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 habe; 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

I Kurzfassung

In der folgenden Ausarbeitung wird beschrieben wie mit hilfe von mikrokontrollern eine Wetterstation für den Außenbereich mit verschiedenen Einzelmodulen erstellt und programmiert werden kann. Sie soll dazu in der Lage sein, die Lufttemperatur, Bodentemperatur, Luftfeuchtigkeit, Regen, Windgeschwindigkeit, Helligkeit, Luftdruck zu messen und auszuwerden. 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.

II Inhaltsverzeichnis

III Abbildungsverzeichnis

IV Tabellenverzeichnis

V Listing-Verzeichnis

1 Aufgabenstellung

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

1.1 Verwendete Hardware

Zur Umsetzung des Projekts wurde folgende Hardware ausgewählt:

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

1.2 Verwendete Software

Zur Porgrammierung der Mikrokontroller wird die Arduino-Entwicklungsumgebung genutzt. Das Hausautomatisierungssystem wird mittels des OpenSource Projektes HomeAssistant erstellt und die Wetterstation darin eingebettet.

1.2.1 Verwendete Bibliotheken

Es wurden folgende Bibliotheken für die Programmierung 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

1.3 Grundlegende Überlegungen

Generell gibt es eine einheitliche Belegung der Kabelfarben und Anschlüsse:

Kabelfarbe	Belegung
rot	vcc
schwarz	gnd

2 Sensoren

Das Auslesen der Sensoren sowie eine kurzbeschreibung zu den Einzelnen Sensoren steht im Folgenden.

2.1 Luftdruck

Zur Luftdruckmessung haben wir einen BMP180 entschieden. Dieser sensor ist sehr gut zur Verwendung in Wetterstationen oder in kleineren Modellfluggeräten geeignet, da er einen geringen Stromverbrauch(ca. 5mA) hat und mit seinen 4x4cm sehr klein ist. Sein Messbereich ist von 300-1100hPA und ist für unseren gebrauch ausreichend. Die Betriebsspannung liegt zwischen 1,8V und 3,6V. Er funktioniert in einem Temperaturbereich von 0 - 65 grad Celsius. Die Werte müssen mittels eins I2C-Bus ausgelesen werden.

Abbildung 1: Luftdrucksensor - BMP180

```
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 2: getPressure()

2.2 Luftfeuchte

Als Luftfeuchte Sensor wird ein DHT22 hierbei handelt es sich um einen digitalen Temperatur und Luftfeuchtesensor. Hier wird mittels eins 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 Humidity liegt im Bereich von 0 bis 100% relativer Feuchtigkeit und hat eine Messgenauigkeit von +- 2% rF.

Abbildung 2: Luftfeuchtigkeits- & Temperatursensor DHT22

```
int getAirHum(){
  int humidity = dht.readHumidity();          // Read humidity (percent)
  if(isnan(humidity)){
    humidity = dht.readHumidity();
  }
  if(isnan(humidity)){
    return -1;
  }
  Serial.print("Air Humidity: ");
  Serial.println(humidity);
  return humidity;
}
```

Listing 3: getAirHum()

2.3 Lufttemperatur

Hierfür wird der o. G. DHT22 ebenfalls genutzt. Es wird mittels eins 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 +- 0,5 grad Celsius und arbeitet im Bereich von -40 bis +80 grad Celsius.

```
float getAirTemp(){
  float temp_f = dht.readTemperature(true);     // Read temperature as
     Fahrenheit
  float temp_c = (((temp_f-32.0)*5/9)-1);       // Convert temperature
     to Celsius + Correction

  if (isnan(temp_c)){
    temp_f = dht.readTemperature(true);     // if read fails try one
       more time
    temp_c = (((temp_f-32.0)*5/9)-1);
  }
  if (isnan(temp_c)){
    return -100.0;
  }
  Serial.print("Air Temperature: ");
  Serial.println(temp_c);
  return temp_c;
}
```

Listing 4: getAirTemp()

2.4 Licht

Hierzu nutzen wir eine einfache Photozelle, 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)))
        +100)*20);
3   boolean bright = digitalRead(BRIGHTNESSPIN);
4   Serial.print("Brightness: ");
5   Serial.println(brightness);
6   Serial.print("Bright: ");
7   Serial.println(bright);
8   return brightness;
9 }
```

Listing 5: getBrightness()

2.5 Regen

Für den Regensensor gibt es auch wieder kein spezifisches Datenblatt. Wir verwenden hierzu einen 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 3,3-5V.

