

WPF-Projekt:

Wetterstation

Von

Manuel Yates (AW)

Felix Becker (SI)

Michael Schkalei (AW)

Peter Rietz (AW)

Lukas Sänger (AW)

Luca Grethen (SI)

Zuletzt geändert am:

15.07.2022

Inhalt

[Vorwort 3](#_Toc109037439)

[Projektbeschreibung 3](#_Toc109037440)

[Projektumsetzung 3](#_Toc109037441)

[Aufgabenverteilung 3](#_Toc109037442)

[Projektumgebung (Hardware) 3](#_Toc109037443)

[Projektumgebung (Software) 4](#_Toc109037444)

[Arduino-Studio (Ardublock) 4](#_Toc109037445)

[Betriebssystem (Raspberry Pi) 4](#_Toc109037446)

[Mosquitto (Raspberry Pi) 4](#_Toc109037447)

[NodeRed (Raspberry Pi) 4](#_Toc109037448)

[Datenbank (Raspberry Pi) 5](#_Toc109037449)

[Programmiersprache & Frameworks 5](#_Toc109037450)

[Entwicklungsumgebung 5](#_Toc109037451)

[Versionsverwaltung 6](#_Toc109037452)

[Allgemeine Struktur und Ablauf 6](#_Toc109037453)

[Installation und Einrichtung 7](#_Toc109037454)

[Schritt 1: Flashen des Betriebssystems für den Raspberry Pi 7](#_Toc109037455)

[Schritt 2: Festlegen einer statischen IP-Adresse für den Raspberry Pi 7](#_Toc109037456)

[Schritt 3: Verbindung zum Raspberry Pi über PuTTY 7](#_Toc109037457)

[Schritt 4: Einrichtung der Wetterstation 8](#_Toc109037458)

[Schritt 5: Einrichtung des Octopus Boards 11](#_Toc109037459)

[Schritt 6: Starten des Webservers 12](#_Toc109037460)

[Begründung dieser Umsetzung 12](#_Toc109037461)

[Einen Blick in die Zukunft 12](#_Toc109037462)

[Anmerkungen 13](#_Toc109037463)

# Vorwort

Das Projekt wurde 2021/22 in eigenständiger Arbeit geplant, vorbereitet und durchgeführt.

# Projektbeschreibung

Anforderung war es eine vollständige Umgebung einer Wetterstation zu erstellen.  
Diese sollte folgende Funktionen beinhalten:  
- Aufnahme der Sensordaten von Octopus-Boards über das MQTT Protokoll  
- Speicherung der aufgenommenen Daten in einer Datenbank  
- Visualisierung über eine Internetseite

Es gab keine Einschränkungen in Hinblick auf verwendete Architektur, Entwicklungssprachen und verwendete Programme.

Die gesamte Projektarbeit fand unter Eigenarbeit ohne konkrete Aufsicht eines Lehrers statt.  
Zusätzlich zur Umsetzung des Projektes wurden einige Unterrichtsstunden zur Vorstellung von Programmiersprachen, Architekturen sowie auch Datenbank-Grundlagen verwendet.

# Projektumsetzung

## Aufgabenverteilung

Manuel Yates: Projektleitung, Softwareentwicklung, System-Architektur

Peter Rietz: Software-Entwicklung, Dokumentation(Software)

Michael Schkalei: Softwareentwicklung, Frontend

Lukas Sänger: Softwareentwicklung, Datenbank

Felix Becker: Hardware, Dokumentation (Hardware)

Luca Grethen: Hardware, System-Architektur

## Projektumgebung (Hardware)

Zur Aufnahme von Sensordaten wurden die sogenannten „Octopus-Board“ der BBS verwendet.  
Hierbei handelt es sich um Microcontroller Boards, welche standardmäßig über einen Temperatur-, Luftdruck- und Luftfeuchtigkeitssensor verfügen. Zusätzlich können diese über WLAN über das MQTT-Protokoll kommunizieren und Daten austauschen.

Zusätzlich wurde in der finalen Version ein Raspberry Pi 4 mit 2GB Arbeitsspeicher zum Speichern der Daten und zum Hosten des Webservers verwendet.  
  
