Projekt Smart-Home

Das Projekt Smart-Home soll einen kurzen Einblick in die Möglichkeiten bieten, dass eigene Haus bzw. Klassenzimmer mit einem Raspberry Pi intelligenter zu machen. In diesem Projekt geht es mehr um die Vermittlung von Konzepten als um eine konkrete IoT Anwendung. Den Lernenden ist selbst überlassen, welche weiteren Projekte mit dem Know-How umgesetzt werden sollen.

Im Modul B2 - Die Internetversteher wurde bereits thematisiert, wie das Internet funktioniert und das Client-Server Modell vorgestellt. Ein weiteres Modell, welches im Kontext Internet-of-Things verwendet werden kann, ist das Message Passing Modell. Bei diesem Konzept reiht ein Sender Nachrichten in eine Warteschlange ein, welche von einem Nachrichtenbroker ausgewertet und an die entsprechenden Empfänger verteilt werden. Sobald ein Empfänger die Nachricht empfängt, unterbricht er seine Tätigkeit. Im Folgenden wird ein Protokoll vorgestellt, welches diesem Paradigma folgt.

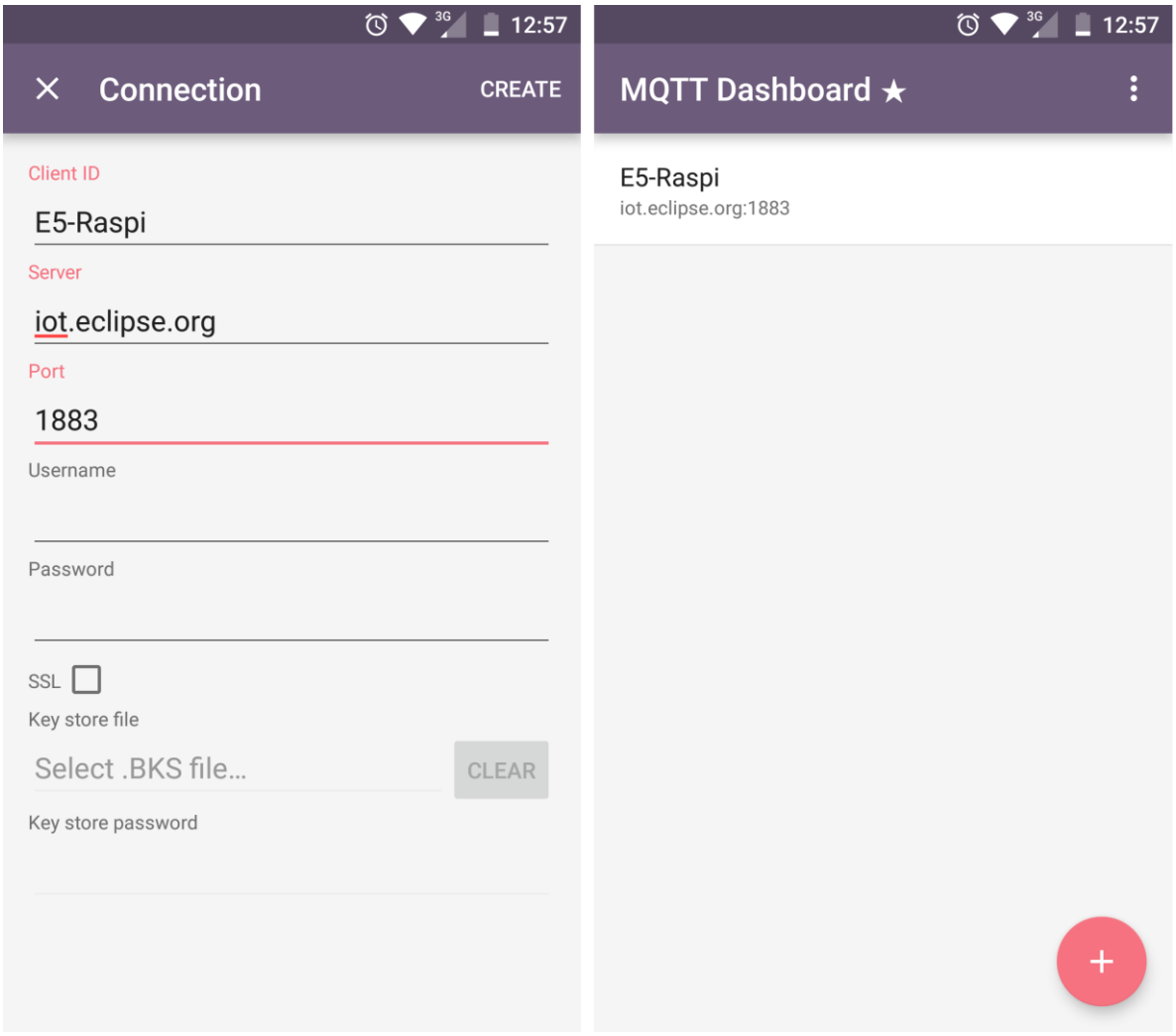
