



Angewandte Informatik

Projektbericht

über das Thema

Entwicklung einer mikrokontrollergesteuerten Wetterstation

Autor: Patrick Frey, Patrik Donauer, Julian Meisel, Luca Harthmuth

Prüfer: Dr. Hubert Zitt

Abgabedatum: 09.02.2017

Erklärung zur Ausarbeitung

Hiermit erklären wir, *Patrick Frey 863457, Patrik Donauer 868038, Julian Meisel 868312, Luca Harthmuth 868262*, dass wir die vorliegende Ausarbeitung selbstständig und ohne fremde Hilfe angefertigt haben und keine anderen als in der Abhandlung angegebenen Hilfen benutzt haben; dass wir die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet haben. Wir sind uns bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.

Ort, Datum

Unterschrift

Ort, Datum

Unterschrift

Ort, Datum

Unterschrift

Ort, Datum

Unterschrift

1 Kurzfassung

Im Rahmen des Studienprojektes wird mit Hilfe von Mikrocontrollerboards und verschiedenen Sensoren eine Wetterstation für den Außenbereich entwickelt. Sie soll dazu in der Lage sein, die Lufttemperatur, Bodentemperatur, Luftfeuchtigkeit, Regen, Windgeschwindigkeit, Helligkeit und Luftdruck zu messen und auszuwerten. Außerdem soll mit Hilfe von Steuerbaren Relais eine Pumpenanlage an- und ausgeschaltet werden können die den Garten bewässern soll. Diese Wetterstation wird dann mittels einer REST-Anbindung in ein OpenSource HomeAutomationSystem eingebettet.

2 Inhaltsverzeichnis

1	Kurzfassung	1
2	Inhaltsverzeichnis	2
3	Einleitung	4
4	Aufgabenstellung	4
4.1	Verwendete Hardware	4
4.2	Verwendete Software	4
4.2.1	Verwendete Bibliotheken	4
4.3	Grundlegende Überlegungen	5
5	Sensoren	5
5.1	Luftdruck	5
5.2	Luftfeuchte	5
5.3	Lufttemperatur	6
5.4	Licht	7
5.5	Regen	7
5.6	Bodentemperatur	8
5.6.1	1-Wire	8
5.7	Bodenfeuchte	9
5.8	Anemometer	10
5.9	Relais	10
6	Moduleinheiten	12
6.1	Modul 1 - Boden, Luft	12
6.2	Modul 2 - Regen, Licht	12
6.3	Modul 3 - Relais	12
6.4	Modul 4 - Bewegungsmelder	13
7	Hauptprogramm	14
7.1	HomeAssistant	14
7.2	Einbindung in HomeAssistant	14
7.3	HomeAssistant - Anzeige	15
8	Probleme	16
8.1	Fehlende Datenblätter	16
8.2	Kurzschluss	16
9	Fazit	16
10	Listings	18
11	Abbildungsverzeichnis	20
12	Tabellenverzeichnis	20

13 Listing-Verzeichnis

20

3 Einleitung

Mit heute verfügbaren Mikrocontrollern und Sensoren können allerhand Dinge erstellt werden. Hieraus entstand die Idee, eine Wetterstation zu entwickeln. Hauptaugenmerk hierbei sollte auf die Integration in ein bestehendes Hausautomatisierungssystem - HomeAssistant - gelegt werden.

4 Aufgabenstellung

Es soll eine Wetterstation erstellt werden, die mit Hilfe von Mikrocontrollern gesteuert wird. Diese wird dann in ein bestehendes OpenSource HomeAutomationSystem integriert.

4.1 Verwendete Hardware

Zur Umsetzung des Projekts wurde folgende Hardware ausgewählt:

- ArduinoMega mit Ethernetshield
- Regensensor
- Bodenfeuchtesensor
- Lichtsensor
- Temperatursensor
- Luftdrucksensor
- Anemometer (Windsensor)
- Relais

4.2 Verwendete Software

Zur Programmierung der Mikrocontroller wird die Arduino-Entwicklungsumgebung genutzt. Das Hausautomatisierungssystem wird mittels des OpenSource Projektes HomeAssistant erstellt und die Wetterstation darin eingebettet. Hierzu wird eine Custom-Komponente mittels Python erstellt.