Abseits davon wurden lediglich USB-Kabel zur Stromversorgung, Netzwerkkabel für die Kommunikation und eine MicroSD Karte für das Betriebssystem des Raspberry Pi’s verwendet. Die MicroSD Karte sollte mindestens 16 GB Speicherplatz haben. Zu beachten ist ebenfalls die Lese- und Schreibgeschwindigkeit, da diese Geschwindigkeit des Raspberry Pis stark beeinflusst.

## Projektumgebung (Software)

### Arduino-Studio (Ardublock)

Für die Entwicklung der Software auf dem Octopus Board wurde die Software Arduino-Studio in Kombination mit der Erweiterung Ardu-Block verwendet. Hierbei handelt es sich um eine Erweiterung die eine „Low-Code“ Programmierung mit Hilfe von Blöcken ermöglicht. So können schnell und übersichtlich Programme erstellt, kompiliert und auf die Octopus Boards geladen werden. Zusätzlich ermöglicht es einen leichten Einstieg in die Programmlogik, ohne vom Anwender die benötigte Programmiersprache zu verlangen.

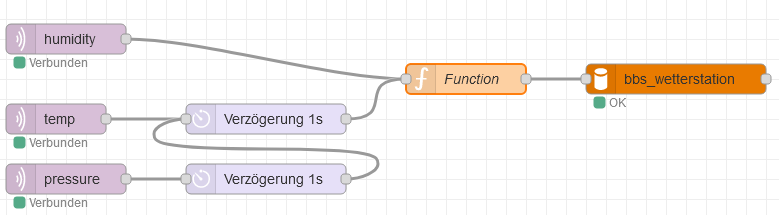
### Betriebssystem (Raspberry Pi)

Als Betriebssystem auf dem Raspberry Pi kommt „Raspberry Pi OS Lite“ in der 64 Bit Variante zum Einsatz. Dieses Betriebssystem ist eine modifizierte Version der Linux-Distribution „Debian“, welche Weltweit einen Großteil der auf Server verwendeten Betriebssystemen ausmacht. Raspberry Pi OS Lite bietet sich in diesem Kontext hervorragend an, da auf eine Desktop Oberfläche vollkommen verzichtet wird und so Ressourcen eingespart werden. Der Zugriff auf das Betriebssystem findet über das Netzwerkprotokoll SSH statt. Dieses bietet einen gesicherten Zugriff auf den Raspberry Pi und stellt die Systemkonsole (Terminal) zur Verfügung. Über das Terminal können alle wichtigen Schritte für die Einrichtung der benötigten Programme vollzogen werden. Hierfür wird die Software PuTTY oder der Standard Linux Terminal verwendet.

### Mosquitto (Raspberry Pi)

Als MQTT Broker wurde die Software Mosquitto von Eclipse verwendet. Hierbei handelt es sich um einen der verbreittesten MQTT Broker. Die Funktionalität des Brokers wird im Verlaufe dieser Dokumentation noch weiter behandelt.

### NodeRed (Raspberry Pi)

Da eine direkte Kommunikation zwischen MQTT Broker und Datenbank nicht möglich ist, wurde das Programm NodeRed verwendet. Hierbei handelt es sich um ein „Workflow“-Tool welches häufig im IoT Bereich Verwendung findet. Mit diesem Tool können vordefinierte Abläufe realisiert und gesteuert werden. Das Programm verwendet wie auch ArduBlock einen „Low-Code“ Ansatz. Hier können über kleine Blöcke und Pfeile gezielte Abläufe optisch strukturiert realisiert werden.

### Datenbank (Raspberry Pi)

Als Datenbank wurde eine MariaDB mit MySQL verwendet. Dieses relationale Datenbanksystem bietet sich an, da es problemlos unter Linux funktioniert und einen guten Kompromiss aus Geschwindigkeit und Benutzerfreundlichkeit gibt. Die verwendete Datenbanksprache ist SQL.

