

Ein Servo per Arduino und Raspberry ansteuern

Autoren: Akouvi Sewovi, Shahed Zaidi

Inhaltsverzeichnis

| | |
|-------------------------------|----------|
| Ziel | 1 |
| Hardware | 1 |
| Material | 1 |
| Arduino | 1 |
| Ein Servo, drei Steckkabel | 2 |
| Raspberry Pi | 3 |
| Software | 4 |
| Development environment | 4 |
| Installation von Raspbian | 6 |
| Programmieren | 6 |
| Programmcode | 6 |
| Telegram-Bots | 8 |
| Erstellen eines Telegram-Bots | 8 |
| Anwendung | 10 |

1. Ziel

Ein Servo soll von einem Arduino-Mikrocontroller angesteuert werden. Der Servo soll dazu in folgenden Beispielen drei verschiedene Positionen ansteuern und zwischen den Positionen eine kurze Zeit warten und auch durch eine Nachricht von Telegram an den Bot in dieser Form `/set [Zahl]` wird der Servo in der drei verschiedenen Positionen angesteuert.

2. Hardware

2.1. Material

2.1.1. Arduino

Der „Arduino“ ist ein sogenanntes Mikrocontroller-Board (im weiteren Verlauf „Board“ genannt). Also im Grunde eine Leiterplatte (Board) mit jeder Menge Elektronik rund um den eigentlichen Mikrocontroller. Am Rand des Boards befinden sich viele Steckplätze (Pins genannt), an denen man die unterschiedlichsten Dinge anschließen kann. Dazu gehören: Schalter, LEDs, Ultraschallsensoren, Temperatursensoren, Drehregler, Displays, Motoren, Servos usw.

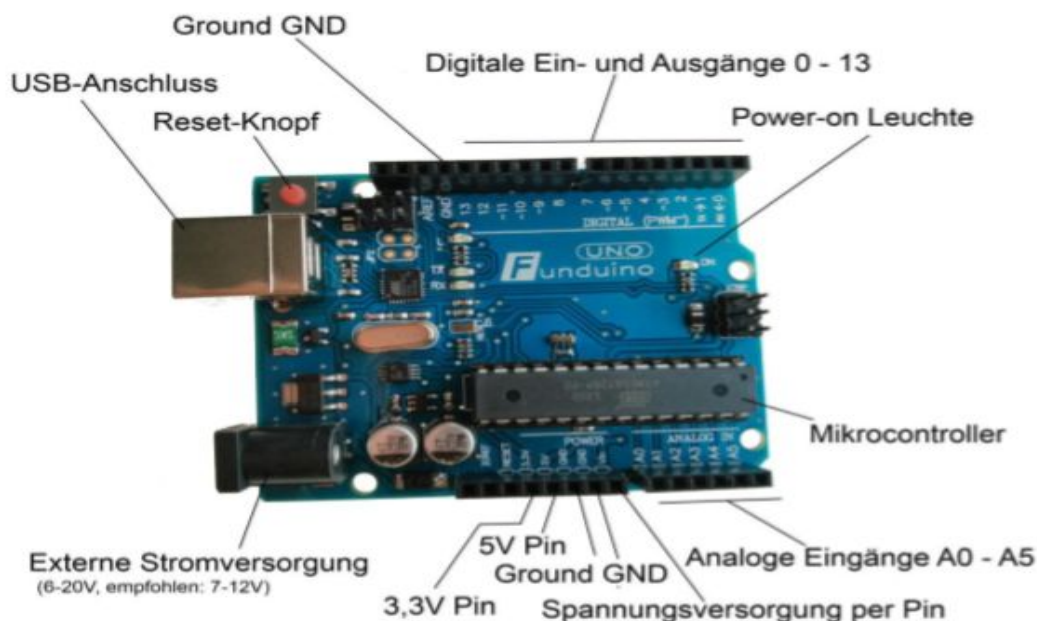


Abbildung 1

Ein Breadboard oder auch „Steckbrett“ ist ein gutes Hilfsmittel, um Schaltungen aufzubauen ohne zu löten. In einem Breadboard sind immer mehrere Kontakte miteinander verbunden. Daher können an

diesen Stellen viele Kabel miteinander verbunden werden, ohne dass sie verlötet oder verschraubt werden müssen.

