein sehr aufwändiger, komplizierter und zeitintensiver Weg. Einfacher ist es, wenn der Kipplöffel extra zu gross dimensioniert und die Füllmenge im Nachhinein justiert wird. Die Justierung erfolgt mittels einer in der Höhe verstellbaren Lagerung, sowie mit in der Höhe verstellbaren Schrauben im Gehäuseboden, welche die Neigung der Endposition des Kipplöffels beeinflusst. Ein weiterer Vorteil dieser Nachjustierung ist, dass auch eine andere Füllmenge einstellbar wäre.



**Abbildung 6.2:** Selbsterstellter Kipplöffel.

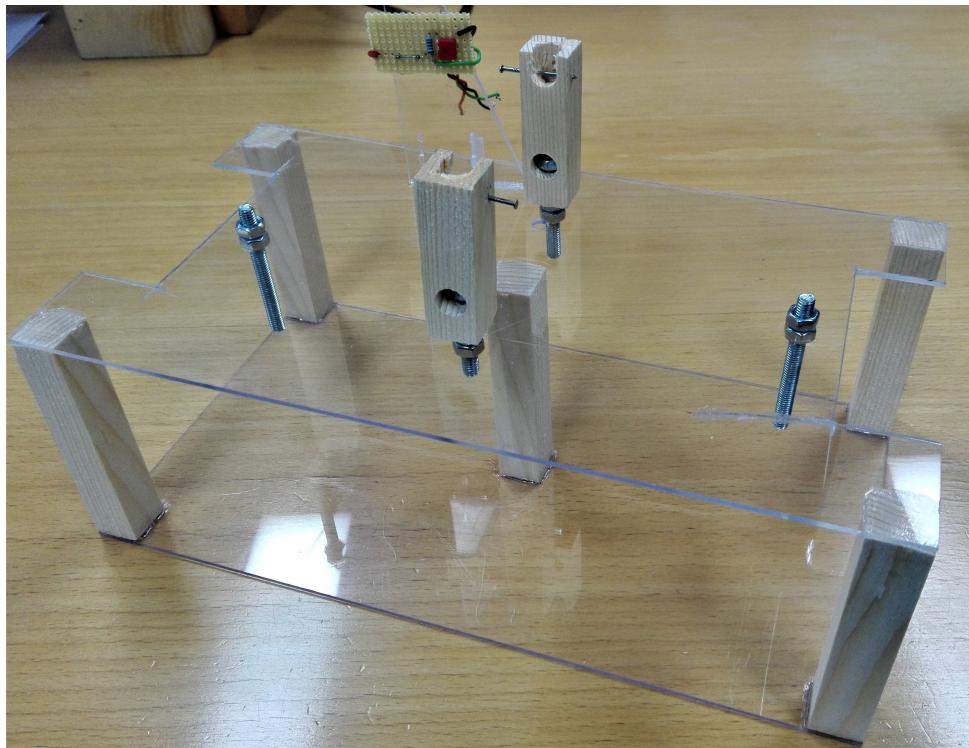
Abbildung 6.2 zeigt den selbsterstellten Kipplöffel aus Acrylglas. Die Drehachse ist mittig unter dem Kipplöffel befestigt und besteht aus einem Holzklotz mit je einer Schraube pro Seite.

### Etappe 2: Realisierung der drehbaren Lagerung

Die drehbare Lagerung des Kipplöffels ist wichtig, damit der Kipplöffel auf beide Seiten kippen kann. Die Drehachse soll direkt unterhalb der Mitte des Kipplöffels befestigt sein um ein gleich-

mässiges Kippen zu ermöglichen. Die Höhe des Kipplöffels wird definiert durch die einstellbare Höhe der Drehachsenlagerung.

Die Drehachse wird aus einem Stück Holz und zwei Schrauben gefertigt, wobei das Holz direkt am Kipplöffel befestigt wird. Die zwei Schrauben werden auf einem höhenverstellbarem Gerüst gelagert, so dass ein drehen möglich ist. Dieses Gerüst wird auch aus Holz gefertigt und enthält eine Metallische Fläche an der Kontaktstelle der zuvor erwähnten Schrauben, um aufkommende Reibkräfte zu verringern. Ausserdem ist dieses Gerüst höhenverstellbar über zwei mit Muttern feststellbaren Gewinden (für jede Seite eine).



**Abbildung 6.3:** Selbsterstelltes Gerüst.

Abbildung 6.3 zeigt das selbsterstellte Gerüst für die Höhenverstellbare, drehbare Lagerung des Kipplöffels. Es kann sowohl die Höhe des Kipplöffels, sowie dessen Neigung in den Endpositionen über Gewinden mit Muttern eingestellt werden. Die Schrauben des Kipplöffels kommen auf einen Stahlnagel zu liegen, womit Reibungsverluste gering gehalten werden.