Als Verwaltungssoftware wurden PhpMyAdmin und MySQLWorkbench verwendet. PhpMyAdmin ist ein Standarddienst der XAMPP Programmfamilie. Über die Weboberfläche können schnell und übersichtlich Zugriffe auf die Datenbank stattfinden. MySQLWorkbench ist ein Desktop-Programm zur Verwaltung von MySQL Datenbanken. Im Vergleich zu phpMyAdmin ist hier die Benutzer- und Einsteigerfreundlichkeit geringer, dafür aber die Zugriffs- und Verwaltungsmöglichkeiten höher.

### Programmiersprache & Frameworks

Als Programmiersprache wurde die objektorientierte Programmiersprache C# verwendet. C# wird von Microsoft entwickelt und befindet sich aktuell in der Version 10. Mit C# können neben Desktop (WPF) und Mobile-Applikationen (Xamarin) auch Webdienste (ASP, Blazor) und Multiplattform-Programme (.NET MAUI) realisiert werden. Verwendet wird Sie ebenfalls in Bereichen wie Machine-Learning (ML.NET) und Spieleentwicklung (Unity-Engine). Sie gehört damit zu den zukunftssichersten und populärsten Programmiersprachen und ist laut „Tiobe-Index“ Platz 5 der am meisten verwendeten Programmiersprachen.  
Im Projekt wurde das .NET Framework in der Version 6 verwendet. Dies ist die aktuelle Version und beinhaltet viele Hilfreiche Bibliotheken und Funktionen.  
Da die Wetterstation über eine Oberfläche im Internet verfügen sollte, wurde eine Blazor ServerSide App verwendet. Die Komponenten-Frameworks MudBlazor sowie RadzenBlazor wurden zur Darstellung der Oberfläche verwendet. Für die Anbindung an die Datenbank kam das Entity Framework als Objekt-Relationaler-Mapper zum Einsatz.

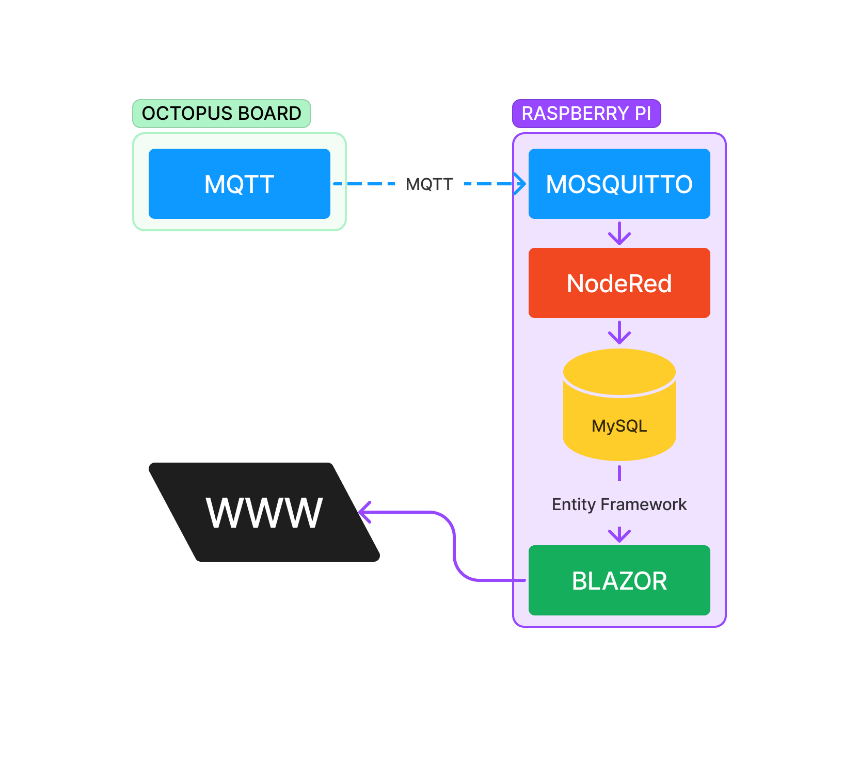
### Entwicklungsumgebung

Als sogenannte IDE (Integrated Development Environment) wird Visual Studio Community 2022 verwendet. Diese IDE ist speziell auf C# zugeschnitten und bietet neben Debugging einen großen Pool an nützlichen Funktionen und Tools.