Das *Message Queue Telemtery Transport* Protokoll, kurz MQTT, wurde für den Einsatz in Netzwerken mit kleiner Bandbreite und hohen Latenzen optimiert und findet daher Verwendung in M2M (*Machine-to-Machine*) Szenerien, in welchem Geräte automatisiert untereinander kommunizieren. Ein Sender kann Daten veröffentlichen (*publish*), indem diese an einem Broker, dessen funktionsweise bereits im Abschnitt zum Message Passing erläutert wurde, sendet. Diese Nachricht enthält nicht nur die zu übertragenden Daten (*payload*), sondern auch Metadaten. In diesen Metadaten ist das sogenannte Topic enthalten. Dieses Topic wird zur Adressierung der Daten im Netzwerk benötigt. Jeder Client kann ein oder mehrere Topics gleichzeitig abonnieren (*subscribe*). Empfängt nun der Broker die Daten, prüft dieser, welche Clients das in den Nachrichten spezifizierte Topic abonniert haben und leitet die Daten die Abonennten weiter. Zusätzlich kann der Sender eine Dienstgüte (*Quality of Service* oder kurz *QoS*) bestimmen, die festlegt mit wie viel Aufwand der Broker versuchen soll die Nachricht bei den Abonennten zuzustellen. Eine gute Übersicht mit weiteren Information zum Protokoll MQTT findet sich auf folgender Seite: http://www.hivemq.com/mqtt-essentials/

