Dokumentation

Projekt Stromverbrauch Monitoring für Werkstattmaschinen

Studiengang: Media Systems an der HAW Hamburg
Modul: Projekt B
Betreuer: Prof. Dr. Plaß
31.08.2022

Projektteilnehmer:

Name	**Matrikelnummer**
Martin Geibel	2405117
Firas Lutfy	2439522
Jan Torge Schneider	2307342

Projektleiter:
Jan Torge Schneider (jantorge.schneider@haw-hamburg.de)

Repository:
https://github.com/paulsteingaesser/Werkstatt-Monitoring

Inhaltsverzeichnis

Einleitung

Ziel des Projektes war die Konzeption und Umsetzung eines Systems, das ein Monitoring des Stromverbrauchs und der Laufzeit von großen Werkstattmaschinen ermöglicht. Das System wurde in Kooperation mit der Kreativtischlerei bauer+planer entwickelt und soll dort zukünftig zum Einsatz kommen.

bauer+planer wird im Folgenden als b+p abgekürzt oder als Stakeholder bezeichnet.

Das Interesse der Tischlerei an einem solchen System sind die hohen Stromkosten der großen Maschinen und ein Konzept, bei dem die Mitarbeiter und auch externe sich in die Werkstatt einmieten können. Bisher kamen dabei Pauschalverträge zum Einsatz. Dabei werden aber Mieter benachteiligt, die kaum an den Maschinen tätig sind.

Ein Monitoring der Nutzungszeiten und des verbrauchten Stroms soll also flexiblere Vertragsmodelle erlauben.

Das System ist aus verschiedenen Komponenten aufgebaut.

Direkt an den Maschinen sind kleine Mikrocontroller angebracht, die mit einigen Sensoren und Aktoren wie einem Pinpad, einer LED-Anzeige und einem Stromsensor kommunizieren. Dieses in sich geschlossene System registriert Stromänderungen von den Maschinen oder eine Anmeldung und gibt über die LEDs ein Feedback zum aktuellen Status. Die Messwerte werden dann über eine Netzwerkschnittstelle der Mikrocontroller an einen Server geschickt.

Der Server ist eine weitere Komponente und empfängt Anfragen und Daten aller Mikrocontroller. Die Daten wertet er aus und legt sie in einer Datenbank ab. Zudem ist ein Webserver eingerichtet, der die Website für das Monitoring bereitstellt. Der Webserver zieht sich die darzustellenden Inhalte aus einer Datenbank. Zudem ist über die Website eine Nutzer- und Maschinenverwaltung möglich.

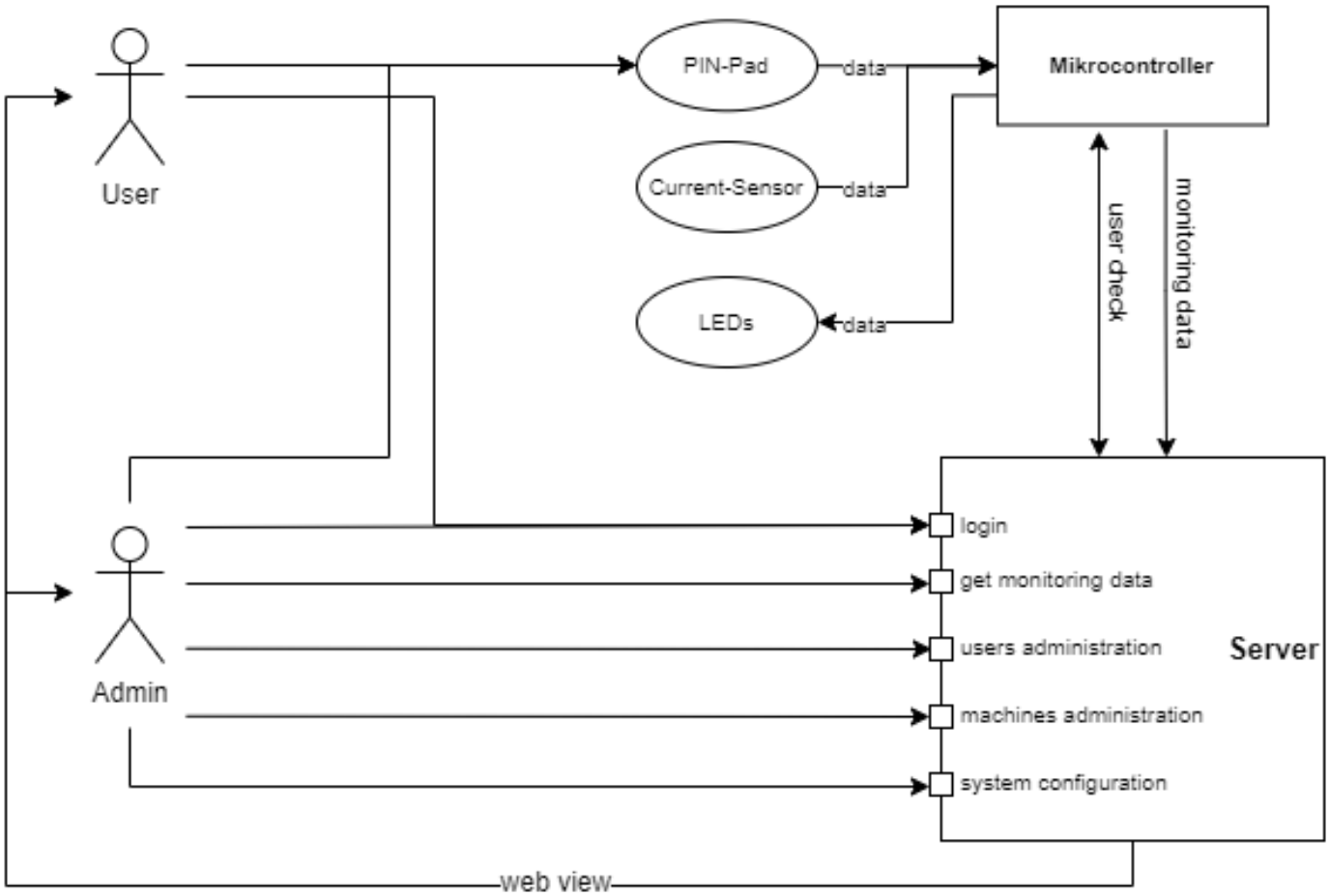
Anforderungsdefinition

Zum Start der Projektphase haben wir mehrere Meetings mit unseren Kontaktpersonen von b+p abgehalten, um eine möglichst detaillierte Anforderungsdefinition für das System aufzunehmen.

Dabei wurden uns auch die ersten eigenen Versuche und Prototypen von b+p vorgestellt. Da sich diese aber noch in einem sehr geringen Entwicklungsstadium befanden und auch nicht wirklich praxistauglich und nachhaltig umgesetzt wurden, haben wir diese verworfen. Sie haben uns aber einen guten Einblick verschafft, was der Stakeholder eigentlich vor hat und mit welchen Problemen er bisher konfrontiert war. Dies ist dann mit in unsere Bestandsaufnahme mit eingeflossen.

Aus der Anforderungsdefinition haben wir das nachfolgende Konzeptdiagramm entwickelt.

Konzeptdiagramm



Es soll zwei Nutzertypen geben, eine normale Nutzerklasse (User) und einen Nutzer mit erhöhten Zugangsrechten (Admin). Der normale Nutzer kann sich sowohl an den PIN-Pads, als auch über einen Browser am Webserver einloggen. Der Admin hat zusätzlich einen Zugriff auf die Nutzer- und Maschinenverwaltung am Server und kann dort auch noch verschiedene Daten auslesen sowie Systemeinstellungen vornehmen.