4.2.1 Verwendete Bibliotheken

Es wurden folgende Bibliotheken für die Programmierung des Mikrocontrollers verwendet:

```
1  #include <SFE_BMP180.h>
2  #include <SPI.h>
3  #include <Ethernet.h>
4  #include <aREST.h>
5  #include <DHT.h>
```

```
6  #include <OneWire.h>
7  #include <avr/wdt.h>
```

Listing 1: Bibliotheken

4.3 Grundlegende Überlegungen

Generell gibt es eine einheitliche Belegung der Kabelfarben und Anschlüsse:

Kabelfarbe	Belegung
rot	vcc
schwarz	gnd

5 Sensoren

Das Auslesen der Sensoren sowie eine Kurzbeschreibung zu den einzelnen Sensoren steht im Folgenden.

5.1 Luftdruck

Zur Luftdruckmessung wird ein BMP180 verwendet. Dieser Sensor wird von Bosch gefertigt und ist daher sehr genau, außerdem ist er sehr gut zur Verwendung in Wetterstationen oder in kleineren Modellfluggeräten geeignet, da er einen geringen Stromverbrauch (ca. 5 mA) hat und mit seinen 4x4cm sehr klein ist. Sein Messbereich ist von 300-1100hPA und ist für unseren Gebrauch ausreichend. Die Betriebsspannung liegt zwischen 1,8 V und 3,6 V. Er funktioniert in einem Temperaturbereich von 0 - 65 Grad Celsius. Die Werte müssen mittels eines I2C-Bus ausgelesen werden. Die Programmierung des Sensors ist in Listing 8 zu sehen.



Abbildung 1: Luftdrucksensor - BMP180

5.2 Luftfeuchte

Als Luftfeuchtesensor wird ein DHT22 verwendet, hierbei handelt es sich um einen digitalen Temperatur- und Luftfeuchtesensor. Hier wird mittels eines kapazitiven Feuchtigkeitsfühlers die Luftfeuchte und über einen Thermistor die Lufttemperatur gemessen. Der

Sensor gibt ein digitales Signal weiter, welches dann ausgelesen werden kann. Wichtig bei diesem Sensor ist, dass der zeitliche Ablauf stimmt um die Daten genau zu erfassen. Der Messbereich der Feuchtigkeit liegt im Bereich von 0 bis 100% relativer Feuchtigkeit und hat eine Messgenauigkeit von $\pm 2\%$ rF.



Abbildung 2: Luftfeuchtigkeits- & Temperatursensor DHT22

```
1 int getAirHum() {
2     int humidity = dht.readHumidity();           // Read humidity (percent)
3     if (isnan(humidity)) {
4         humidity = dht.readHumidity();
5     }
6     if (isnan(humidity)) {
7         return -1;
8     }
9     Serial.print(" Air Humidity: ");
10    Serial.println(humidity);
11    return humidity;
12 }
```

Listing 2: getAirHum()

5.3 Lufttemperatur

Hierfür wird ebenfalls der oben Genannte DHT22 ebenfalls genutzt. Es wird mittels eines kapazitiven Feuchtigkeitsfühlers die Luftfeuchte und über einen Thermistor die Lufttemperatur gemessen. Der Sensor gibt ein digitales Signal weiter, welches dann ausgelesen werden kann. Wichtig bei diesem Sensor ist, dass der zeitliche Ablauf stimmt, um die Daten genau zu erfassen. Der Sensor hat eine Temperatur Messgenauigkeit von $\pm 0,5$ Grad Celsius und arbeitet im Bereich von -40 bis $+80$ Grad Celsius.

```
1 float getAirTemp() {
2     float temp_f = dht.readTemperature(true);    // Read temperature as
        Fahrenheit
3     float temp_c = (((temp_f - 32.0) * 5 / 9) - 1); // Convert temperature
        to Celsius + Correction
4     if (isnan(temp_c)) {
5         temp_f = dht.readTemperature(true);      // if read fails try one
        more time
6         temp_c = (((temp_f - 32.0) * 5 / 9) - 1);
7     }
```

```
8   if (isnan(temp_c)){
9       return -100.0;
10  }
11  Serial.print(" Air Temperature: ");
12  Serial.println(temp_c);
13  return temp_c;
14 }
```