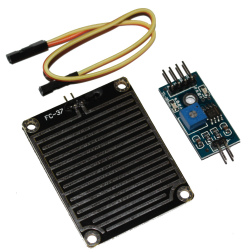

Abbildung 3: Regensensor

```
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  }
```

2.6 Bodentemperatur

2.6.1 1-Wire

Zum messen der Bodentemperatur wird ein wasserdichter Temperatursensor basierend auf dem 1-Wire DS18B20 mit einer Sonde aus Edelstahl. Er 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. Der Sensor hat einen Arbeitsbereich von -55 Grad Celsius und +125 Grad Celsius. Er läuft mit einer Spannung von 3,3-5V und hat eine Messgenauigkeit von +/- 0,5 Grad Celsius im Bereich von -10Grad Celsius bis +85 Grad Celsius.

Abbildung 4: Bodentemperatursensor DS18B20

```
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 6: getSoilTemp()

2.7 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   boolean soilMoisDry = digitalRead(SOILMOISPIN);
4   Serial.print("Soil Moisture: ");
5   Serial.println(soilMois);
6   Serial.print("Moist: ");
7   Serial.println(soilMoisDry);
8   return soilMois;
9 }
```

Listing 7: getSoilMois()

2.8 Anemometer

Hierbei handelt es sich um einen Handelsüblichen Windstärkesensor der auch für andere Wetterstationen genutzt werden kann. Dieser kann über die eingebaute Schnittstelle ein-

Abbildung 7: Anemometer

fach ausgelesen werden.

```
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 8: MeasureWind()

Außerdem wurde eine Interrupt-Routine für das Zählen der Umdrehungen des Anemometers initialisiert:

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

Listing 9: countWind()

Um eine einigermaßen genaue Messung zu erhalten müssen Sensoren kalibriert werden. Um zu Testen ob das Anemometer richtig funktoniert haben wir 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.

2.9 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 }
```

3 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.

3.1 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

3.2 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 er ausschließt 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

3.3 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.

3.4 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.

4 Hauptprogramm

4.1 HomeAssistant

HomeAssistant ist ein OpenSource Hausautomatisierungssystem, dass auf Python3 basiert. Es läuft auf verschiedenen Betriebssystemen wie z.B: Windows10, MacOS, Linux. Hier hat man die möglichkeit die verschiedensten Geräte mit einander in den HomeAssistant einzubinden um sein Zuhause zu Überwachen, zu Kontrollieren und auch um bestimmte automatisierte Abläufe einzustellen.

