Heterogeneous Computing
Übung 3, Ergebnisbericht

Tillmann Faust

29.9.2024

Contents

1 Einleitung

1.1 Projektziele

Ziel dieses Projekts war es, unter Verwendung eines Mikrocontrollers, z.B. Arduino oder ESP32, eine funktionierende Wetterstation zu bauen, welche in der Lage ist Temperatur, Luftdruck, Luftfeuchtigkeit und Regenfall zu messen und diese Daten zur Weiterverarbeitung an einen zu Senden. Weiterhin soll ein Server aufgesetzt werden, welcher in der Lage ist die Daten zu empfangen, zu speichern und anschließend für Interessenten in einer simplen Web-Ansicht zur Verfügung zu stellen.

1.2 Motivation

"Heterogeneous Computing", als Teilbereich der praktischen Informatik, beschäftigt sich mit der Nutzung verschiedener Rechenarchitekturen, und auf diese speziell zugeschnittene Software in einem System, um Probleme und Aufgaben effizienter und/oder effektiver zu lösen. Die Motivation hinter diesem Projekt liegt in der praktischen Umsetzung des Heterogeneous Computing durch die Verknüpfung eines eingebetteten Systems, genauer eines Mikrocontrollers mit einem Server um die Vorteiler beider Plattformen bestmöglich auszunutzen.
Dieses Projekt entstand als Abschlussübung im Rahmen der Veranstaltung "Heterogeneous Computing" an der Universität Trier im Sommersemester 2024.

2 Projektplanung

2.1 Funktionale Anforderungen

Die Wetterstation soll dazu in der Lage sein in einem Intervall von 10 Minuten Daten über Temperatur, Luftfeuchtigkeit, Luftdruck und Regenfall zu Sammeln, vorzuverarbeiten und an einen Server zu senden. Die Station soll sich mit dem lokalen Netzwerk verbinden können um die interne Uhr mit einem Zeitserver zu synchronisieren und um das Senden der Daten zu ermöglichen. Weiterhin soll es eine Möglichkeit für Beobachter geben den aktuellen Status der Station im Betrieb zu beurteilen. Dazu sollen eine rote und eine grüne LED verwendet werden, welche nach den 3 Hauptteilen des Programms, nämlich der Initialisierung, der Datensammlung und der Datensendung entsprechend von Erfolg und Misserfolg leuchten. Der Server soll einen MQTT-Broker bereitstellen, welcher die Daten der Station empfangen und weiterleiten kann. Zudem sollen die Daten lokal auf dem Server in einer Datenbank gespeichert werden. Dazu kann ein simples Verarbeitungsskript verwendet werden. Zuletzt soll auf dem Server eine Web-UI laufen, welche Teile der Wetterdaten für Interessenten abfragt und graphisch darstellt.

2.2 Ressourcen

2.2.1 Mikrocontroller

Als Mikrocontroller wird ein ESP32-D0WDQ6 mit einem Tensilica LX6 Dual-Core@240MHz und 4MB Flash Speicher verwendet. Das Board verfügt über ein Wlan Modul und unterstützt Bluetooth 4.2 und 5.0.

2.2.2 Sensoren

Temperatur, Luftfeuchtigkeit und -druck Zum messen von Temperatur, Feuchtigkeit und Druck gibt es eine große Auswahl an verschiedenen einzelnen Komponenten oder auch Kombinationen von Sensoren zur Verwendung in Hardwareprojekten. Beispielhaft aufzuführen wären der DHT11 oder DHT22 als Kombination aus Temperatur- und Feuchtigkeitssensor, der BMP280 ist eine Kombination aus Temperatur- und Luftdrucksensor. In diesem Projekt wurde sich für den BME280 von Bosch entschieden welcher in der Lage ist sowohl Temperatur, Luftdruck, als auch Feuchtigkeit mit einer hohen Präzision zu messen. Der Temperatursensor misst Temperaturen zwischen von -40°C bis +85°C und hat eine Genauigkeit von ±1.0°C zwischen 0°C und 60°C. Der Luftdrucksensor hat einen Messbereich von 300hPa bis 1100hPa und misst mit einer absoluten Genauigkeit von ±1.0hPa, kann unter idealen Bedingungen allerdings auch zu noch genaueren Werten gelangen. Der Luftfeuchtigkeitssensor misst von 0% bis 100% relative Luftfeuchtigkeit und besitzt eine Genauigkeit von ±3% zwischen 20% und 80%.