### **Etappe 3: Realisierung des Trichters**

Der Trichter sorgt dafür, dass der Regen, welcher auf die Trichterfläche fällt, über der Mitte des Kipplöffels in den Löffel fliesst. Die Trichterfläche stellt gleichzeitig die Referenzfläche dar, da die gesamte Regenmenge dieser Fläche über den Kipplöffel erfasst wird. Ist diese Fläche von  $1 m^2$  abweichend, so muss in der Firmware ein Skalierungsfaktor implementiert werden, damit die Regenmenge wie gewünscht gemäss Pflichtenheft ermittelt werden kann. Der Trichter wird aus demselben Material gefertigt wie der Kipplöffel, da hier die gleichen Anforderungen gelten. Es sei angemerkt, dass der Trichter nur bei weiteren Verwendung des selbst erstellten Kipplöffels, zusammen mit dem Gehäuse der gesamten Wetterstation, erstellt wird.

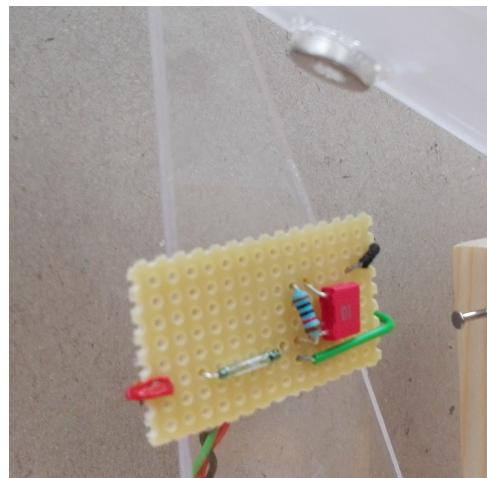
### **Etappe 4: Realisierung des Gehäuses**

Das Gehäuse soll den Sensor vor ungewollten äusseren Einflüssen schützen, sowie umgebende Elektronik vor eventuellen Regenwasserspritzern. Ausserdem soll ein Schaltkreis mit Reedrelais implementiert werden, damit die Kippbewegungen von der Elektronik erfasst werden können.

Es sei erwähnt, dass das Gehäuse nur bei weiteren Verwendung des selbst erstellten Sensors, zusammen mit dem Gehäuse der Wetterstation, konstruiert wird.

### **Implementierung des Schaltkreises**

Der Schaltkreis, welcher die Kippbewegungen feststellen soll, besteht im wesentlichen aus einem Reedrelais und einem Permanentmagneten. Das Reedrelais ist NO (Normally Open) und wirkt als stromkreisschliessender Schalter, sobald ein magnetisches Feld (z.B. das eines Permanentmagneten) sich in unmittelbarer Nähe befindet. Der Permanentmagnet wird auf dem Kipplöffel befestigt und das Reedrelais als Gegenstück an einem Fixpunkt in der Nähe. Wichtig dabei ist, dass das Reedrelais bei den Endpositionen des Kipplöffels nicht geschlossen ist, damit der Stromkreis geöffnet ist und Strom gespart werden kann. Das Reedrelais benötigt einen seriellen Widerstand, damit bei einem schliessen des Stromkreises kein Kurzschluss auftritt. Außerdem soll ein Kondensator parallel zum Widerstand sein, um die Speisespannung zu glätten und so ein nutzbares Signal zu erhalten. Die Speisespannung stellt den Pegel für ein schliessen des Reedrelais, und somit auch für eine Kippbewegung dar. Um die Kippbewegungen zu zählen, kann somit entweder jede steigende oder jede fallende Flanke des Signals gezählt werden.



**Abbildung 6.4:** Schaltkreis zur detektion der Kippbewegung.

Abbildung 6.4 zeigt den Schaltkreis zur detektion der Kippbewegung. Benutzt wurde ein  $10\text{k}\Omega$  Widerstand mit einem parallel angeschlossenen  $100\text{nF}$  Kondensator. Der Reedkontakt reagiert auf den ebenso sichtbaren Permanentmagneten, welcher an der Unterseite des Kipplöffels befestigt ist.

### **Nachteile des Selbstgebauten Niederschlagsmengensensor**

Der selbstgebaute Niederschlagssensor beweist, dass das Prinzip des Kipplöffels funktioniert. Dennoch weist der Selbstbau Mängel auf. Der verwendete Permanentmagnet muss geklebt werden, weshalb dessen Magnetfeld massiv an Stärke verliert und die Schaltung deshalb äusserst nahe angebracht werden muss. Außerdem kam es, dadurch dass keine Werkstatt zugänglich war, zu Improvisation bei nahezu allen Fertigungsschritten, was zu unkalkulierbaren Abweichungen führt. Als Beispiel sei das Spiel der drehbaren Lagerung des Kipplöffels auf dem Gerüst angeführt, was jegliche Justierungsversuche der Niederschlagsmenge beeinflusst. Aus den genannten Gründen wird vorgefertigter Sensor mit Kipplöffelprinzip verwendet.