Nun soll ein kleines Internet-of-Things Projekt mit Hilfe von MQTT und dem Raspberry vorgestellt werden. Eine LED soll über das Internet an und aus geschaltet werden. Das Prinzip lässt sich im Anschluss auch auf andere Aktoren, wie Motoren oder Relais übertragen.

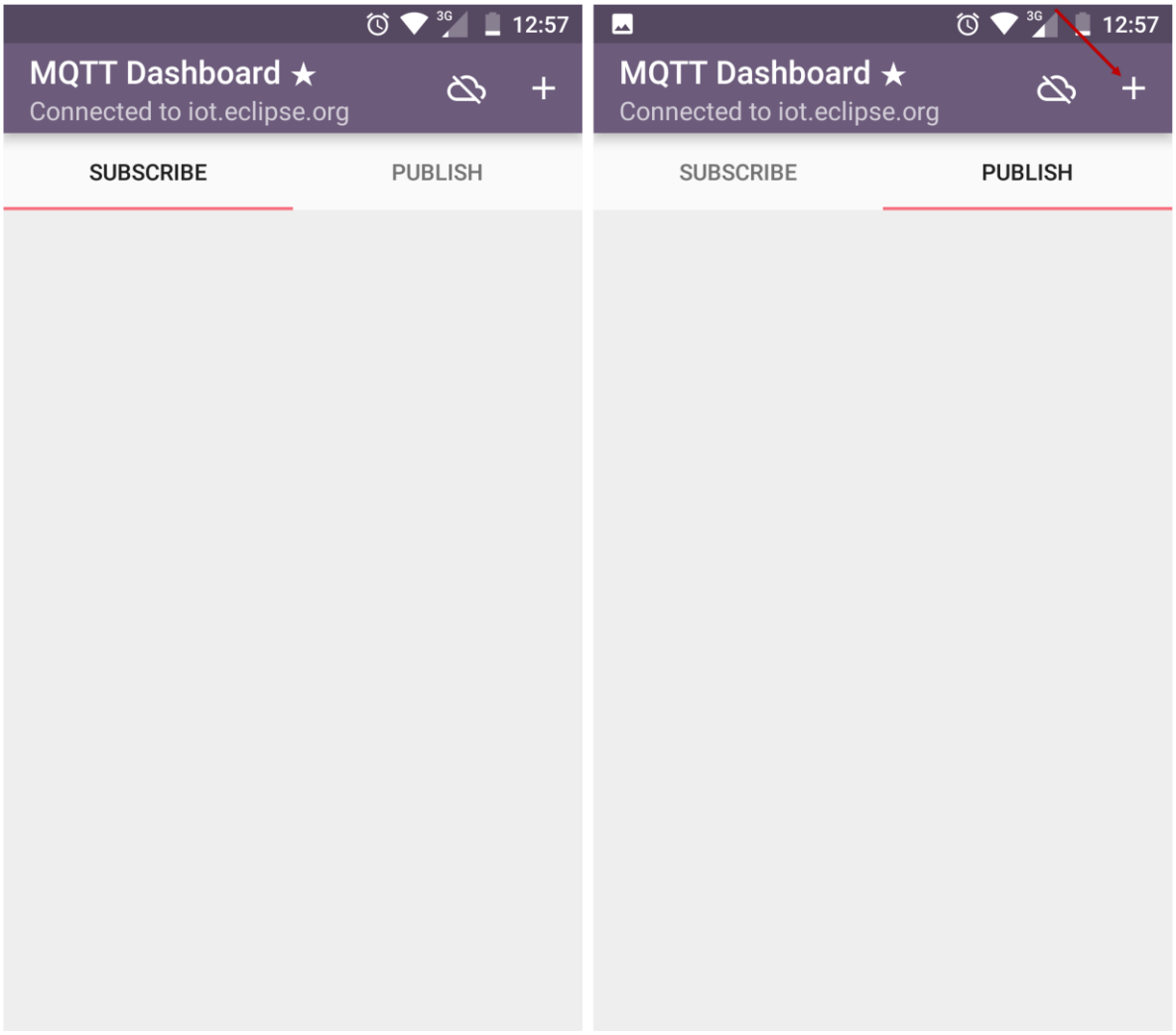
Es werden folgende Komponenten benötigt:

- 1x LED
- 1x 220 Ohm Widerstand
- 1x Steckbrett
- 2x Jumper Kabel
- 1x Android Smartphone mit Zugang zum Internet

Zunächst soll das Android Smartphone eingerichtet werden. Dazu wird die App IoT MQTT-Dashboard benötigt. Beim ersten Starten der App muss man dem Smartphone eine beliebige Client-ID zuweisen. Als Server wird iot.eclipse.org eingetragen. Dies ist ein kostenloser MQTT-Broker, welchen man über das Internet erreichen kann. Bei Bedarf kann auch ein eigener Broker, wie beispielsweise Mosquitto (https://mosquitto.org) auf dem Raspberry Pi eingerichtet werden. Als Port wird 1883 gewählt. Durch einen Klick auf *CREATE* oben rechts, werden die Servereinstellungen gespeichert.



