

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. Zudem soll der Servo durch eine Nachricht von dem Telegram an den Bot in dieser Form `"/set [Zahl]"` in den 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). Im Grunde ist es eine Leiterplatte (Board) mit jeder Menge Elektronik rund um den eigentlichen Mikrocontroller. Am Rand des Boards befinden sich viele Steckplätze (Pins genannt), an denen die unterschiedlichsten Sensoren angeschlossen werden können. Dazu gehören: Schalter, LEDs, Ultraschallsensoren, Temperatursensoren, Drehregler, Displays, Motoren, Servos usw.

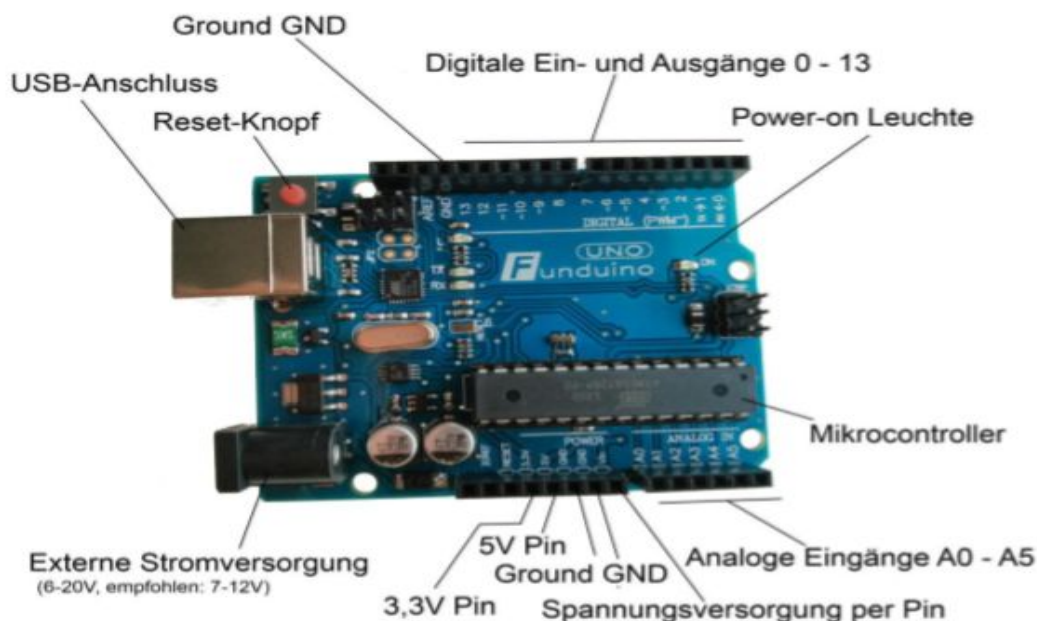


Abbildung 1

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.