Zudem gab es den Wunsch, dass wir für die Entwicklung ein möglichst einfaches und leicht verständliches Framework wählen. Der Stakeholder verspricht sich davon, in Zukunft gegebenenfalls selbständig in der Lage zu sein, kleine Änderungen am System vorzunehmen, ohne uns oder andere Softwareentwickler beauftragen zu müssen.

Unsere Wahl fiel dann auf das von IBM entwickelte grafische Entwicklungswerkzeug Node-RED. Dieses lässt sich einfach über einen Webbrowser öffnen und ermöglicht im Baukastenprinzip auch komplexe Programmierlogiken einfach umzusetzen. Dieses wird auf dem Server verwendet.

https://nodered.org

Server

Für den Server haben wir von bauer+planer bereits zu Beginn des Projektes einen Raspberry Pi 4B gestellt bekommen.

Im ersten Schritt haben wir uns damit beschäftigt, den Raspberry Pi zu konfigurieren. Dabei ging es primär um Systemeinstellungen, wie den automatischen Neustart nach einem Stromausfall oder das Festlegen einer statischen IP Adresse, damit der Server im lokalen Netz immer unter der gleichen Adresse erreichbar ist. Zudem haben wir einen SSH Tunnel für die Fernwartung eingerichtet und einen eigenen Git Account erstellt, um auf unser Git-Repository auf GitHub zuzugreifen und ggf. Änderungen auch pushen zu können.

Bedienungsanleitung für den Server und die Webanwendung:

Installationsanweisung für Node Red

Wie auch auf unseren eigenen Windows Rechnern, die wir für die Entwicklung genutzt haben, ist auf dem Rasperry Pi eine Node.js Version und der Node Package Manager (npm) zu installieren, damit anschließend Node-Red erfolgreich installiert werden kann.

Eine Anleitung dazu ist hier zu finden: https://nodered.org/docs/faq/node-versions

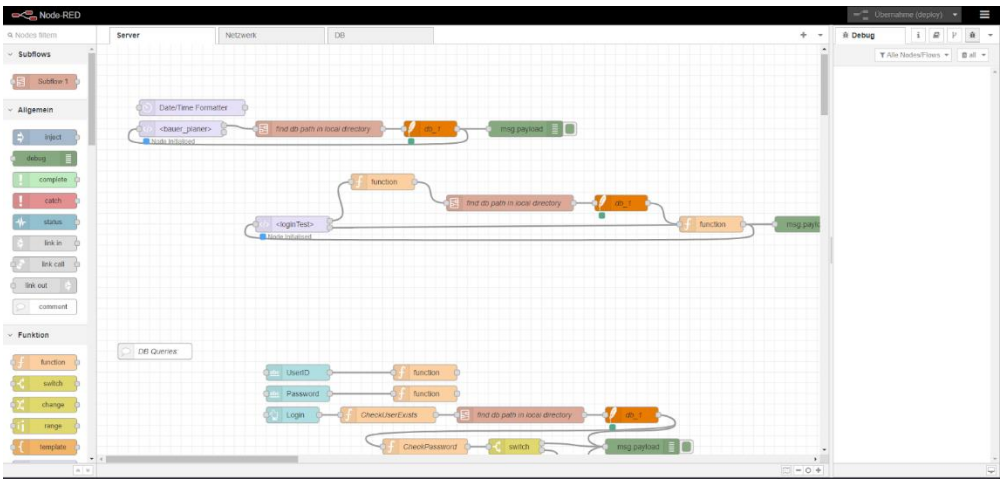
Mit Hilfe des Package Managers lässt sich dann sehr einfach das Node-RED Modul installieren:

npm install -g --unsafe-perm node-red (Windows)
sudo npm install -g --unsafe-perm node-red

Unter Windows sollte nun im Benutzerverzeichnis ein Ordner mit dem Namen ".node-red" erstellt worden sein. Dieser dient auch als Projektverzeichnis.

Für das Starten von Node-RED muss nur in einer Konsole oder einem Terminal der Befehl "node-red" eingegeben werden. Anschließend wird auch die Adresse angezeigt, unter welcher der Server erreichbar ist. Für unser Projekt ist das folgende Adresse: http://127.0.0.1:1880/

Gibt man die Adresse im Browser ein oder folgt dem Link, nach dem Node-RED gestartet wurde, erreicht man den grafischen Editor von Node-RED.



Grafischer Editor von Node-RED im Browser

Weitere Informationen zur Installation und zum Starten sind der Dokumentation von Node-RED zu entnehmen: https://nodered.org/docs/getting-started/local

Repository klonen

Auschecken des Repositorys über Node-RED:
Um unser Projekt von GitHub auszuchecken, muss zuerst die Git-Integration von Node-RED aktiviert werden. Dazu öffnet man die Datei settings.js im lokalen .node-red Ordner, sucht dort das Property "projects" und setzt "enabled" auf "true". Anschließend muss Node-RED einmal neu gestartet werden und es erscheint beim Aufrufen des Servers im Browser eine Maske. Über diese kann dann ein Git-Repository ausgecheckt werden.
(SSH Key empfohlen, aber nicht zwangsläufig nötig: https://git-scm.com/book/de/v2/Git-auf-dem-Server-Erstellung-eines-SSH-Public-Keys)

Im .node-red Ordner ist nun ein Verzeichnis "projects/Werkstatt-Monitoring/" angelegt, unter dem unser Repository zu finden ist.

Kopieren des Repositorys
Alternativ kann der Inhalt unseres Repositories auch kopiert und im node-red Ordner eingefügt werden. Wichtig ist dabei, dass nur gleiche Dateien ersetzt oder überschrieben werden! Ordner wie z.B. "node_modules" sollten erhalten bleiben. Anschließend sollte der Server auch über den node-red Befehl in einer Konsole gestartet werden können und einen Zugriff auf unser Projekt gewähren.

Weitere Infos sind über folgendem Link unter "Clone a project repository" zu finden :
https://nodered.org/docs/user-guide/projects/

Verwendete Plugins

Zusätzlich verwenden wir einige Plug-ins und Node-RED Erweiterungen. Diese müssen lokal hinzugefügt werden, ansonsten weist Node-RED über eine Warnung auch darauf hin!

Folgendes gilt es zu installieren:

-node-red-contrib-uibuilder
https://flows.nodered.org/node/node-red-contrib-uibuilder

-SQLite Plugins:
https://flows.nodered.org/node/sqlite-plugin-red
https://flows.nodered.org/node/node-red-contrib-queued-sqlite-fix
https://flows.nodered.org/node/node-red-node-sqlite

-node-red-contrib-moment
https://flows.nodered.org/node/node-red-contrib-moment

-node-red-contrib-projectdir
https://flows.nodered.org/node/node-red-contrib-projectdir

-node-red-dashboard

https://flows.nodered.org/node/node-red-dashboard

Die Installation kann über ein Terminal mit Hilfe des npm install Befehls erfolgen oder über Node-RED in den Einstellungen über “Palette” und dort im “Installation” Tab.

Aufrufen des Frontend

Für unser Frontend haben wir in Node-RED das Web User-Interface “uibuilder” integriert. Dieses baut und managt unser Frontend und startet dafür einen eigenen Webserver. Als Client erreicht man diesen unter folgender Adresse: http://127.0.0.1:1880/bauer_planer

Folgende Credentials stehen zum Testen bereit:

	User ID	Passwort
Admin	100	12345
Normaler Nutzer	101	

Wir haben mit Hilfe von GIFs ein paar der Kernfunktionen als Durchlauf aufgenommen.