### Versionsverwaltung

Als Versionsverwaltung kam Git über GitHub zum Einsatz. Git ist ein weit verbreitetes Open-Source-Tool zur verteilten Versionskontrolle von Software. Es wird sowohl in Unternehmen als auch von privaten Entwicklern weltweit genutzt und funktioniert zusammen mit verschiedensten Plattformen und IDEs. Über dieses Tool ist ein einfaches und übersichtliches Verwalten von Programmcode und Änderungen an diesem möglich. GitHub ist ein gewinnorientiertes Unternehmen, das einen Cloud-basierten Git Repository Hosting Service anbietet. Seit 2018 gehört das Unternehmen zu Microsoft.

## Allgemeine Struktur und Ablauf

Die grundlegende Struktur der Wetterstation besteht aus den oben beschriebenen Ocotopus Boards, welche in regelmäßigen Abständen ihre Sensordaten über Temperatur, Luftfeuchtigkeit und Luftdruck über das MQTT Protokoll an den MQTT Broker übermittelt („publisht). Das Programm NodeRed wurde so konfiguriert, dass so bald neue Werte übermittelt werden, diese in die Datenbank geschrieben werden.

Der Blazor-Webserver, welcher verantwortlich für den Internetauftritt ist, erhält nun diese Daten über das Entity Framework als Objekte. Im Backend des Servers werden diese nun an die Nutzeranfragen angepasst (Zeitraum, Genauigkeit etc.) und anschließend an eine Blazor-Komponente übergeben, welche die Diagramme darstellt.

# Installation und Einrichtung

### Schritt 1: Flashen des Betriebssystems für den Raspberry Pi

Zur Installation verwenden wir das Programm Raspberry Pi Imager. Dies ist speziell auf die üblichen Betriebssysteme und Konfigurationen eines Raspberry Pis ausgelegt.  
Zuerst wählen wir über die Schaltfläche OS WÄHLEN das gewünschte Betriebssystem aus. In unserem Fall Raspberry Pi OS Lite (64-bit). Anschließend öffnen wir die Einstellungen über die Schaltfläche mit dem Zahnrad-Symbol. Hier können wir einen Hostnamen vergeben sowie auch den Zugriff über SSH einstellen. Dies ist für den Remote-Zugriff später essenziell. Nach Vergabe eines Benutzernamens und Passwort können ebenfalls die Zugangsdaten für eine Wifi-Verbindung eingegeben werden. Abschließend klicken wir auf „SPEICHERN“ und wählen über die Schaltfläche „SD-KARTE WÄHLEN“ unsere MicroSD-Karte aus. Mit einem Klick auf „SCHREIBEN“ startet nun der Flash-Vorgang. Dieser kann einige Minuten in Anspruch nehmen.

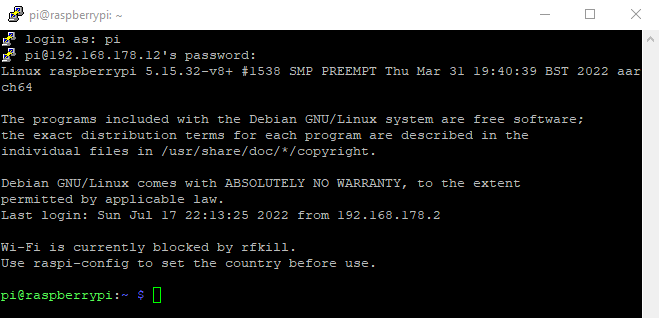
Anschließend schieben wir die MicroSD-Karte in den Speicherslot und verbinden den Raspberry Pi mit dem Netzstecker und bei Bedarf einem Netzwerkkabel.

### Schritt 2: Festlegen einer statischen IP-Adresse für den Raspberry Pi

Um eine reibungslose Verbindung mit dem Raspberry Pi sicherzustellen, wird von uns dringendst empfohlen dem Gerät eine statische IP-Adresse im Router zu vergeben. Hier werden wir nicht weiter auf die benötigten Schritte eingehen, da diese stark vom jeweiligen Router abhängen.