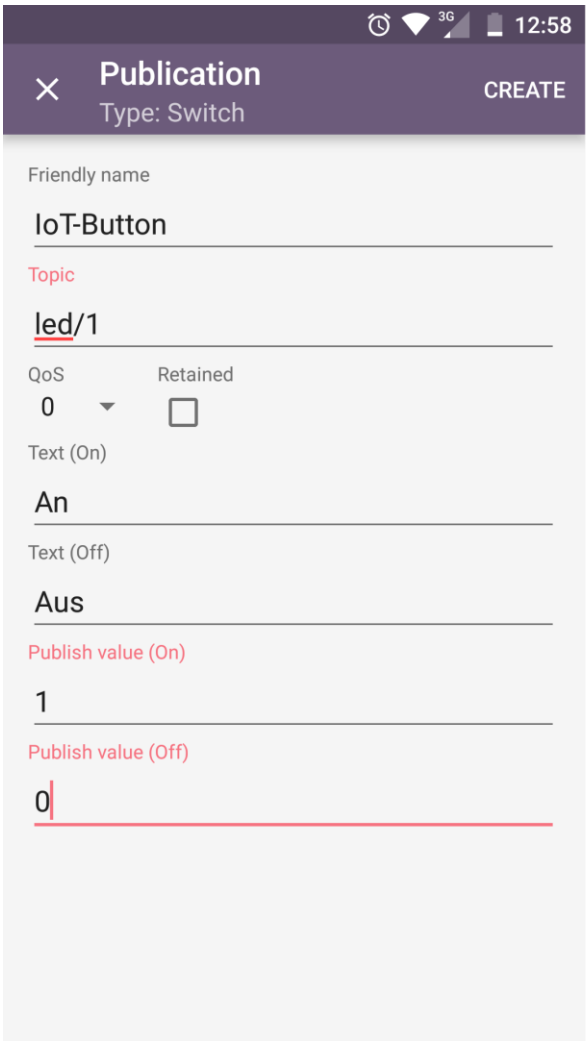
Im nächsten Schritt soll ein Schalter in der App eingerichtet werden. Dazu klickt man einfach auf E5-Raspi (Abbildung oben rechts) und gelangt in das nächste Menü (siehe unten).

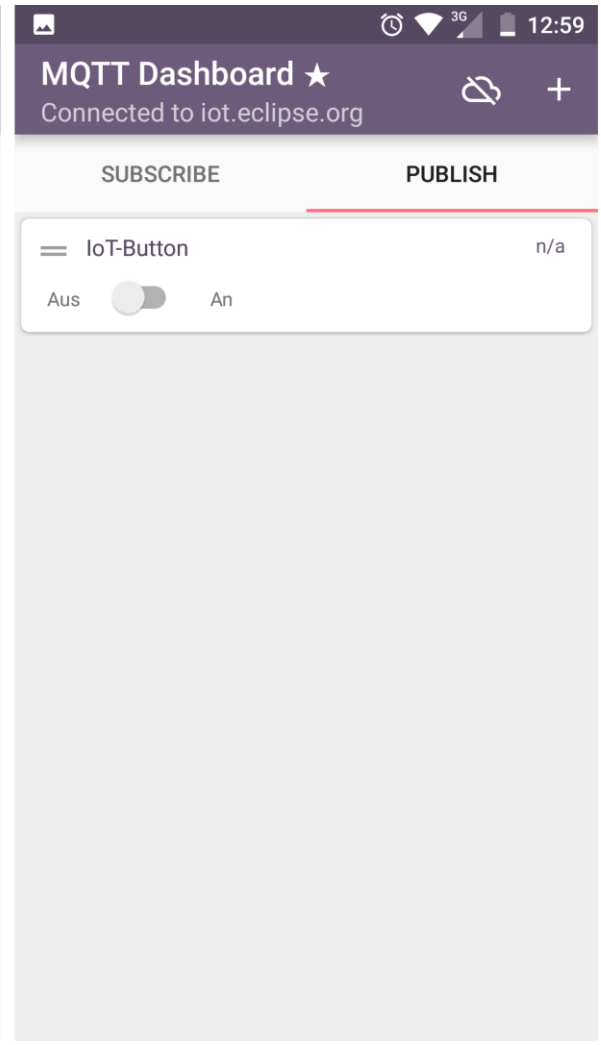


Man kann nun auswählen, ob man Nachrichten abonnieren (Abbildung links) oder veröffentlichen möchte (Abbildung rechts). Um Daten zu versenden, muss man zunächst festlegen, wie das Interface in der App aussehen soll. Dazu klickt auf das + Zeichen oben rechts in der rechten Abbildung. Es öffnet sich ein weiteres Menü, in welchem man die Art der Komponente auswählen muss. Für den oben definierten Anwendungsfall einer LED, die man an und aus schalten kann, eignet sich die Komponente *Switch*.