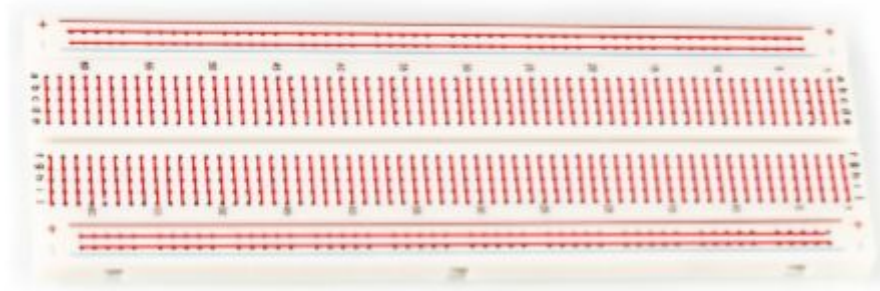


Abbildung 2: Das Breadboard

2.1.2. Ein Servo, drei Steckkabel

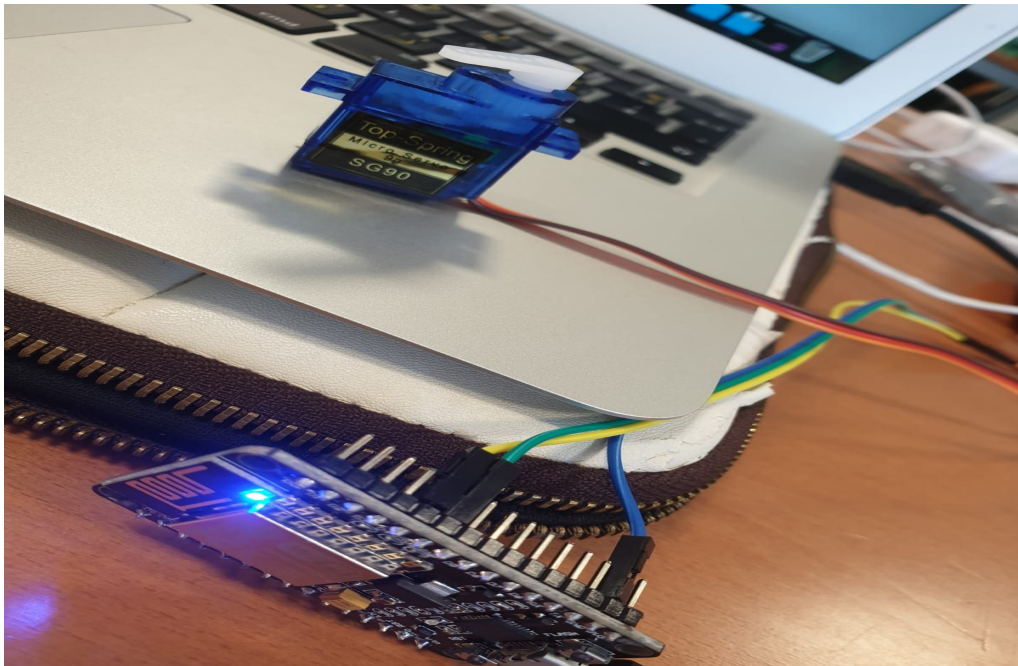


Abbildung 3 ein dreipoliges Kabel und Servo SG90

Anschluss : Gebräuchlich sind die Kombinationen

Orange → D4
Braun → GND
Rot → 3V3

2.1.3. Raspberry Pi

Ein Raspberry Pi wird benötigt für den Datenaustausch per MQTT