### **Implentation in der Firmware**

### **Validierung der Niederschlagsmessung**

### 6.3 Anemometer

Für die Windgeschwindigkeitsmessung wurde ein Ersatz Anemometer von Froggit genommen (Abb. 6.5). Das Anschlusskabel hat einen vier poligen RJ-11 Stecker, dessen Signal über eine Buchse zum MCU geführt wird. Das Anemometer selbst hat allerdings nur zwei Anschlüsse, die Speisung (rot) und das durch einen mit einem Dauermagneten schließbaren Reedkontakt modulierte pulsförmige Ausgangssignal (grün, Abb. 6.6). In der Abb. 6.7 ist ersichtlich, dass das Ausgangssignal über R1 abfällt und C1 als Spannungsstabilisierung dient. Das daraus resultierende Signal ist in der Abb. 6.8 aufgezeigt. Die Windgeschwindigkeit ist nun aus der Anzahl Rechteckpulsen direkt interpretierbar:

Wenn über einen Zeitraum  $T$  die Anzahl Pulse  $A$  gemessen werden, dann kann auf die Winkelgeschwindigkeit  $\omega$  nach

$$\omega = \frac{A}{T} \quad [s^{-1}] \quad (6.1)$$

geschlossen werden. Da allerdings verschiedene Faktoren wie das Trägheitsmoment des Schalenkreuzes, Reibungsverluste bei der Drehbewegung, Verfälschung bei wechselnder Windrichtung usw. zusätzlich auf das Anemometer wirken, wird es sehr komplex die Windgeschwindigkeit exakt zu berechnen. Deshalb wird nur ein Näherungswert ermittelt und mit einem Skalierungsfaktor  $SF$  korrigiert. Somit ergibt sich für die Windgeschwindigkeit  $v_{Wind}$  mit Radius  $r$  des Schalenkreuzes

$$v_{Wind} = \frac{A * r * SF}{T} \quad [m/s]. \quad (6.2)$$

Der Skalierungsfaktor  $SF$  wird mittels Referenzmessungen der Windgeschwindigkeit eines digitalen Anemometers eruiert.



Abbildung 6.5: Anemometer [2]

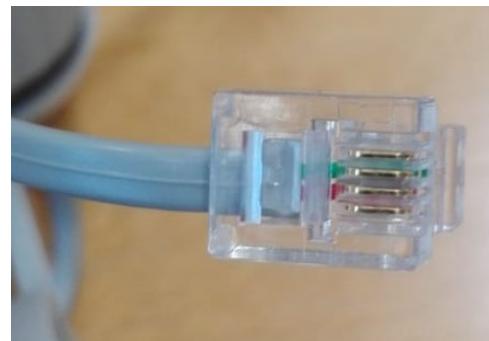


Abbildung 6.6: RJ-11 Stecker

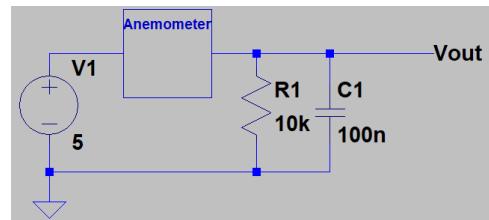


Abbildung 6.7: Beschaltung des Ausgangs des Anemometers.

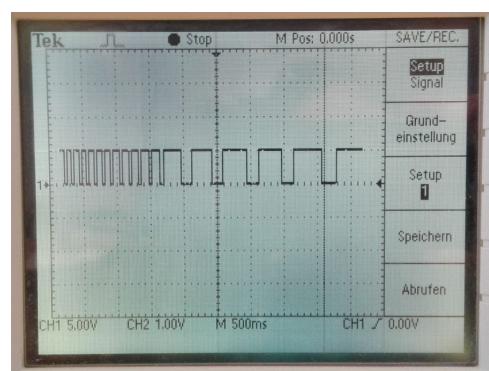


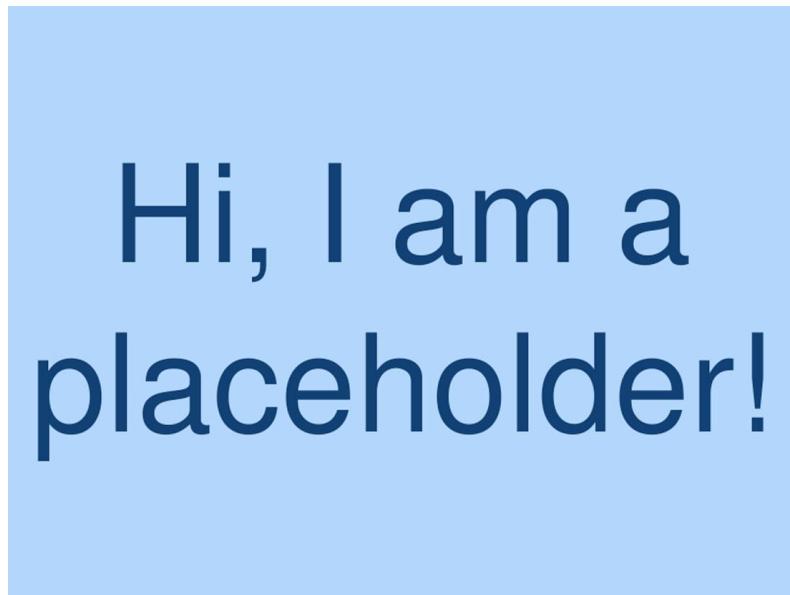
Abbildung 6.8: Ausgangssignal  $V_{out}$