Regenmessung In den Recherchen zur Messung von Regenmengen sind einige Methoden wiederholt aufgetaucht: Die Trichter-Methode funktioniert, indem das Regenwasser auf einer gewissen Fläche über einen Trichter in einem Container gesammelt wird. Anhand der Veränderung des Wasserstandes in dem Behältnis kann der Regenfall anschließend gemessen werden. Die Kipp-Trichter-Methode ist eine Erweiterung der Trichter-Methode, in welcher das Wasser auf eine Art Waage geleitet wird auf deren Seiten sich kleine Auffangbehältnisse befinden. Das Wasser fließt immer auf eine Seite, bis die Schale voll ist umkippt und geleert wird. Die zweite Schale ist derweil zum Sammeln von Wasser in Position gebracht.

Für das Projekt wurde sich gegen Methoden zur Messung genauer Regenmengen entschieden und stattdessen ein simpler MH-RD Regensensor verwendet, welcher aus einer Platte mit zwei Leiterbahnen besteht. Ein zweiter IC kann mit dieser Platte verbunden werden und den Widerstand der Platte bestimmen. Bei Regenfall sorgen Tropfen auf der Platte für eine messbare Verringerung des Widerstandes. Diese Methode ist deutlich weniger aufwendig, als die zuvor aufgeführten, liefert allerdings auch deutlich weniger interessante Daten. Der Regensensor ist in der Lange den Widerstand über einen analogen Ausgang auszugeben wodurch sich die Stärke des Regenfalls inferieren lässt, genauere Daten lassen sich allerdings nicht erheben.

2.2.3 Server Hardware

Als Server dient ein Raspberry Pi 2 Model B mit einem ARM Cortex-A7 Quad-Core@900MHz, 1GB Arbeitsspeicher und ist mit einer 512GB Micro SD Karte als persistenten Speicher ausgestattet.

3 Umsetzung

3.1 Systemarchitektur

Das System besteht aus zwei primären Komponenten, der Wetterstation, welche die Aufgabe hat Wetterdaten zu erheben und diese an den Server weiter zu leiten.

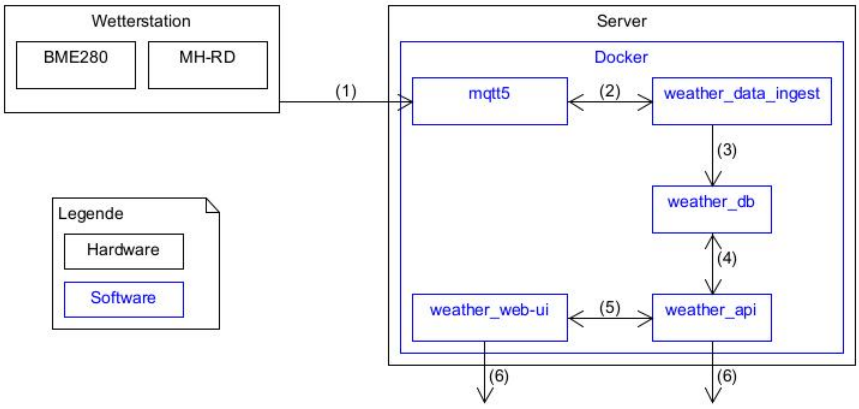

Figure 1: Grobe Übersicht der Komponenten und Interaktion

1. Die Wetterstation sendet Wetterdaten an den MQTT-Broker
2. Das Skript subscribed beim Broker und erhält die Wetterdaten
3. Die Wetterdaten werden in die Datenbank geschrieben
4. Die API fragt die Wetterdaten ab und Stellt diese der Web-UI und anderen Interessenten auf Port 3000 zur Verfügung
5. Die Web-UI Fragt die API ab und stellt die Ergebnisse der Abfrage auf Port 8080 graphisch dar

Um eine möglichst gute Isolierung der einzelnen Serverkomponenten zu ermöglichen wurde Docker verwendet. Alle Container befinden sich in einem Docker-Netzwerk was eine einfachere interne Kommunikation ermöglicht, welche nicht über das interne Netzwerk des Servers laufen muss. Weiterhin werden

auf dem Server die Ports 3000 für die API und 8080 für die Web-UI geöffnet, damit jeder Rechner im Netzwerk die Möglichkeit hat auf beide zuzugreifen. Die genauen Ports für die Interaktion zwischen Containern im Netzwerk und Komponenten außerhalb sind in der ”docker-compose.yml” definiert.

