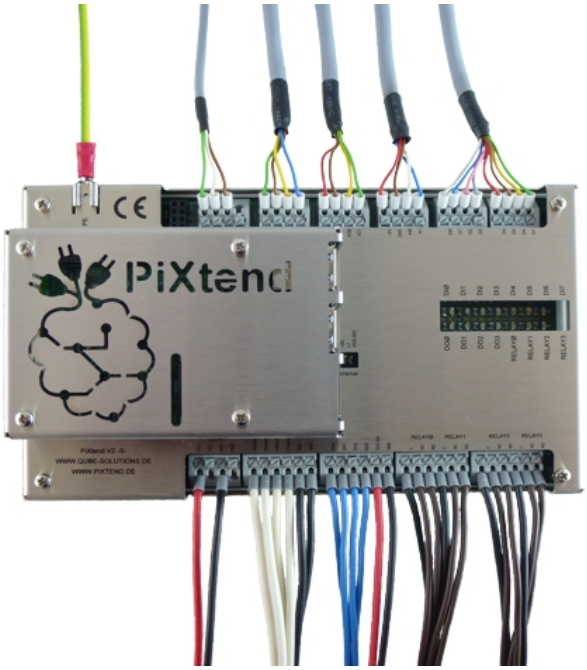
Application-Note
CODESYS - Node-RED Web-Visu

Installation, Einrichtung, Programmierung



APP-PX2-570

Stand 28.08.2018, V1.00

Qube Solutions GmbH
Arbachtalstr. 6, 72800 Eningen, Germany
http://www.qube-solutions.de/
https://www.pixtend.de

Versionshistorie

Version	Beschreibung	Bearbeiter
1.00	Dokument erstellt	RT

Inhaltsverzeichnis

1. Einleitung

Die ***CODESYS*** Plattform bietet dem Anwender ein umfangreiches SPS-Programmierwerkzeug für viele CODESYS kompatible Steuerungen, darunter auch der *Raspberry Pi* mit der *PiXtend*-Erweiterung. Obwohl CODESYS eine Web-Visualisierung anbietet, kann der Wunsch entstehen eine zusätzliche Web-Visualisierung zu haben die der IoT-Welt (Internet of Things) entstammt und weitere bzw. andere Visualisierungsmöglichkeiten bietet.

Was liegt näher als ***Node-RED***, den IoT-Editor, für einen solchen zweck zu verwenden.

Eine kurze Einführung zu *Node-RED*:

Node-RED ist ein Programmierwerkzeug mit dem es möglich ist verschiedene Geräte, APIs[1] und Onlinedienste auf neue und interessante Art und Weise „*virtuell*“ zu verbinden.
Es bietet einen Webbrowser basierten Editor an, der es einem einfach macht verschiedene Datenflüsse miteinander zu „*verkabeln*“, unter Verwendung verschiedenster „*Knoten (Nodes)*“ die das System bereitstellt.[2]

Fluss- bzw. Datenfluss basierte Programmierung wurde in den 1970er Jahren von J. Paul Morrison[3] erfunden und bietet die Möglichkeit das Verhalten einer Applikation unter Verwendung eines Netzwerks bestehend aus „Black-Boxen“, in *Node-RED* heißen Sie *Knoten (Nodes)*, zu beschreiben.
Jeder Knoten hat dabei eine festgelegte und eindeutige Aufgabe. Werden an einen Knoten Daten übermittelt, so kann der Knoten diese Daten verarbeiten und gibt sie dann an den nächsten Knoten weiter. Es ist Aufgabe des Netzwerks, in dem diese Knoten enthalten sind, für den richten Datenfluss zu sorgen.

Node-RED selbst basiert auf einer Runtime die in *Node.js* geschrieben wurde, der Fluss-Editor kann über einen Webbrowser aufgerufen und bedient werden. Eine Anwendung kann man erstellen, in dem man aus der „*Palette*“ den bzw. die gewünschten *Knoten* auf die *Arbeitsfläche* zieht und beginnt diese miteinander per „*Kabel*“ zu verbinden. Mit einem Klick auf die Schaltfläche „*Deploy*“ wird die Applikation an den *Node-RED* Server übertragen und aktiv geschaltet.

1 API – Abkürzung für: Application Programming Interface. z.D. in etwa: Applikations-Programmierschnittstelle
2 Definition laut Projektbeschreibung auf https://nodered.org/, siehe auch https://nodered.org/about/
3 Siehe https://en.wikipedia.org/wiki/Flow-based_programming

Da *Node-RED* für das IoT-Zeitalter gemacht ist, darf auch eine grafische Bedienoberfläche für den PC bzw. das Smartphone und Tablet nicht fehlen. Es gibt mittlerweilse viele verschiedene *Node-RED* Module, die es dem Anwender sehr leicht machen, durch *Node-RED* eine web-basierte grafische Bedienoberfläche zu erstellten.

Node-RED wurde Anfang 2013 als Nebenprojekt von *Nick O´Leary* und *Dave Conway-Jones* bei der *IBM Emerging Technology Services Gruppe* ins Leben gerufen.

Was als *Proof-of-Concept* für Visualisierung und Manipulation von MQTT Themen begann, entwickelte sich schnell zu einem allgemeinen Tool, dass sich leicht erweitern lässt. Im September 2013 wurde *Node-RED* Open-Source und im Oktober 2016 eines der Mitgründungsprojekte der *JS Foundation,* wo es seitdem für alle frei zugänglich ist und weiterentwickelt wird.[4]

Node-RED wird mit dem Raspbian Betriebssystem für den Raspberry Pi mit ausgeliefert und kann sofort verwendet werden. Es gehört neben Python und vielen anderen Komponenten zur Grundausstattung des Raspberry Pi, was den Einstieg in die flussbasierte Programmierung sehr einfach macht.

Wie bereits erwähnt bietet *Node-RED* die Möglichkeit eine Web-Visualisierung zu erstellen, dies geschieht z.B. unter Verwendung des Moduls ***node-red-dashboard***.
Der Datenaustausch mit *CODESYS* lässt sich schnell und einfach per ModBus TCP erledigen, dazu verwenden wir in dieser App-Note in *Node-RED* das Modul ***node-red-contrib-modbus***, in *CODESYS* selbst ist ModBus TCP schon vorhanden und muss nur eingebunden und konfiguriert werden.

Auf den folgenden Seiten möchten wir als Ideenanregung kurz darstellen, wie *Node-RED* in Verbindung mit *CODESYS* genutzt werden kann, um eine eigene Web-Visualisierung zu erstellen.

Hinweis: *In dieser App-Note wird ein PiXtend V2 -S- als Demonstrationsgerät eingesetzt, Sie können aber genauso ein* ***PiXtend V2 -L-*** *verwenden.*

Wir wünschen viel Spaß beim Visualisieren!

Viele weitere Informationen, Tipps und Tricks finden Sie auch in unserem Support-Forum unter: https://www.pixtend.de/forum/

Die jeweils neusten Versionen aller Dokumente und Software-Komponenten finden Sie im Download-Bereich unserer Homepage: https://www.pixtend.de/downloads/

4 Siehe https://nodered.org/about/

1.1 Voraussetzungen

Diese App-Note wurde für *Node-RED* v0.18.x, das *PiXtend V2 -S-* und *CODESYS SP13* erstellt. Kenntnisse über die Funktion von *Node-RED*, die Verwendung der Palette und wie die Einstellungsdatei zu bearbeiten ist, sowie Erfahrung im Umgang und Programmierung von *CODESYS,* ModBus TCP und JavaScript werden vorausgesetzt. Es müssen zusätzliche Module in *Node-RED* installiert werden und in *CODESYS* sind verschiedene Datentypen (Stichwort: Konvertierung) im Einsatz, damit ein Datenaustausch über ModBus TCP reibungslos gelingt.

Diese App-Note richtet sich an *Experten*. Die zur Installation notwendigen Schritte werden im Folgenden exemplarisch für einen Anwendungsfall beschrieben, abweichende Anforderungen müssen vom Anwender selbst erarbeitet oder bei Qube Solutions UG beauftragt werden.

Es gibt keine spezielle Festlegung auf ein Raspberry Pi Modell.
Wir empfehlen jedoch eines der folgenden Modelle: B+, 2 B, 3 B und 3 B+.

Laden Sie das *PiXtend **CODESYS Image*** Version 2.x SD-Karten Image aus unserem Download-Bereich herunter und verwenden Sie dieses Image als Ausgangspunkt für die gezeigten Schritte in dieser App-Note.

Auf den Raspberry Pi können Sie entweder mit direkt angeschlossener Tastatur und Monitor oder per SSH (TeraTerm / Putty) von einem PC aus zugreifen. Der Raspberry Pi benötigt zum Update von *Node-RED* bzw. zur Installation weiterer Komponenten, wie *npm*, *node-red-dashboard* und dem *node-red-contrib-modbus* Modul eine aktive Internetverbindung.