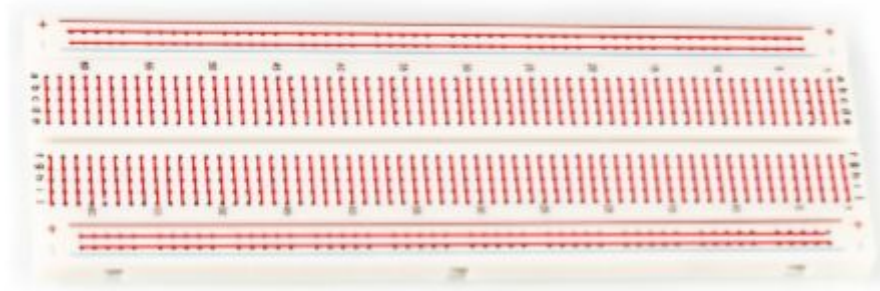


Abbildung 2: Das Breadboard

2.1.2. Ein Servo, drei Steckkabel

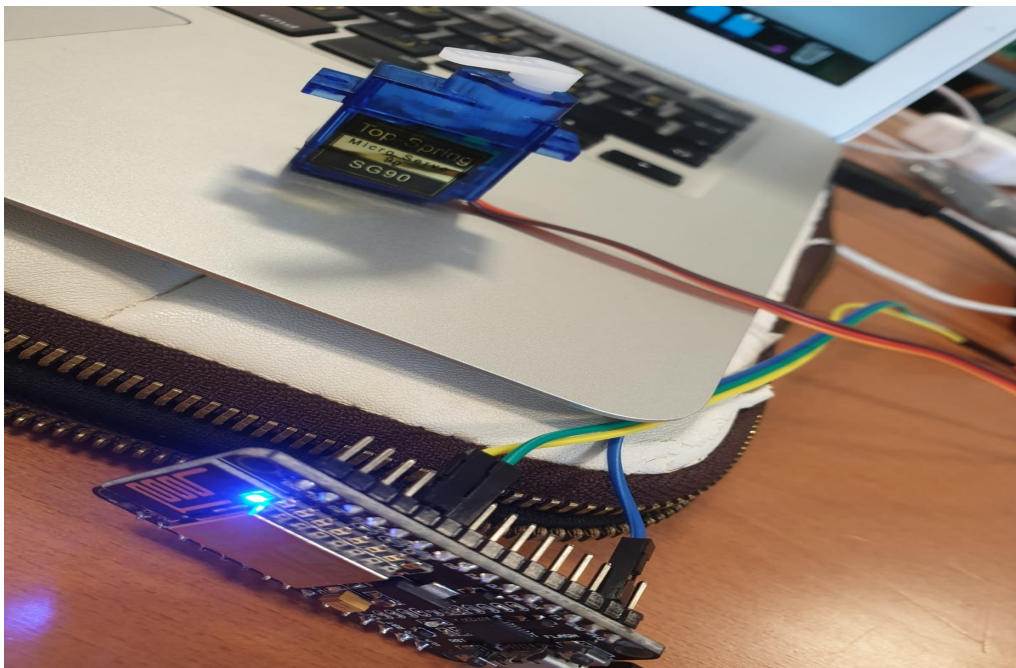


Abbildung 3: Dreipoliges Kabel und Servo SG90

Anschluss : Gebräuchlich sind die Kombinationen

Orange → D4
Braun → GND
Rot → 3V3

2.1.3. Raspberry Pi

Ein Raspberry Pi wird für den Datenaustausch per MQTT benötigt

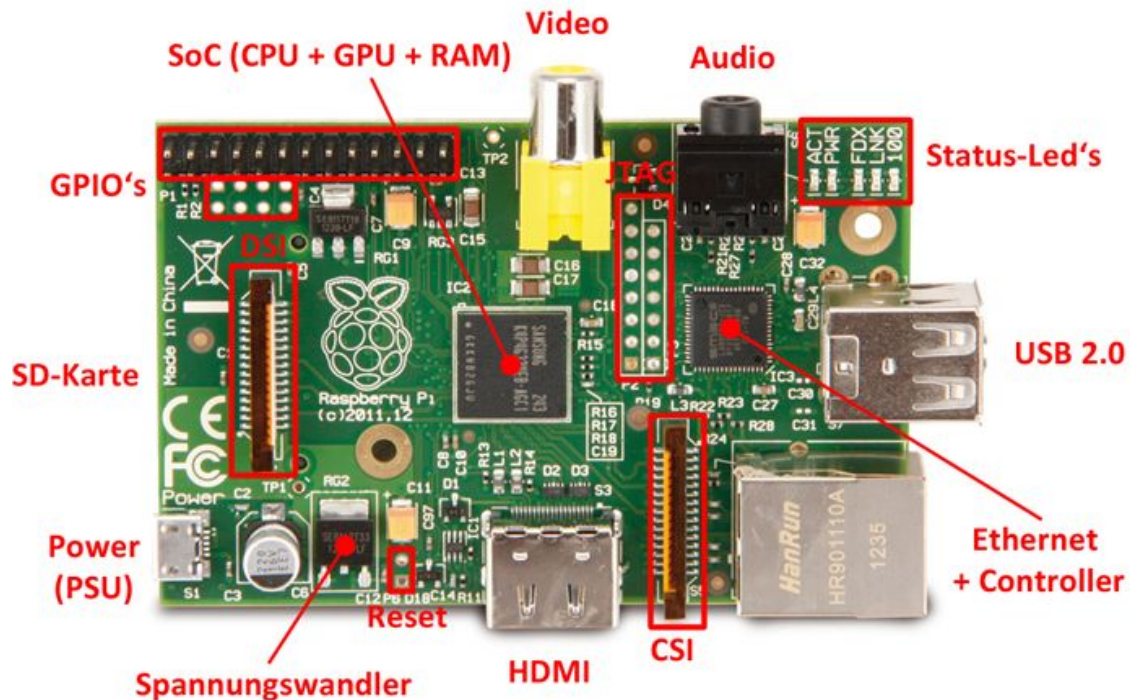


Abbildung 4:Raspberry Aufbau

3. Software

3.1. Development environment

- Arduino IDE Plattform mit Module Esp8266