Diese sind leider nicht in PDF-Dateien exportierbar. Daher haben wir die GIFs in die README.md Datei unseres Repositorys integriert. Diese sind dann hier einsehbar:

https://github.com/paulsteingaesser/Werkstatt-Monitoring

Entwicklerhandbuch Node-RED Server:

Beschreibung der Anwendung

Der Node-Red Server kommuniziert über eine HTTP Schnittstelle mit den Mikrocontrollern an den Maschinen. Diese Fragen beim Server an, ob die an den PIN-Pads eingegebenen IDs in der Datenbank vorhanden sind und ob die zugehörige Berechtigungsstufe des Users die Benutzung der Maschine erlaubt.
Zudem erhält der Server die erfassten Stromdaten und zugehörigen Zeitintervalle von den Mikrocontrollern über die Schnittstelle, formatiert und persistiert diese.

Über ein Web User-Interface können die Daten dann abgerufen und exportiert werden. Zudem können User mit einer Admin Berechtigung Nutzer und Maschinen verwalten. Admins sind auch in der Lage, Grundeinstellungen wie z.B. den vom Stromanbieter vorgegebenen Strompreis pro kWh zu editieren, der im Monitoring für die Berechnungen verwendet wird.

Struktur der Anwendung

uibuilder

In Node-RED haben wir die Nodes gezielt nach Funktionalität in drei Flows (Tabs) aufgeteilt. Der Server Flow beinhaltet den uibuilder Node “<bauer_planer>” und damit die Schnittstelle zwischen Frontend und Node-RED. Im Frontend werden die Datenbankbefehle zusammengesetzt und über das “Topic” Property an das Datenbank-Node gesendet. Dieses stellt bereits beim Starten des Servers eine Verbindung zur SQLite Datenbank her und managed beim Erhalt einer Nachricht aus dem Frontend die Datenbankabfrage. Das Datenbank-Node schickt anschließend die Ausgabe zurück an den uibuilder Node.

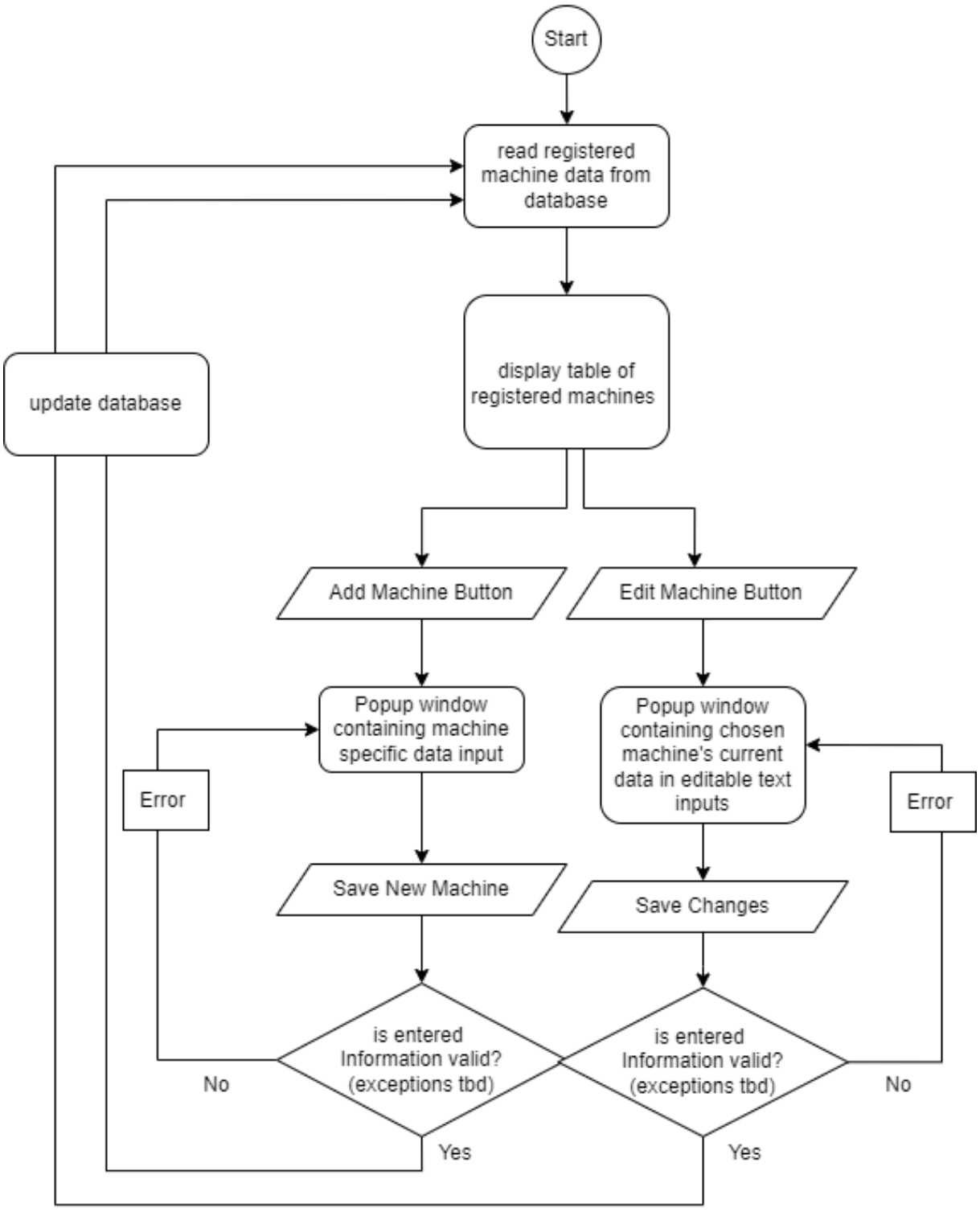
Flow von uibuilder und SQLite Datenbank

Im uibuilder Node kann neben ein paar Netzwerkkonfigurationen festgelegt werden, ob man reines HTML und CSS oder ein Framework für das Frontend verwenden möchte. Zuerst kam der Gedanke, eine moderne Single-Page-Application zu bauen und dafür das vom uibuilder empfohlene Vue.js zu verwenden. Aufgrund von Zeitknappheit und dem Umstand, dass nur ein Teammitglied bisher Erfahrung mit der Arbeit an Single-Page-Applications und Frameworks wie Vue, react oder Angular hat, mussten wir uns dagegen entscheiden.

Dafür konnten wir ein HTML Template finden, das sich gut für unsere Zwecke verwenden lässt und uns einiges an Arbeit abgenommen hat. Für die Frontend-Logik haben wir das Template dann noch um JavaScript-Code ergänzt.

Für alle aus der Anforderungsdefinition entnommenen Kernfunktionen haben wir vor der Implementierung ein Flow-Chart angelegt. Diese konnten wir dann dem Stakeholder vorlegen und prüfen lassen, ob wir die Anforderungen korrekt aufgenommen haben. Anschließend dienten die Flow-Charts dann als Vorlage für die Frontend-Logik.

Nachfolgend ist das Flow-Chart für das Anlegen oder Bearbeiten einer Maschine zu sehen. Der Prozess für das Anlegen und Bearbeiten eines neuen Nutzers verläuft sehr ähnlich.