Listing 3: getAirTemp()

5.4 Licht

Hierzu nutzen wir eine einfache Photozelle, mit einem vorgeschalteten A/D-Wandler, ein eigenes Datenblatt gibt es hierfür für die von uns bestellte Zelle leider nicht. Eine Photozelle kann die Intensität von Licht mit einer geeigneten Wellenlänge messen. Die aktuelle Helligkeit wird folgendermaßen ausgewertet:

```
1  int getBrightness(){
2      double brightness = ((((-0.0975)*(analogRead(BRIGHTNESSANALOGPIN))
3          +100)*20);
4      boolean bright = digitalRead(BRIGHTNESSPIN);
5      Serial.print(" Brightness: ");
6      Serial.println(brightness);
7      Serial.print(" Bright: ");
8      Serial.println(bright);
9      return brightness;
10 }
```

Listing 4: getBrightness()

5.5 Regen

Für den Regensensor gibt es auch wieder kein spezifisches Datenblatt. Wir verwenden hierzu einen kapazitiven Regen- / Feuchtigkeitssensor inklusive Auswerteelektronik. Je nach Befeuchtung des Sensors gibt er hier eine Ausgabe über den analogen und den digitalen TTL-Ausgang ab. Der Sensor arbeitet mit einer Betriebsspannung von 3,3-5 V.

```
1  int getPrecip(){
2      int rain1 = ! digitalRead(RAIN1PIN);
3      int rain2 = ! digitalRead(RAIN2PIN);
4
5      int rain1a = analogRead(RAIN1ANALOGPIN);
6      int rain2a = analogRead(RAIN2ANALOGPIN);
7      Serial.print(" Raining: ");
8      Serial.println(rain1 && rain2);
9      return (rain1 && rain2);
10 }
```

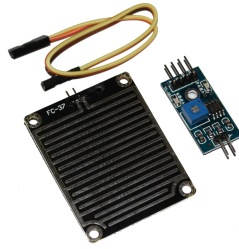


Abbildung 3: Regensensor

5.6 Bodentemperatur

5.6.1 1-Wire

Zum Messen der Bodentemperatur wird ein wasserdichter Temperatursensor basierend auf dem 1-Wire DS18B20 mit einer Sonde aus Edelstahl verwendet. Hiermit kann man die Temperatur in feuchter und nasser Umgebung messen. Dank dem 1-Wire Protokoll wird nur ein digitaler Anschluss zum ansteuern des Sensors benötigt und eine Spannungsversorgung. Der Sensor hat einen Arbeitsbereich von -55 Grad Celsius und +125 Grad Celsius. Er läuft mit einer Spannung von 3,3-5 V und hat eine Messgenauigkeit von +/- 0,5 Grad Celsius im Bereich von -10 Grad Celsius bis +85 Grad Celsius. Die Programmierung des Sensors ist in Listing 9 zu sehen.



Abbildung 4: Bodentempertursensor DS18B20

5.7 Bodenfeuchte

Der Sensor zum messen der Bodenfeuchtigkeit ist ein YL-69 Feuchtigkeitssensor, es handelt sich hierbei um ein Hygrometer, dass zum Messen von Bodenfeuchte und für Füllstandsmessungen verwendet werden kann. Leider liegt uns hierzu auch kein Datenblatt vor. Der Sensor verfügt über einen digitalen- und einen analogen Output und ein Potentiometer. Zum Auswerten der Daten wird noch zusätzlich eine Auswertungselektronik benötigt, hier wird ein LM393-IC basierter AD-Wandler verwendet.