Zunächst musste Visual Studio Code installiert werden, über Extensions wurde anschließend die kostenlose Alternative des Open-Source-Projekt Plattform IO für die Microcontroller-Entwicklung installiert.(siehe Abbildung 5)

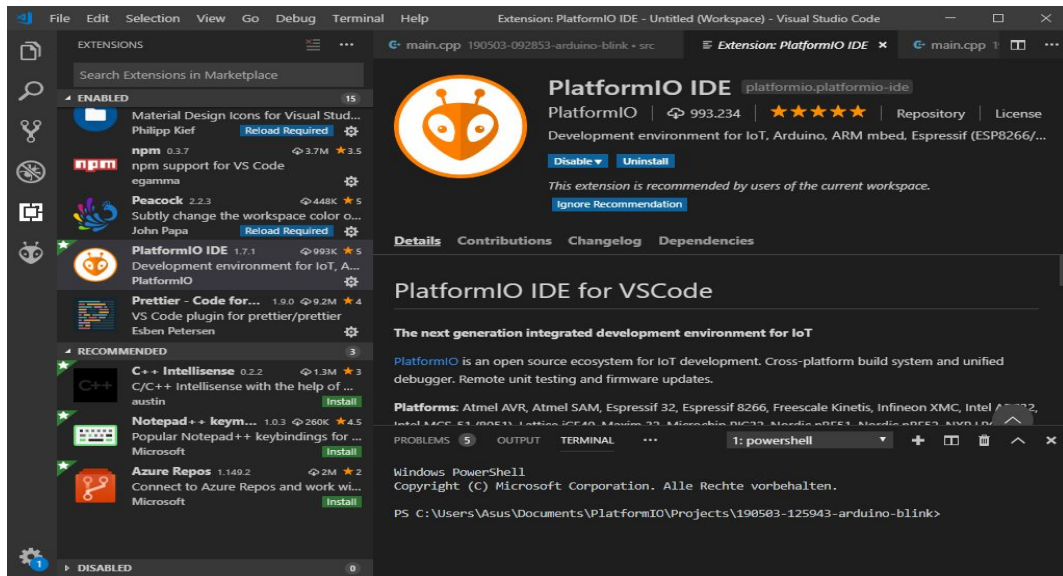


Abbildung 5 Installation

- Projektstruktur

Um möglichst schnell mit funktionsfähigem Code zu starten, wurde auf PROJECT EXAMPLES ein Beispiel ausgewählt. PlatformIO hat Beispiele für verschiedene Plattformen. In der Kategorie ESP8266 wurde anschließend “arduino-blink” ausgewählt. (siehe Abbildung 6)