Zunächst gibt man der Komponente einen Namen, wie zum Beispiel IoT-Button. Als nächstes muss das Topic festgelegt werden. In diesem Beispiel soll das Topic led/1 lauten. Jeder Client, der dieses Topic abonniert, wird sofort benachrichtigt, falls neue Daten vorliegen. QoS kann vorerst auf 0 bleiben. Das bedeutet, dass der Broker genau ein einziges mal versucht, die Nachricht bei den Abonnenten zuzustellen. Die Checkbox *Retained* kann man unverändert lassen. Sollte man jedoch die Option *Retained* ausgewählt haben, bekommt jeder Client, der das Topic led/1 nachträglich abonniert hat, die zu letzt versendete Nachricht automatisch zugestellt und muss nicht warten, bis eine neue Nachricht mit dem Topic led/1 versendet wurde. Die Textfelder für Text (On) und Text (Off) sind optional. Im Gegensatz dazu sind die letzten beiden Textfelder essentiel, da in diesen die Nutzdaten bzw. der Payload beim An- und Ausschalten festgelegt wird. Um es übersichtlich zu halten, soll beim Anschalten der Wert 1 und beim Ausschalten der Wert 0 versendet werden.

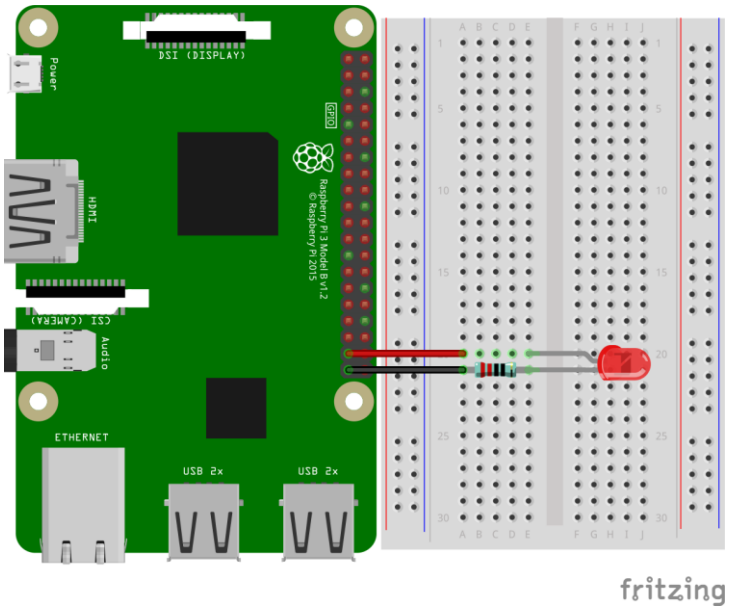
Wenn man die Konfiguration abgeschlossen hat, klickt man oben rechts auf *CREATE* und die Komponente sollte nun in der App zu sehen sein.





Für erste Tests kann man die Lernenden bitten das Topic led/1 mit der App zu abonnieren. Klickt man nun auf den Schalter, bekommen die Schülerinnen und Schüler eine 0 bzw. 1 auf ihr Smartphone zugesendet.

Nun muss der Raspberry Pi eingerichtet werden. Zunächst soll eine LED mit einem GPIO verbunden werden. **Achtung**: Raspberry Pi vorher von der Spannungsquelle trennen!