3.2 Kommunikation

Die Kommunikation zwischen der Station und dem Server erfolgt über das MQTT-Protokoll über das lokale Netzwerk. MQTT oder "Message Query Telemetry Transport" ist Netzwerkprotokoll, welches sich durch seine geringe Bandbreitennutzung auszeichnet. Es eignet sich ideal für eingeschränkte Netzwerke und Geräte mit einer hohen Latenz und findet daher häufig Verwendung in der Maschine zu Maschine Kommunikation und IoT Anwendungen. Das Protokoll funktioniert über ein Publisher-Subscriber-System, in welchem Gerate oder Programme, so genannte Publisher, Daten zu einem gewissen Topic an einen Broker senden, welcher diese dann an Subscriber weiterleitet. Alle Daten laufen über den Broker, es gibt also keine Direkte Verbindung zwischen Publisher und Subscriber. In diesem Projekt stellt die Wetterstation den Publisher dar, welcher die vor-verarbeiteten Wetterdaten unter dem Topic 'weather' published. Auf dem Server laufen ein MQTT-Broker, welcher dauerhaft auf eingehende Verbindungen horcht und die Daten anschließend weiterleitet und ein Subscriber-Script zur Weiterverarbeitung.

3.3 Hardware-Design

3.3.1 Wetterstation

Für die Wetterstation wurde der esp-wroom-32 auf zwei zusammengesteckten Breadboards mit den Sensoren verbunden. Tabelle 1 gibt eine Übersicht über alle verwendeten Teile und deren Verbindung mit dem ESP32. Figur 2 ist ein Bild des Abgabefertigen Prototypen zur Referenz im weiteren Verlauf. Auf die Komponenten soll nun im folgenden genauer eingegangen werden:

Part	Part Pin	ESP Pin
BME280	V_{IN}	3.3V
	GND	GND
	SDA	GPIO21
	SCL	GPIO22
MH-RD Regensensor	VCC	3.3V
	GND	GND
	DO	—
	AO	GPIO33
LED rot	V_{IN}	GPIO5
	GND	GND
LED grün	V_{IN}	GPIO19
	GND	GND

Table 1: Liste der benutzen ICs mit den zugehörigen Pin-Verbindungen

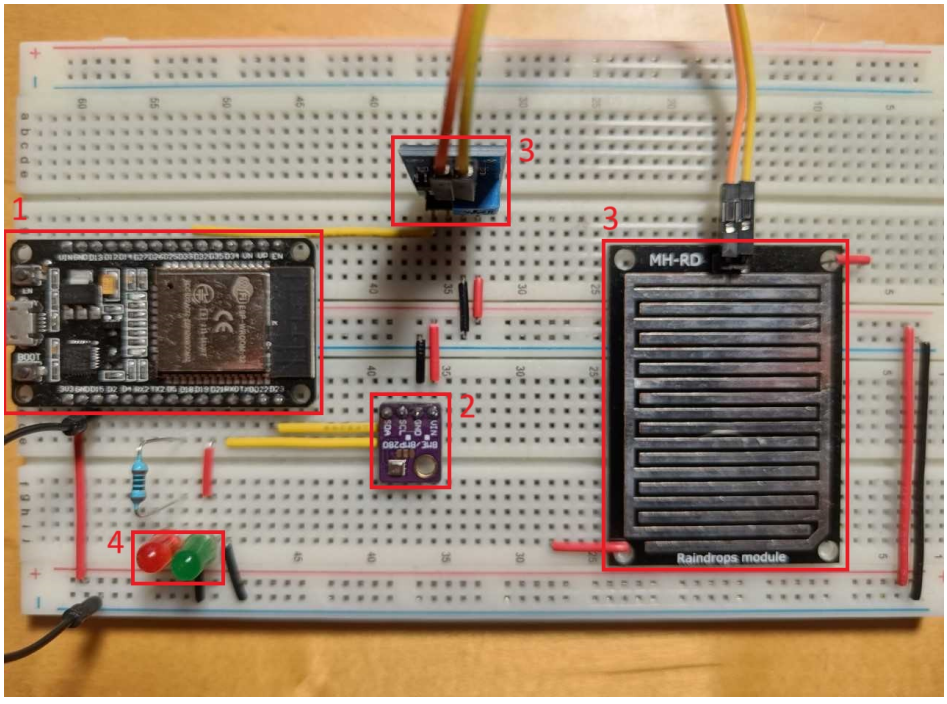

Figure 2: Prototyp der Wetterstation

(1) ESP32 Der ESP32 stellt das Herzstück der Wetterstation dar, mit welchem die Restlichem Sensoren verbunden sind und welcher die Aufgabe hat die gemessenen Daten zu verarbeiten und an den Server weiter zu leiten.