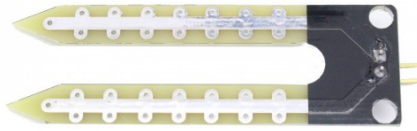


Abbildung 5: Feuchtigkeitssensor YL-69



Abbildung 6: AD-Wandler

```
1 double getSoilMois() {  
2     double soilMois = (log10(analogRead(SOILMOISANALOGPIN)) / 0.06930720659)  
3     ;  
4     boolean soilMoisDry = digitalRead(SOILMOISPIN);  
5     Serial.print(" Soil Moisture: ");  
6     Serial.println(soilMois);  
7     Serial.print(" Moist: ");  
8     Serial.println(soilMoisDry);  
9     return soilMois;  
}
```

Listing 5: getSoilMois()

5.8 Anemometer

Hierbei handelt es sich um einen handelsüblichen Windstärkesensor, der auch für andere Wetterstationen genutzt werden kann. Dieser kann über die eingebaute Schnittstelle einfach ausgelesen werden, wie in Listing 10 zu sehen ist.

Außerdem wurde eine Interrupt-Routine für das Zählen der Umdrehungen des Anemo-



Abbildung 7: Anemometer

meters initialisiert:

```
1 void countWind() {
2     windCounter ++;
3 }
```

Listing 6: countWind()

Um eine einigermaßen genaue Messung zu erhalten, müssen Sensoren kalibriert werden. Um zu testen ob das Anemometer richtig funktioniert, haben wurde es, nach dem die Auswertung programmiert wurde, auf einem Autodach befestigt um dann den Fahrtwind zu messen. Hier kann man natürlich nur testen ob die Werte ungefähr in die richtige Richtung gehen, da andere Aspekte wie z. B. Windböen die Messung verfälschen können.

5.9 Relais

Es wurden zwei Relais verbaut mit denen weitere Zusatzkomponenten angesteuert werden können, wie z.B. eine Bewässerungspumpe zur Bewässerung der Gartenanlage.

```
1 int switch1(String param){
2     Serial.println("switch1.param : "+param);
3
4     // Get state from param
5     if (param == "n"){
6         digitalWrite(SWITCH1PIN, LOW);
7         switch1_state = "on";
8         Serial.println("switch1 : on");
9     }else if (param == "ff"){
10        digitalWrite(SWITCH1PIN, HIGH);
11        switch1_state = "off";
```

```
12     Serial.println(" switch1  :  off");  
13   }  
14 }
```

6 Moduleinheiten

Es werden mehrere verschiedene Sensoren, die einander zugehörig sind zu einzelnen Moduleinheiten zusammen gesetzt. Innerhalb der Moduleinheiten sind die Sensoren nur mit Steckverbindungen miteinander verbunden, damit falls ein Defekt vorliegt, dieser einfach ausgetauscht werden kann. Die weiteren Leitungen, die von den Modulen zur Zentraleinheit gehen wurden fest miteinander verbunden. Um zu vermeiden, dass Feuchtigkeit die Sensoren zerstört, wurden diese alle in wasserdichte Außengehäuse verbaut.

6.1 Modul 1 - Boden, Luft

Bei Modul 1 handelt es sich um eine Kombination aus Luftdruck-, Luftfeuchte- und Temperatursensor. Die Verteilung der Anschlüsse ist wie folgt:

Kabelfarbe	Belegung
grün	dht22
orange	bmpsda
blau	bmpsdl

6.2 Modul 2 - Regen, Licht

Modul 2 beinhaltet die Überwachung von Regen und Licht. Hierin sind die Regen- und Lichtsensoren enthalten. Der Regensensor wurde so konfiguriert, dass ausgeschlossen wird, dass sich einzelne Regentropfen auf dem Sensor abgelagert haben. Er löst also nur aus, wenn beide Sensoren auslösen.