1.2 Haftungsausschluss

Qube Solutions UG kann nicht für etwaige Schäden verantwortlich gemacht werden die unter Umständen durch die Verwendung der zur Verfügung gestellten Software, Hardware, Treiber oder der hier beschriebenen Schritte oder Software von Drittherstellern entstehen können.

1.3 Sicherheitshinweise

PiXtend darf nicht in sicherheitskritischen Systemen eingesetzt werden.

Prüfen Sie vor der Verwendung die Eignung von Raspberry Pi und PiXtend für Ihre Anwendung.

2. Node-RED Installation und Einrichtung

Zur Verwendung von *Node-RED* mit *CODESYS* und dem *PiXtend V2 -S-* wird eine SD-Karte mit dem *PiXtend CODESYS* Image ab Version 2.1.1.x benötigt. In diesem Image ist eine *CODESYS Demo* und eine *Node-RED Version* bereits vorinstalliert.

*Führen Sie alle Schritte mit dem „**pi**“ **Benutzer** durch, verwenden Sie keinesfalls „root“, dies führt zu einem Installationschaos bei Node-RED, das sich nicht mehr entwirren lässt.*

Um *Node-RED* auf dem Raspberry Pi auf den aktuellen Stand zu bringen, führen Sie folgende Schritte durch. Denken Sie daran den Raspberry Pi mit dem Internet zu verbinden, bevor Sie mit der Durchführung beginnen. Sie können die gezeigten Schritte entweder direkt am Raspberry Pi Computer durchführen mit Bildschirm, Maus und Tastatur oder per SSH-Client Programm (z.B. TeraTerm oder Putty) über das Netzwerk von Ihrem PC aus.

1. *Node-RED* Update:

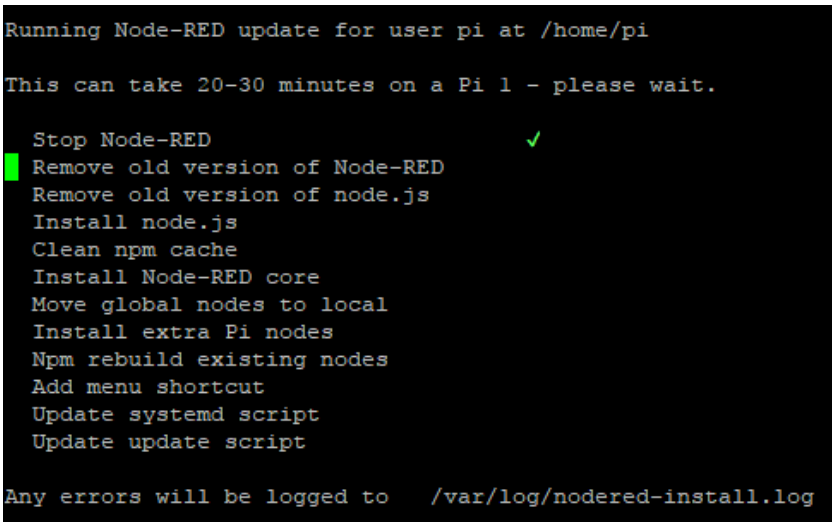
```
bash <(curl -sL https://raw.githubusercontent.com/node-red/raspbian-deb-package/master/resources/update-nodejs-and-nodered)
```

Der oben stehende Befehl muss in einer Zeile stehen, damit dieser fehlerfrei kopiert werden kann.

2. Alle Fragen des Installations- und Updateprogramms mit „y“ (yes/ja) beantworten.

3. Die Frage ob die Raspberry Pi spezifischen Knoten installiert werden sollen auch mit „y“ (yes/ja) beantworten.

4. Je nach SD-Karte, Raspberry Pi Typ und Geschwindigkeit der Internetverbindung kann die Installation *20-30 min* in Anspruch nehmen.



5. Weitere Informationen zu *Node-RED* und dem Raspberry Pi finden Sie hier:

https://nodered.org/docs/hardware/raspberrypi

6. Führen Sie nach dem *Node-RED* Update ein Reboot aus:

```
sudo reboot
```

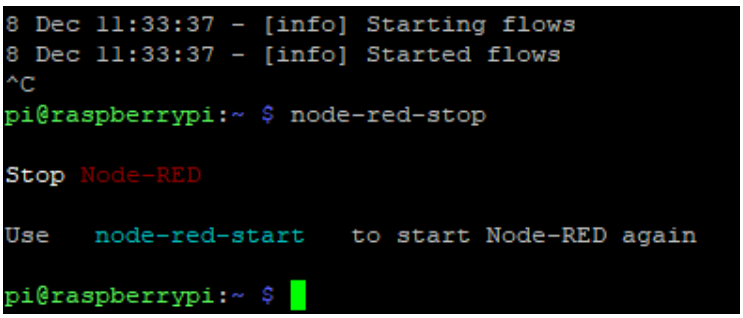
7. Nach dem Neustart können Sie mit dem folgenden Befehl *Node-RED* starten:

```
node-red-start
```

```
pi@raspberrypi:~ $ node-red-start

Start Node-RED

Once Node-RED has started, point a browser at http:// xxx.xxxx.xxx.xxx :1880
On Pi Node-RED works better with the Firefox or Chrome browser

Use   node-red-stop                         to stop Node-RED
Use   node-red-start                        to start Node-RED again
Use   node-red-log                          to view the recent log output
Use   sudo systemctl enable nodered.service  to autostart Node-RED at every boot
Use   sudo systemctl disable nodered.service to disable autostart on boot

To find more nodes and example flows - go to http://flows.nodered.org

Starting as a systemd service.
Started Node-RED graphical event wiring tool.
2 Jul 09:25:43 - [info]
Welcome to Node-RED
===================
2 Jul 09:25:43 - [info] Node-RED version: v0.18.7
2 Jul 09:25:43 - [info] Node.js  version: v8.11.3
2 Jul 09:25:43 - [info] Linux 4.9.80-v7+ arm LE
2 Jul 09:25:44 - [info] Loading palette nodes
2 Jul 09:25:51 - [info] Dashboard version 2.9.4 started at /ui
2 Jul 09:25:53 - [info] Settings file  : /home/pi/.node-red/settings.js
2 Jul 09:25:53 - [info] User directory : /home/pi/.node-red
2 Jul 09:25:53 - [warn] Projects disabled : editorTheme.projects.enabled=false
2 Jul 09:25:53 - [info] Flows file     : /home/pi/.node-red/flows_raspberrypi.json
2 Jul 09:25:53 - [info] Server now running at http://127.0.0.1:1880/
```

8. Möchten Sie *Node-RED* nicht weiter verwenden, dann können Sie mit dem Tastendruck *Ctrl+C (Strg+C)* zur Konsole zurückkehren und *Node-RED* mit folgendem Befehl stoppen:

```
node-red-stop
```



9. Für die Übungen mit *CODESYS* und *Node-RED*, die in dieser App-Note gezeigt werden, kann es von Vorteil sein, wenn Sie *Node-RED* automatisch bei jedem Start des Raspberry Pi mit starten lassen, ähnlich wie es bei *CODESYS* der Fall ist.

Node-RED hat beim ersten Start bereits alle notwendigen Informationen angezeigt, die wir brauchen, um es automatisch starten zu lassen.

Mit dem folgenden Befehl wird *Node-RED* automatisch gestartet:

```
sudo systemctl enable nodered.service
```

Soll *Node-RED* nicht mehr automatisch starten, dann lässt sich mit diesem Befehl der Autostart von *Node-RED* rückgängig machen:

```
sudo systemctl disable nodered.service
```

Wird *Node-RED* beim Start vom Raspberry Pi Computer nicht automatisch gestartet, so muss man dies nach jedem Neustart manuell tun, z.B. wenn die *CODESYS Demo* nach 2 Stunden stoppt und ein Neustart notwendig ist um weitermachen zu können. In diesen Fällen muss *Node-RED* jedes Mal manuell gestartet werden, daher empfiehlt es sich *Node-RED* automatisch starten zu lassen.

3. Node-RED vorbereiten

Bevor es mit der Visualisierung und *CODESYS* Programmierung losgehen kann, müssen in *Node-RED* die beiden Module *node-red-dashboard* und *node-red-contrib-modbus* installiert werden.