### Schritt 3: Verbindung zum Raspberry Pi über PuTTY

Für die Verbindung über SSH öffnen wir zuerst das Programm PuTTY. Im Eingabefeld „HostName (or IP adress)“ tragen wir nun die festgelegte IP-Adresse ein. Anschließend starten wir die Verbindung mit einem Klick auf „Open“.



Es öffnet sich nun ein neues Terminal-Fenster in welchem wir nach unseren Anmeldedaten gefragt werden. Nach der erfolgreichen Anmeldung sollte folgendes Terminal zu sehen sein.

### Schritt 4: Einrichtung der Wetterstation

Um nun sicherzustellen ob das Betriebssystem sowie alle zugehörigen Pakete auf dem aktuellen Stand sind geben wir folgende Befehle in den Terminal ein:

sudo apt update

sudo apt upgrade

Nachdem die Updates erfolgreich installiert wurden, beginnen wir nun mit der Installation von Mosquitto. Hierfür verwenden wir den Befehl:

sudo apt install mosquitto

Auf Abfragen im Installationsprozess antworten wir mit Y.  
Wenn die Installation abgeschlossen ist, müssen wir nun die Standardeinstellungen überschreiben. Hierzu verwenden wir den Terminal Editor Nano welcher standardmäßig installiert ist. Mit dem Befehl:

sudo nano /etc/mosquitto/mosquitto.conf

öffnen wir die Konfigurationsdatei von Mosquitto. In dieser tragen wir ans Ende der Datei folgende zwei Dinge ein:

listener 1883

allow\_anonymous true

Mit diesen Einstellungen legen wir fest, dass Mosquitto über den Port 1883 aus dem Netzwerk erreichbar ist und Werte empfängt, ohne das ein Benutzer oder Passwort benötigt wird. Wir verlassen Nano mit STRG + X und bestätigen mit einem Y und Enter, dass wir die Datei überschreiben wollen. Nun installieren wir NodeRed mit diesem Befehl:

bash <(curl -sL <https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered>)

Nachdem das Installationsskript erfolgreich beendet wurde. Können wir nun unser Datenbank-System installieren.

sudo apt install mariadb-server

Ein Bild, das Text enthält.

Automatisch generierte BeschreibungAuch hier müssen wir mit Nano die Standard-Konfiguration erweitern.

sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf

Um von anderen Geräten auf die Datenbank zuzugreifen müssen wir in der Konfigurationsdatei im Eintrag bind-adress die IP-Adresse 0.0.0.0 eintragen.  
Nun erstellen wir über die Konsole einen „Super-User“. Dieser soll uns die Möglichkeit geben über eine Remote-Verbindung unsere Datenbank vollständig zu managen.

Zunächst loggen wir uns über folgende Befehle in die Datenbank ein:

sudo mysql -u root -p

Da der Standard-User root über kein Passwort verfügt, können wir bei der Passwort-Abfrage einfach mit Enter bestätigen. Anschließend befinden wir uns im Terminal Menü der Datenbank. Um hier unseren User zu erstellen, verwenden wir folgende Befehle:

CREATE USER 'Admin'@'%' IDENTIFIED BY 'root';

GRANT ALL PRIVILEGES ON \*.\* TO 'Admin'@'%' WITH GRANT OPTION;

FLUSH PRIVILEGES;

Nun verfügen wir über einen User mit allen Berechtigungen. Das „%“ steht in diesem Falle für eine Wildcard und bedeutet, dass sich von jedem Gerät im Netzwerk auf diesen Account angemeldet werden kann. Um hierfür eine einfache Verwaltungsoberfläche im Browser zu nutzen installieren wir phpMyAdmin:

sudo apt install phpmyadmin

Während des Installationsvorgang werden wir nach unserem Server-Typen gefragt, welchen wir installieren möchten. Hier wählen wir apache2 aus. Nach dem vollendeten Installationsvorgang müssen wir nun auch den Apache Webserver im Netz erreichbar machen.