Kabelfarbe	Belegung
grün	analogRegen
orange	digitalRegen
blau	analogLicht
gelb	digitalLicht

6.3 Modul 3 - Relais

Die Relais wurden in ein einzelnes Modul gepackt, damit dies individuell einsetzbar ist. Somit kann sichergestellt werden, dass die Stromversorgung für bestimmte Geräte, wie z. B. die Bewässerungspumpe unabhängig vom Standort der Wetterstation im Garten installiert werden kann.

6.4 Modul 4 - Bewegungsmelder

Der Bewegungsmelder wurde in die Wetterstation unter anderem wegen der Gartenbewässerung eingebaut, so soll diese - falls sie gerade aktiv ist - stoppen, sobald jemand in ihre Nähe kommt.

7 Hauptprogramm

7.1 HomeAssistant

HomeAssistant ist ein OpenSource Hausautomatisierungssystem das auf Python3 basiert. Es läuft auf verschiedenen Betriebssystemen wie z.B: Windows10, MacOS, Linux. Hier hat man die Möglichkeit die verschiedensten Geräte miteinander in den HomeAssistant einzubinden um sein Zuhause zu überwachen, zu kontrollieren und auch um bestimmte automatisierte Abläufe einzustellen.

7.2 Einbindung in HomeAssistant

Die Daten werden über eine REST-Schnittstelle dem HomeAssistant-System zur Verfügung gestellt, somit können immer die aktuellsten Werte der Sensoren ausgelesen werden und hier weiter verarbeitet werden.

```
1 // rest init
2 rest.variable("wind_speed",&windSpeed);
3 rest.variable("air_temperature",&airTemp);
4 rest.variable("air_humidity",&airHum);
5 rest.variable("air_pressure", &airPres);
6 rest.variable("soil_temperature", &soilTemp);
7 rest.variable("soil_moisture", &soilMois);
8 rest.variable("motion_detector", &motion);
9 rest.variable("brightness", &brightness);
10 rest.variable("precipitation", &precip);
11 rest.variable("switch1", &switch1_state);
12 rest.variable("switch2", &switch2_state);
13 rest.function("switch1",switch1);
14 rest.function("switch2",switch2);
15 // Give name & ID to the device (ID should be 6 characters long)
16 rest.set_id("000001");
17 rest.set_name("arduino_weatherstation");
```

Listing 7: RestInit

Diese Werte werden dann unter anderem auf einer Übersichtsseite angezeigt, sowie für verschiedene Szenarien, die man benutzerdefiniert einstellen kann verwendet werden. Ein Beispiel: "Wenn es beginnt zu regnen sollen die Fenster im Haus geschlossen werden".