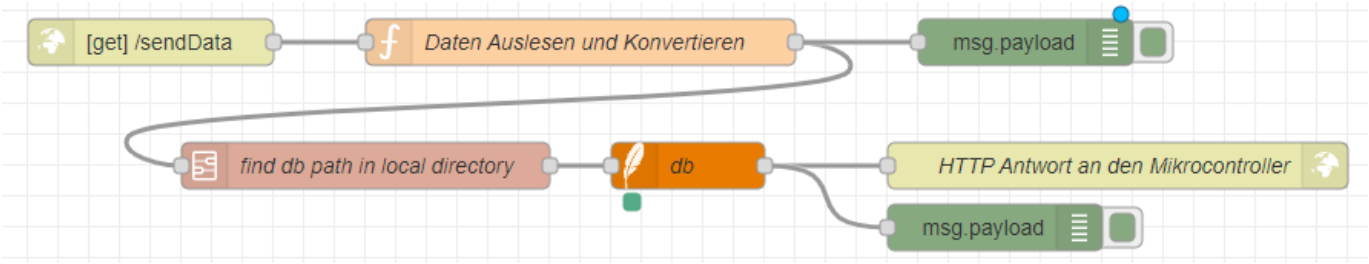


Flow-Chart zum Anlegen und Bearbeiten einer Maschine

Der Frontend-Code ist in folgendem Verzeichnis einseh- und editierbar "../Werkstatt-Monitoring/uibuilder/bauer_planer/src".

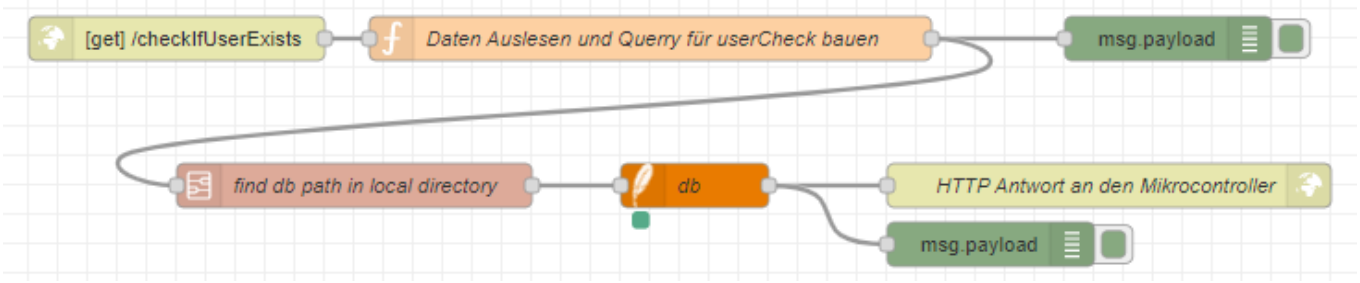
Netzwerk

Der Netzwerk-Flow beinhaltet zwei HTTP-IN Nodes, die auf GET Anfragen lauschen. Der “/sendData” Node empfängt über das HTTP Protokoll Daten von den Mikrocontrollern, die in URL-Parametern mitgeschickt werden. Anschließend werden die Daten formatiert und in die “data” Tabelle der Datenbank geschrieben.



HTTP-IN Node mit Flow zum Persistieren der Daten

Wollen die Mikrocontroller überprüfen, ob eine PIN-Pad Eingabe valide ist, schicken sie eine Anfrage mit den entsprechenden URL-Parametern an den “/checkIfUserExists” Node. Aus den Parametern wird eine Datenbank Query gebaut und an die SQLite Datenbank geschickt. Das Ergebnis wird über die gleiche Verbindung an den Mikrocontroller zurückgeschickt.

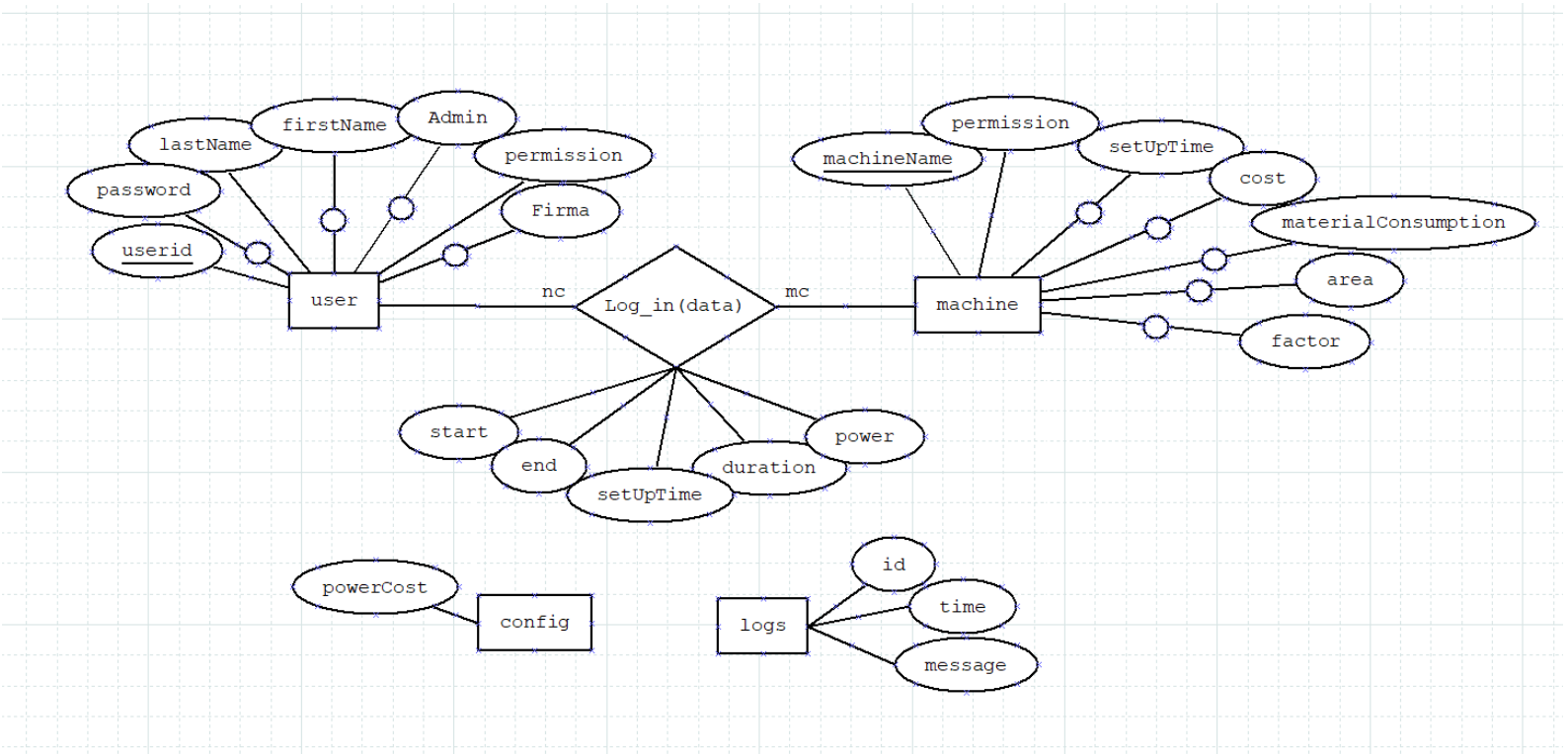


HTTP-IN Node mit Flow zur Datenabfrage

Datenbank

Zum Speichern der Daten verwenden wir eine SQLite Datenbank. Die Anbindung wird von Node-RED gut unterstützt und ist auch gut dokumentiert. Zudem haben wir in dem Modul “Angewandte Programmierung” bereits Erfahrung im Umgang mit SQLite sammeln können. Durch das zusätzlich gewonnen Wissen aus dem Wahlpflichtkurs “Relationale Datenbanken” war das Erstellen eines Datenbank Schemas mit den zugehörigen Diagrammen schnell erledigt.