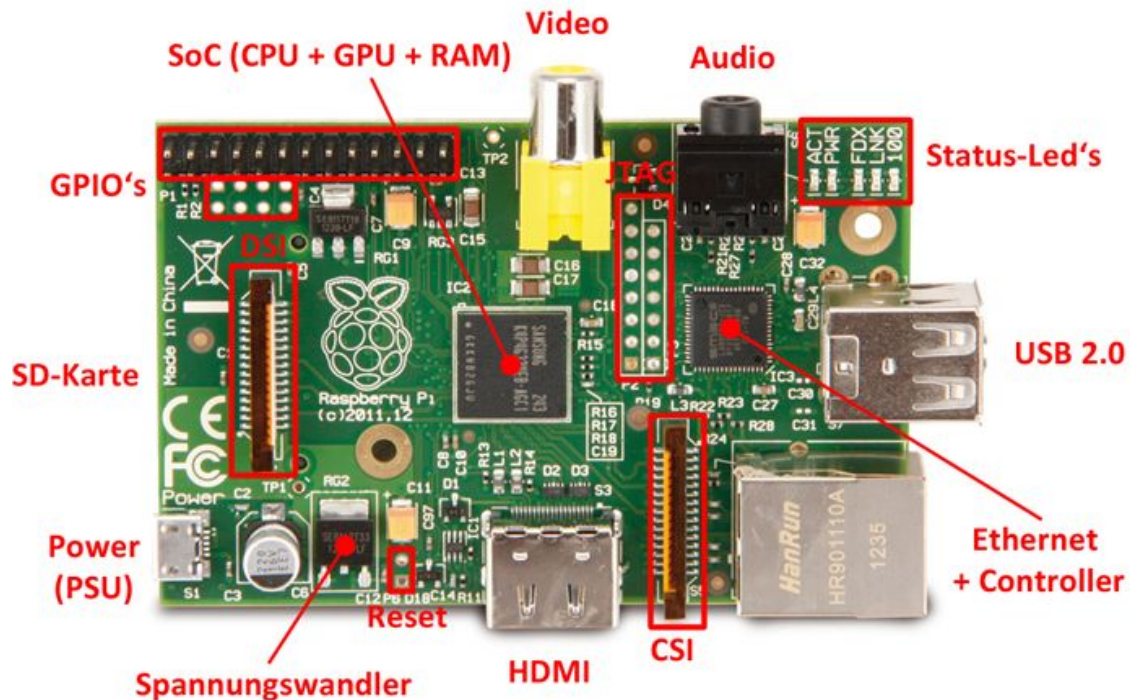


Abbildung 4:Raspberry Aufbau

3. Software

3.1. Development environment

- Arduino IDE Plattform mit Module Esp8266

Zuerst müssen wir Visual Studio Code installieren, über Extensions können wir das Open-Source-Projekt Platform IO die kostenlose Alternative zur Arduino-IDE für die Microcontroller-Entwicklung auch installieren. (Abbildung 5)

