Appgesteuerte Bewässerungsanlage für den Garten

Technik

Essen, NRW – JuFo-Projekt 2021

Lukas Krinke
Q1 Gymnasium Essen-Werden

Kurzfassung

Das Projekt beschäftigt sich mit dem Thema, wie man eine Bewässerungsanlage realisieren kann, welche mit dem Handy von überall steuerbar ist, dabei jedoch alle Teilprogramme im lokalen Netzwerk betreibt. Als Kernstück wurde dafür ein Raspberry Pi benutzt, welcher mit Magnetventilen und verschiedenen Sensoren ausgestattet wurde.

Die Sensoren sammeln verschieden Daten, welche dem Nutzer in einer App als Diagramm dargestellt werden, wodurch er entscheiden kann, ob die Fläche bewässert werden muss.

Ebenfalls fungiert der Raspberry Pi als Server, da er die Anfragen der Clients verarbeitet und die entsprechende Tätigkeit, wie zum Beispiel das Bewässern eines Gartens ausführt. Um an den Raspberry Pi Anfragen zu senden, wird die Public IP benutzt, welche jedes lokale Netzwerk besitzt und es wurden zwei Ports freigeschaltet. Alle Anfragen an diese Ports werden an den Raspberry Pi weitergeleitet, wodurch die Anlage auch von außerhalb steuerbar ist.

Begriffserklärung

Client [1]:

Ein Client ist ein Gerät, welches sich mit einem Server verbinden kann. Er kann Daten bei einem Server abfragen wie auch Daten senden.

API [2]:

Ein Application Programming Interface, kurz API, ist eine Programmierschnittstelle, welche dazu dient, zwischen unterschiedlichen Programmteilen Information auszutauschen. Durch die API wird definiert, in welcher Form Information gesendet und empfangen werden müssen.

Framework [3]:

Ein Framework ist eine Sammlung an bereits geschriebenem Codeteilen, welche man in seinen eigenen Projekten benutzen kann, um nicht immer alles von Grund auf neu zu programmieren.

GPIO-Pins:

Der Raspberry Pi hat eine gewisse Anzahl an GPIO-Pins, an die man verschiedene Sensoren und Aktoren anschließen kann. Dadurch lassen sich die Sensoren auslesen und die Aktoren steuern.

Inhaltsverzeichnis

1. Ideenfindung und Anforderungen

Im ersten Lockdown haben mein Vater und ich bei uns neuen Rasen ausgesät, weil unser letzter vertrocknet ist. Daraufhin bekam ich die Idee, dass man eine Bewässerungsanlage bauen könnte, mit der man den Garten bewässern kann, damit er nicht wieder vertrocknet.

Dazu wurden folgende Anforderungen aufgestellt:

- Mit Handy-App steuerbar
    - Mit Server verbinden und trennen
        - Meldung ob Verbindung erfolgreich war
        - Meldung, wenn die Bewässerung startet
    - Startzeit festlegen
    - Dauer
    - „Notfall-Aus“
- Daten speichern und auswerten
- Möglichst kosteneffizient
    - Keinen externen Server kaufen/mieten
    - Nicht unnötig bewässern

Aus diesen Anforderungen ergab sich folgende Leitfrage:

Kann man eine Bewässerungsanlage bauen, welche man von überall steuern kann und dabei auf externe Server verzichten?

2. Hardware

2.1 Materialen:

1. Raspberry Pi 3B
2. Regentropfen Sensor
3. Bodenfeuchtesensor Hygrometer Modul V1.2 kapazitiv
4. Temperatursensor 3M Kabel DS18B20 digitaler Edelstahl
5. ADS1115 Modul 16bit 4 Kanäle
6. 3 Magnetventile
7. 4-Kanal Relais Modul 5V/230V

2.2 Microkontroller

Als Kernstück der Anlage wurde sich für ein Raspberry Pi 3 B entschieden, da er viele GPIO-Pins hat, welche sich einfach mit der Programmiersprache Python ansteuern lassen. Zudem hat Python den Vorteil, dass man die Serverkommunikation sehr einfach programmieren kann. Auf dem Raspberry Pi läuft das Skript, welches die Bewässerung steuert, ein MQTT-Broker für die Kommunikation zwischen Pi und Client, eine Datenbank und eine API zur Datenabfrage.