7.3 HomeAssistant - Anzeige

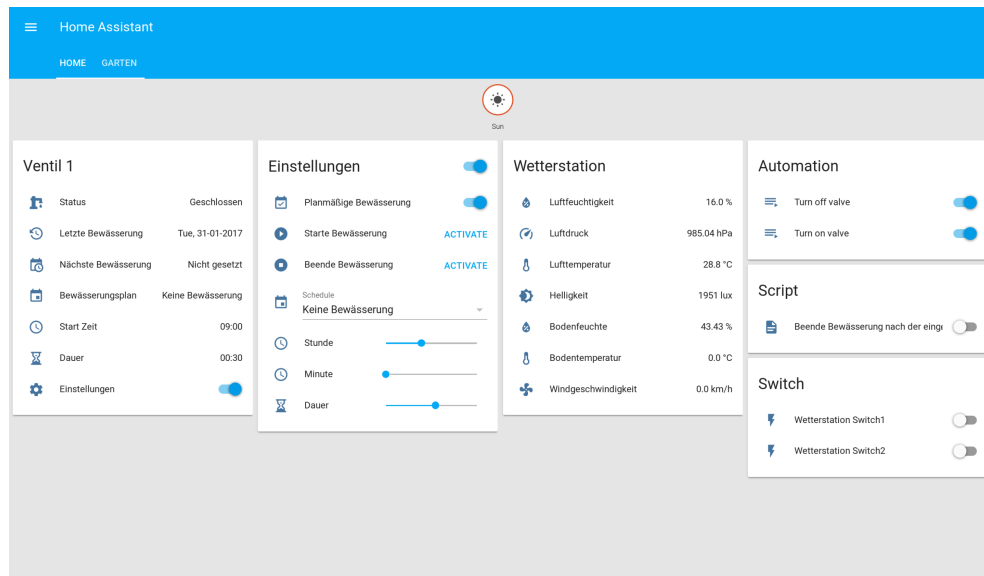


Abbildung 8: HomeAssistant - Home

Die in Abbildung 8 gezeigte Übersicht bietet das HomeAssistant-System im Standardfall. Es handelt sich hierbei um eine Gesamtübersicht des Systems wenn man es über das Netzwerk anwählt. Diese Übersicht kann nach belieben konfiguriert werden. Etwa in eine kleinere Übersicht über die Gartenanlage, wie zum Beispiel: Ist die Bewässerung gerade aktiv? oder andere. Zu sehen in Abbildung 9.

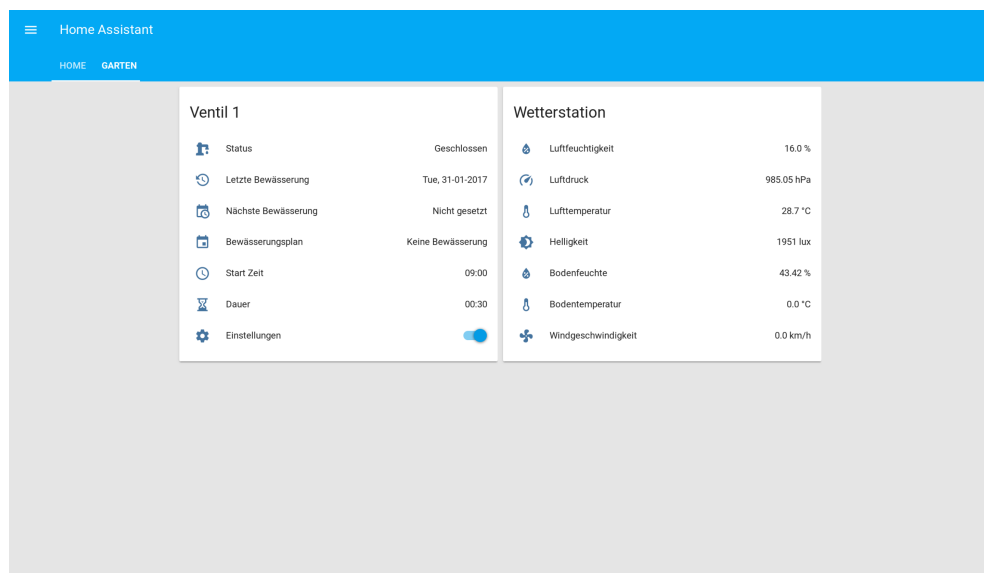


Abbildung 9: HomeAssistant - Garten

Die Bewässerung kann individuell angepasst werden. Es kann eine genaue Planung, wann bewässert werden soll, festgelegt werden mit Hilfe eines einstellbaren Szenarios, ähnlich angesetzt wie ein Termineintrag im Kalender.