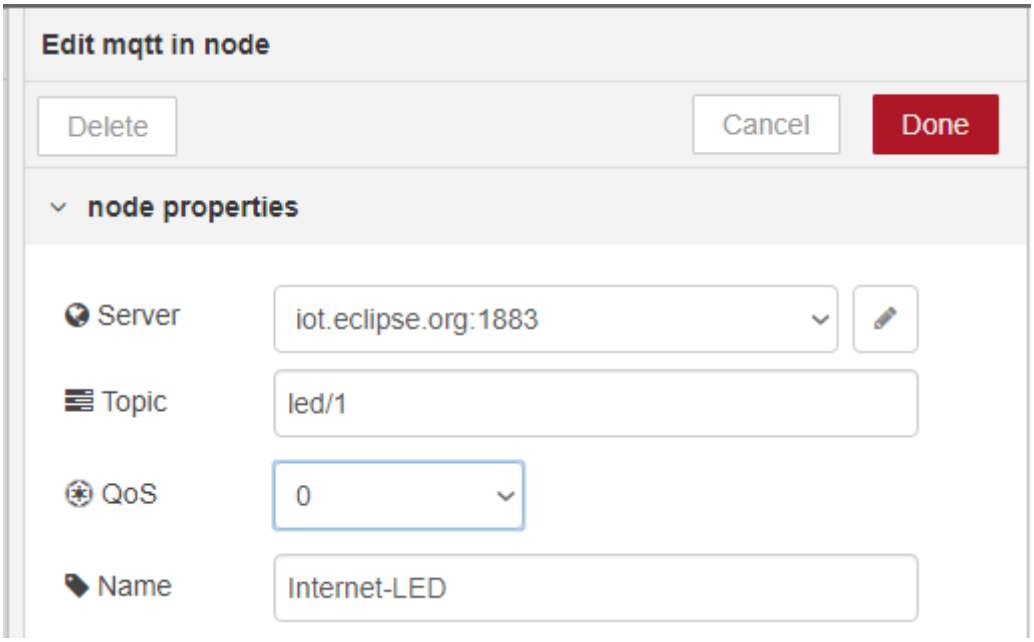
Die Anode (das lange Beinchen) wird mit GPIO 26 und die Kathode (kurzes Beinchen) wird über einen 220 Ohm Widerstand mit Masse verbunden. Sobald man GPIO 26 nun auf HIGH schaltet, leuchtet die LED auf. Wenn die Verkabelung abgeschlossen ist, kann der Raspberry Pi wieder angeschaltet werden.

Als nächstes müssen die MQTT Nachrichten des Smartphones empfangen und die LED entsprechend des Payloads geschaltet werden. Um dies zu realisieren, wird Node RED verwendet. Node RED ist ein Framework mit welchem man diverse Geräte und APIs, unabhängig vom Hersteller und Anbieter, mit einander verknüpfen kann um eigene Internet-of-Things-Projekte zu realisieren. So bietet die Software eine im Webbrowser aufrufbare Programmierumgebung, welche es dem Anwender ermöglicht Hardware, Schnittstellen, Webservices und vieles mehr graphisch miteinander zu verbinden. Mittels *Drag-and-Drop* kann man Knoten (*Nodes*) in die Programmieroberfläche hineinziehen und im Anschluss konfigurieren. Ein Knoten erfüllt eine bestimmte Aufgabe innerhalb des Programmablaufs (*Flow*), wie zum Beispiel das Empfangen von MQTT Nachrichten, das Anbieten von Webservices oder das Speichern von Daten in einer Datenbank. Um zwei Knoten mit einander zu verknüpfen, zieht man eine Verbindung (*Wire*) von einem Knotenausgang zu einem Knoteneingang. Auf der Projektwebseite ist ein informatives Video verlinkt, in welchem einige Funktionalitäten kurz erläutert und vorgeführt werden.

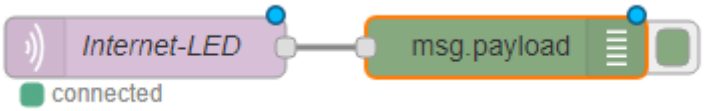
Um Node RED zu starten, öffnet man das Terminal, gibt node-red-pi ein und bestätigt mit der Enter Taste. Als nächstes startet man den Webbrowser und verbindet sich mit 127.0.0.1:1880. Nun sollte die Programmieroberfläche sichtbar sein.

Auf der linken Seite sieht man diverse Reiter, wie *input*, *function* oder *Raspberry Pi*. Dort befinden sich die Knoten, die man zum Programmieren benötigt. Um MQTT Nachrichten zu empfangen, zieht man den Knoten *mqtt* aus dem Reiter *input* in die Oberfläche. Mit einem Doppelklick kann der Knoten konfiguriert werden.