sudo nano /etc/apache2/apache2.conf

Ans Ende dieser Datei fügen wir die Zeile Include /etc/phpmyadmin/apache.conf ein. So wird nach einem Neustart unser Apache Server im Netzwerk erreichbar sein. Erfahrungsgemäß muss allerdings noch folgende Bibliothek installiert werden:

sudo apt-get install libapache2-mod-php

Bevor wir uns phpMyAdmin widmen, sollten wir noch git und die .NET Runtime installieren.

sudo apt install git

curl -sSL https://dot.net/v1/dotnet-install.sh | bash /dev/stdin --channel Current

echo 'export DOTNET\_ROOT=$HOME/.dotnet' >> ~/.bashrc

echo 'export PATH=$PATH:$HOME/.dotnet' >> ~/.bashrc

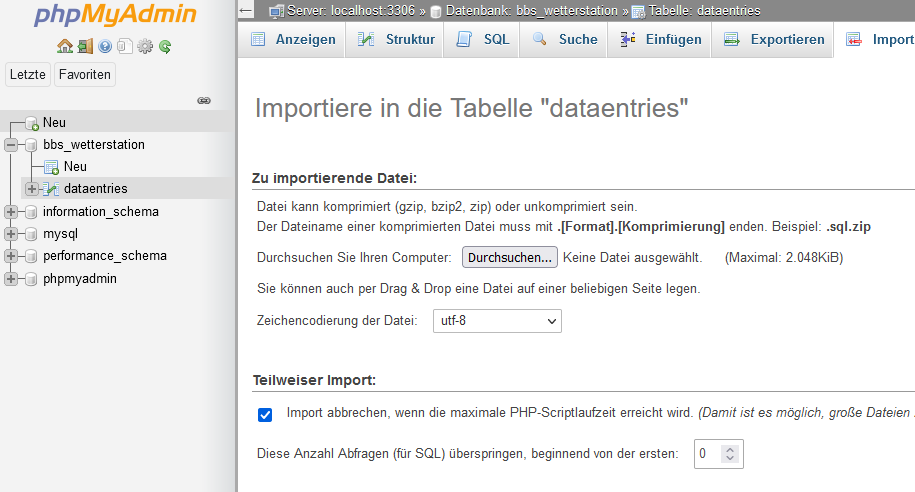
source ~/.bashrc

Um unsere Dienste direkt dem Autostart hinzuzufügen, verwenden wir die Befehle:

sudo systemctl enable mosquitto

sudo systemctl enable nodered.service

Nun können wir mit dem Befehl sudo reboot unseren Pi neustarten.

Nachdem der Pi neu gestartet ist legen wir nun unsere Datenbank an. Wir melden uns mit unserem Super-User Daten bei phpMyAdmin an und klicken in der Datenbankübersicht links auf Neu. Als Datenbank-Namen geben wir bbs\_wetterstation an und bestätigen mit Anlegen.

Nun wählen wir den Menüpunkt Importieren. Hier wählen wir die Datei dataentries.sql aus dem Repository aus. Dies ist unsere Vorlage für die Datenbank. Mit OK rechts bestätigen wir den Vorgang.

Nachdem der Import erfolgreich abgeschlossen ist, wählen wir im Reiter SQL aus und geben dort folgende Statements ein:

CREATE USER 'NodeRed'@'%' IDENTIFIED BY 'nodered';

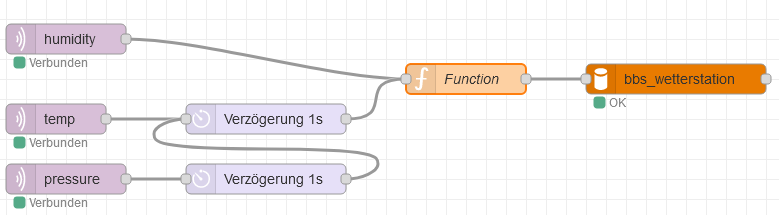
CREATE USER 'WebServer'@'%' IDENTIFIED BY 'Wittlich';

GRANT ALL PRIVILEGES ON bbs\_wetterstation.\* to 'NodeRed'@'%' IDENTIFIED BY 'nodered';

GRANT ALL PRIVILEGES ON bbs\_wetterstation.\* to 'WebServer'@'%' IDENTIFIED BY 'Wittlich';

FLUSH PRIVILEGES;

Nun existieren die benötigten Benutzer auf der Datenbank und diese verfügen auch über die entsprechenden Berechtigungen.