1. Nach dem Start des Raspberry Pi kann *Node-RED* mit folgenden Befehl gestaret werden, sollte es nicht bereits automatisch starten:

```
node-red-start
```

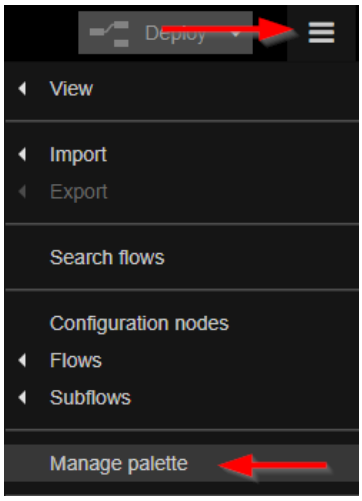
2. Der *Node-RED* Editor kann jetzt im Browser aufgerufen werden:

```
http://ip-adresse-rpi:1880
```

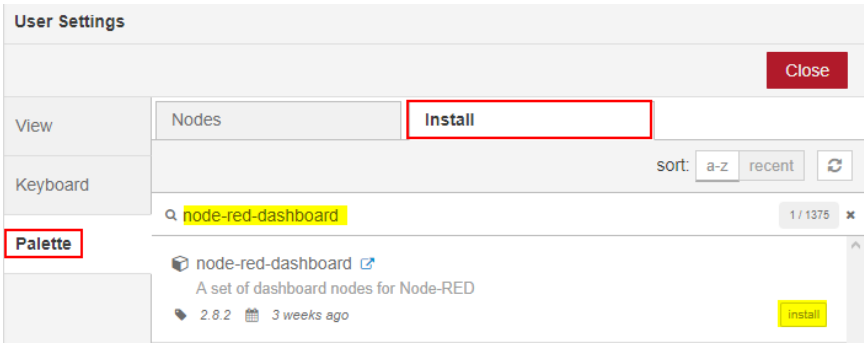
Für „*ip-adresse-rpi*“ die Netzwerkadresse des Raspberry Pi eingeben, üblicherweise wird diese von *Node-RED* beim Start auf der Konsole ausgegeben.

Der *Node-RED* Web-Server selbst ist über den Port *1880* zu erreichen.

3. Wurde der *Node-RED* Editor im Web-Browser geladen, kann es mit der Installation der benötigten Module los gehen. Zunächst öffnen wir das *Node-RED* Menü und wählen hier den Menüpunkt „*Manage palette*“ aus.

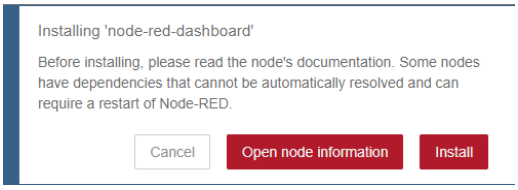


4. In den „*User Settings*“ aktivieren wir den Reiter „*Palette*“ und auf der rechten Seite in der *Palette*-Ansicht klicken wir auf den Reiter „*Install*“. Im Suchfeld tragen wir den Namen des zu installierenden Moduls ein.
In unserem Fall ist dies: *node-red-dashboard*



Wurde das Modul im *Node-RED* Verzeichnis gefunden, dann kann es mit einem Klick auf „*Install*“ in *Node-RED* installiert werden.
Bevor die Installation tatsächlich startet, erhalten wir noch einen Hinweis auf die Dokumentation und dass wir über die zu installierenden Knoten weitere Informationen erhalten können.

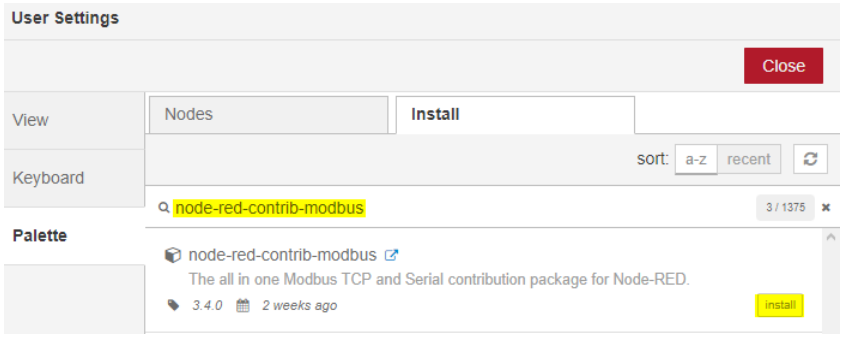
Mit einem Klick auf „*Install*“ wird das *Node-RED node-red-dashboard* Modul installiert.



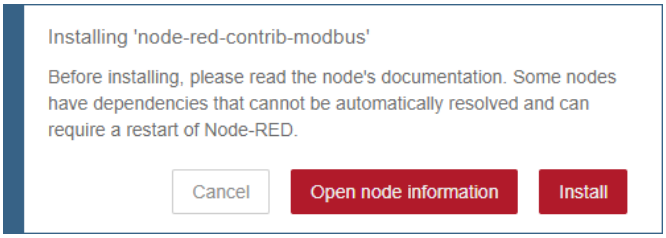
Die Installation kann *einige Zeit* in Anspruch nehmen, an dieser Stelle müssen wir gedultig warten bis wir den Hinweis erhalten, dass die Installation erfolgreich abgeschlossen wurde.

Im Browser erscheint ein grünes Hinweisfenster, das uns alle gerade installieren Knoten anzeigt.

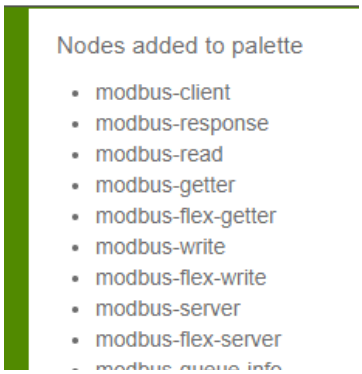
5. Bei der Installation des Moduls *node-red-contrib-modbus* verfahren wir genauso wie mit dem *node-red-dashboard* Modul, nur mit dem Unterschied, dass wir auf dem „*Install*“ Reiter den Suchbegriff „*node-red-contrib-modbus*“ *verwenden.*



Nach dem Klick auf *Install* muss die Installation für die ModBus Knoten ebenfalls noch mal bestätigt werden, bevor diese tatsächlich startet.



Die Installation aller ModBus Knoten kann einige Minuten in anspruchnehmen. Hat alles geklappt, dann zeigt uns *Node-RED* ein grünes Fenster an, das alle gerade installierten Knoten aufführt.



Die Vorbereitungen in *Node-RED* sind jetzt abgeschlossen und alles ist bereit, so dass wir mit der Programmierung in *CODESYS* loslegen können. Im nächsten Kapitel erstellen wir ein kleines *CODESYS* Projekt und verbinden dieses später mit *Node-RED* zur Visualisierung.

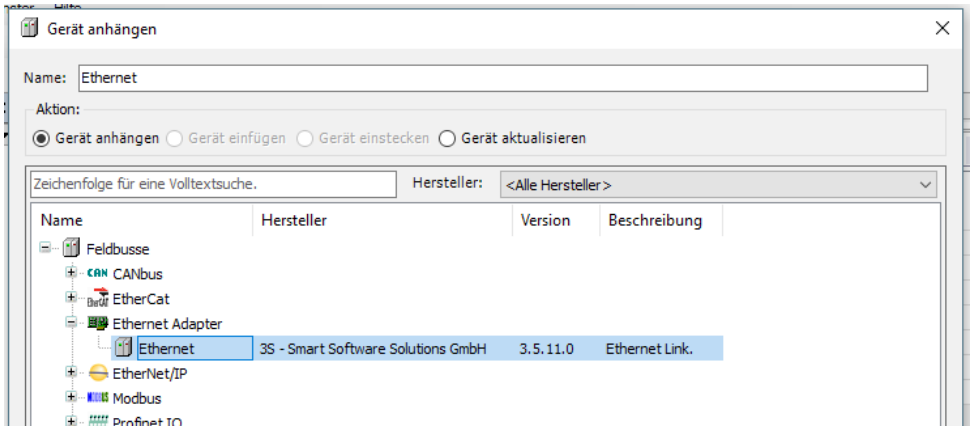
4. CODESYS Projekt erstellen

Bevor wir in *Node-RED* eine Web-Visualisierung erstellen können, brauchen wir in *CODESYS* ein kleines Programm, das sich um den Informationsaustausch mit *Node-RED* kümmert, zusätzlich wir müssen einen ModBus TCP Slave anlegen und konfigurieren. Mehr ist im ersten Schritt nicht notwendig.

1. Als erstes brauchen wir ein neues leeres *CODESYS* Projekt mit dem Raspberry Pi Computer als Steuerung. Der Projektname kann z.B. „*Node-RED Web-Visu“* lauten. Ferner muss der *GPIO 24* des Raspberry Pi als *Output* eingestellt werden und braucht im E/A-Abbild Reiter einen Namen, z.B. *SPI_EN*
Zusätzlich brauchen wir am SPI-Bus einen SPI-Master und darunter ein *PiXtend V2 -S-* als SPI-Slave.

2. Jetzt brauchen wir einen *Ethernet-Adapter* damit wir einen ModBus TCP Slave unter *CODESYS* erstellen können.

In *CODESYS* im Gerätebaum auf der SPS (dem Raspberry Pi) einen Rechtsklick ausführen und im Menü den Eintrag „*Gerät anhängen*“ wählen.