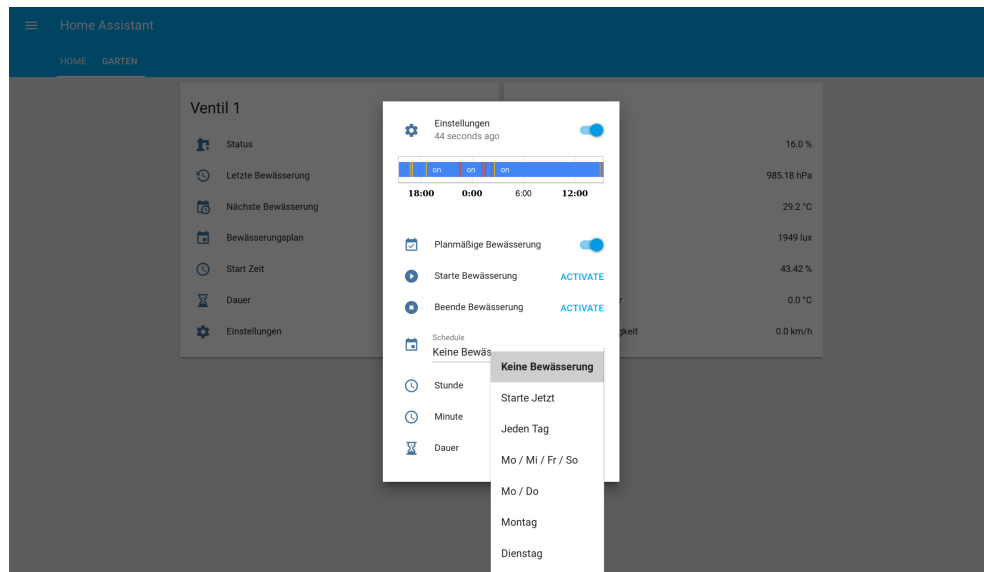


Abbildung 10: HomeAssistant - Bewässerung

8 Probleme

8.1 Fehlende Datenblätter

Zu ein paar der von genutzten Sensoren gibt es keine Datenblätter, so musste sich hier auf die Informationen verlassen werden, die vom Händler bereit gestellten Informationen verlassen werden.

8.2 Kurzschluss

Da die Sensoren mittels Steckverbindungen - RJ11 - mit einander verbunden werden sollten, wurden diese in die Verkabelung mit eingebaut, allerdings wurde nicht darauf geachtet, dass ein Stecker auch "falsch herum" eingesteckt werden kann. Dadurch mussten dann einige der Sensoren ausgetauscht werden. Um dies in Zukunft zu vermeiden, wurde die Verkabelung derart geändert, dass ein falsches Einstecken der Verbindungen keine Kurzschlüsse erzeugen kann. Der Sensor funktioniert dann lediglich nur nicht.

9 Fazit

Abschließend ist zu sagen, dass das Projekt Wetterstation einen sehr guten Einblick in die arbeit mit Mikrokontrollern sowie die zusammenarbeit verschiedener Systeme geliefert

hat. Die Wetterstation arbeitet jetzt autark in einem bestehenden Hausautomatisierungssystem.

Anschließend möchten wir noch ein bisschen in die Zukunft schauen. Die neuen Systeme mit Sprachsteuerung wie Alexa, Siri und GoogleHome bieten im Bereich der Heimautomatisierung neue Möglichkeiten. So wäre es praktisch, wenn einem der Wecker gleich noch das aktuelle Wettersituation und die Vorhersagen mitteilen würde um sich entsprechend zu kleiden. Als Reaktion auf aufkommenden Regen die Markise einzufahren und die Rolläden und die Dachfenster zu schließen. Mit einem interaktiven Beleuchtungssysteme wäre es zum Beispiel auch möglich, die Beleuchtungsstimmung so auf eine angenehme Lichtverhältnis und Lichtfarbe anzupassen um bei schlechtem Wetter die Bewohner positiv zu beeinflussen und so Depressionen vorzubeugen und an sonnigen Tagen Energie zu sparen.

10 Listings

```
1 double getPressure() {
2   char status;
3   double T,P,p0,a;
4   status = pressure.startTemperature();
5   if (status != 0) {
6     delay(status);
7     status = pressure.getTemperature(T);
8     if (status != 0) {
9       Serial.print("temperature: ");
10      Serial.print(T,2);
11      Serial.print(" deg C, ");
12      Serial.print((9.0/5.0)*T+32.0,2);
13      Serial.println(" deg F");
14      status = pressure.startPressure(3);
15      if (status != 0) {
16        delay(status);
17        status = pressure.getPressure(P,T);
18        if (status != 0) {
19          Serial.print("absolute pressure: ");
20          Serial.print(P,2);
21          Serial.print(" mb, ");
22          Serial.print(P*0.0295333727,2);
23          Serial.println(" inHg");
24        }
25        else Serial.println("error retrieving pressure measurement\n");
26      }
27      else Serial.println("error starting pressure measurement\n");
28    }
29    else Serial.println("error retrieving temperature measurement\n");
30  }
31  else Serial.println("error starting temperature measurement\n");
32  return P;
33 }
```