### 6.3.1 Implementation in die Firmware

Die Implementation wurde recht simpel gehalten. Der gesamte implementierte Code für das Anemometer ist im Headerfile "Anemometer.h" extern deklariert und im File Anemometer.cpp initialisiert. Das Signal  $V_{out}$  ist mit einen digital Pin des Atmega 2560 (Pinnummer 2 des Arduino Mega Boards) verbunden. Über einen Zeitraum von  $5000ms$ , auf die steigende Flanke getriggert, wird die Anzahl von Pulsen mittels Interrupt<sup>1</sup> gezählt. Dabei wird zuerst der Interrupt auf der Pinnummer 2 aktiviert, mit einem Delay von  $5000ms$  gewartet, wobei bei jedem ausgelösten Interrupt die Funktion `countWind()` ausgeführt und somit bei jeder steigenden Flanke um eins inkrementiert wird. Zum Schluss folgt die Deaktivierung des Interrupts und die Berechnung der Windgeschwindigkeit nach der Gleichung 6.2.

### 6.3.2 Validierung

Über eine einigermaßen konstanten Windgeschwindigkeit eines Heizlüfters/Ventilators (mit verschiedenen Stärkestufen) wurden Messpunkte des Anemometers (Abb. 6.5), sowie auch des digitalen Anemometers (Abb. 6.10) erfasst. In der Abb. 6.9 sind diese Messwerte graphisch dargestellt.



**Abbildung 6.10:** Digitales Anemometer (Benetech GM8908) mit einer Auflösung von  $0.1m/s$  und einer Unsicherheit von  $\pm 5\%$  [3]

**Abbildung 6.9:** Graph der Messwerte

Hier muss noch die Interpretation der gemessenen Werte geschrieben werden.

<sup>1</sup>es handelt sich hierbei um *external Interrupts*.

#### **6.4 Zählung der Sonnenstunden**

## 7 Datenspeicherung

Die Datenspeicherung beinhaltet die gespeicherten Messwerte der Sensoren. Dafür wird als Speichermedium eine  $\mu$ SD-Karte verwendet, welche direkt in das 254  $\mu$ SD-Breakoutboard von Adafruit gesteckt wird. Die Daten werden dann in einem \*.txt-File nicht flüchtig gespeichert. Bei Beschädigung der Hardware können dann die zuletzt erfassten Daten immer noch mittels eines SD-Karten-Adapters von einem Computer ausgelesen werden. Als Kommunikationsprotokoll für das Schreiben und Auslesen der Karte wird SPI verwendet.

### 7.1 Breakoutboard

Das Breakoutboard (siehe Abb. 7.1) kann wegen des intern implementierten *CD74HC4050 high-speed logic level translators*<sup>2</sup> mit 5V betrieben werden. Das Arduino Mega Board und das Breakoutboard werden über SPI (siehe Kapitel 7.1.2) nach dem Master-Slave Kommunikationsprinzip miteinander verbunden.

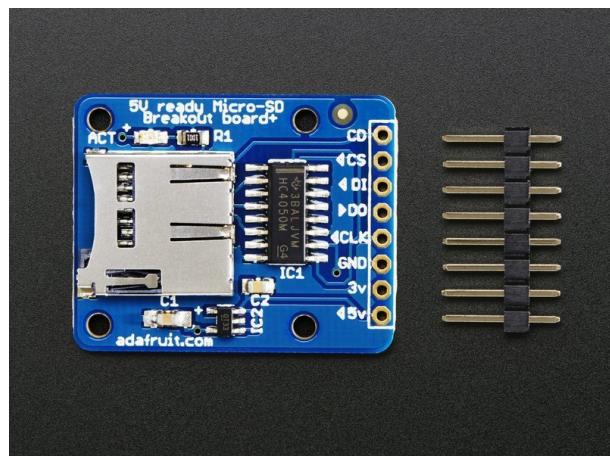


Abbildung 7.1: 254  $\mu$ SD-Breakoutboard von Adafruit [4]

#### 7.1.1 Verdrahtung

SD-Karten erfordern viel Datenübertragung. Deshalb kann die beste Leistung erbracht werden, wenn sie an die Hardware-SPI-Pins eines Mikrocontrollers angeschlossen werden. Dabei wird es wie folgt miteinander verbunden: [4]

Hier vielleicht noch das Schema hinzufügen wie es hardwaremäßig implementiert wird.