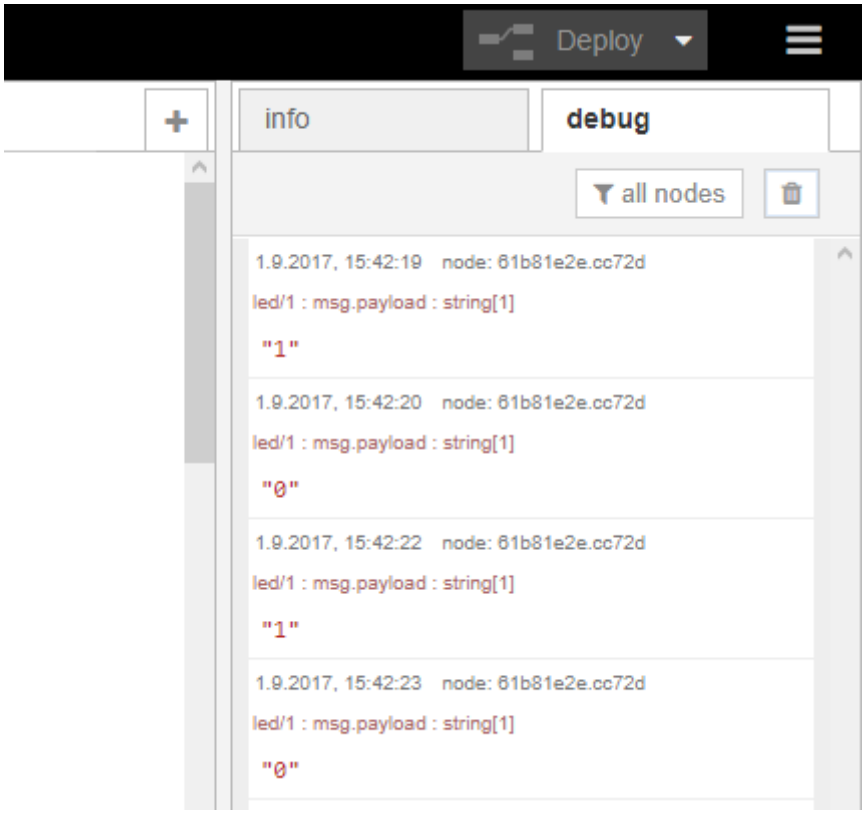
Als Server muss, wie beim Smartphone auch, iot.eclipse.org und der Port 1883 gewählt werden. Das Topic lautet, wie in der App festgelegt, led/1. Optional kann man dem Knoten einen Namen zuweisen. Mit einem Klick auf *Done* speichert man den Knoten.



Für einen schnellen Test, kann der Ausgang des MQTT Knoten mit dem Eingang des Debug Knoten verbunden werden. Der Debug Knoten befindet sich im Reiter *output.*



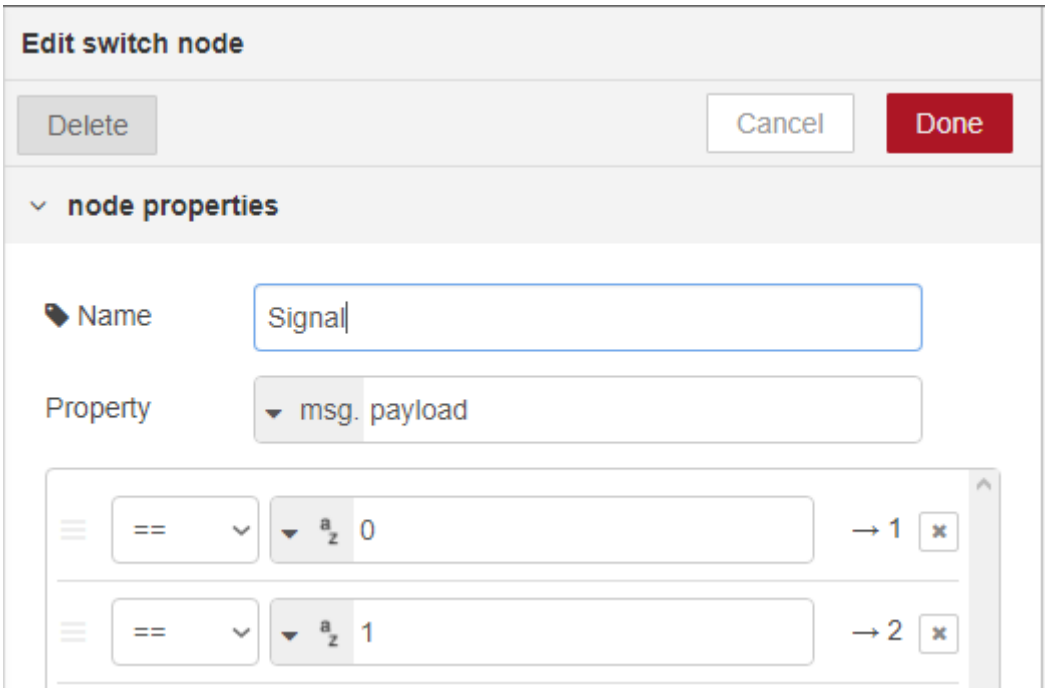
Mit einem Klick auf den roten Deploy Button oben rechts im Fenster, wird das Programm übernommen und die blauen Punkte oberhalb der Knoten verschwinden.