Listing 8: getPressure()

```
1 double getSoilTemp() {
2   oneWire.reset_search();
3   double temp = 0.00;
4   byte i;
5   byte present = 0;
6   byte data[12];
7   byte addr[8];
8   //find a device
9   if (!oneWire.search(addr)) {
10    oneWire.reset_search();
11    Serial.println("Error SoilTemp");
12  }
13
14  if (OneWire::crc8( addr, 7) != addr[7]) {
15    Serial.println("Error SoilTemp");
16  }
17
18  if (addr[0] != oneWire18S20_ID && addr[0] != oneWire18B20_ID) {
```

```
19 Serial.println(" Error SoilTemp");
20 }
21
22 oneWire.reset();
23 oneWire.select(addr);
24 // Start conversion
25 oneWire.write(0x44, 1);
26 // Wait some time...
27 delay(850);
28 present = oneWire.reset();
29 oneWire.select(addr);
30 // Issue Read scratchpad command
31 oneWire.write(0xBE);
32 // Receive 9 bytes
33 for ( i = 0; i < 9; i++) {
34   data[i] = oneWire.read();
35 }
36 // Calculate temperature value
37 temp = ( (data[1] << 8) + data[0] ) * 0.0625;
38 Serial.print(" Soil Temperature: ");
39 Serial.println(temp);
40 return temp;
41 }
```

Listing 9: getSoilTemp()

```
1 void startMeasureWind(){
2   //starten des messzeitraums
3   Serial.print("starting wind-counter for ");
4   Serial.print(measureTime);
5   Serial.println(" seconds");
6   //zaehler auf 0 stellen
7   windCounter = 0;
8   time = millis();
9   //zaehl-interrupt aktivieren
10  attachInterrupt(digitalPinToInterrupt(WINDPIN),countWind,FALLING);
11 }
12
13 void stopMeasureWind(){
14   //zaehl-interrupt deaktivieren
15   detachInterrupt(1);
16   //zeit bestimmen
17   time = (millis() - time) / 1000;
18
19   Serial.print("Time elapsed : ");
20   Serial.println(time);
21   Serial.print("Wind counts = ");
22   Serial.println(windCounter);
23   //berechnen der geschwindigkeit
24   windSpeed = (float)windCounter/4 / (float)measureTime * windFactor;
25   Serial.print("Wind Speed = ");
26   Serial.print(windSpeed);
27   Serial.println(" km/h");
28 }
```

Listing 10: MeasureWind()

11 Abbildungsverzeichnis

Abb. 1	Luftdrucksensor - BMP180	5
Abb. 2	Luftfeuchtigkeits- & Temperatursensor DHT22	6
Abb. 3	Regensensor	8
Abb. 4	Bodentemperatursensor DS18B20	8
Abb. 5	Feuchtigkeitssensor YL-69	9
Abb. 6	AD-Wandler	9
Abb. 7	Anemometer	10
Abb. 8	HomeAssistant - Home	15
Abb. 9	HomeAssistant - Garten	15
Abb. 10	HomeAssistant - Bewässerung	16

12 Tabellenverzeichnis

13 Listing-Verzeichnis

Lst. 1	Bibliotheken	4
Lst. 2	getAirHum()	6
Lst. 3	getAirTemp()	6
Lst. 4	getBrightness()	7
Lst. 5	getSoilMois()	9
Lst. 6	countWind()	10
Lst. 7	RestInit	14
Lst. 8	getPressure()	18
Lst. 9	getSoilTemp()	18
Lst. 10	MeasureWind()	19