2.3 Sensoren

Für die Datenerfassung, welche benötigt wird, damit die Bewässerung optimal ablaufen kann, wurden verschiedene Sensoren eingesetzt, welche im Folgenden beschrieben werden.

Abbildung 1 Regensensor

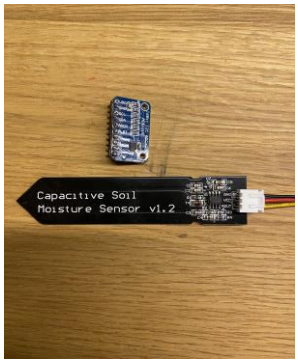

Abbildung 2
Bodenfeuchtesensor mit ADC

Abbildung 3 Temperatursensor

2.3.1 Regensensor - 35in1 [4]

Der Sensor erkennt Regen, indem das Wasser, welches die Oberfläche des Sensors berührt, einen Stromkreis schließt. Wenn dies eintritt, wird solange ein Signal gesendet, bis der Stromkreis wieder unterbrochen wird, es also aufgehört hat zu regnen. Theoretisch liefert der Sensor auch die Intensität des Regens. Dies funktioniert, da durch stärkeren Regen der Stromkreis an mehr Stellen geschlossen wird, wodurch der Widerstand sinkt und die Spannung steigt. Um dies zu testen, wurde er mit einer Sprühflasche unterschiedlich stark besprüht. Dabei kam jedoch keine unterschiedliche Signalstärke zustande. Dies liegt vermutlich daran, dass auf Grund des Raspberry Pis, ein digitaler Pin benutzt werden muss, welcher jedoch vermutlich nur ON/OFF als Output liefern kann. Dieses Problem lässt sich lösen, indem der Sensor an das, im Folgenden noch erklärte, ADC-Modul angeschlossen wird, welches dem Raspberry PI vier analoge Pins gibt. Jedoch wurde dieses Problem vernachlässigt, da es hauptsächlich darauf ankommt, ob es regnet oder nicht. Dies wird sehr zuverlässig zurückgegeben.

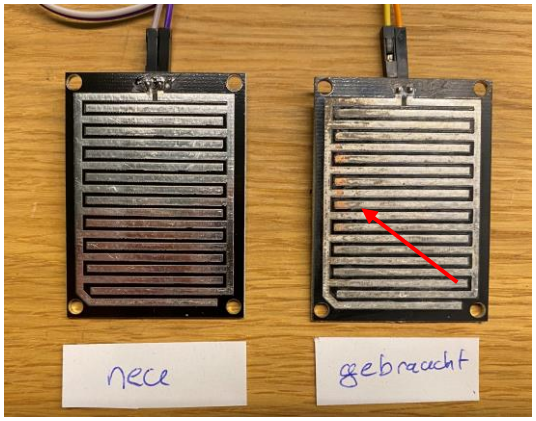

Abbildung 4 Vergleich Regensensoren

Ein weiteres Problem ist, dass die Lebensdauer des Sensors sehr gering zu sein scheint. Wie man in Abbildung 4 leicht erkennt, ist der gebrauchte Sensor nach circa einem Monat Nutzung bereits angerostet. Dadurch ist es notwendig ihn oft zu ersetzen, wodurch ein teurerer, aber auch langlebigerer Sensor eine mögliche Alternative ist.

2.3.2 Feuchtesensor – Hygrometer Modul V.1.2 kapazitiv [5]

Dieser Sensor misst die Kapazität der Erde, welche bei feuchter Erde steigt. Um die Ausgabe einzuordnen, wurde der Sensor einmal in Erde gesteckt, welche eine Woche im Haus getrocknet wurde sowie in Erde, welche mit Wasser durchtränkt war, sodass diese Erde kein Wasser mehr aufnehmen kann. So lässt sich herausfinden, ab wann die Ausgabe eine trockene Erde darstellt und wann eine feuchte Erde. Um ein möglichst genaues Resultat zu erhalten, misst der Sensor zwei Minuten lang jede Sekunde und addiert die Werte. Daraufhin wird dieser Wert durch die Durchgänge, insgesamt 120, geteilt, wodurch ein Durchschnitt gebildet wird.

	Sehr Trocken	Sehr Feucht
Ausgabe:	440	632

Aus diesen Ergebnissen lässt sich ableiten, dass eine mittel feuchte Erde einen Wert von 536 hat. Wenn dieser Wert unterschritten wird, kann man eine Bewässerung in Betracht ziehen.