Nun können wir unseren Flow in NodeRed installieren. Hierfür rufen wir im Browser die IP-Adresse unseres Pis über den Port 1880 auf. Mit einem Klick auf „Hamburger-Menü“(rechte obere Ecke) => „Palette verwalten“ => Reiter „Installation“ können wir mit dem Suchbegriff „MySQL“ die Erweiterung „node-red-node-mysql“ installieren. Diese wird für unseren Flow benötigt. Über den Menüpunkt „Import“ können wir nun auch die Datei „flows.json“ aus dem Repository hochladen. Nach dem Import erscheint im oberen Reiter nun ein neues „Blatt“ mit dem Namen „Flow 1“.

Nun ist das System bereit um Daten zu Empfangen.

### Schritt 5: Einrichtung des Octopus Boards

Zum Hochladen des Programmes auf das Board öffnen wir die Arduino-Studio Software und wählen im Reiter „Werkzeuge“ den Punkt Ardublock aus. Es öffnet sich nun die Ardublock Oberfläche. In dieser wählen wir über Öffnen die Datei bbs\_wetterstation.abp aus. In der Übersicht tragen wir folgende Werte ein:

* Netzname = WLAN-Name
* Passwort = WLAN-Passwort
* Broker = IP-Adresse des Pis

Anschließend klicken wir auf „Hochladen auf den Octopus“. Wichtig ist, dass im Arduino-Studio der richtige COM-Port ausgewählt ist.  
Wenn nun das Board angeschlossen wird, sollte es im Abstand von 60 Sekunden Grün blinken und Daten senden.

### Schritt 6: Starten des Webservers

Nun sind alle Vorkehrungen getroffen, um den Webserver in Betrieb zu nehmen. Hierzu verbinden wir uns wieder mit PuTTY auf unseren Raspberry Pi. Nun legen wir uns zur Übersichtlichkeit ein separates Verzeichnis für Github an.

mkdir Github

cd Github/

Nun klonen wir das Repository von Github auf den Pi:

git clone <https://github.com/PorscheManu/BBS_WeatherStation.git>

Nachdem der Download abgeschlossen ist, navigieren wir ins Verzeichnis Releases/raspi. Hier führen wir folgenden Befehl aus:

dotnet Server.dll --urls [https://\*:5001](https://*:5001)

Nun sollte der Blazor-Server hochfahren und über die IP-Adresse des Raspberry Pis und den Port 5001 sollte die Website auch im Netzwerk erreichbar sein.

# Begründung dieser Umsetzung

Die Gruppe hatte vor der Umsetzung in C# einen Versuch unternommen, mit Hilfe der Programmiersprachen Python und JavaScript das Projekt umzusetzen. Nach ersten erfolgreichen Ansätzen (auch zu finden im Repository unter ./) wurde jedoch von Seiten der Software-Entwickler gewünscht möglichst sich auf eine Programmiersprache zu beschränken. Nach einiger Abwägung wurde sich dann auf C# geeinigt.

Bei der Erarbeitung der System-Architektur wurde nach ausführlicher Test-Phase das obige Konzept erstellt. Durch dieses ist neben einer plattformunabhängigen Lauffähigkeit auch eine Verteilung auf mehrere Systeme problemlos möglich. So kann grundsätzlich unabhängig von Betriebssystem oder Hardware das System in Betrieb genommen werden.