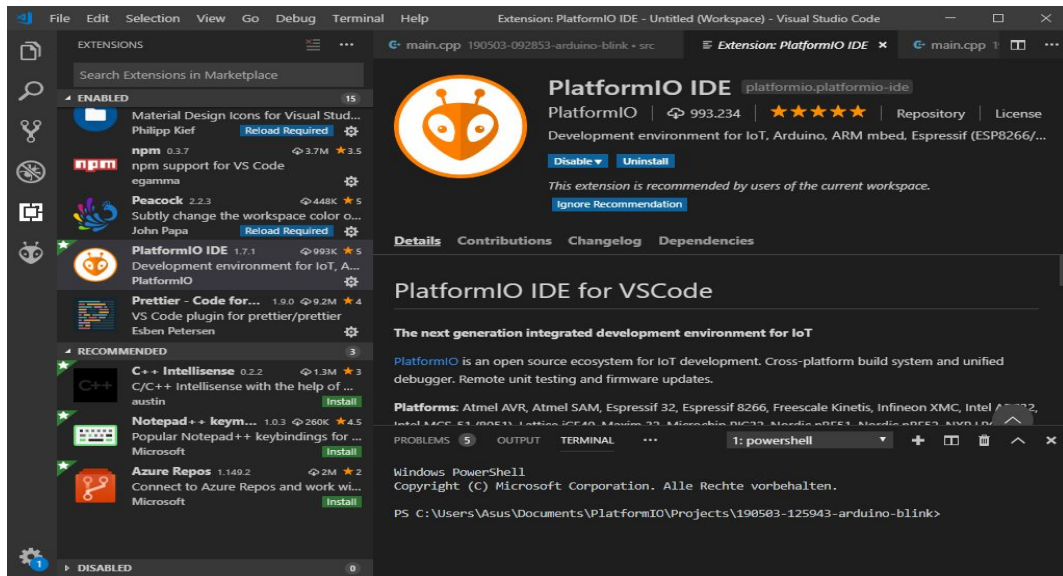


Abbildung 5 Installation

- Projektstruktur

Um möglichst schnell mit funktionsfähigem Code zu starten, wählen wir mit einem Klick auf PROJECT EXAMPLES ein Beispiel aus. PlatformIO hat Beispiele für verschiedene Plattformen. In der Kategorie ESP8266 wählen wir “arduino-blink” aus. (Abbildung 6)

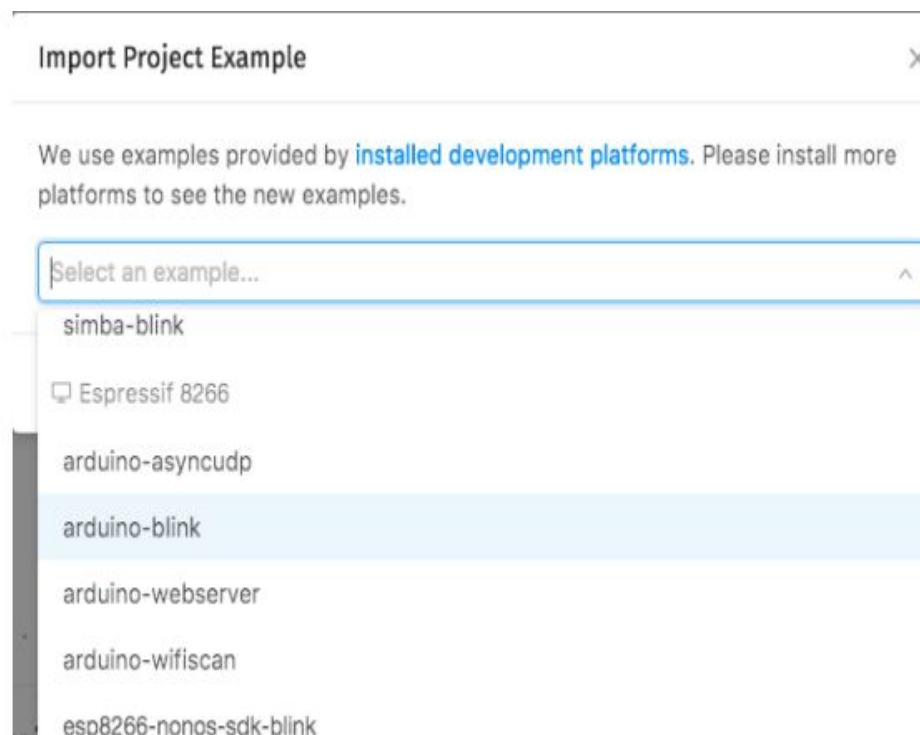


Abbildung 6: Projektbeispiel auswählen

3.2. Installation von Raspbian

Der Link zur Installation des Betriebssystems:

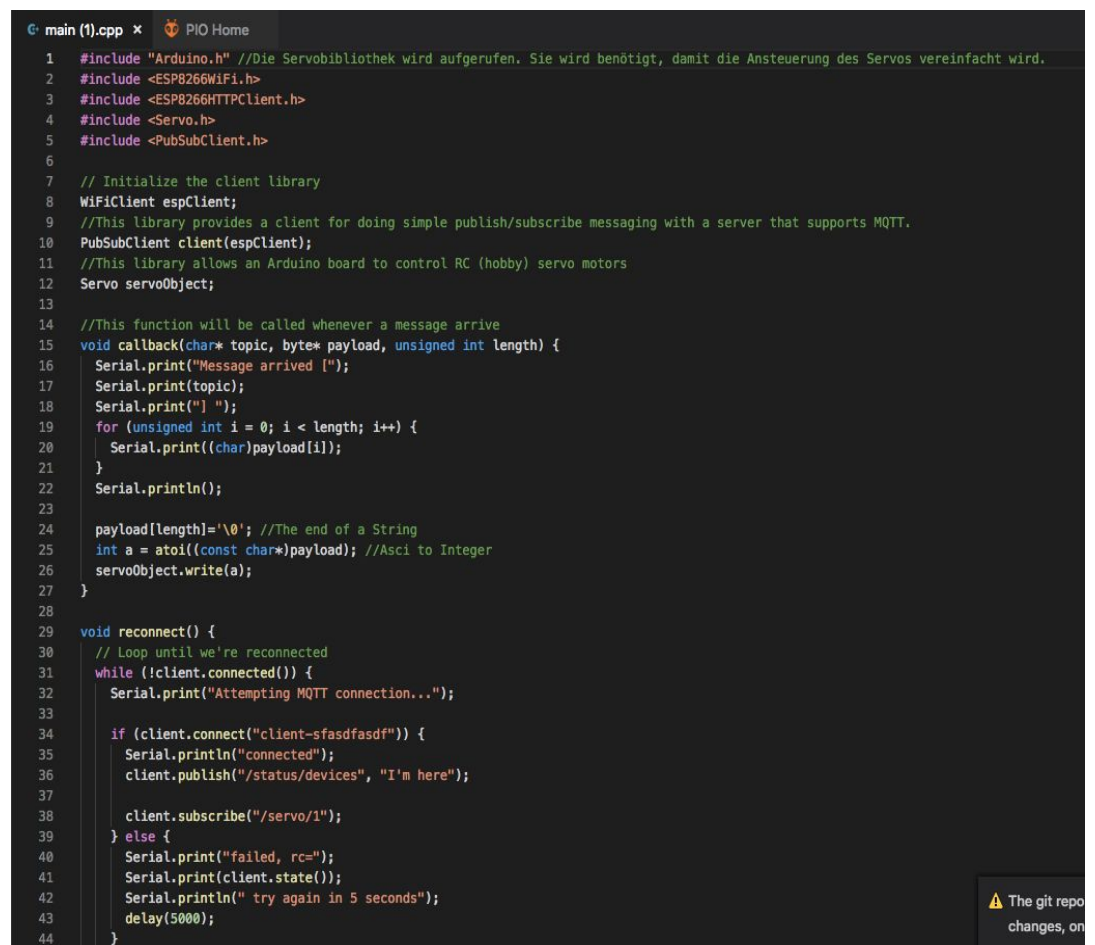
https://downloads.raspberrypi.org/raspbian_lite_latest

Für Windows - Benutzer müssen wir Putty installieren um Raspberry zu konfigurieren.

4. Programmieren

4.1. Programmcode

Die Main.cpp Datei im Projekt, den wir oben erstellt haben, soll so aussehen, damit wir unser Servo ansteuern können. Siehe Datei (Main(1).cpp).



```
1 #include "Arduino.h" //Die Servobibliothek wird aufgerufen. Sie wird benötigt, damit die Ansteuerung des Servos vereinfacht wird.
2 #include <ESP8266WiFi.h>
3 #include <ESP8266HTTPClient.h>
4 #include <Servo.h>
5 #include <PubSubClient.h>
6
7 // Initialize the client library
8 WiFiClient espClient;
9 //This library provides a client for doing simple publish/subscribe messaging with a server that supports MQTT.
10 PubSubClient client(espClient);
11 //This library allows an Arduino board to control RC (hobby) servo motors
12 Servo servoObject;
13
14 //This function will be called whenever a message arrive
15 void callback(char* topic, byte* payload, unsigned int length) {
16     Serial.print("Message arrived ");
17     Serial.print(topic);
18     Serial.print(" ");
19     for (unsigned int i = 0; i < length; i++) {
20         Serial.print((char)payload[i]);
21     }
22     Serial.println();
23
24     payload[length]='\0'; //The end of a String
25     int a = atoi((const char*)payload); //Ascii to Integer
26     servoObject.write(a);
27 }
28
29 void reconnect() {
30     // Loop until we're reconnected
31     while (!client.connected()) {
32         Serial.print("Attempting MQTT connection...");
33
34         if (client.connect("client-sfasdfasdf")) {
35             Serial.println("connected");
36             client.publish("/status/devices", "I'm here");
37
38             client.subscribe("/servo/1");
39         } else {
40             Serial.print("failed, rc=");
41             Serial.print(client.state());
42             Serial.println(" try again in 5 seconds");
43             delay(5000);
44         }
45     }
46 }
```