- **5V** und **GND** Pins jeweils auf die **5V** und **GND** Pins des Arduino Mega Boards
- **CLK** auf die Pinnummer **52**
- **DO** auf die Pinnummer **50**
- **DI** auf die Pinnummer **51**
- **CS** auf die Pinnummer **53**

<sup>2</sup>konvertiert eine high-level logik in eine low-level logik

### 7.1.2 SPI (Serial Peripheral Interface)

Das *Serial Peripheral Interface* ist ein synchrones serielles Datenprotokoll (Datenbus) bestehend aus drei Datenleitungen zur Datenübertragung. Diese sind, wie in Abbildung 7.2 zu sehen, **MISO** (Master In Slave Out), **MOSI** (Master Out Slave In) und **SCLK** (Serial Clock). Auf dem Breakoutboard (Abb. 7.1) sind die Pins mit **DI** (Data In), **DO** (Data Out) und **CLK** (Clock) beschriftet. Zu den Datenleitungen wird noch eine **SS**- (Slave Select) oder auch **CS**-Leitung (Chip Select) benötigt. Damit wird vom Master aus den zur momentanen Kommunikation nötigen Slave selektiert. Große Vorteile von SPI sind die Vollduplexfähigkeit und das Taktfrequenzen bis in den MHz-Bereich reichen. [5][6]

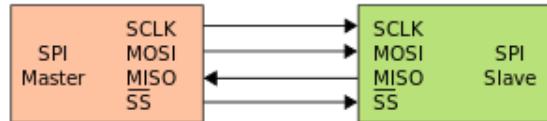


Abbildung 7.2: Einfacher SPI-Datenbus [6]

Möglicherweise noch die Taktfrequenz angeben, resp. die Einstellungen des SPI-Protokolls

## 7.2 $\mu$ SD-Karte



Abbildung 7.3: 16 GB  $\mu$ SD-Karte [7]

Bei der  $\mu$ SD-Karte muss auf die Kompatibilität mit dem Breakoutboard geachtet werden. Dafür sind folgende Kriterien zu beachten:

- Die  $\mu$ SD-Karte muss FAT16 oder FAT 32 formatiert sein.
- Es sind nur die SD und SD High Capacity (SDHC) kompatibel.

Für die Umsetzung dieses Projektes wurde eine  $\mu$ SD-Karte der SD-Familie SDHC I verwendet (siehe Abb. 7.3). SDHC sind Kapazitäten bis zu 32GB möglich und FAT32 formatiert. [8]

Es könnte noch auf die Anschlüsse der  $\mu$ SD-Karte eingegangen werden. Gäbe aber nur Sinn wenn ein Print erstellt werden muss wo das Breakoutboard nachkonstruiert wird.

### 7.3 Implementation in die Firmware

Für die Implementation in die Firmware, um mit dem Breakoutboard über SPI zu kommunizieren und die  $\mu$ SD-Karte zu beschreiben, resp. zu lesen, wurden direkt die bereits existierenden Librarys `<SPI.h>`<sup>3</sup> und `<SD.h>` von Arduino inkludiert. In der Arduino IDE können bereits vorgefertigte Example-Codes (siehe Abb. 7.4) zur weiteren Interpretation, wie mit diesen Librarys  $\mu$ SD-Karten gelesen und geschrieben werden können, verwendet werden.

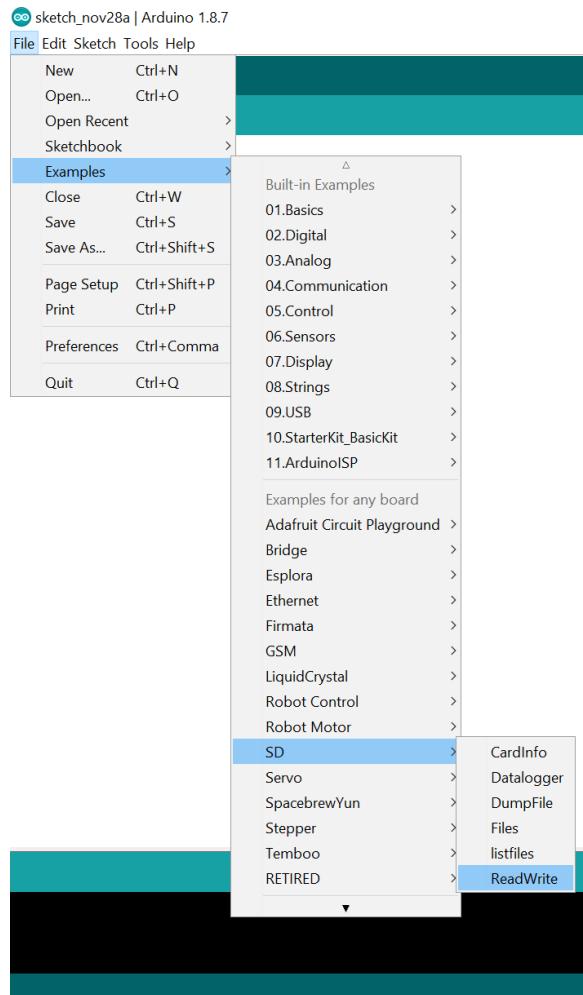


Abbildung 7.4: Example-Codes der Arduino IDE zum Lesen und Schreiben von SD-Karten.

Für eine übersichtlichere Programmstruktur und einfachere Handhabung wurden die Example-Codes für die korrekte Implementation angepasst und in Funktionen verpackt. Die Funktionen sind extern im Headerfile "SDCard.h"<sup>4</sup> deklariert und im SDCard.cpp initialisiert.

- `void getCardInformations();`
- `void readFileSDCard(String filename);`
- `void writeFileSDCard(double value2save, String filename);`
- `void deleteFileSDCard(String filename);`

Funktionen könnten noch beschrieben werden. Noch Abklären, ob die Fußzeilen nötig sind!