Ein Problem des Sensors ist, dass der Output analog ist und deshalb nicht vom Raspberry PI verarbeitet werden kann, weil die GPIO-Pins nur digital sind. Leider findet man jedoch keine Feuchtsensoren mit digitalem Output. Um dieses Problem zu beheben, wurde der Analog-Digital Wandler ADS1115 benutzt. Dieser wird an den SDA0 und SCLO Pin vom Raspberry PI geschlossen, wodurch dieser vier analoge Pins bekommt. Auf diese Weise lassen sich insgesamt vier Feuchtesensoren gleichzeitig anschließen, wodurch die Messwerte genauer werden, da man an verschiedenen Positionen messen kann.

2.3.3 Temperatursensor - DS18B201 [6]

Dieser kann Temperaturen von -55°C bis 125°C aufnehmen und hat dabei nur eine Abweichung von um die 0.5°C. Aus den Temperaturdaten lässt sich Folgendes ableiten:

- Ist es besonders heiß, benötigt der Garten mehr Wasser
- Ist die Temperatur unter 0° C, muss das Wasser aus den Schläuchen entfernt werden, da sie ansonsten platzen können

Bei der Inbetriebnahme wurde zuerst eine Fehlermessung durchgeführt, um die angegeben Abweichung zu überprüfen. Dazu wurde der Sensor in Wasser mit Eiswürfeln gehalten. In diesem Zustand sollte er eine Temperatur von genau 0°C anzeigen. In einer Messreihe von 100 Messungen, waren alle Werte knapp bei 0°C, wodurch die Abweichung nicht näher betrachtet wurde. Ansonsten hätte man bei der Temperaturmessung eine Konstante addiert, beziehungsweise subtrahiert.

2.4 Ventile

Die Magnetventile öffnen sich, wenn eine Spannung von 24 Volt anliegt. Da der Raspberry Pi jedoch nur eine Spannung von maximal 5 Volt liefert, wurde eine Relais dazwischengeschaltet. Die Spannung des Raspberry Pis schaltet das Relais, wodurch der Stromkreis des entsprechenden Ventils geschlossen wird und sich das Ventil öffnet. Insgesamt wurden drei Ventile benutzt, welche an eine Verteilung angeschlossen sind. An jedem Ventil hängen drei bis vier Sprenger, welche bei Wasserdruck automatisch hochfahren und den Rasen bewässern. Die Bewässerung wurde in mehrere Kreisläufe aufgeteilt, da der Wasserdruck nicht ausgereicht hat, um alle Sprenger gleichzeitig hochfahren zulassen. Es wird immer nur ein Kreislauf geöffnet, welcher eine geringe Anzahl an Sprengern hat. Auf diese Weise reicht der Druck aus.