Abbildung 7: Code

```

47
48  /*
49  *Das Setup enthält die Information, dass das Servo an der Steuerleitung mit Pin 2 verbunden wird.
50  *Hier ist natürlich auch ein anderer Pin möglich.
51  */
52  void setup() {
53      servoObject.attach(2);
54      servoObject.write(0);
55      delay(2000); // Das Programm stopt für 2 Sekunden
56      Serial.begin(9600);
57      Serial.println("Hallo Welt");
58      //Setting WIFI mode to STA
59      WiFi.mode(WIFI_STA);
60      //Setting the host name
61      WiFi.hostname("dpsasdf");
62      /*
63      Connecting to WIFI
64      *username : hsb-labor
65      *password : 6MVfNSqdMr5Szo6d
66      */
67      WiFi.begin("hsb-labor", "6MVfNSqdMr5Szo6d");
68      //While we're not connected , delay and try again
69      while (WiFi.status() != WL_CONNECTED) {
70          delay(500);
71          Serial.print(".");
72      }
73      //Print the IP of the arduino
74      Serial.println(WiFi.localIP());
75      //Set the server and the callback function
76      // This IP-Address is IP of the Raspbery
77      client.setServer("192.168.206.19", 1883);
78      client.setCallback(callback);
79  }
80  //This function will keep try to connect to the Arduino
81  /*Im „loop“ wird über den write-Befehl „servoblau.write(Grad)“ das Servo angesteuert.
82  *Zwischen den einzelnen Positionen gibt es eine Pause, damit das Servo genug Zeit hat,
83  *die gewünschten Positionen zu erreichen.
84  */
85  void loop() {
86      if (!client.connected())
87          reconnect();
88      client.loop();
89  }
90

```

Abbildung 8: Code

4.2. Telegram-Bots

4.2.1. Erstellen eines Telegram-Bots

In Telegram Web "<https://web.telegram.org/>" müssen wir nach "BotFather " suchen.

Durch "BotFather" kann man alle bots editieren , löschen , erstellen.

Es wird /start als Nachrichten an BotFather gesendet, indem die Interaktion mit dem Benutzer begonnen hat und man die mögliche Befehle hat, kann man mit der Erstellung einen Bot mit dem Befehl /newbot beginnen.

Man wird nach Name und Benutzername nachgefragt.

Zum Schluss erhält man ein Token, das von Node-RED für den Zugriff auf das Telegramm verwendet wird.(Abbildung 9)