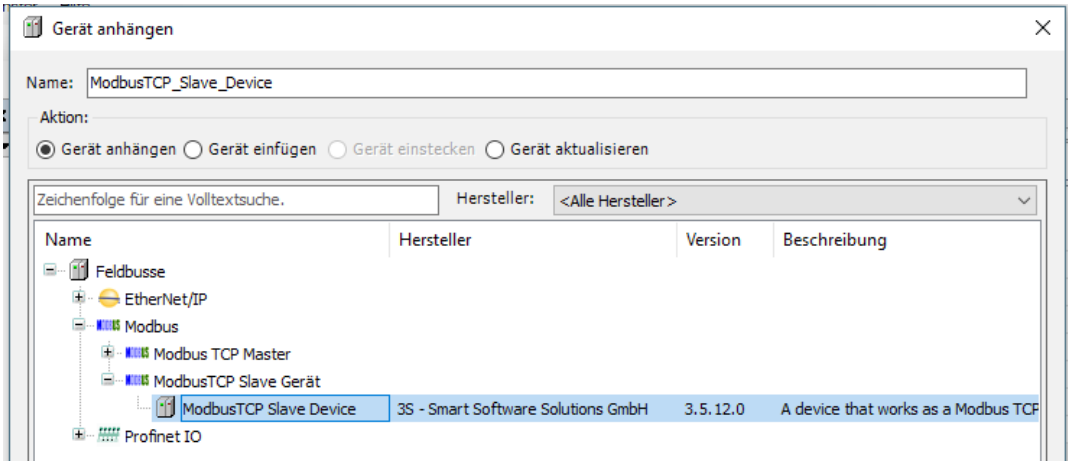
Im „*Gerät anhängen*“ Fenster den Eintrag *Feldbusse* aufklappen und den Eintrag „*Ethernet Adapter*“ finden und auch diesen aufklappen. Nach dem Anklicken des Eintrags „*Ethernet*“, können wir jetzt einen „*Ethernet Adapter*“ zu unserer Steuerung hinzufügen, dies geschieht mit einem Klick auf die Schaltfläche „*Gerät anhängen*“ unten rechts im Fenster.

CODESYS braucht einen Moment um diese Aktion durchzuführen, das Fenster „*Gerät anhängen“* können wir *offen* lassen, da wir es gleich noch mal brauchen.

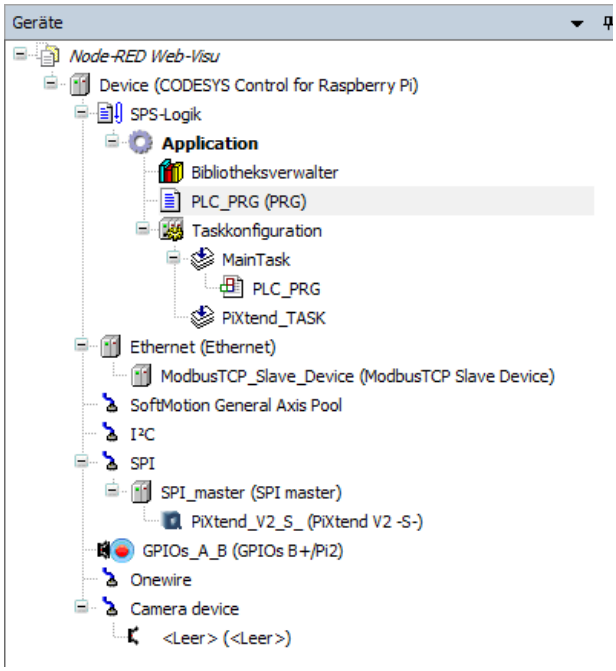
Hat *CODESYS* den *Ethernet Adapter* unserer Steuerung hinzugefügt, dann können wir diesen jetzt einfach im Gerätebaum anklicken, während das „*Gerät anhängen“* Fenster

noch offen ist, und der Inhalt des Fensters verändert sich.

Wir bekommen jetzt eine Auswahl an möglichen Bussystemen angezeigt, die mit dem *Ethernet Adapter* kompatibel sind.
In der Liste suchen wir das Gerät „*ModbusTCP Slave Device*“ und fügen es dem *Ethernet Adapter* hinzu.



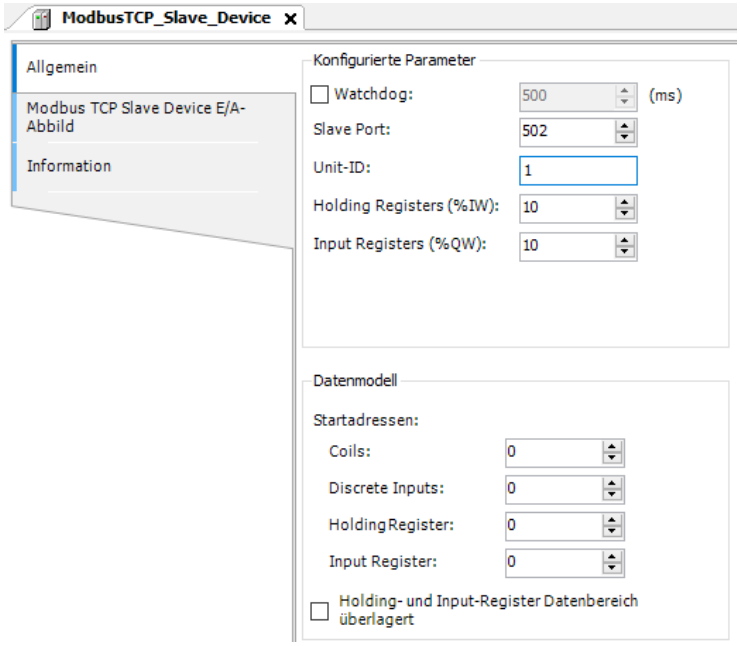
Unsere Steuerung, der Raspberry Pi Computer, ist jetzt ein ModBus TCP Slave und unser Projekt sieht jetzt so aus wie im nachfolgenden Bild:



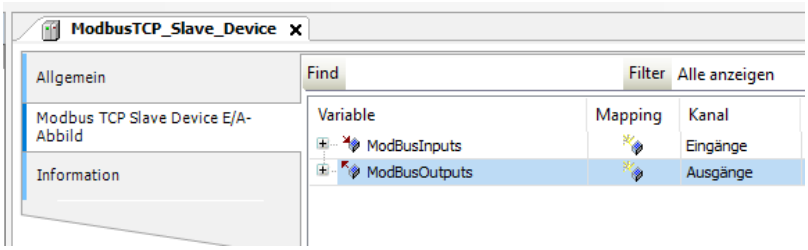
3. Um den ModBus später nutzen zu können, müssen wir noch 2 Einstellungen vornehmen.

Mit einem Doppelklick auf das Gerät *ModbusTCP_Slave_Device* im Gerätebaum von *CODESYS,* können wir die Einstellungen für dieses Gerät öffnen.

Im Reiter „*Allgemein*“ tragen wir bei „*Unit-ID*“ die Zahl *1* ein. Die Anzahl der *Holding*- und *Input*-Register lassen wir auf *10* stehen, da wir in diese App-Note nicht mehr brauchen. Sollten später dennoch mehr Register gebraucht werden, so lässt sich die Zahl in diesem Fenster einfach erhöhen. Sonst muss auf dieser Seite nicht veändert werden.



Jetzt wechseln wir auf die Seite „*Modbus TCP Slave Device E/A-Abbild*“ und tragen für die *ModBus Holding Register (Eingänge)* den Namen ***ModBusInputs*** und den *Modbus Input Registern (Ausgänge)* geben wir den Variablennamen ***ModBusOutputs***.