Abbildung 5 Verteilung - Relais mit Magnetventilen

2.5 Gesamtanlage

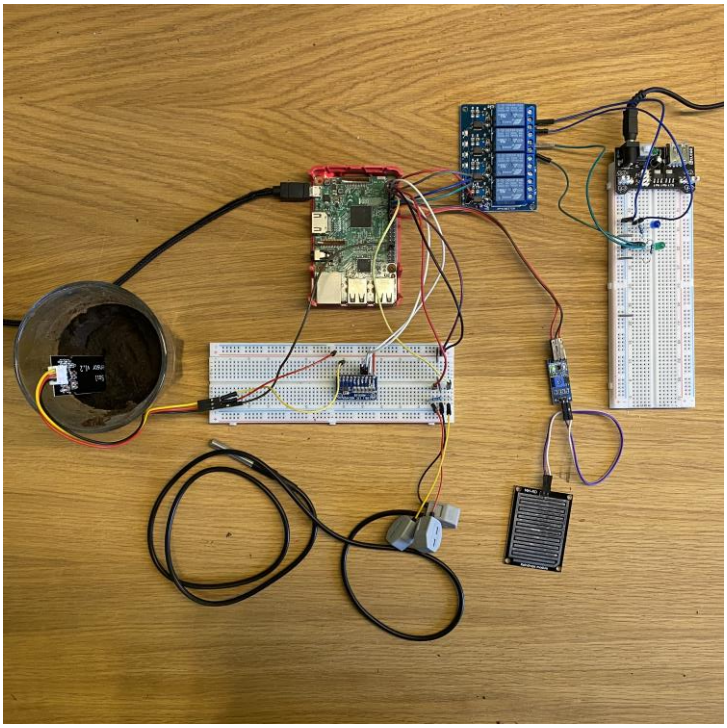

Abbildung 6 Modell der Anlage

In Abbildung 6 sieht man das Modell der Bewässerungsanlage. In der Praxis werden anstelle der LEDs(rechts) die Magnetventile angeschlossen (Abb. 5). Wenn die Anlage in Betrieb genommen wird, befinden sich die Sensoren und die Magnetventile im Garten. Da der Raspberry Pi jedoch im Haus angebracht wird, ist es nötig, dass es eine Möglichkeit gibt, dass die Sensoren an den Raspberry Pi angeschlossen werden können. Dafür eignet sich zum Beispiel ein Flachbandkabel, welches unter der Tür hindurch passt, ohne dass es dabei kaputt geht.

3. Software

Die Software für das Projekt teilt sich in drei Hauptbereiche, Backend, Frontend und Datenbank auf. Der wichtigste Teil ist das Programm, welches die Bewässerung steuert. Um jedoch zu wissen, wann es bewässern soll, benötigt es bestimmte Daten, welche vom Client festgelegt werden. Dies ist der zweite Bereich und ist ebenfalls essentiell. Es ist notwendig, dass der Nutzer seine gewünschten Werte an einen Server senden kann und ebenfalls eine Nachricht bekommt, wenn der Rasen gesprengt wird. Der dritte Bereich ist die Datenbank, welche die verschiedenen Messwerte der Sensoren speichert. Um diese Daten später abzufragen, ist ebenfalls eine API nötigt, welche Daten abruft und diese an den Client zurückschickt. Diese einzelnen Bereiche werden nun genauer erläutert.

3.1 Backend [7-8]

Das Backend regelt die Bewässerung des Gartens. Um dies zu realisieren, wurde die Programmiersprache Python benutzt, da man mit ihr sehr gut die Kommunikation zwischen dem Raspberry Pi und den Clients programmieren kann. Ebenfalls lassen sich in ihr mit dem Framework *Flask* sehr einfach eigene Server schreiben.

Insgesamt besteht die Anwendung aus vier Klassen. Hierbei sind die Klassen *Communication* und *Water* am wichtigsten. Koordiniert wird alles über das Programm *main.py*.

Communication:

Dieses Objekt wird im Hauptskript *main.py* erzeugt und regelt die Kommunikation zwischen Raspberry Pi und Client. Dazu wird das MQTT-Protokoll verwendet, welches sehr einfach zu verwenden ist. Dabei gibt es einen MQTT-Broker, mit dem sich die einzelnen Clients verbinden können. In diesem Projekt ist der Raspberry Pi sowohl Broker als auch Client, da *main.py* sich als Client in den Broker einwählt.

Eine Nachricht besteht immer aus einem *Topic* und einem *Body*, welcher die eigentliche Nachricht beinhaltet.

Clients können nun bestimmte *Topics subscriben* und werden benachrichtigt, wenn ein anderer Client eine Nachricht mit dem entsprechenden *Topic* versendet. Der Broker leitet diese dann an den Client weiter, welcher den Inhalt dann verarbeiten kann.

Beispiel:

```
elif msg.topic == "rpi/gpio":
    message_active = str(msg.payload.decode(encoding='UTF-8'))

    if message_active == "on":
        self.active = True
        print(active)
    elif message_active == "off":
        self.active = False
        print(active)
    else:
        print("Unknown message!")
```

Abbildung 7 Beispiel MQTT – empfangen

In diesem Beispiel hat der Raspberry PI unter anderem bereits das *Topic rpi/gpio* subscribt. Da er auf mehrere *Topics* hört, muss zuerst über *if-else-Abfragen* die richtige Aktion zu dem empfangenen *Topic* herausgefiltert werden. Bei *rpi/gpio* ist der Inhalt, also der *Body*, entweder „on“ oder „off“ und steuert, ob die Bewässerungsanlage an oder aus sein soll. Zuvor muss die Nachricht jedoch noch zu einem String decodiert werden, da die Nachrichten im JSON-Format versandt werden und sonst nicht interpretiert werden können.

Insgesamt hört diese Klasse auf drei *Topics*, also An/Aus, Zeitpunkt und Dauer, welche der Nutzer einstellen kann. Diese Daten werden in der folgenden Klasse *Water* weiterverarbeitet.