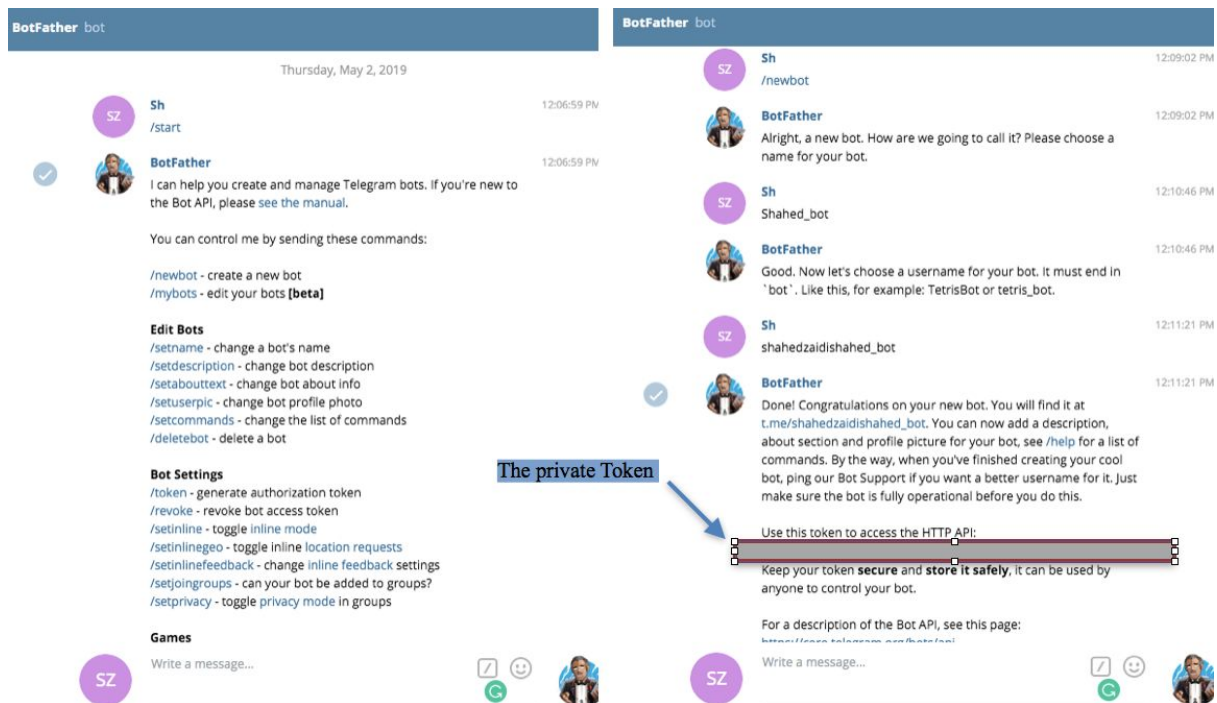


Abbildung 9

- Telegramm-Plugin auf Node-RED installieren:
im Node-RED wenn man auf das Menüsymbol in der rechten oberen Ecke und dann auf "Palette verwalten" klickt, kann man nun nach "telegrambot" suchen und installieren, danach werden 5 weitere Nodes über die linken Palette hinzugefügt.
- Konfiguration eines Bot:

Es wird einen "Telegram receiver node" zu dem "Flow" hinzugefügt und darauf geklickt. Als nächstes gibt man die "Bot-Namen" und der private "Token" ein.
Abbildung(10)



Abbildung 10

in Telegram wird nochmal eine Gruppe mit dem bot erzeugt.(Abbildung 11)



Abbildung 11

4.2.2. Anwendung

Es wird in Node RED diese Nodes wie auf dem Bild erzeugt : (Abbildung 12)

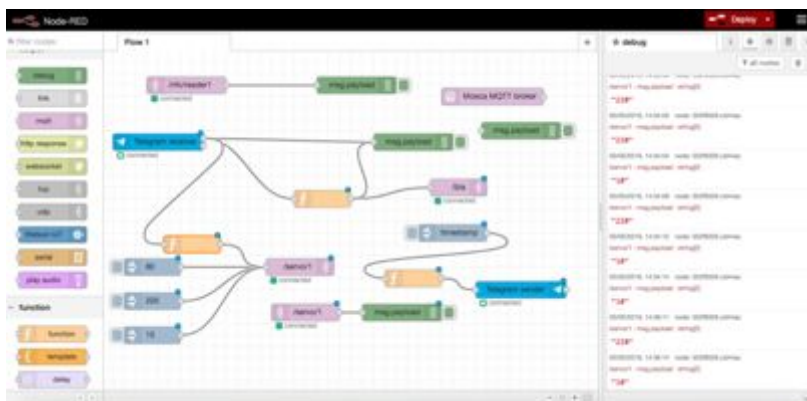


Abbildung 12

Für unsere Aufgabe wurden die folgenden Nodes gebraucht:

debug, Inject, mqtt von output und input, Mosca MQTT broker und function.