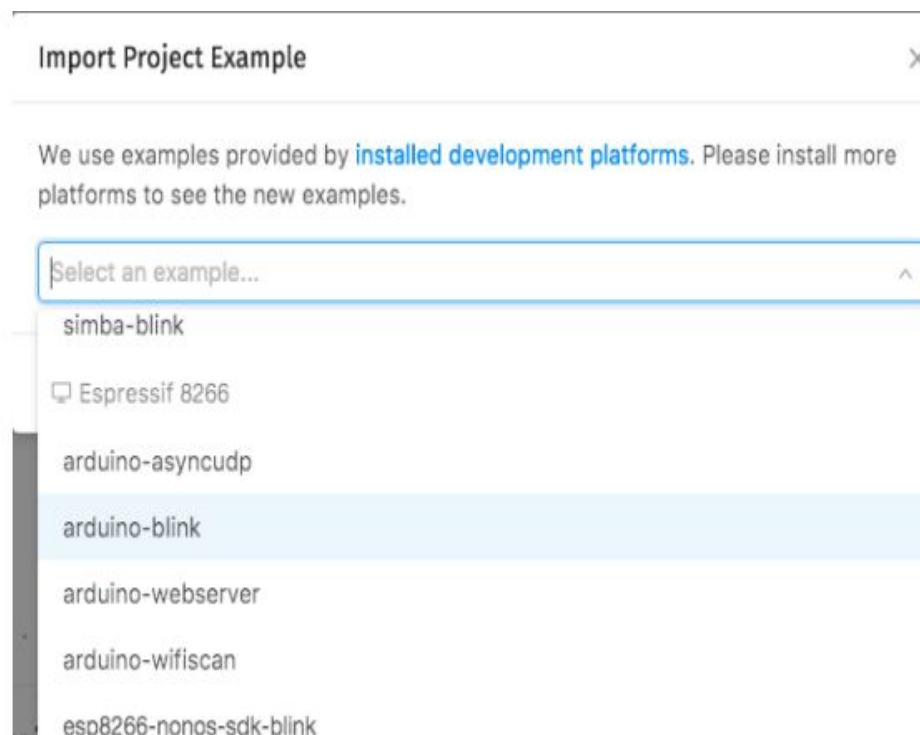


Abbildung 6: Projektbeispiel auswählen

3.2. Installation von Raspbian

Der Link zur Installation des Betriebssystems:

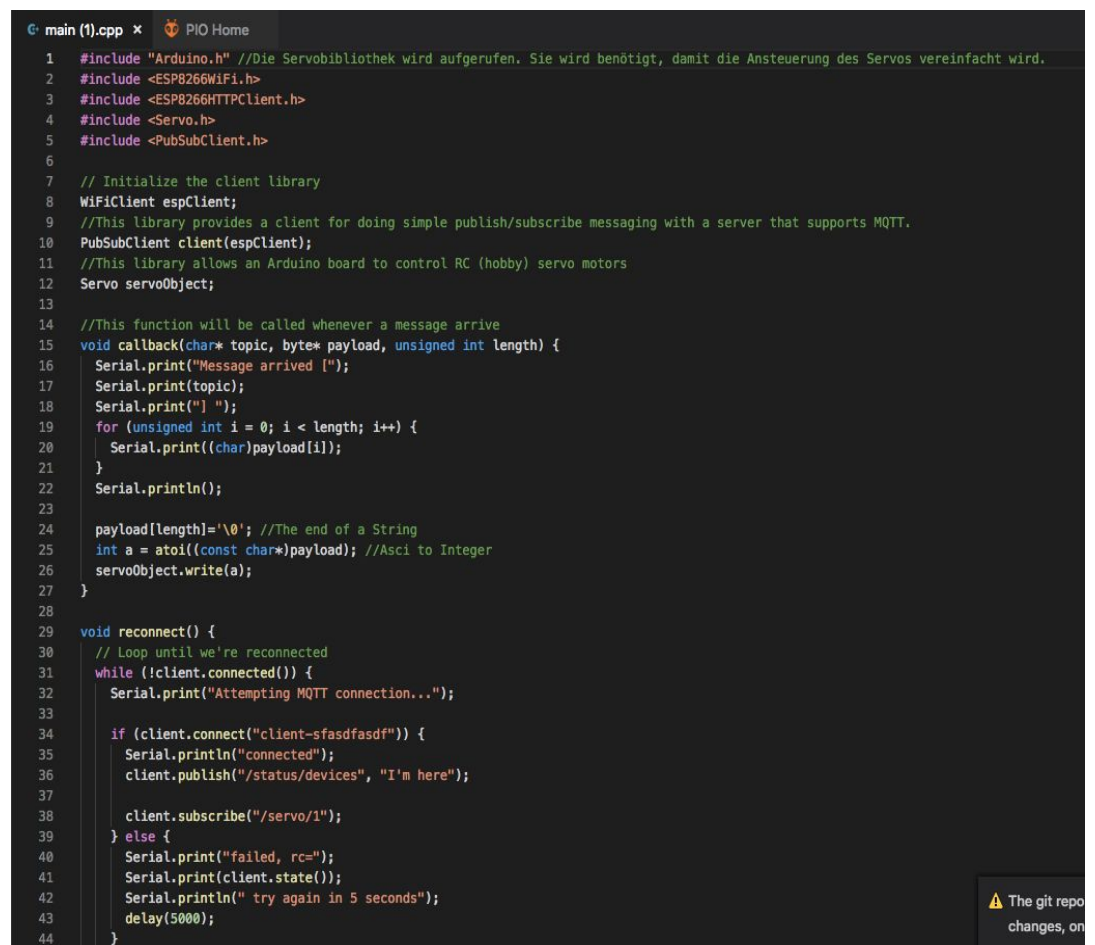
https://downloads.raspberrypi.org/raspbian_lite_latest