Diese sind nachfolgend dargestellt.



Erarbeitetes Entity-Relationship-Modell

DB:	user	userid	password	lastname	firstName	admin	permission	company
	User	UserID	Paswort	Nachname	Vorname	Admin	Berechtigungstufe	Firma
		num	zk	zk	zk	bool	num	zk
		pk	nullable					nullable

DB:	machine	machineName	permission	setUpTime	cost	materialConsumption	area	factor
	Maschine	MaschinenName	Berechtigungsstufe	Rüstzeiten	Kosten	Materialverbrauch	Bereich	Faktor
		zk	num	num	num	num	zk	num
		pk				nullable	nullable	

DB:	data	userid	machineName	start	end	setUpTime	duration	power
	Data	UserID	MaschinenName	beginn	Ende	Rüstzeiten	Dauer	Strom
		num	zk	num	num	num	num	num
		pk, fk	pk, fk	pk				

DB:	config	powerCost
	Config	powerCost
		real

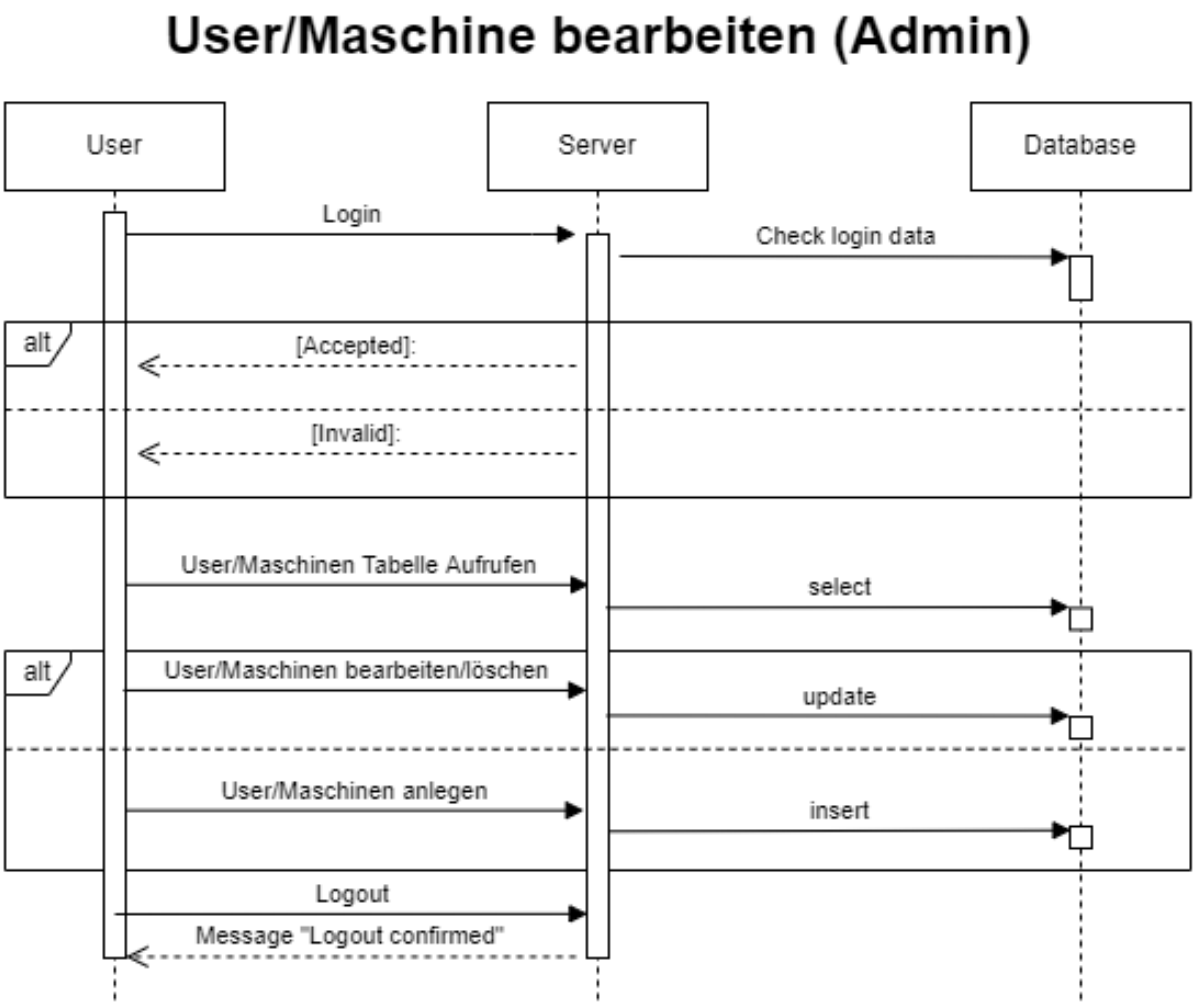
DB:	logs	id	time	message
	Logs	ID	Time	Message
		num	num	zk

Aus dem ERM abgeleitetes Relationen Modell

Mit Hilfe des RMs haben wir anschließend die Tabellen angelegt. Die dafür verwendeten Nodes sind in dem Datenbank-Flow “DB” zu finden. Dort sind auch die INSERT und DROP Befehle sowie einige Test-Querys angelegt, die das Bearbeiten der Tabellen und Einfügen von Testdaten sehr einfach gestalten.

Um einen besseren Überblick über die Datenbankaufrufe aus dem Frontend zu erhalten, haben wir versucht, die Kommunikation mit der Datenbank in einem Sequenzdiagramm zu skizzieren.

Nachfolgend ist ein Ausschnitt aus dem Sequenzdiagramm für das Anlegen, Bearbeiten und Löschen von einem Nutzer oder einer Maschine zu sehen.



Ausschnitt aus dem Sequenzdiagramm

Um den Inhalt der Datenbank Tabellen anzuzeigen kann ein Hilfsprogramm wie z.B. der DB-Browser verwendet werden: https://sqlitebrowser.org

Maschinenmodul:

Installationsanweisung Arduino IDE

Der zweite wichtige Bestandteil unseres Systems sind kleine Mikrocontroller, die an den Maschinen den Stromverbrauch messen und eine Authentifizierung an der Maschine ermöglichen. Dafür verwenden wir Entwicklungsboards auf Basis von ESP32 Chips. Diese können über eine Arduino Schnittstelle programmiert und geflasht werden. Dafür ist die Installation der Arduino IDE nützlich. (Download unter: https://www.arduino.cc/en/software)

Es kann aber auch ein normaler Code Editor verwendet werden, um den Inhalt der .ino Dateien anzuzeigen. Das Hauptskript "mikrocontroller_script.ino" befindet sich in dem Verzeichnis "../Werkstatt-Monitoring/mikrocontroller/mikrocontroller_script/". Ist die Arduino IDE installiert, kann das Skript einfach über einen Doppelklick aufgerufen werden.

Für das erfolgreiche Kompilieren des Skriptes müssen noch die benötigten Bibliotheken installiert werden:

- WiFi.h (https://www.arduino.cc/reference/en/libraries/wifi/)
- HTTPClient.h (https://www.arduino.cc/reference/en/libraries/httpclient/)
- FastLED.h (https://www.arduino.cc/reference/en/libraries/fastled/)
- Keypad.h (https://playground.arduino.cc/Code/Keypad/)

Wie das Installieren der Bibliotheken funktioniert ist in der Anleitung unter folgendem Link erklärt: https://docs.arduino.cc/software/ide-v1/tutorials/installing-libraries

Entwicklerhandbuch des Maschinenmodul

Beschreibung des Mikrocontrollers

Ganz zu Beginn der Projektphase war geplant, WLAN oder Netzwerkfähige Mini-Computer wie die Raspberry Pi Zero Ws für die Maschinen-Module zu verwenden. Darauf sollten dann auch Performance getrimmte Versionen von Node-RED zum Einsatz kommen. Allerdings wurde der Plan durch den zu der Zeit vorherrschenden Chip shortage durchkreuzt, da die Geräte auf nicht absehbare Zeit ausverkauft waren.