(2) BME280 Der Bosch BME280 ist ein 2,5 x 2,5 x 0,93 mm^3 kleiner Sensor, welcher Temperatur, Luftdruck sowie Luftfeuchtigkeit präzise messen kann und dazu entworfen wurde, durch seine Größe und geringen Stromverbrauch, auf einer Vielzahl an IoT Geräten Anwendung finden zu können. Der Sensor verfügt über eine I2C-, sowie eine SPI-Schnittstelle. Verwendet wurde ein vorgefertigtes Breakout-Board, welches die 8 Pins des Sensors auf 4 Pins bündelt und die Verwendung der I2C Schnittstelle ermöglicht. Die Standard-I2C-Adresse des Chips ist 0x76.

(3) MH-RD Regensensor Als Regensensor wurde ein MH-RD Sensor von AZD-Delivery verwendet, welcher aus einem PCB mit zwei Leiterbahnen und einem kleinen Modul zum Messen der Spannungsdifferenz besteht. Der ESP32 wird anschließend mit dem Auslesemodul verbunden. Der MH-RD kann sowohl per digitalem Signal (DO) ausgeben ob gerade Regenfall besteht oder über einen analogen Output (AO) den aktuell gemessenen Widerstand übermitteln. Um im weiteren Verlauf selbst entscheiden zu können, was als "Regenfall" betrachtet wird, wurde nur der AO-Pin mit dem ESP32 verbunden.

(4) LEDs Die LEDs sind an eigene GPIO-Pins angeschlossen und sollen während einer Aktiven Datensammel und -sende Phase mögliche Beobachter über den erfolgreichen, oder nicht erfolgreichen, Programmablauf informieren. Die rote LED ist zudem über einen 100Ω Widerstand an GND gebunden um eine Überspannung zu verhindern.

3.4 Software-Design

Das Softwaredesign des Projekts setzt sich zusammen aus der Software für die Station selbst, und Docker auf Seiten des Servers, auf welchem der MQTT Broker, die Datenbank, das Skript zur Datenaufnahme und die Web-Ansicht in je einem Container laufen. Im folgenden explizit benannt und unterstrichene Dateien und Ordner befinden sich ebenfalls in den Abgabedateien. Login Informationen, wie Nutzernamen und Passwörter wurden vor Abgabe abgeändert.

3.4.1 Wetterstation

Der Code für die Wetterstation wurde in Arduino geschrieben und befindet sich in der Datei "weather_station_v.0.1.ino". Der Code folgt einer simplen linearen Struktur. Zu Beginn werden Variablen, Konstanten und Structs zur späteren Verwendung und Vereinfachung des Codes definiert. Weiterhin werden die einzelnen Schritte, als das Networking, der Datenauslese und das Senden, in einigen separaten Methoden verpackt und sollen im folgenden gelistet und erklärt werden:

systemInit() Entsprechend des Namens wird in dieser Methode eine Verbindung mit dem Lokalen Netzwerk hergestellt, die lokale Zeit mit einem Zeitserver synchronisiert und anschließend versucht eine Verbindung mit dem BME280 über

I2C aufzubauen. Sollte keine Netzwerkverbindung möglich sein, so endet der ESP32 in einer Sisyphos artigen Schleife des endlosen Versuchs, schlägt die Initalisierung des BME280 fehl so gibt die Methode einen 'false' Wert zurück.

systemDecon() Diese Methode soll alle möglichen Variablen und Zuweisungen dekonstruieren, sobald die loop-Methode erfolgreich durchlaufen wurde. Sie ist in dem Versuch entstanden den Deep-Sleep-Mode des ESP32 zu verwenden. Da in dieser Methode die Netzwerkverbindung getrennt und das W-Lan Modul abgeschaltet wird, führt die Methode zu einem insgesamt geringeren Stromverbrauchs des Geräts.

sleepToFullSixth() Da die Wetterstation die Wetterdaten alle 10 Minuten erfassen soll, war das einfachste die aktuelle Zeit nach einem Durchlauf abzufragen und die Differenz bis zur nächsten vollen Minute zu berechnen. Anschließend wird der ESP32 für die Differenz in Millisekunden delayed. Ursprünglich sollte der ESP32 in den Deep-Sleep-Mode, dies hat allerdings bis zur Abgabe nicht geklappt und wird daher außerhalb der Abgabe wieder aufgegriffen.