<sup>3</sup>`<*.h>` bezieht sich auf ein include-Verzeichnis unter dem Compiler-Installationsverzeichnis

<sup>4</sup>`"*.h"` bezieht sich relativ auf das aktive Projektverzeichnis

## 8 Kommunikationsmodul

## 9 Energieversorgung

## 10 Konzeptvalidierung

## Literatur

- [1] Arduino, *Arduino Mega 2560 Datasheet*, Techn. Ber., keine Angabe. Adresse: <https://www.robotshop.com/media/files/pdf/arduinomega2560datasheet.pdf> (besucht am 30. Nov. 2018).
- [2] Amazon. (keine Angabe). Ersatz Sensor Windgeschwindigkeit für Froggit WH1080 WH3080 WH1090, Adresse: [https://www.amazon.de/Ersatz-Sensor-Windgeschwindigkeit-Froggit-WH1080/dp/B00GGM5HEA/ref=pd\\_rhf\\_eetyp\\_p\\_img\\_1?encoding=UTF8&psc=1&refRID=4R5P7RRF8H7RC3A4KAHQ](https://www.amazon.de/Ersatz-Sensor-Windgeschwindigkeit-Froggit-WH1080/dp/B00GGM5HEA/ref=pd_rhf_eetyp_p_img_1?encoding=UTF8&psc=1&refRID=4R5P7RRF8H7RC3A4KAHQ) (besucht am 30. Nov. 2018).
- [3] ——, (keine Angabe). Benetech GM8908 Digital Anemometer, Adresse: <https://www.amazon.com/Benetech-GM8908-Digital-Anemometer-10/dp/B074TDR5YQ> (besucht am 30. Nov. 2018).
- [4] Lady Ada, „Micro SD Card Breakout Board Tutorial“, Adafruit Industries, Bericht, 17. Sep. 2018. Adresse: <https://cdn-learn.adafruit.com/downloads/pdf/adafruit-micro-sd-breakout-board-card-tutorial.pdf?timestamp=1543256513> (besucht am 28. Nov. 2018).
- [5] mikrocontroller.net. (28. Nov. 2018). Serial Peripheral Interface, Adresse: [https://www.mikrocontroller.net/articles/Serial\\_Peripheral\\_Interface](https://www.mikrocontroller.net/articles/Serial_Peripheral_Interface) (besucht am 28. Nov. 2018).
- [6] Wikipedia. (28. Nov. 2018). Serial Peripheral Interface, Adresse: [https://de.wikipedia.org/wiki/Serial\\_Peripheral\\_Interface](https://de.wikipedia.org/wiki/Serial_Peripheral_Interface) (besucht am 28. Nov. 2018).
- [7] SanDisk. (keine Angabe). SanDisk microSDHC Speicherkarte, Adresse: <https://www.sandisk.de/home/memory-cards/microsd-cards/sandisk-microsd> (besucht am 27. Nov. 2018).
- [8] ——, (keine Angabe). SD/SDHC/SDXC Spezifikationen und Kompatibilitäten, Adresse: [https://kb-de.sandisk.com/app/answers/detail/a\\_id/8317/~/sd%2Fsdhc%2Fsdxc-spezifikationen-und-kompatibilit%C3%A4ten](https://kb-de.sandisk.com/app/answers/detail/a_id/8317/~/sd%2Fsdhc%2Fsdxc-spezifikationen-und-kompatibilit%C3%A4ten) (besucht am 27. Nov. 2018).
- [9] Arduino. (2018). ATmega2560-Arduino Pin Mapping, Adresse: <https://www.arduino.cc/en/Hacking/PinMapping2560> (besucht am 30. Nov. 2018).

## A Lastenheft



Fachhochschule Nordwestschweiz  
Hochschule für Technik

### **Ausschreibung Studierendenprojekt P5/P6 Studiengang Elektro- und Informationstechnik**

<b>Titel:</b>
Wetterstation mit Solar Energie
<b>Betreuer:</b>
Prof. Dr. Taoufik Nouri (Institut für Mobile und Verteilte Systeme)
<b>Auftraggeber:</b>
Prof. Dr. Taoufik Nouri (Institut für Mobile und Verteilte Systeme)
<b>Aufgabenbeschreibung:</b>
<p>Ausgangslage:          Wetterstation sind viele verlangt besonders im Gebiete ohne Strom. Wir schlagen solche Möglichkeit zu realisieren.</p> <p>Zielsetzung:</p> <ol style="list-style-type: none"> <li>1. Diese Wetterstation misst Regen, Wind- Geschwindigkeit, -Richtung, Temperatur, Sonnenlicht, Feuchtigkeit, Zeit usw.</li> <li>2. Sie ist dotiert mit verschiedener Kommunikation Module wie GPS, SIM Karte.</li> <li>3. Sie ist fern abfragbar durch Handy</li> <li>4. Sie speichert regelmässig die verschiedenen Parameter (Journal).</li> <li>5. Sie ist komplett automatisiert z.B. Regenwasser wird automatisch ausgeleert.</li> </ol> <p>Schlüsselwörter: Energie, Mikrokontroller, Programmierung, Elektronik</p>