# Einen Blick in die Zukunft

Als nächsten Schritt wurde überlegt die Notwendigkeit von Drittanbieter Software, speziell die Software NodeRed & Mosquitto bestmöglich oder vollständig durch Eigenentwicklung zu lösen. Hierfür würde die Gruppe einen eigenen MQTT Broker in C# programmieren, welcher dann direkt gekoppelt an den Webserver Einträge in die Datenbank vollziehen könnte. So wäre vollkommen unabhängige Datenverarbeitung möglich und Erweiterungen z.B.: in Zusammenarbeit mit anderen Instituten leichter und einfacher zu handhaben. Hierfür müssten ebenfalls neue Oberflächen für Administratoren hinzugefügt werden. Außerdem könnten so Ressourcen eingespart werden, da für NodeRed eine Hintergrund-Instanz des Webservers Node.js installiert und aktiv sein muss.

Eine weitere Überlegung ist, die benötigten Dienste als „Docker-Container“ zu verwenden. Docker kann man als „light“ Version einer virtuellen Maschine verstehen. Hierbei werden jedoch nicht Bestandteile der Hardware virtualisiert, sondern lediglich Software-Dienste oder Betriebssysteme.  
So könnte man den MQTT-Broker, die Datenbank und den Webserver als einzelne Container virtualisieren, damit vollständig voneinander entkoppeln und so auch nach Belieben auswechseln. Hierzu wurden bereits einige erfolgreiche Testversuche zu Beginn des Projektes unternommen, allerdings aus zeitlichen Gründen nicht weiterverfolgt.

Abseits von Änderungen an dem Projekt gibt es die Überlegung eine virtuelle interaktive Dokumentation des Projektes anzulegen. So könnten Schüler der Schule einen umfangreichen Einblick in das Projekt bekommen und mögliches Interesse an den Fachbereichen wecken.

Als letzte Überlegung wurde die Nutzung von Cloud-Basierten Server-Diensten wie Microsoft Azure oder Amazon AWS in Kombination mit einem automatischen Test- und Release-Mechanismus angesprochen. Dies wurde bisher allerdings noch nicht weiterverfolgt.

# Anmerkungen

Dieses Projekt wurde zum aktuellen Zeitpunkt lediglich unter den Gesichtspunkten von Funktion und Aussehen erstellt. Etwaige Sicherheitsmängel sind nicht ausgeschlossen und wir weisen eindeutig darauf hin, dass vor Inbetriebnahme sämtliche Mängel behoben werden sollten.   
Weiter wurde während der Projektarbeit Drittanbieter-Software, Frameworks und Erweiterungen verwendet. Es folgt nun eine Liste jeder externen Software, welche nicht von unserer Gruppe entwickelt wurde und die jeweiligen Lizenzen.

|  |  |  |
| --- | --- | --- |
| **Software(-Bestandteil)** | **Herausgeber** | **Lizenz** |
| MySQL | Oracle | GPL-2.0 |
| Node.js | Node.js Foundation, Ryan Dahl | MIT |
| NodeRed | IBM Emerging Technology | Apache 2.0 |
| phpMyAdmin | The phpMyAdmin Project | GPL-2.0+ |
| Blazor | Microsoft | Apache 2.0 |
| Entity Framework | Microsoft | MIT |
| MudBlazor | Gardnet AB | MIT |
| RadzenBlazor | Radzen | Apache 2.0 |
| *Nuget:* Pomelo.EntityFrameworkCore.MySql | Laurents Meyer, Caleb Lloyd, Yuko Zheng | MIT |

Zusätzlich wurde das Projekt über das GitHub-Repository während der Entwicklungsphase veröffentlich und ist über diesen [Link](https://github.com/porschemanu/BBS_WeatherStation) erreichbar.

Schließlich wurde zur besseren Demonstration die Webseite der Wetterstation unter der Adresse „weatherstation.pm913.de“ erreichbar gemacht. Hierbei sei angemerkt, dass dies über einen privaten Raspberry Pi, eine private Portfreigabe und einen virtuellen Linux Server bei der STRATO AG erfolgt. Der erwähnte Server dient lediglich als Relay-Server zum externen IPV4 Zugriff. Die oben genannte Sub- sowie Hauptdomain sind im Besitz von Manuel Yates, ebenfalls eingerichtet bei der STRATO AG. Sollte die Deaktivierung gewünscht werden, so senden Sie bitte eine E-Mail an [manuel@pm913.de](mailto:manuel@pm913.de).