Datenauslese Die Methoden 'readBME280Data', 'readRainSensor' und 'getCurrentTime' lesen die gefragten Daten ein und geben diese in geeigneten Datenstrukturen zurück.

Datenvorverarbeitung und senden Die Methode 'createJsonPayload' nimmt die zuvor gesammelten Daten und bereitet diese als Json-String zum Senden auf. Anschließend wird dieser Json-String in 'sendJsonPayload' - Sofern zuvor eine erfolgreiche Verbindung in 'connectTOMQTT' mit dem MQTT Broker hergestellt wurde - an den Broker unter dem Topic 'weather' weiter.

3.4.2 Docker

Docker ist eine Software für Linux, welche es erlaubt Anwendungen als isolierte Container auf einem Rechner laufen zu lassen. Jeder Container fungiert dabei als seine eigene abgekapselte Instanz was die Portabilität und Erweiterbarkeit enorm erhöht. Zum einfacheren Management kann Docker-Compose verwendet werden, damit Containerstrukturen gebündelt und damit einfacher definiert und verwaltet werden können. Dies geschieht über eine sogenannte "docker-compose.yml". Dieses Projekt verwendet Docker-Compose um die Server-seitigen Komponenten, namentlich der MQTT-Broker, die Datenbank, das Datenverarbetungs-Skript, die Web API und die Web-UI zu verwalten. Die Komponenten laufen also insgesamt in fünf Containern, welche sich ein Docker-Netzwerk, zur einfachen internen Kommunikation, teilen.

3.4.3 Mosquitto

Der MQTT-Broker Mosquitto wurde als Image in der docker-compose.yml definiert und hat die Aufgabe die von der Wetterstation gesendeten Json-Strings an die

Subscriber, speziell das Subscriber-Script weiterzuleiten.

3.4.4 MariaDB

Der ursprüngliche Plan war ein MySQL image für die Datenbank zu benutzen, da dies allerdings nicht funktioniert hat wurde stattdessen ein MariaDB Derivat, was wiederum selbst ein MySQL Derivat ist und über dieselben Funktionalitäten verfügt. Die Datenbank 'wather_db' besteht, neben den Standardtabellen für Metadaten, über eine Tabelle 'weather_data' mit den Spalten id, year, month, day, hour, minute, temperature, pressure, humidity und rain.

id	year	month	day	hour	minute	temperature	pressure	humidity	rain
1	2024	9	13	21	51	25.27	1012.71	48.76	0
2	2024	9	13	22	0	24.70	1012.78	48.82	0
3	2024	9	13	22	10	24.34	1012.87	48.06	0
4	2024	9	13	22	20	24.66	1013.00	48.59	0
5	2024	9	13	22	30	24.17	1013.07	48.00	0
6	2024	9	13	22	40	23.92	1013.18	47.46	0
7	2024	9	13	22	50	23.80	1013.30	47.95	0
8	2024	9	13	23	0	24.10	1013.33	47.85	0
9	2024	9	13	23	10	21.43	1013.40	46.78	0
10	2024	9	13	23	20	21.63	1013.51	46.97	0

Figure 3: Zeilen 1-10 der Tabelle 'weather_data'

Zum Zeitpunkt der Verfassung des Berichts umfasst die Tabelle 1896 Zeilen.

3.4.5 Python Skript

Als Bindeglied zwischen Mosquitto und der Datenbank wird ein Python-Skript "weather_data_ingest.py" verwendet. Das Skript ist simpel gehalten und besteht lediglich aus einem MQTT Subscriber und einer sql-connection, mit write Zugriff auf die Datenbank. Die notwendigen Packages werden in einer "requirements.txt" gelistet. Mittels einer Dockerfile wird ein, speziell auf dieses Skript zugeschnittenes, Image definiert. Die Dateien befinden sich in den Abgabedateien im Ordner "script"

3.5 Web API

Die API ist in JavaScript geschrieben und benutzt Node.js. Sie Stellt bei Aufruf eine Select-Anfrage an die weather_db für die Wetterdaten der letzten 24 Studen und leitet das Ergebnis dieser Anfrage als Json Dokument weiter. Die API läuft auf dem internen Port 3000 mit der subdomain '/api/weather/last24hours'. Die Dateien der API, also "index.js" und die "Dockerfile" befinden sich in dem Ordner "web-api".