Als eine etwas aufwändigere, aber deutlich kostengünstigere Alternative boten sich mit ESP32 Chips bestückte Mikrocontroller. Diese bieten nicht nur zuverlässige Netzwerkschnittstellen, sondern auch genügend Performance für unsere Anforderungen. Entschieden haben wir uns am Ende für das "ESP-32 Dev Kit C V4" Entwicklungsboard von der Firma AZDelivery.

Link zum Board: https://www.az-delivery.de/products/esp-32-dev-kit-c-v4?

Gründe die für die Wahl des Boards:

- Im Gegensatz zu den meisten anderen ESP32 Boards besitzt dieses einen 5V Output, was einen direkten Anschluss von programmierbaren WS2812B Leds ermöglicht.
- Gute Qualität aus deutscher Produktion.
- Eine langfristige Versorgungssicherheit für Ersatz Boards und Zubehör.
- Eine spezielle Bauweise, die die WLAN-Konnektivität im Vergleich zu anderen Boards erhöht.

Zu der Arbeit mit ESP32 Entwicklungsboards gibt es sehr gute Dokumentationen. Dabei ist folgende Seite besonders zu empfehlen:

https://randomnerdtutorials.com/getting-started-with-esp32/

Dort sind neben grundlegenden Erklärungen auch für sehr viele Netzwerkprotokolle Beispielanwendungen zu finden. Unter anderem wird in einigen Beispielen sogar eine Verbindung mit einem Node-RED Flow verwendet!

Wie man über die Arduino IDE oder einen Code Editor wie Visual Studio Code ein Skript auf einen ESP32 flasht, wird dort ebenfalls ausführlich erläutert.

Beschreibung der weiteren Komponenten (Sensoren und Aktoren)

PIN-Pads

Damit die Mitarbeiter von b+p sich an den Maschinen anmelden können, sollen dort so genannte Sturm PIN-Pads zum Einsatz kommen, die durch ihren extra Schutz auch sicher vor dem Dreck und Schmutz der Werkstatt sind. Der Stakeholder hatte sich bereits im Vorfeld des Projektes auf folgendes Modell festgelegt:

https://www.storm-interface.com/keypads/720-series/storm-720-blk-series-12-button.html

Dort ist auch in Form eines PDFs eine Dokumentation zur Pin-Belegung zu finden.

Stromwandlerklemmen (Stromsensoren)

Die eigentliche Strommessung soll dann über Stromwandlerklemmen erfolgen. Dies sind Eisenkernspulen, die über einen Leiter oder eine Phase geklemmt werden können. Der Stromfluss in der Phase induziert dann ein Magnetfeld in den Eisenkern, welches durch einen vorgeschalteten Stromteiler in eine Spannung umgewandelt werden kann. Durch die Wahl der Bauteile des Stromteilers und bekannten Parametern der Stromwandlerklemme kann bereits im Vorfeld das Maximum und Minimum der entstehenden Spannung bestimmt werden. So kann der Stromfluss in der Phase sehr einfach auf den bekannten Spannungsbereich am Ausgang des Stromteilers skaliert werden.

Hinweis: Die elektronischen Prozesse wurden nur grundlegend veranschaulicht, da eine detailliertere Beschreibung den Rahmen dieser Dokumentation sprengen würde.

Eine sehr ausführliche Erklärung ist unter folgendem Link zu finden:

https://learn.openenergymonitor.org/electricity-monitoring/ac-power-theory/introduction

Als Stromwandlerklemmen setzten wir die Baureihe SCT-013 der Firma YHDC ein. Diese sind marktführend im Bereich Stromwandlerklemmen und finden sich auch in diversen Anwendungen zur Strommessung wieder, die wir bei unserer Recherche ausfindig machen konnten. Konkret verwenden wir das Modell SCT-013-000 da dieses mit einem Messbereich von 50 mA bis maximal 100 Ampere die größte Bandbreite bietet. Da wir mit großen Werkstatt Maschinen arbeiten, sind auch Werte in höheren Ampere Bereichen zu erwarten.

Link zu der Herstellerseite: https://en.yhdc.com/product/SCT013-401.html

Auch die bereits oben verlinkte Seite von openenergymonitor.org verwendet die Stromwandlerklemmen von YHDC und hat eine umfassende Dokumentation erstellt:

https://learn.openenergymonitor.org/electricity-monitoring/ct-sensors/introduction

Bei unserer Recherche sind wir auch auf eine deutsche Beschreibung zu einem Projekt mit ähnlichen Zielen wie unseren gestoßen.

https://technik-fan.de/index.php/Open_Energy_Monitor_mit_dem_ESP32

Dort wird auch auf den Grundlagen von openenergymonitor.org aufgebaut und diese noch erweitert. Das hat dann als Vorlage für die Schaltung unseres Prototypen gedient.

LEDs

Um dem Anwender direkt an der Maschine ein Feedback geben zu können, wünscht sich der Stakeholder eine Statusanzeige über einen LED-Stripe, der um das PIN-Pad gewickelt ist. Die LEDs sollen z.B. signalisieren, ob eine Anmeldung vorliegt, ein Anmeldeversuch erfolgreich war, ob die Berechtigungsstufe des Anwenders für die Maschine ausreichend ist oder ob ein Fehler vorliegt und beispielsweise der Server nicht erreicht werden kann.

Wir verwenden LEDs vom Typ WS2812B, da diese weit verbreitet und damit gut kompatibel zu vielen LED-Bibliotheken sind. Bei längeren LED-Stripes ist es ratsam, diese über eine eigene Stromquelle zu versorgen. Da bei uns maximal 10 LEDs zum Einsatz kommen, genügt es vollkommen, die 5V Spannung des Mikrocontrollers zu verwenden. Zusätzlich schalten wir einen 1000µF Kondensator vor die LEDs, um Spannungsschwankungen vom Mikrocontroller auszugleichen. Ein kleiner Vorwiderstand am Datenpin der LEDs schützt zudem die ersten LEDs vor möglichen Spannungsspitzen.

Da in vergangenen Projekten bereits gute Erfahrung mit der FastLED Bibliothek gesammelt wurde, kommt diese auch in diesem Projekt zum Einsatz.

https://github.com/FastLED/FastLED und https://fastled.io

Mit der Bibliothek lassen sich sehr einfach komplexe Farbkombinationen auf den LED-Stripe spielen und auch ein Blinken der LEDs für Warnsignale realisieren.

Struktur der Anwendung

Wird ein neues Maschinenmodul erstellt, muss neben dem Schaltungsbau auch das Hauptskript auf den Mikrocontroller gespielt werden.

Im oberen Teil des Skriptes sind diverse Einstellungsparameter, die angepasst werden können. Neben der Adresse des Servers, dem SSID Namen und dem zugehörigen Passwort muss dort z.B. auch der

Maschinenname eingetragen werden. Dieser muss identisch mit dem Namen sein, der beim Anlegen einer Maschine im Frontend anschließend in die Datenbank geschrieben wird! Denn der Maschinenname wird auch immer als URL-Parameter mit an den Server gesendet, damit dieser einordnen kann, von welcher Maschine er den Datensatz erhalten hat.