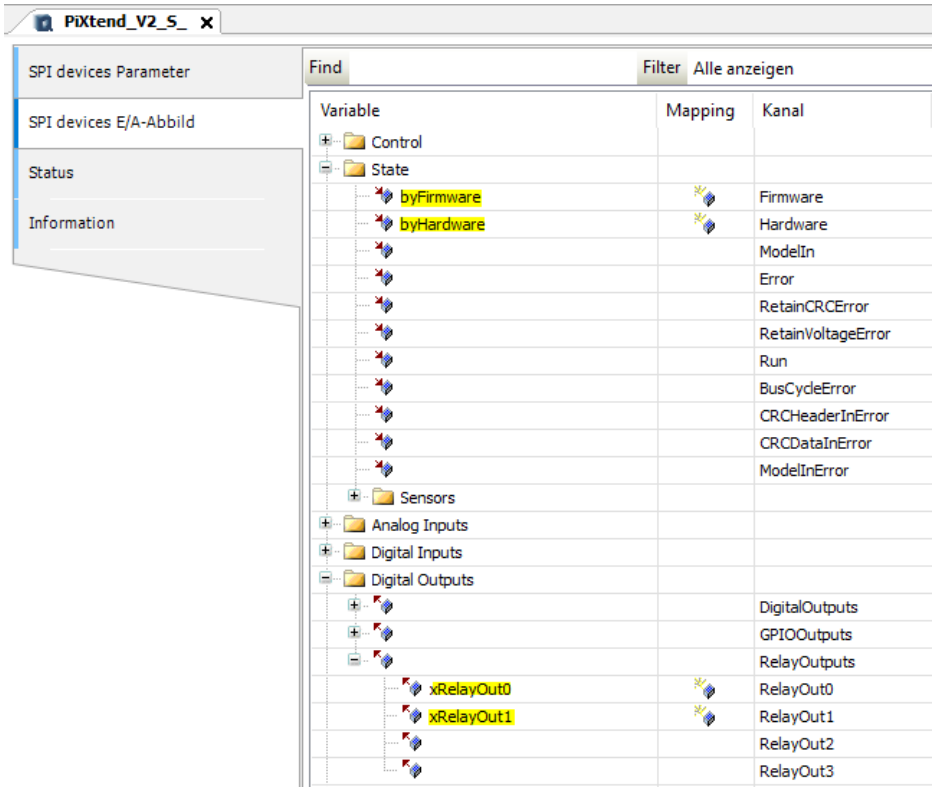


Über diese beiden Namen erhalten wir später im *CODESYS* Programm Zugriff auf die Daten die wir von *Node-RED* erhalten bzw. die wir *Node-RED* als Information zur Visualisierung zur Verfügung stellen.

4. Als letzten Schritt, bevor wir ein Programm schreiben, vergeben wir im *PiXtend V2 -S-* Gerät noch 4 Variablennamen. Auf diesem Weg können wir in *Node-RED* etwas anzeigen, aber auch etwas steuern, z.B. Relais ein- und ausschalten.

Mit einem Doppelklick auf das „*PiXtend_V2_S_*“ Gerät können wir die *PiXtend* Einstellungen aufrufen und wechseln hier in den Reiter „*SPI devices E/A-Abbild*“.
Auf der rechten Seite klappen wir den Ordner „*State*“ auf und tragen bei den Kanälen „*Firmware*“ und „*Hardware*“ jeweils in der Spalte *Variable* die Namen „*byFirmware*“ und „*byHardware*“ ein.

Beim Ordner „*Digital Outputs*“ machen wir das gleiche und öffnen diesen, zusätzlich klappen wir noch den Kanal „*RelayOutputs*“ auf und geben den Kanälen „*RelayOut0*“ und „*RelayOut1*“ die Variablennamen „*xRelayOut0*“ und „*xRelayOut1*“.



Alle Geräte sind jetzt eingestellt und konfiguriert. Im nächsten Schritt können wir unser Programm schreiben, das die Daten für *Node-RED* verarbeitet und über ModBus TCP weitergibt bzw. die Daten die von *Node-RED* kommen in die richtigen Datentypen umwandelt und an die richtige Stelle weiterleitet.

5. Im Baustein „*PLC_PRG*“ deklarieren wir noch eine neue Variable:

wLifeCounter : WORD;

Diese Variable verwenden wir um in *Node-RED* ein kontinuierliches Lebenszeichen unserer Steuerung zu haben. Für den Datenaustausch mit *Node-RED* über ModBus TCP verwenden wir folgende Register:

Register (Adresse)	ModBus Holding Register (*Eingänge*) Array-Variable: *ModBusInputs*	Modbus Input Register (*Ausgänge*) Array-Variable: *ModBusOutputs*
0	Relais 0	Lebenszeichenzähler
1	Relais 1	Firmware Version
2	-	Hardware Version

Der Datentyp der ModBus Register für die Ein- und Ausgänge ist jeweils vom Typ *WORD*, daher müssen alle Werte entsprechend umgewandelt bzw. später in *Node-RED* richtig gesetzt werden, damit *CODESYS* diese verarbeiten kann.

Für unser Beispiel verwenden wir folgendes Programm:

```
SPI_ENABLE := TRUE;

xRelayOut0 := WORD_TO_BOOL(ModBusInputs[0]);
xRelayOut1 := WORD_TO_BOOL(ModBusInputs[1]);

wLifeCounter := wLifeCounter + 1;
ModBusOutputs[0] := wLifeCounter;
ModBusOutputs[1] := BYTE_TO_WORD(byFirmware);
ModBusOutputs[2] := BYTE_TO_WORD(byHardware);
```

PLC_PRG

```
1  PROGRAM PLC_PRG
2  VAR
3      wLifeCounter : WORD;
4  END_VAR
```

```
1  SPI_ENABLE := TRUE;
2
3  //Assign ModBus inputs to the Relays, register 0 is Relay 0 and register 1
4  //is Relay 1. Note: A value of 0 = False and a value of 1 = True.
5  //It is important in Node-RED to set the correct values!
6  xRelayOut0 := WORD_TO_BOOL(ModBusInputs[0]);
7  xRelayOut1 := WORD_TO_BOOL(ModBusInputs[1]);
8
9  //Set ModBus outputs for Node-RED
10 //Set the life counter to ModBus register 0
11 wLifeCounter := wLifeCounter + 1;
12 ModBusOutputs[0] := wLifeCounter;
13 //Set the firmware and hardware version to register 1 and 2
14 ModBusOutputs[1] := BYTE_TO_WORD(byFirmware);
15 ModBusOutputs[2] := BYTE_TO_WORD(byHardware);
```

In der ersten Zeile wird die SPI-Kommunikation eingeschaltet (GPIO 24), ggf. haben Sie sich für einen anderen Namen entschieden, dann verwenden Sie diesen und setzen ihn auf *TRUE*.

Für die Relais erwarten wir von *Node-RED* Zahlenwerte im Bereich von *0* bis *1*, diese entsprechen in *CODESYS* den Werten *FALSE bzw. TRUE*, wenn wir die Umwandlungsfunktion *WORD_TO_BOOL* verwenden.

Andersherum müssen wir die Werte unserer *Firmware*- und *Hardware-Versionen* vom Typ *BYTE* nach *WORD* umwandeln bevor wir diese in das *ModBusOutputs* Datenarray schreiben können.
Der Lebenszeichenzähler (*wLifeCounter*) hingegen liegt schon im richtigen Datentyp vor und kann daher direkt zugewiesen werden.

6. Damit sind in *CODESYS* alle Arbeiten abgeschlossen, hier müssen wir nur noch das Programm übersetzen und auf unsere Steuerung, den Raspberry Pi, laden und starten.

Das war‘s.
Die nächsten Schritte führen wir in *Node-RED* durch und erstellen uns eine Web-Visualisierung.

5. Node-RED Flow mit Web-Visualisierung erstellen

Nach den Arbeiten in *CODESYS* können wir jetzt mit *Node-RED* eine Web-Visualisierung erstellen. Die nachfolgenden Schritte zeigen die Verwendung der *Modbus read* und *Modbus write* Knoten, sowie der *node-red-dashboard* Elemente „*text*“ und „*switch*“ zusammen mit ihrer Konfiguration. Besuchen Sie auch die *Node-RED dashboard* Seite im Internet für weitere Informationen.

1. Einen *Node-RED* kompatiblen Browser öffnen, z.B. Firefox oder Chrome.

2. Den *Node-RED* Editor im Browser aufgerufen:

http://ip-adresse-rpi:1880

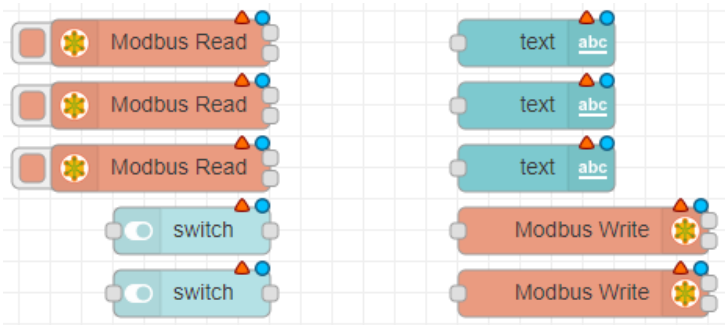
Für „*ip-adresse-rpi*“ die Netzwerkadresse des Raspberry Pi eingeben, üblicherweise wird diese von *Node-RED* beim Start auf der Konsole ausgegeben.

Der *Node-RED* Web-Server selbst ist über den Port *1880* zu erreichen.

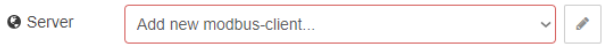
3. Hat der Web-Browser den *Node-RED* Editor geladen, können wir mit der Erstellung unseres *Flows* und der damit verbundenen Web-Visualisierung loslegen.

Wir brauchen 3 *Modbus read* und 2 *Modbus write* Knoten. Damit wir die gelesenen Informationen aus den *3 ModBus Input Registern* (0 = LifeCounter, 1 = Firmware-Version und 2 = Hardware-Version) auch anzeigen können, ziehen wir uns aus der Kategorie „*dashboard*“ 3 „*text*“-Knoten auf die Arbeitsfläche.