Die Windows - Benutzer müssen zunächst Putty installieren um Raspberry konfigurieren zu können.

4. Programmieren

4.1. Programmcode

Die Main.cpp Datei im Projekt, die erstellt wurde, sollte wie in der Abbildung aufgebaut sein, damit der Servo angesteuert werden kann. 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: Main(1).cpp

```

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: Main(1).cpp

4.2. Telegram-Bots

4.2.1. Erstellen eines Telegram-Bots

In Telegram Web "<https://web.telegram.org/>" musste nach "BotFather " gesucht werden. Durch "BotFather" können alle bots editiert ,gelöscht und erstellt werden. Es wurde /start als Nachricht an BotFather gesendet, indem die Interaktion mit dem Benutzer begonnen hat und die möglichen Befehle vorhanden sind, kann mit der Erstellung eines Bots mit dem Befehl /newbot begonnen werden.

Es wird nach Name und Benutzername gefragt.

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

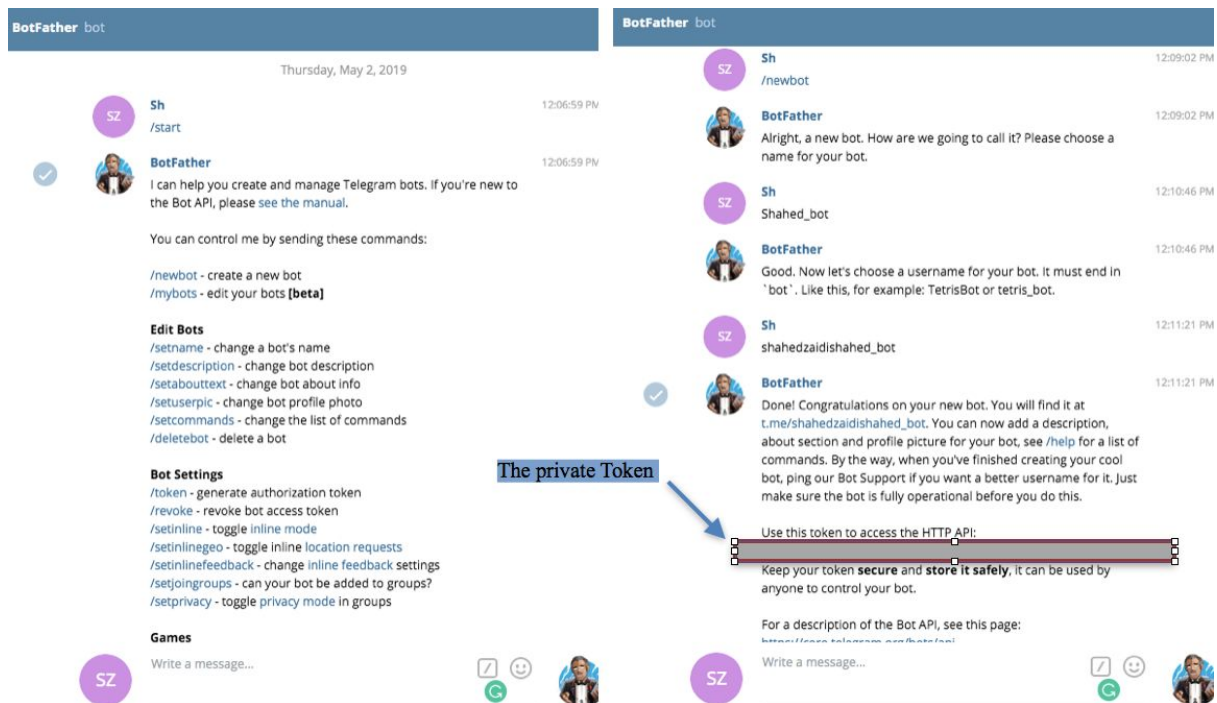


Abbildung 9

- Telegram-Plugin auf Node-RED installieren:
Über das Menüsymbol im Node-RED, was sich in der rechten oberen Ecke befindet wird "Palette verwalten" gewählt. Hier wird nach "telegrambot" gesucht und dieses installiert. Anschließend mussten fünf weitere Nodes über die linke Palette hinzugefügt werden.
- Konfiguration eines Bot:

Es wurde ein "Telegram receiver node" zu dem "Flow" hinzugefügt. Als nächstes wird der "Bot-Namen" und der private "Token" eingegeben. (siehe Abbildung 10)