Water:

Bei der Initialisierung dieser Klasse wird von *main.py* ein bereits erzeugtes Sensoren-Objekt, welches jeweils Setter und Getter für jeden Sensorwert hat sowie das Communication-Objekt (im Folgenden *com*), wie oben beschrieben, übergeben.

Daraufhin wird im *main.py* die Funktion *water.main()* auf einem neuen Thread aufgerufen.

```
def main(self):
    print("Thread3 activated")
    while True:
        while self.com.get_isActive():

            state_rain = self.sensoren.get_state_rain()
            now = datetime.datetime.today()

            if self.com.get_time_to_water() != None:
                if now.hour == self.com.get_time_to_water().hour and now.minute ==  self.com.get_time_to_water().minute and state_rain == 1:
                    print("Water now!")

                    self.com.send_succes(temp=self.sensoren.get_temperature(), hum=self.sensoren.get_humidity())

                    self.water_now(self.com.get_dauer())
```

Abbildung 8 Steuerung der Bewässerung

Diese Funktion beinhaltet eine *while-Schleife*, welche sich immer wieder aufruft. In ihr wird überprüft, ob das Objekt *com* den Befehl vom Client bekommen hat, dass die Bewässerungsanlage angeschaltet werden soll. Wenn dies der Fall ist, geht das Objekt in die innere *while-Schleife* und es wird überprüft, ob die Startzeit zum Sprengen mit dem jetzigen Zeitpunkt übereinstimmt. Hierbei wird nur auf die Stunde und Minute geachtet, da es theoretisch möglich ist, dass der Zeitpunkt verpasst wird und so gar nicht gesprengt wird, wenn ebenfalls auf die Sekunde geachtet wird. Wenn die Zeitpunkte übereinstimmen, wird der Client über eine MQTT-Message benachrichtigt und das erste Ventil öffnet sich. Dort wird alle 15 Sekunden überprüft, ob es angefangen hat zu regen, um möglicherweise die Bewässerung automatisch zu stoppen und Wasser zu sparen. Wenn das erste Ventil für die angegebene Dauer offen war, werden nach dem oben beschriebenen Prinzip alle weitern Ventile nacheinander geöffnet.

3.2 Datenbank und API

In *main.py* wird ebenfalls eine Klasse erzeugt, welche die Daten der Sensoren alle fünf Minuten in einer Datenbank sichert. Diese Datenbank läuft auf einem Apache Webserver, welcher ebenfalls auf dem Raspberry Pi läuft. Dafür wird zuerst eine Verbindung zur Datenbank hergestellt und daraufhin mit SQL die jeweiligen Daten eingetragen.

Diese Daten sollen ebenfalls in der App dargestellt werden, wofür die Daten abrufbar sein müssen. Da man das Frontend nicht direkte mit der Datenbank verbinden sollte, wurde eine API, basierend auf dem Framework *Flask* und *flask_restful*, geschrieben, welche die gewünschten Daten in der Datenbank abruft und an den Client sendet.

```python
class SensorData(Resource):
    def get(self, sensor_id):
        sql = f"SELECT {sensor_id} FROM SensorData ORDER BY SensorData.DatumUhrzeit DESC LIMIT 576"
        try:
            cursor.execute(sql)
            results = cursor.fetchall()
            temp = [] # DatumUhrzeit geht momentan nict, da es nicht als JSON dirket geht
            for result in results:
                temp.append(result[0])
            return {"data": temp}
        except:
            print("Error")
            return "Error"

api.add_resource(SensorData, "/alldata/<string:sensor_id>")
```

Abbildung 9 API

In der Route, welche die App aufruft, wird der Name der Spalte für den entsprechenden Datensatz übergeben, um möglichst einfach die Daten abzurufen. Der Tabellenkopf in der Datenbank für die Temperaturen heißt zum Beispiel „Temperatur“, also wird in der Route ebenfalls „Temperatur“ übergeben. Die zurückgegebenen Daten der Datenbank werden in das Array *temp* geschrieben, welches daraufhin im JSON-Format mit dem

Key „data“ an den Client zurückgesendet wird. Dort werden die Daten weiterverarbeitet.

3.3 Frontend

Als Frontend wurde eine App für das Smartphone geschrieben. Dafür wurde die Programmiersprache Swift benutzt. Dabei unterteilt sich die App in zwei Bereiche; erstens die Steuerung der Bewässerungsanlage und zweitens die Visualisierung der Messwerte.