4.2 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
7     rest.variable("soil_temperature", &soilTemp);
8     rest.variable("soil_moisture", &soilMois);
9
10    rest.variable("motion_detector", &motion);
11
12    rest.variable("brightness", &brightness);
13    rest.variable("precipation", &precip);
14
15    rest.variable("switch1", &switch1_state);
16    rest.variable("switch2", &switch2_state);
17    rest.function("switch1",switch1);
18    rest.function("switch2",switch2);
19
20    // Give name & ID to the device (ID should be 6 characters long)
21    rest.set_id("000001");
22    rest.set_name("arduino_weatherstation");
```

Listing 10: 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".

4.3 HomeAssistant - Anzeige

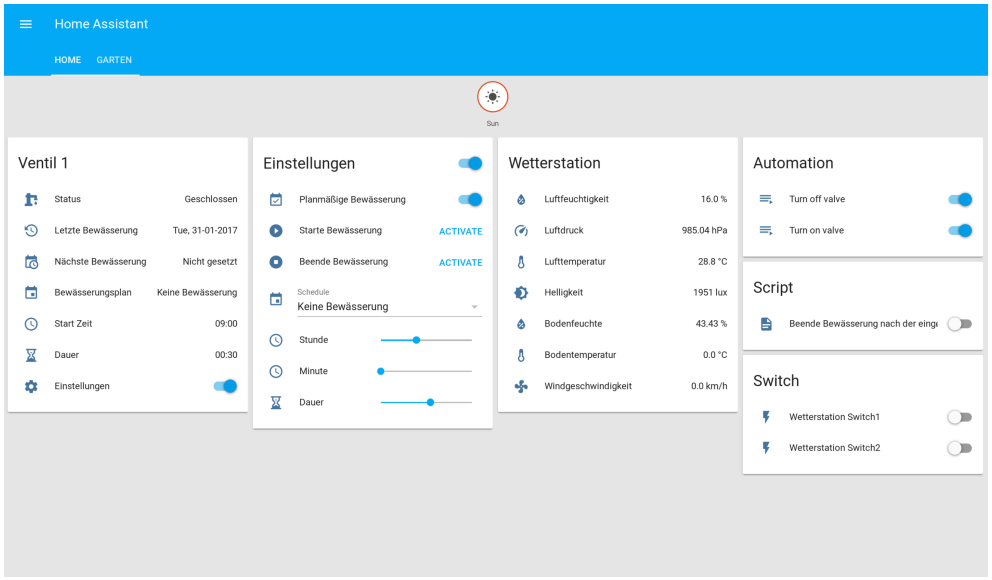

Abbildung 8: HomeAssistant - Home

Die in Abbildung 8 gezeigte Übersicht bietet das HomeAssistant-System im standard Fall. 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.

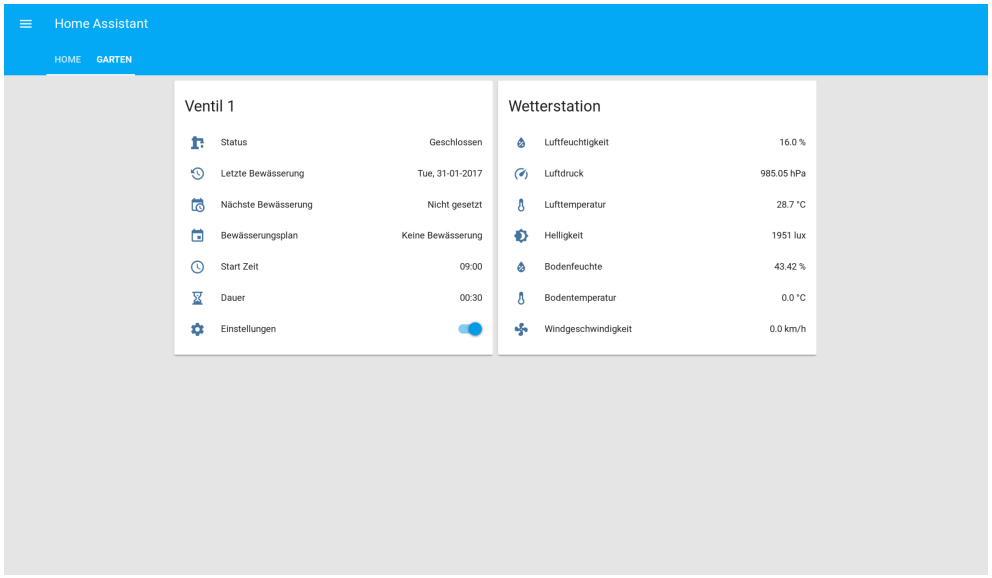

Abbildung 9: HomeAssistant - Garten

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.

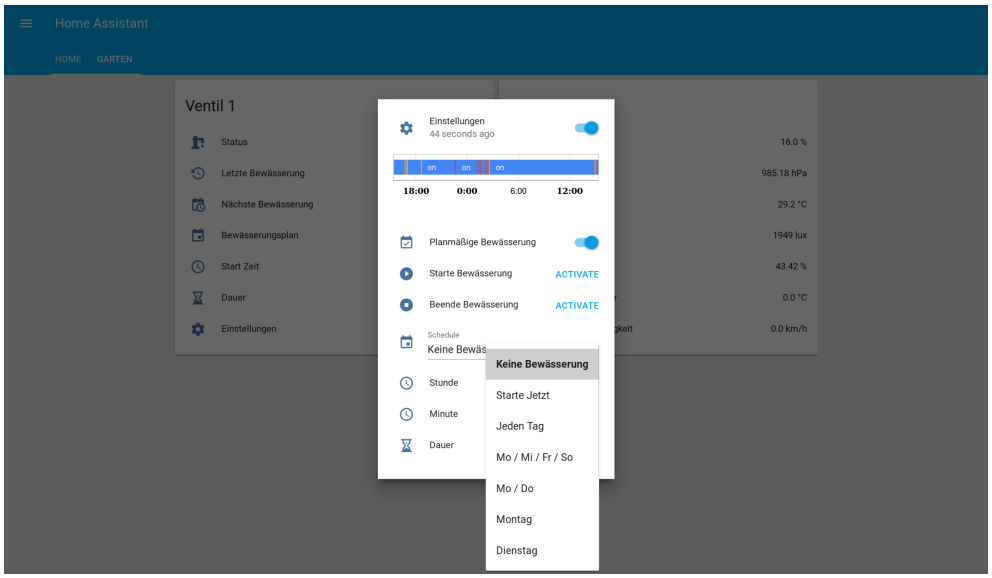

Abbildung 10: HomeAssistant - Bewässerung

5 Probleme

5.1 Fehlende Datenbläter

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

5.2 Kurzschluss

Da die Sensoren mittels Steckverbindungen - RJ11 - mit einander verbunden werden sollten haben wir diese in die Verkabelung mit eingebaut, allerdings achteten wir zunächst nicht darauf, dass ein Stecker auch "falsch herumëingesteckt 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.