## B ATmega2560-Arduino Pin Mapping

**Tabelle B.1:** ATmega2560-Arduino Pin Mapping Tabelle [9]

Pin Number	Pin Name	Mapped Pin Name
1	PG5 ( OC0B )	Digital pin 4 (PWM)
2	PE0 ( RXD0/PCINT8 )	Digital pin 0 (RX0)
3	PE1 ( TXD0 )	Digital pin 1 (TX0)
4	PE2 ( XCK0/AIN0 )	
5	PE3 ( OC3A/AIN1 )	Digital pin 5 (PWM)
6	PE4 ( OC3B/INT4 )	Digital pin 2 (PWM)
7	PE5 ( OC3C/INT5 )	Digital pin 3 (PWM)
8	PE6 ( T3/INT6 )	
9	PE7 ( CLKO/ICP3/INT7 )	
10	VCC	VCC
11	GND	GND
12	PH0 ( RXD2 )	Digital pin 17 (RX2)
13	PH1 ( TXD2 )	Digital pin 16 (TX2)
14	PH2 ( XCK2 )	
15	PH3 ( OC4A )	Digital pin 6 (PWM)
16	PH4 ( OC4B )	Digital pin 7 (PWM)
17	PH5 ( OC4C )	Digital pin 8 (PWM)
18	PH6 ( OC2B )	Digital pin 9 (PWM)
19	PB0 ( SS/PCINT0 )	Digital pin 53 (SS)
20	PB1 ( SCK/PCINT1 )	Digital pin 52 (SCK)
21	PB2 ( MOSI/PCINT2 )	Digital pin 51 (MOSI)
22	PB3 ( MISO/PCINT3 )	Digital pin 50 (MISO)
23	PB4 ( OC2A/PCINT4 )	Digital pin 10 (PWM)
24	PB5 ( OC1A/PCINT5 )	Digital pin 11 (PWM)
25	PB6 ( OC1B/PCINT6 )	Digital pin 12 (PWM)
26	PB7 ( OC0A/OC1C/PCINT7 )	Digital pin 13 (PWM)
27	PH7 ( T4 )	
28	PG3 ( TOSC2 )	
29	PG4 ( TOSC1 )	
30	RESET	RESET
31	VCC	VCC
32	GND	GND
33	XTAL2	XTAL2
34	XTAL1	XTAL1
35	PL0 ( ICP4 )	Digital pin 49

<b>Pin Number</b>	<b>Pin Name</b>	<b>Mapped Pin Name</b>
36	PL1 ( ICP5 )	Digital pin 48
37	PL2 ( T5 )	Digital pin 47
38	PL3 ( OC5A )	Digital pin 46 (PWM)
39	PL4 ( OC5B )	Digital pin 45 (PWM)
40	PL5 ( OC5C )	Digital pin 44 (PWM)
41	PL6	Digital pin 43
42	PL7	Digital pin 42
43	PD0 ( SCL/INT0 )	Digital pin 21 (SCL)
44	PD1 ( SDA/INT1 )	Digital pin 20 (SDA)
45	PD2 ( RXDI/INT2 )	Digital pin 19 (RX1)
46	PD3 ( TXD1/INT3 )	Digital pin 18 (TX1)
47	PD4 ( ICP1 )	
48	PD5 ( XCK1 )	
49	PD6 ( T1 )	
50	PD7 ( T0 )	Digital pin 38
51	PG0 ( WR )	Digital pin 41
52	PG1 ( RD )	Digital pin 40
53	PC0 ( A8 )	Digital pin 37
54	PC1 ( A9 )	Digital pin 36
55	PC2 ( A10 )	Digital pin 35
56	PC3 ( A11 )	Digital pin 34
57	PC4 ( A12 )	Digital pin 33
58	PC5 ( A13 )	Digital pin 32
59	PC6 ( A14 )	Digital pin 31
60	PC7 ( A15 )	Digital pin 30
61	VCC	VCC
62	GND	GND
63	PJ0 ( RXD3/PCINT9 )	Digital pin 15 (RX3)
64	PJ1 ( TXD3/PCINT10 )	Digital pin 14 (TX3)
65	PJ2 ( XCK3/PCINT11 )	
66	PJ3 ( PCINT12 )	
67	PJ4 ( PCINT13 )	
68	PJ5 ( PCINT14 )	
69	PJ6 ( PCINT 15 )	
70	PG2 ( ALE )	Digital pin 39
71	PA7 ( AD7 )	Digital pin 29
72	PA6 ( AD6 )	Digital pin 28
73	PA5 ( AD5 )	Digital pin 27
74	PA4 ( AD4 )	Digital pin 26
75	PA3 ( AD3 )	Digital pin 25
76	PA2 ( AD2 )	Digital pin 24
77	PA1 ( AD1 )	Digital pin 23
78	PA0 ( AD0 )	Digital pin 22
79	PJ7	
80	VCC	VCC