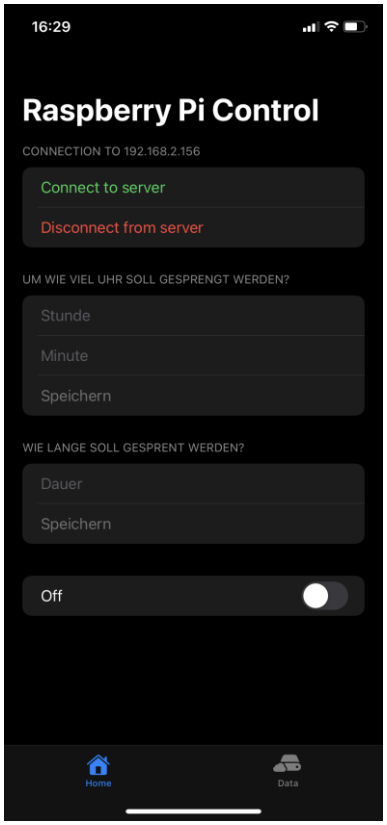

Abbildung 10 App - Steuerung

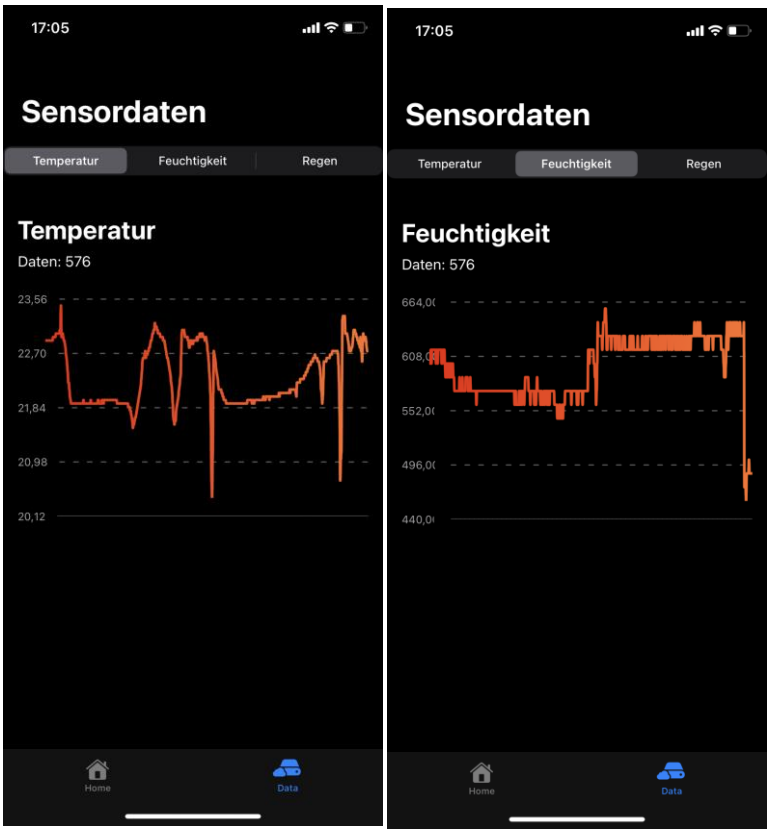

Abbildung 11 Diagramme der Messdaten

Für die erste Ansicht (Abb.10) wird ein Objekt, erzeugt, welches die Kommunikation regelt. In diesem Objekt wird mit Hilfe des Frameworks *CocoaMQTT* ein MQTT-Client initialisiert, welcher die Daten an den Broker senden kann. Dieser Client verbindet sich mit dem Broker auf dem Raspberry Pi und wenn dies erfolgreich ist, werden auch hier verschiedene *Topics subscribt*.

Beispiel:

```
func sendState(state: Bool){
    if isConnected{
        if state{
            mqttClient.publish("rpi/gpio", withString: "on")
            print("Published ON")
        }else{
            mqttClient.publish("rpi/gpio", withString: "off")
            print("Published OFF")
        }
    }
}
```

Abbildung 12 MQTT - senden

Die Funktion *sendState()* wird aufgerufen, wenn der Nutzer den Schalter mit der Beschriftung „Off“ betätigt. Die Funktion sendet dann den entsprechenden Zustand, mit dem *Topic* „rpi/gpio“, auf welches der Raspberry Pi hört.