- Debug: Zeigt Nachrichten, Fehler und Warnungen rechts im Debug-Bereich an.
- Mqtt von output: Ist ein MQTT-Subscriber eines definierten MQTT-Brokers. Veröffentlicht 'msg.payload' der eingehenden Nachrichten auf einem definierten Topic.
- Mqtt von input: Ist ein MQTT-Subscriber eines definierten MQTT-Brokers und lauscht auf einem definierten Topic. Gibt alle Daten, die auf jenem Topic veröffentlicht werden, in einer neuen Nachricht zurück.
- Function: Hier kann eigener JavaScript-Code stehen. Um den Flow fortzusetzen, sendet man mit 'return' eine oder mehrere Nachrichten an den

Ausgang. Um den Nachrichtenfluss zu unterbrechen, verwendet man 'return null'. Nodes ignorieren null-Nachrichten.

- Inject : Sendet als 'msg.payload' beispielsweise einen Zeitstempel oder einen benutzerdefinierten Text. Die Nachricht kann auf Mausklick, in bestimmten Intervallen oder zu definierten Zeiten in den Flow gespeist werden. Inject Nodes dienen somit als Trigger für den Flow.
- Mqtt In: Ist ein MQTT-Subscriber eines definierten MQTT-Brokers und lauscht auf einem definierten Topic. Gibt alle Daten, die auf jenem Topic veröffentlicht werden, in einer neuen Nachricht zurück.

Dadurch konnten wir entweder:

1. bei den Inject Node (80 , 220 oder 10) klicken , dann dreht der Servo (80 , 220 oder 10 grad)



Abbildung 13

2. Oder durch eine Nachricht in Telegramm an den Bot in dieser Form “/set 90”, die in der folgenden Funktion Node definiert ist :

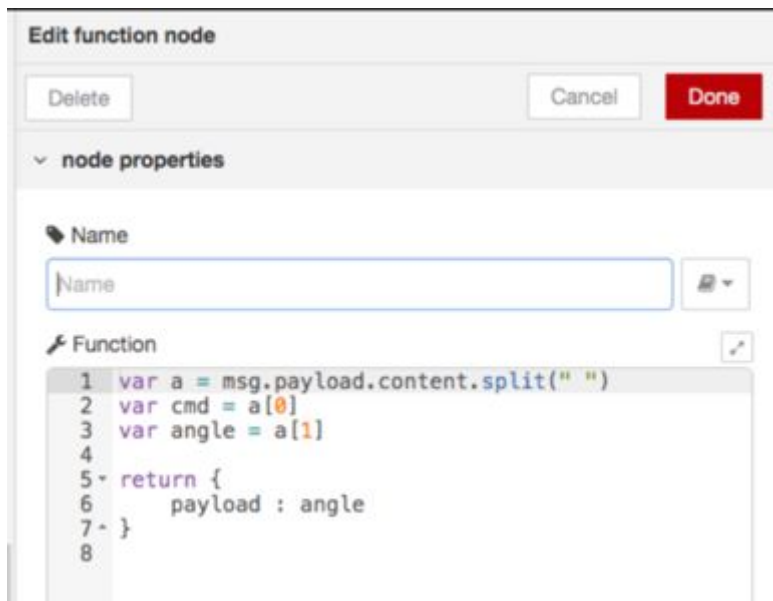


Abbildung 14

In der Funktion erzeugt man einen Variable "a", der aus dem Inhalt von dem Telegramm Nachricht besteht, "angle" ist der zweite teil von der Nachricht (der Grad vom Winkel) , in unserem Beispiel ist 90. Es wird diese Wert zurückgegeben, die der Servo nimmt als Grad.

Quellenverzeichnis

<http://raspberrypiguide.de/>

<https://farberg.de/talks/hb-iot-2019/#/agenda>

<https://entwickler.de/online/iot/node-red-iot-prototypen-2-579809637.html>