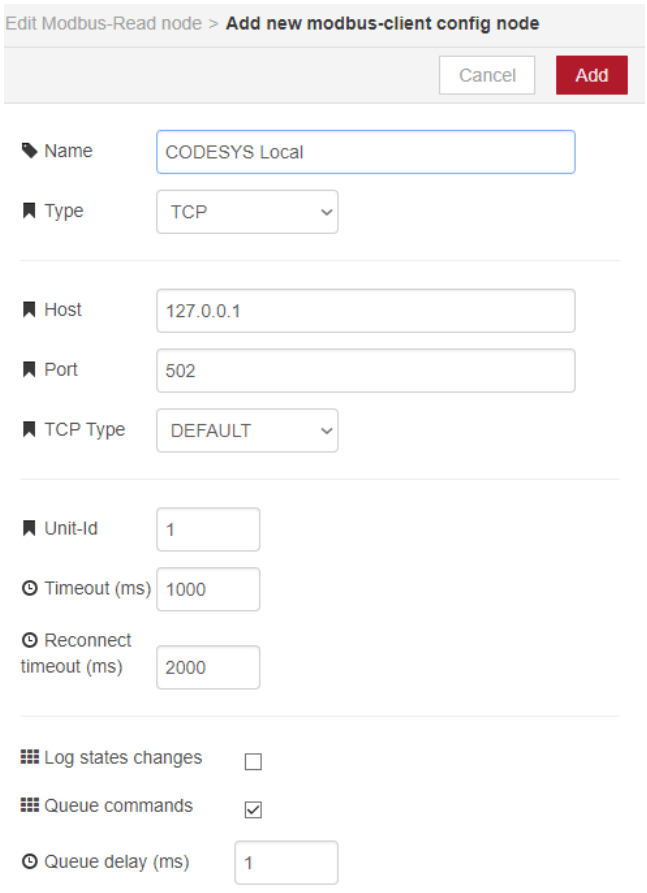
Um auch etwas bewegen bzw. steuern zu können holen wir uns zusätzlich noch 2 „*switch*“-Knoten dazu.



4. Durch einen Doppelklick auf den ersten *Modbus read* Knoten öffnet sich dessen Konfigurationsseite und als erste Aufgabe müssen wir eine neue ModBus Verbindung bzw. einen neuen Server einrichten.



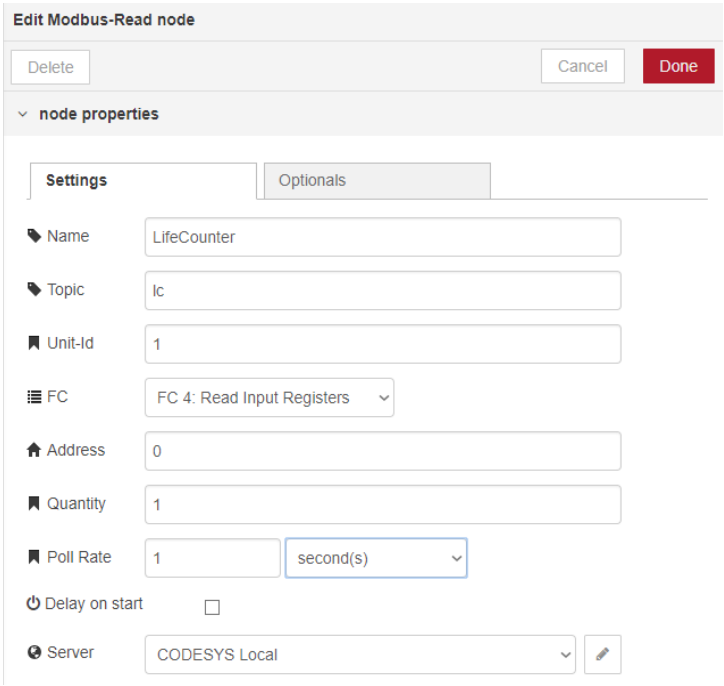
Ganz unten auf der Konfigurationseite ist das *Server* Auswahlfeld zu finden und rechts daneben können wir über einen Klick auf das *Stift-Symbol* eine neue Verbindung bzw. eine neue ModBus Client Konfiguration anlegen. Der *Name* kann frei gewählt werden, jedoch unter *Host* muss die Addresse *127.0.0.1* stehen, unter *Port* die Zahl 502 und der *TCP Type* muss auf *DEFAULT* bleiben.
Damit ist die ModBus Client Konfiguration abgeschlossen und mit einem Klick auf „*Add*“ wird die neue Server-Verbindung eingerichtet und kann von uns verwendet werden.



Jetzt können wir die 3 *Modbus read* Knoten konfigurieren. Die Einstellugnen für alle 3 Knoten stehen in der nachfolgenden Tabelle. Der *Name* und das *Topic* für jeden Knoten können freigewählt werden.

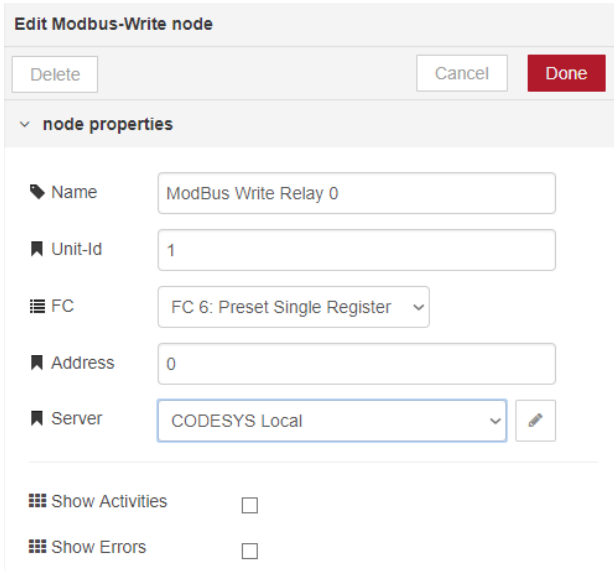
Name	Topic	Unit-Id	FC	Address	Quantity	Poll Rate
LifeCounter	lc	1	4	0	1	1 second
Firmware Version	fw	1	4	1	1	1 second
Hardware Version	hw	1	4	2	1	1 second

Unter *Server* wird bei jedem Knoten die gleiche Verbindung eingestellt. Die Konfiguration des ersten Knotens zum Auslesen des Lebenszeichenzählers könnte wie folgt aussehen:



Damit sind die *Modbus read* Knoten konfiguriert und können verwendet werden. Im nächsten Schritt stellen wir noch die *Modbus write* Knoten ein, bevor wir alle Knoten verbinden und die Einstellungen für unsere Web-Visualisierungs Knoten vornehmen.

5. Durch einen Doppelklick auf den ersten *Modbus write* Knoten öffnet sich dessen Konfigurationsseite und wir nehmen folgende Einstellugnen vor:



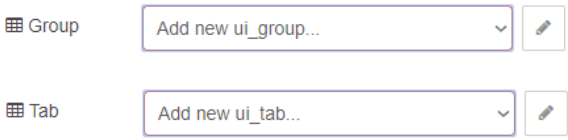
Beim zweiten Knoten nehmen wir die gleichen Einstellungen vor wie beim ersten, mit dem Unterschied, dass wir bei *Address* eine *1* eintragen und im *Name* ebenfalls die *0* durch eine *1* ersetzen. Der Server ist auch hier der gleiche wie bei den *Modbus read* Knoten und muss nicht geändert werden.

Die Einstellungen in der Übersicht:

Name	Unit-Id	FC	Address
ModBus Write Relay 0	1	6	0
ModBus Write Relay 1	1	6	1

Die ModBus Knoten sind alle konfiguriert und eingestellt, jetzt können wir die Knoten für die Web-Visualisierung bearbeiten.

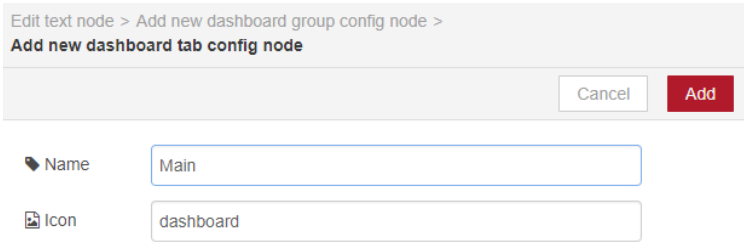
6. Durch einen Doppelklick auf den ersten *text*-Knoten öffnet sich die Konfigurationsseite und wir müssen als erstes eine neue „*ui_group*“ (Anzeigegruppe) anlegen sowie gleich dazu einen neuen „*ui_tab*“ (Anzeigereiter).