In der zweiten Ansicht (Abb. 11) kann der Nutzer die verschiedenen Daten abfragen, welche in der Datenbank gespeichert sind. Dafür wird ein neues Objekt der Klasse ApiCaller erzeugt, welches mit der entsprechenden Route, wie oben erläutert, die Daten bei der API abfragt. Daraufhin kommt ein JSON-Objekt zurück, welches decodiert wird und in dem Diagramm dargestellt wird.

4. Netzwerktechnik [9-12]

Das Ziel der Bewässerungsanlage war es zudem, dass sie von überall steuerbar ist, innerhalb des eigenen Netzwerkes wie auch außerhalb. Dafür wurde die Public IP benutzt, welche jedes lokale Netzwerk hat und von außen angesteuert werden kann. Normalerweise ist es nicht relevant, dass das lokale Netzwerk durch die Public IP für jeden „sichtbar“ ist, da alle Ports geschlossen sind. Jedoch kann man bestimmte Ports öffnen, wodurch die Anfragen an ein Gerät im lokalen Netzwerk weitergeleitet werden. Dies nennt man *Port forwarding,* welches beim Router eingestellt wird. Für die Bewässerungsanlage wurden zwei Ports freigeschaltet; einer für die Befehlserteilung mit Hilfe von MQTT und einer für die eigene API. Dadurch lässt sich nun die Anlage auch von außerhalb steuern.

Jedoch trat das Problem auf, dass, wenn man nun mit dem eigenen Netzwerk verbunden ist, nicht auf seine eigene Public IP zugreifen kann. Dadurch konnte man die Bewässerungsanlage von zu Hause nicht mehr steuern. Um dies zu beheben, bieten einige Router die Funktion des *NAT-Loopbacks* an, wodurch das beschriebene Problem behoben wird. Leider unterstützt jedoch unser Router diese Funktion nicht, wodurch eine andere Lösung gefunden werden musste. Dafür wird in der App kontrolliert, ob das Gerät mit einem WLAN verbunden ist. Wenn dies der Fall ist, wird die lokale IP des Raspberry Pis benutzt, andernfalls die Public IP mit dem entsprechenden Port.

Ebenfalls wichtig ist es, dass die API von den anderen Geräten im lokalen wie auch außerhalb erreichbar ist. Dafür wurde der Host des Flaks-Servers, auf welchem die API läuft, auf 0.0.0.0 gesetzt, wodurch der Server im gesamten lokalen Netzwerk sichtbar wird. Durch das *Port forwarding* kann er auch von außerhalb angesteuert werden.

5. Anwendungsbereiche

Der Anwendungsbereich dieser Anlage liegt vordergründig auf der Bewässerung von Gärten. Mit den verbundenen Sprengern an den Magnetventilen lässt sich sehr einfach eine sehr große Fläche bewässern. Allerdings kann man mit dieser Anlage auch kleine Bete oder zum Beispiel Blumentöpfe auf einem Balkon gießen, indem statt der Sprenger ein anderer Aufsatz gewählt wird, welchen man an die Magnetventile anschließt. Hier muss man jedoch noch überlegen, welcher Aufsatz gewählt wird.

6. Zusammenfassung

Insgesamt lässt sich sagen, dass durch die verwendete Hardware und die geschriebene Software die Leitfrage beantwortet wurde.

Mit Hilfe der App lässt sich genau festlegen, wann und wie lange der Garten gesprengt werden soll. Zudem bekommt man eine Meldung, wenn man sich erfolgreich mit dem Server verbunden hat oder die Bewässerungsanlage startet.

Zudem sammeln die Sensoren wichtige Information, welche einem helfen zu entscheiden, ob der Garten bewässert werden sollte. Der Regensensor stoppt außerdem die Bewässerung automatisch, wenn es während eines Durchgangs anfangen sollte zu regen. Die gesamten Daten werden außerdem in einer Datenbank gespeichert und in der App in schönen Diagrammen visualisiert.

Desweitern wurde das gesamte Projekt ohne externe Server realisiert, da die gesamten Prozesse, von dem Programm für die Steuerung bis hin zur Datenbank, auf dem Raspberry Pi laufen. Dies gelingt durch das Freischalten von zwei Ports und der eigenen Public IP. Dadurch werden keine weiteren monatlichen Kosten für Server erzeugt.