3.5.1 Web-UI

Für die Web-UI wurde VueJS als Framework verwendet. Sie besteht aus einer simplen Web-Ansicht auf welcher die Wetterdaten des bisherigen Tages angezeigt werden. Bei Interesse befinden sich die Dateien "App.vue" und die verwendeten Chart-Komponenten im Ordner "web-ui"

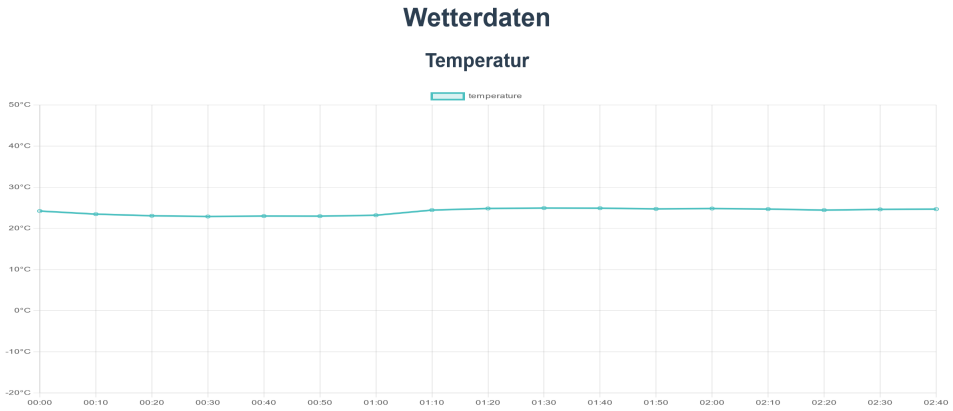

Figure 4: Temperatur-Graph der Web-Ansicht

4 Fazit und Ausblick

4.1 Fazit

Insgesamt lässt sich sagen, dass dieses Projekt eine großartige Lernerfahrung war, welche eine genauere Auseinandersetzung mit einer Reihe besprochener Themen des zugrundeliegenden Moduls ermöglicht hat, aber auch ein paar Aspekte anderer Bereiche der Informatik, wie beispielsweise der Projektplanung oder der Web-Entwicklung beleuchten konnte. Einige Ziele, wie die Verwendung des Deep-Sleep-Modes des ESP32, ein Betrieb der Wetterstation per Akku/Batterie oder das Lösen der Station vom Breadboard und die Gestaltung eines Gehäuses, konnten leider nicht erreicht werden hätten meiner Auffassung nach allerdings auch den Rahmen des Projekts etwas überschritten. Diese Unterfangen werden stattdessen im privaten Rahmen wieder aufgegriffen.

4.2 Ausblick

Wie im Fazit bereits angeschnitten ist das bisherige Ergebnis ein erfolgreicher Prototyp, was bedeutet, dass es noch einige Punkte gibt, welche im weiteren Verlauf aufgegriffen und umgesetzt werden können. Die wichtigsten seien im folgenden genannt:

- Einen Batterie- bzw. Akkubetrieb umsetzten, welcher es der Station erlaubt auch außerhalb der Nähe zu einer Steckdose zu arbeiten.
- Gestaltung und Bau eines wasserdichten Gehäuses um aus dem Prototypen eine tatsächliche Station zu machen, welche auch außerhalb von Innenräumen zum Einsatz kommen kann.
- Umgestaltung der Web-Ansicht in Hinsicht auf die dargestellten Informationen (die Daten der letzten Stunde eignen sich deutlich besser als die Daten des bisherigen Tages), die Positionierung, Größe und Darstellung der Graphen und ein besserer automatischer reload Prozess, welcher nicht jede Minute die gesamte Seite neu lädt sondern nur die relevanten Komponenten
- Wechsel der Regenmessung vom MH-RD Regensensor zu einer anderen Methode, welche in der Lage ist, den tatsächlichen Regenfall zu messen.