<b>Pin Number</b>	<b>Pin Name</b>	<b>Mapped Pin Name</b>
81	GND	GND
82	PK7 ( ADC15/PCINT23 )	Analog pin 15
83	PK6 ( ADC14/PCINT22 )	Analog pin 14
84	PK5 ( ADC13/PCINT21 )	Analog pin 13
85	PK4 ( ADC12/PCINT20 )	Analog pin 12
86	PK3 ( ADC11/PCINT19 )	Analog pin 11
87	PK2 ( ADC10/PCINT18 )	Analog pin 10
88	PK1 ( ADC9/PCINT17 )	Analog pin 9
89	PK0 ( ADC8/PCINT16 )	Analog pin 8
90	PF7 ( ADC7 )	Analog pin 7
91	PF6 ( ADC6 )	Analog pin 6
92	PF5 ( ADC5/TMS )	Analog pin 5
93	PF4 ( ADC4/TMK )	Analog pin 4
94	PF3 ( ADC3 )	Analog pin 3
95	PF2 ( ADC2 )	Analog pin 2
96	PF1 ( ADC1 )	Analog pin 1
97	PF0 ( ADC0 )	Analog pin 0
98	AREF	Analog Reference
99	GND	GND
100	AVCC	VCC

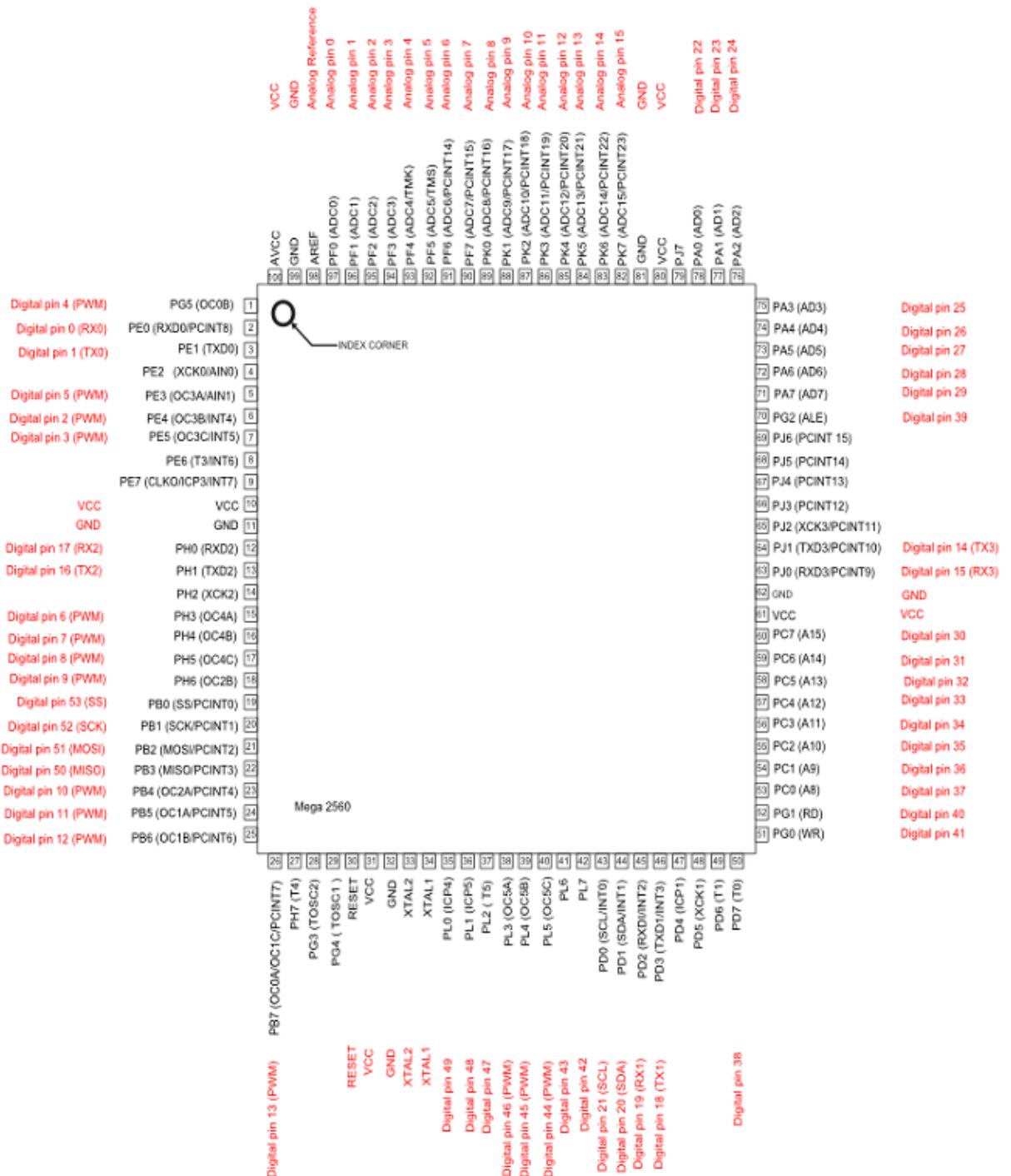


Abbildung B.1: ATmega2560 Pin Layout