Zudem müssen im Script Header die Pinbelegungen des Mikrocontrollers angegeben werden. Sind die Maschinenmodule immer gleich verkabelt, ändert sich an diesen Konfigurationen nichts. Über #define Komponenten lassen sich dann weitere Grundeinstellungen für das Maschinenmodul vornehmen. Über das Ein- oder Auskommentieren der entsprechenden Zeilen kann dann z.B. das Debugging eingeschaltet werden, wenn zusätzlich ein Rechner an den Mikrocontroller angeschlossen ist, um Debugging Nachrichten zu empfangen.

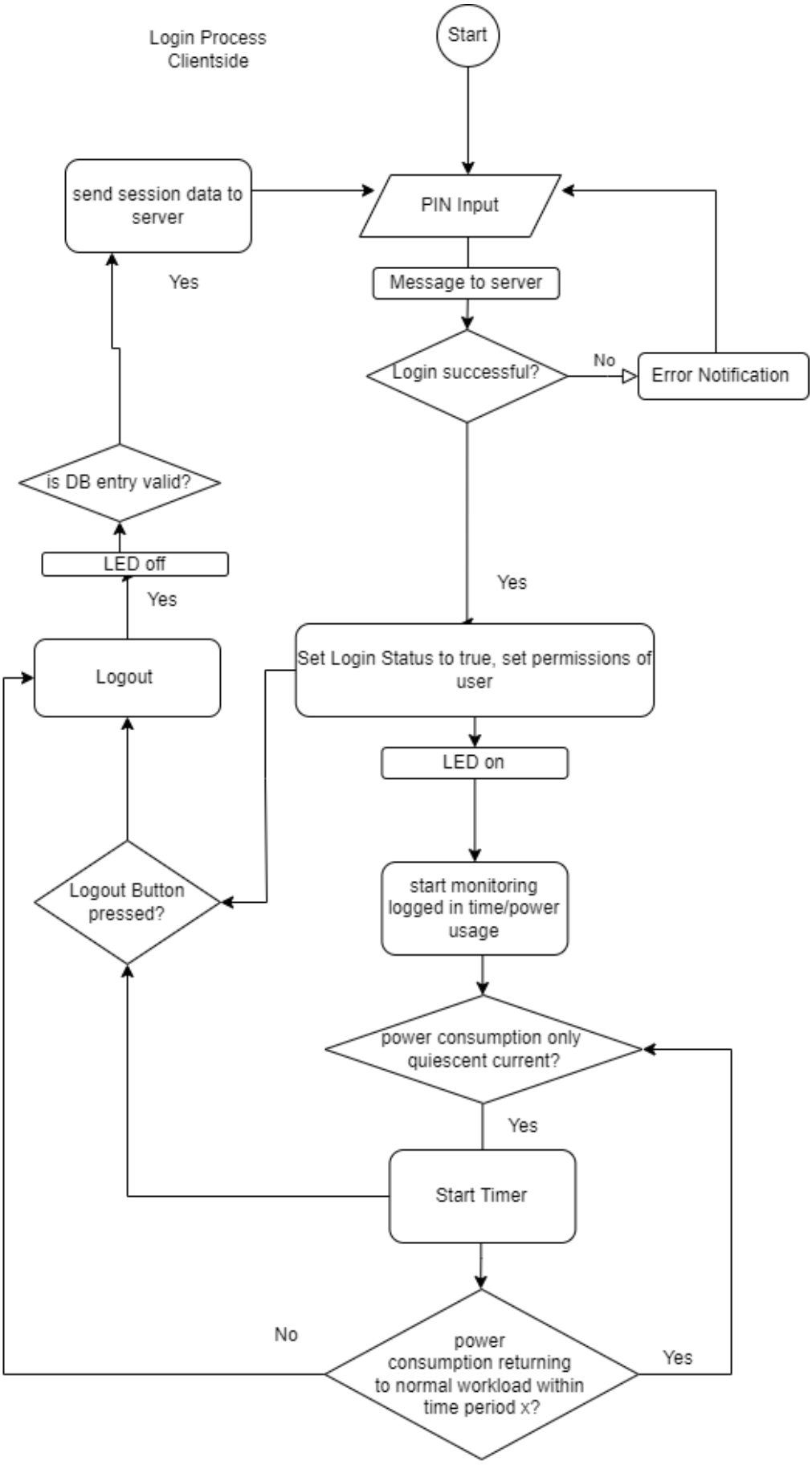
Da einige der Werkstatt Maschinen an Drei-Phasen-Strom angeschlossen sind, kommen an den entsprechenden Modulen auch drei Stromwandlerklemmen als Sensoren zum Einsatz. Mit den entsprechenden #define Komponenten lässt sich in dem Skript schnell einstellen, wie viele Stromwandlerklemmen an das Modul angeschlossen sind.

Weitere Einstellungsmöglichkeiten gibt es bei den Parametern für die Strommessung, mit denen später z.B. der Maschinenspezifische Offset-Strom aus den Messwerten entfernt werden kann. Auch die Helligkeit der LEDs und die PIN-Pad Belegung kann im oberen Skript Teil konfiguriert werden.

Es ist ratsam, die fertig eingestellten Skripte nach dem Flashen der Mikrocontroller abzuspeichern. So lässt sich später genau nachvollziehen, wie die Module konfiguriert sind und man kann Änderungen an einzelnen Maschinen vornehmen!

Nach den Einstellungen folgt eine Setup-Funktion, die einmalig zum Start des Skriptes ausgeführt wird. Dort werden dann die vorher eingestellten Parameter übernommen, die Pinbelegung ausgeführt und versucht, eine WLAN-Verbindung aufzubauen. Anschließend folgt die Arduino typische Loop-Funktion, die wie ein Main-Thread funktioniert und dauerhaft die Sensoren abfragt und die Aktoren und den Server entsprechend informiert.