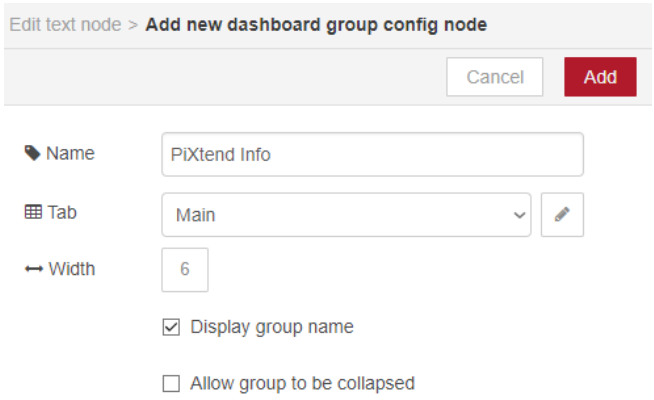
Das Bearbeiten der jeweiligen Felder geschieht durch einen Klick auf das *Stift-Symbol* rechts neben dem jeweiligen Feld.

Für unseren ersten Anzeigereiter (*ui_tab*) tragen wir „*Main*“ ein und für die Anzeigegruppe „*PiXtend Info*“. Auf jeder Konfigurationsseite müssen wir unsere Angaben mit „*Add*“ bestätigen und der *node-red-dashboard* Konfiguration hinzufügen.

Einen neuen Anzeigereiter (*tab*) anlegen:



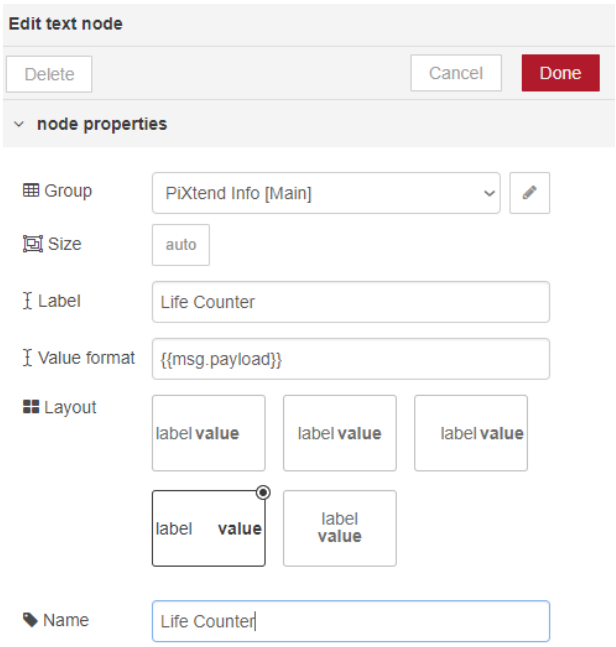
Eine neue Anzeigegruppe anlegen (*group*):



Sind diese ersten Schritte beim ersten *text*-Knoten abgeschlossen, können wir jetzt das eigentliche Element konfigurieren.

Für alle 3 *text*-Knoten können folgende Einstellungen verwendet werden:

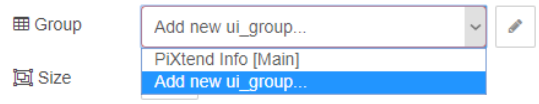
Group	Label	Name
PiXtend Info [Main]	Life Counter	Life Counter
PiXtend Info [Main]	Firmware Version	Firmware Version
PiXtend Info [Main]	Hardware Version	Hardware Version

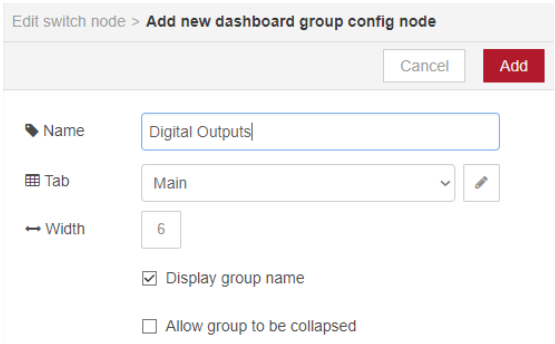


Nach der Eingabe aller Informationen müssen diese durch einen Klick auf die Schaltfläche „*Done*“ in der *node-red-dashboard* Konfiguration gespeichert werden.

7. Für die *switch*-Knoten legen wir eine neue Gruppe an und platzieren sie in dieser Gruppe, getrennt von der *PiXtend Info* Gruppe.

Eine neue Gruppe lässt sich anlegen in dem wir auf dem ersten *switch*-Knoten einen Doppelklick ausführen und dessen Konfigurationsseite öffnen. Klappen wir jetzt das *Group* Feld auf, können wir den Eintrag „*Add new ui_group...*“ auswählen und eine neue Anzeigegruppe anlegen. Der neuen Gruppe geben wir z.B. den Namen „*Digital Outputs*“.

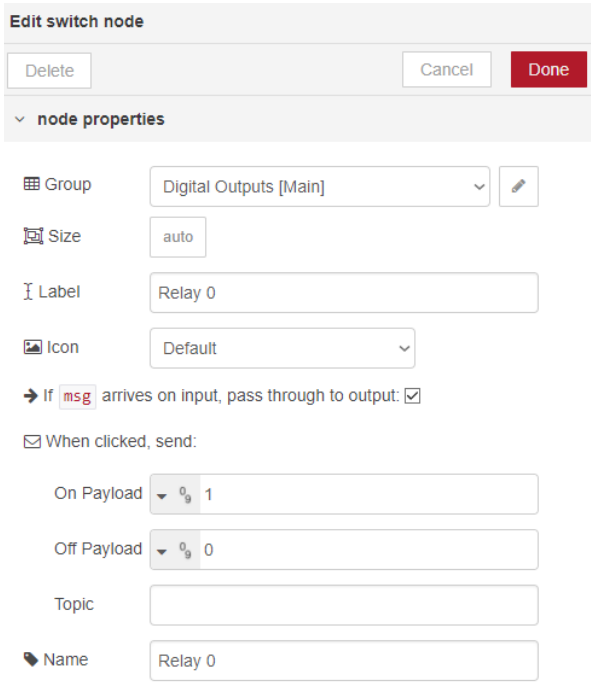




Diese neue Anzeigegruppe fügen wir dem schon bestehenden Reiter (*Tab*) „*Main*“ hinzu.

Mit einem Klick auf die Schaltfläche „*Add*“ bestätigen wir unsere Eingaben und erstellen eine neue Gruppe.

Den ersten *switch*-Knoten stellen wir wiefolgt ein:



Wichtig ist dass der Datentyp von „*On Payload*“ und „*Off Payload*“ eine Zahl (number) ist und für „*On*“ eine *1* und für „*Off*“ eine *0* eingetragen wird.

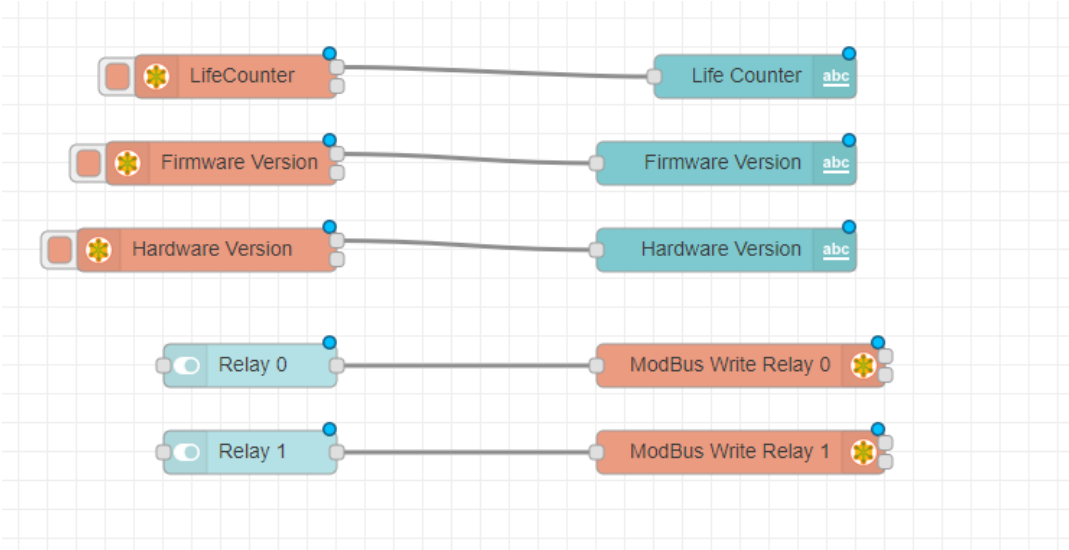
Die gleichen Einstellungen nehmen wir beim zweiten *switch*-Knoten vor.

8. Alles was jetzt noch fehlt sind die Verbindungen zwischen den Knoten und damit wären auch alle Arbeiten in *Node-RED* erledigt.