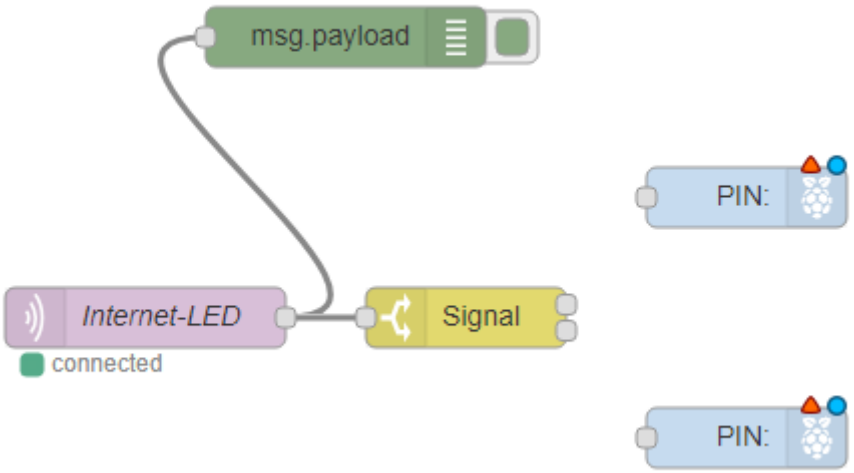
Wählt man nun oben rechts den Reiter *debug* aus und betätigt den Schalter in der Smartphone App, so sieht man, dass Nachrichten beim Raspberry Pi ankommen. Im nächsten Schritt werden diese ausgewertet.

Dafür verwendet man den Knoten *Switch* aus dem Reiter *function.* Dieser Knoten ermöglicht es die Nachricht, abhängig vom Payload auf unterschiedliche Ausgänge weiter zu leiten. Wenn die Nutzdaten eine 0 sind, dann wird die Nachricht an den Ausgang 1 (der obere Ausgang) weiter geleitet. Um weitere Regeln zuzufügen, klickt man im Konfigurationsmenü unten links auf *add.*

Die folgende Abbildung zeigt einen vollständig konfigurierten *Switch* Knoten. Der einen Klick auf *Done* wird die Konfiguration gespeichert. Wenn nun die MQTT Nachricht empfangen wird, prüft der *Switch* Knoten den Inhalt und leitet die Nachricht an einen der beiden Ausgänge weiter.



Als nächstes wird zwei mal der Knoten *rpi gpio* aus dem Reiter *Raspberry Pi* benötigt.



Das rote Dreieck zeigt an, dass die Knoten noch nicht konfiguriert sind. Dies wird nun geändert, indem man einen Doppelklick auf den unkonfigurierten Knoten tätigt. Im Konfigurationsmenü wählt man GPIO 26 aus und bestätigt im Anschluss mit *Done* oben rechts. Analog dazu geht man auch mit dem unteren Raspberry Pi Knoten um. Sobald beide Knoten konfiguriert sind, verbindet man diese mit dem Signal Knoten und betätigt den *Deploy* Button.