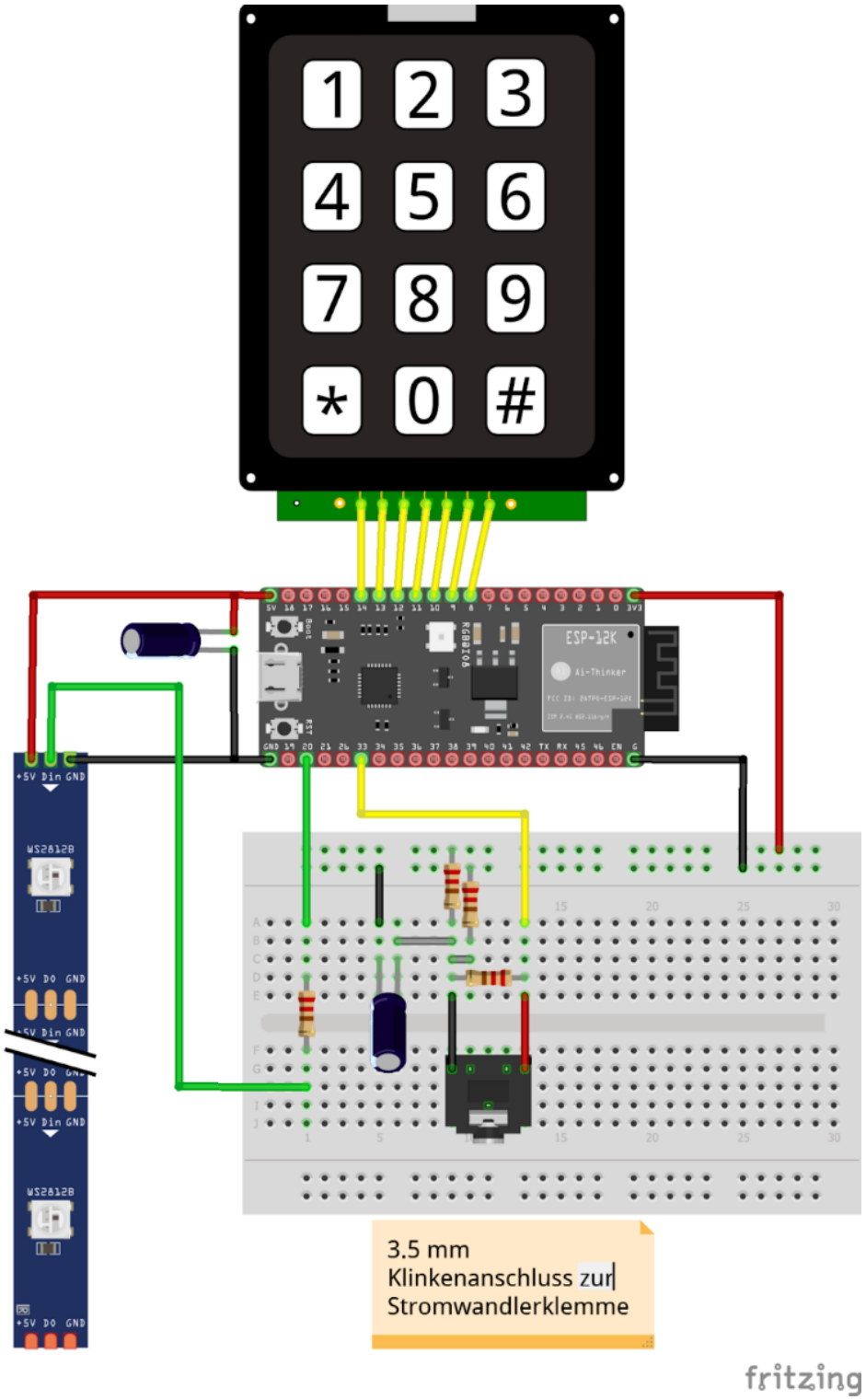
Wie die einzelnen Komponenten zusammenarbeiten, haben wir in einem weiteren Flow-Chart skizziert.



Flow-Chart zum Grundablauf des Maschinenmoduls

Ein in Fritzing selbst entworfener Schaltungsplan zeigt die Verkabelung der verschiedenen Komponenten.

Hinweis: Die Widerstände sind in dem Plan nicht korrekt dimensioniert und auch die Pinbelegung am Mikrocontroller stimmt nicht 1:1 mit der im Skript verwendeten Belegung überein! Der Plan soll lediglich ein Verständnis für die Verkabelung schaffen.



Schaltungsplan des Maschinenmoduls

Die Schaltung mit den oben beschriebenen Komponenten lässt sich dann auch im ersten Prototypen wiederfinden.

Auf dem folgenden Foto erkennt man das Sturm PIN-Pad, um das ein LED-Stripe gewickelt ist. In der Mitte ist das ESP32 Entwicklungsboard von AZDelivery zu erkennen. Links im Bild ist die Stromwandlerklemme, die über einen 3.5mm Klinkenanschluss mit der Stromteiler Schaltung auf dem Steckbrett verbunden ist.

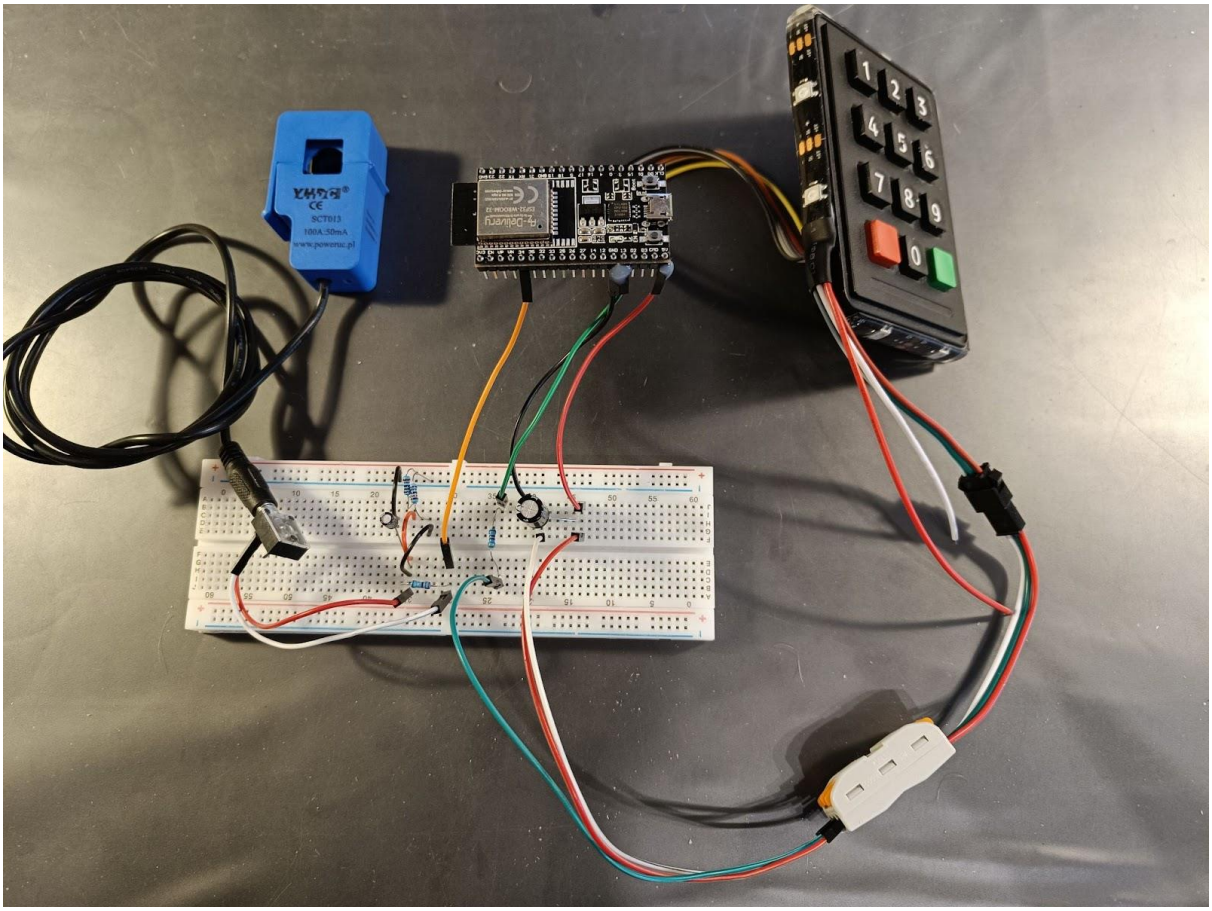


Foto des Prototyps

Im Verzeichnis "../Werkstatt-Monitoring/documentation" ist eine Datei "Strommessung Wasserkocher.txt" hinterlegt. Diese zeigt den Output eines Mikrocontrollers beim Messen des Stroms eines Wasserkochers vor, während und nach dem Erhitzen von Wasser. Die aus gemessenem Strom und bekannter Netzspannung berechenbare Leistung von ~1650 Watt kommt der Angabe von 1600 Watt auf dem Wasserkocher sehr nahe!

Alle Diagramme, Flow-Charts, Messwerte und auch unsere Frontend Mockups, die nicht Teil dieser Dokumentation sind, haben wir in unserem Repository unter dem Ordner "documentation" abgelegt.