Wir stellen folgende Verbindungen zwischen den Knoten her:

- LifeCounter (*ModBus*) → Life Counter (*text*)
- Firmware Version (*ModBus*) → Firmware Version (*text*)
- Hardware Version (*ModBus*) → Hardware Version (*text*)
- Relay 0 (*switch*) → ModBus Write Relay 0 (*ModBus*)
- Realy 1 (*switch*) → ModBus Write Relay 1 (*ModBus*)

Sind alle Knoten verbunden, dann sollte das Ergebnis in etwa so aussehen wie im foglenden Bild:



9. Sind alle Knoten verbunden, so ist der Flow für unsere Web-Visualisierung fertig und mit einem Klick auf „***Deploy***“ wird alles an den *Node-RED* Server übertragen. Der Flow startet automatisch mit seiner Arbeit.

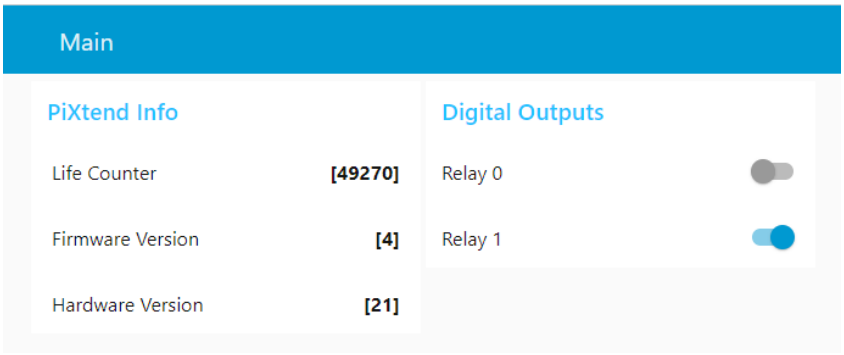
10. Ab jetzt können wir über die nachfolgende Adresse auf unsere neue Web-Visualisierung zugreifen und alle Informationen vom *PiXtend* in Augenschein nehmen bzw. die Relais 0 und 1 fernsteuern.

http://ip-adresse-rpi:1880/ui

Im Abschnitt *PiXtend Info* sehen wir die Textausgaben für unseren Lebenszeichenzähler, sowie die Version der Firmware und der Hardware des verwendeten *PiXtends*.

Daneben befindet sich die Anzeigegruppe *Digital Outputs* wo wir unsere beiden Relais finden. Mit einem Klick auf einen der beiden Schalter können wir das jeweilige Relais ein- bzw. ausschalten.

Node-RED dashboard Ansicht im Web-Browser oder auf dem Smartphone/Tablet.



Ansicht in *CODESYS,* das den Inhalt der Web-Visualisierung widerspiegelt.

```
//Assign ModBus inputs to the Relays, register 0 is Relay 0 and register 1
//is Relay 1. Note: A value of 0 = False and a value of 1 = True.
//It is important in Node-RED to set the correct values!
xRelayOut0 FALSE := WORD_TO_BOOL(ModBusInputs[0] 0 );
xRelayOut1 TRUE  := WORD_TO_BOOL(ModBusInputs[1] 1 );

//Set ModBus outputs for Node-RED
//Set the life counter to ModBus register 0
wLifeCounter 43378 := wLifeCounter 43378 + 1;
ModBusOutputs[0] 43378 := wLifeCounter 43378 ;
//Set the firmware and hardware version to register 1 and 2
ModBusOutputs[1] 4  := BYTE_TO_WORD(byFirmware 4 );
ModBusOutputs[2] 21 := BYTE_TO_WORD(byHardware 21 );
RETURN
```

Geschafft und Glückwunsch!

Sie haben Ihre erste *Node-RED* Web-Visualisierung für CODESYS erstellt.

Weiterführende Informationen finden Sie im nachfolgenden *FAQ* Kapitel.

6. Frequently Asked Questions (FAQ)

Gibt es Einschränkungen bei der Verwendung von *Node-RED* auf dem Raspberry Pi die ich beachten muss?

Uns sind derzeit keine Einschränkungen bekannt, einzig die Rechenleistung der Quad-Core CPU und die Menge an Arbeitsspeicher des Raspberry Pi können Ihre Kreativität begrenzen.

Die verfügbaren Knoten und Module in *Node-RED* scheinen begrenzt zu sein, gibt es da nicht mehr?

Da *Node-RED* auf *Node.js* aufbaut gibt es eine ganze Fülle an weiteren Knoten und Funktionen die man nachinstallieren kann, hierfür wird der *npm* Paket Manager verwendet. Im *Node-RED* Menü einfach den Eintrag „*Manage Palette*“ anklicken und im Reiter „*Install*“ die Suchfunktion nutzen um neue Knoten zu finden und zu installieren.

Um sich erstmal einen Überblick zu verschaffen empfiehlt es sich ggf. zu erst die Web-Seite von *npm* zu besuchen und hier nach *Node-RED* Packages zu suchen, um zu sehen was es alles gibt.

Weitere Informationen gibts z.B. hier:

- https://nodered.org/docs/getting-started/adding-nodes
- https://github.com/node-red/node-red-nodes
- https://www.npmjs.com/

Ich habe viele Informationen die angezeigt werden sollen, gibt es in *Node-RED* eine Obergrenze?

In *Node-RED* selbst ist uns nichts aufgefallen, was auf eine Einschränkung bzw. Obergrenze der Web-Visualisierung hindeutet.
Bei den Modbus Knoten hingegen gibt es vom Ersteller selbst einen Hinweis:

- Möchten man mehr als 10 Knoten (Register-Abfragen) verwenden, dann sollte man entweder eine zweite Verbindung erstellen oder den Knoten „***Modbus Flex Getter***“ verwenden. Hier kann man die Register-Abfragen selbst zusammenstellen und per Nachricht an den Knoten schicken.

Dies geht z.B. unter Verwendung eines *function* Knoten mit dem Code:

```
msg.payload = { value: msg.payload, 'fc': 6, 'unitid': 1, 'address': 0 , 'quantity': 1 }
return msg
```

- Für das Schreiben von Registern gilt das Gleiche. Möchte man mehr als 10 Register einzeln beschreiben bzw. braucht man mehr als 10 Modbus *write* Knoten, dann sollte man auf den Knoten „***Modbus Flex Write***“ zurückgreifen.
- Bei *CODESYS* hingegen scheint es eine Obergrenze bei ModBus TCP zu geben, hier kann man aktuell nicht mehr als *2000* Holding- und *2000* Input-Register einstellen, wenn man das *PiXtend* und den Raspberry Pi GPIO 24 verwenden möchte. Werden mehr Register eingestellt, dann gehen *CODESYS* die Speicherandressen aus und das Programm lässt sich nicht mehr übersetzen.

Kann ich auch andere Datentypen als nur Byte und Bool mit *Node-RED* austauschen?

Dies ist kein Problem, alles was dazu notwendig ist, ist in *CODESYS* die entsprechenden Daten ggf. aufzuteilen und in den WORD-Datentyp umzuwandeln und in *Node-RED* nach dem Empfang der Daten diese wieder zurückzuwandeln, sofern dies notwendig ist.

Beispiel Temperatur:
In CODESYS erhalten Sie eine Temperatur z.B. von einem DHT11 oder DHT22 Sensor in Grad Celcius und als Datentyp REAL. Diesen können Sie z.B. mit 10 oder 100 multiplizieren, jenachdem wieviele Nachkommastellen Sie an *Node-RED* weitergeben möchten, dann wandeln Sie den Wert vom Typ REAL nach WORD um. Nach dem Einlesen in *Node-RED* dividieren Sie den abgerufenen Wert durch 10 oder 100 und Sie erhalten eine Gleitkommazahl mit einer oder zwei Nachkommastellen und diesen Wert können Sie an der Web-Visualisierung anzeigen.

Bei noch größeren Werten wie bei einem DWORD (4 Bytes bzw. 2 Words), muss der Wert in jeweils 2 WORDs aufgeteilt werden, sonst lässt sich dieser nicht übertragen. Dies geht in *CODESYS* bequem per Bit Operation mit AND bzw. OR, oder durch die Verwendung des *Union*-Datentyps. In der *CODESYS* Dokumentation gibt es dazu weitere Informationen.

In *Node-RED* kann man einen *function*-Knoten verwenden und mit etwas *JavaScript* Code lässt aus 2 WORDs wieder eine vollständige Zahl erstellen.

Ein Ansatz könnte z.B. so aussehen:

Zeile 1: *msg.payload = (msg.payload[1]<<16) | msg.payload[0];*

Zeile 2: *return msg;*

Bei *Zeile 1* muss beachtet werden, dass *msg.payload[1]* und *msg.payload[0]* auch vertauscht sein können, dies hängt davon ab, in welche ModBus Register in *CODESYS* das *High-WORD* und das *Low-WORD* geschrieben wurden.

Gerne möchten wir Sie für den Informationsaustausch in die Foren von Qube Solutions einladen:

https://www.pixtend.de/forum/