7. Erweiterungsmöglichkeiten und Optimierung

Die Bewässerungsanlage lässt sich jedoch auch noch in einigen Punkten erweitern und optimieren.

Es wurde sich vor allem auf Grund der Programmiersprache Python für einen Raspberry Pi entschieden, jedoch hätte eine Arduino ebenfalls einige Vorteile. So hat ein Umstieg auf den Arduino zum Beispiel den Vorteil, dass er günstiger ist und zudem analoge Pins hat. So lässt sich auch die Regenintensität messen und für den Bodenfeuchtesensor ist kein ADC Modul mehr nötig. Generell kann man erwägen, ob man auf hochwertigere Sensoren setzt, da die Qualität, wie man bei dem Regensensor sieht, mittelmäßig ist. Dadurch bekommt man noch genauere Messwerte, muss jedoch auch mehr bezahlen.

Auch können weitere Sensoren angeschlossen werden, wie zum Beispiel ein Durchflussmesser. Dieser misst wie viel Wasser durch ihn hindurch fließt, wodurch man viele weitere Möglichkeiten bekommt. So kann man ein bestimmtes Limit an Wasser festlegen, welches pro Woche oder Monat zu Bewässerung genutzt werden darf. Desweitern fungiert dieser Sensor als Sicherheitsmaßnahme, da er erkennen kann, ob gerade Wasser fließt, obwohl gar nicht bewässert wird. Daraufhin kann der Nutzer sofort benachrichtig werden, dass die Anlage eine undichte Stelle hat.

Ebenfalls muss die Netzwerksicherheit noch optimiert werden. Momentan könnte theoretisch jeder die Bewässerungsanlage steuern. Jedoch ist dies sehr unwahrscheinlich, da man die Public IP, den richtigen Port, das richtige Topic und noch den richtigen Body wissen müsste. Um diese jedoch noch weiter auszuschließen, kann man in der Route noch ein festgelegtes Passwort übermitteln, welches vorher verschlüsselt wird.

Momentan ist es so, dass der Nutzer anhand der Messwerte selber entscheiden muss, ob der Garten bewässert werden sollte. In der Zukunft kann man dieses Projekt jedoch mit einem neuronalen Netzwerk erweitern, welches auf Basis der gesammelten Datensätze trainiert werden kann. Daraufhin könnte es selber entscheiden, ob der Garten gesprengt werden sollte.

8. Quellen

1. *https://de.wikipedia.org/wiki/Client*
2. *https://www.dev-insider.de/was-ist-eine-api-a-583923/*
3. *https://de.wikipedia.org/wiki/Framework*
4. *https://www.youtube.com/watch?v=cXLNoqtQhxc&list=PLBz-cd41fguSqHDljTpxGt9JLphtPSHi8&index=9&t=41s&ab_channel=AlleviateITConsultancyPvtLtd*
5. *https://www.youtube.com/watch?v=oda_jsSYqR4&list=PLBz-cd41fguSqHDljTpxGt9JLphtPSHi8&index=7&t=308s&ab_channel=VirusHQ*
6. *https://www.youtube.com/watch?v=Ykx9eacAwIM&list=PLBz-cd41fguSqHDljTpxGt9JLphtPSHi8&index=8&t=809s&ab_channel=WolfgangRaab*
7. *https://www.youtube.com/watch?v=VidqoMsdySs&list=PLBz-cd41fguSqHDljTpxGt9JLphtPSHi8&index=1&ab_channel=Defpom%27sElectronics%26Repair*
8. *https://www.youtube.com/watch?v=0V_zBolP14o&list=PLBz-cd41fguSqHDljTpxGt9JLphtPSHi8&index=2&ab_channel=Industry40tv*
9. *https://www.youtube.com/watch?v=JnEPwWlr8kY&list=PLBz-cd41fguSqHDljTpxGt9JLphtPSHi8&index=3&ab_channel=Luciano%27sTechnologyChannel*
10. *https://www.youtube.com/watch?v=1uackeSKhRM&list=PLBz-cd41fguSqHDljTpxGt9JLphtPSHi8&index=4&t=863s&ab_channel=AksharkumarPatel*
11. *https://www.youtube.com/watch?v=GMppyAPbLYk&list=PLBz-cd41fguSqHDljTpxGt9JLphtPSHi8&index=10&t=351s&ab_channel=TechWithTim*
12. *https://en.wikipedia.org/wiki/Network_address_translation#NAT_loopback*

9. Abbildungen