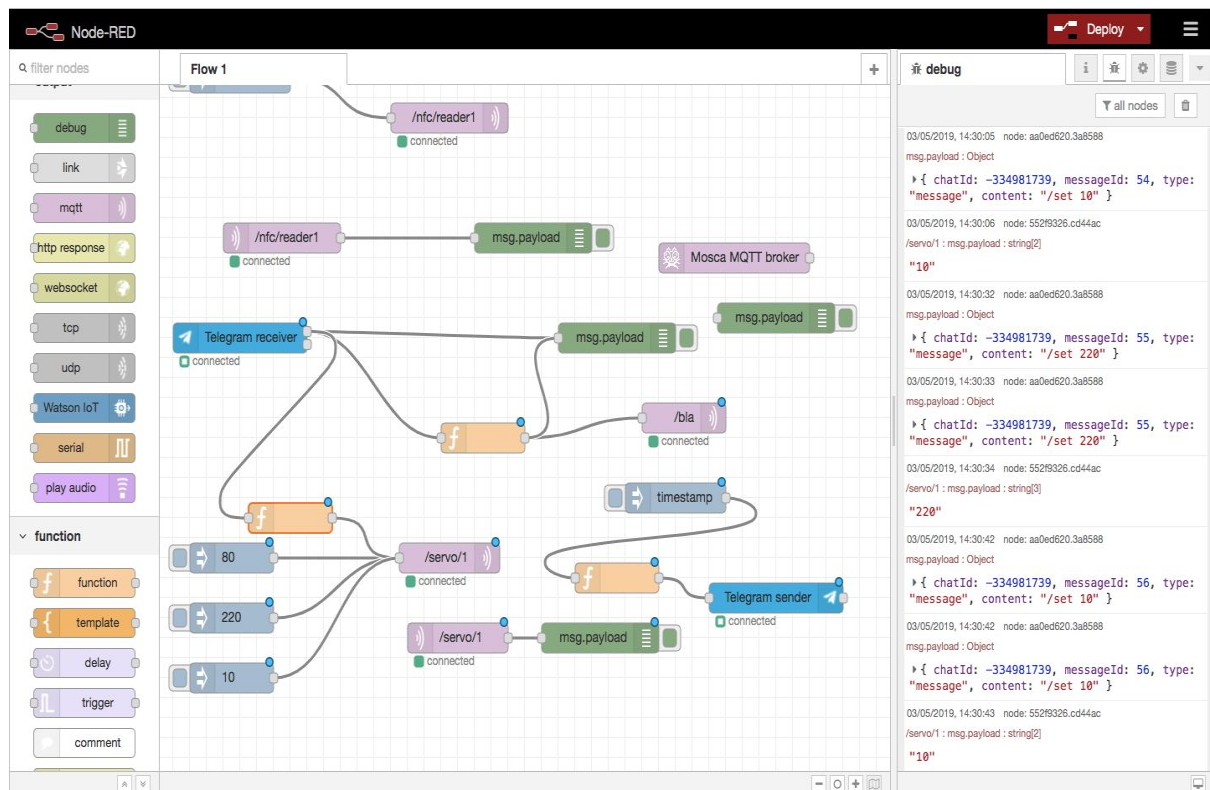


Abbildung 10

In Telegram wurde nochmal eine Gruppe mit dem Account “botfather” erzeugt.(Abbildung 11)

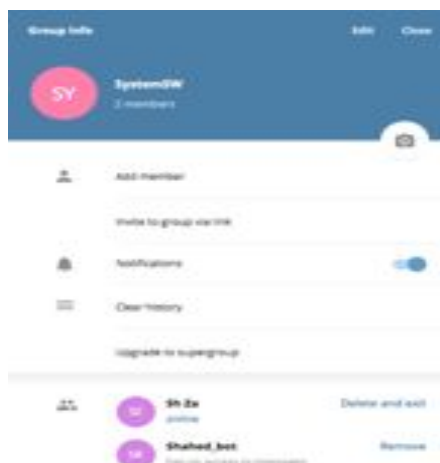


Abbildung 11

4.2.2. Anwendung

In Node RED wurden Nodes wie in der Abbildung zu sehen erzeugt. (siehe Abbildung 12)

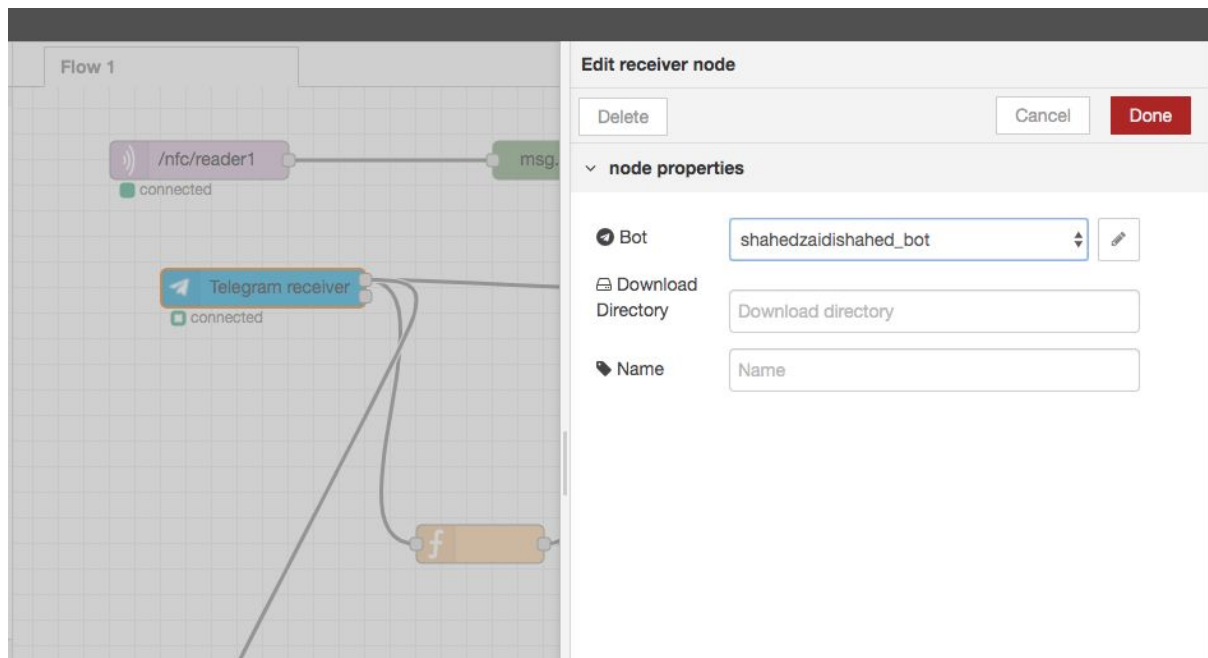


Abbildung 12

Für unsere Aufgabe wurden die folgenden Nodes benötigt:
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, wird 'return null' verwendet. 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 konnte entweder:

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

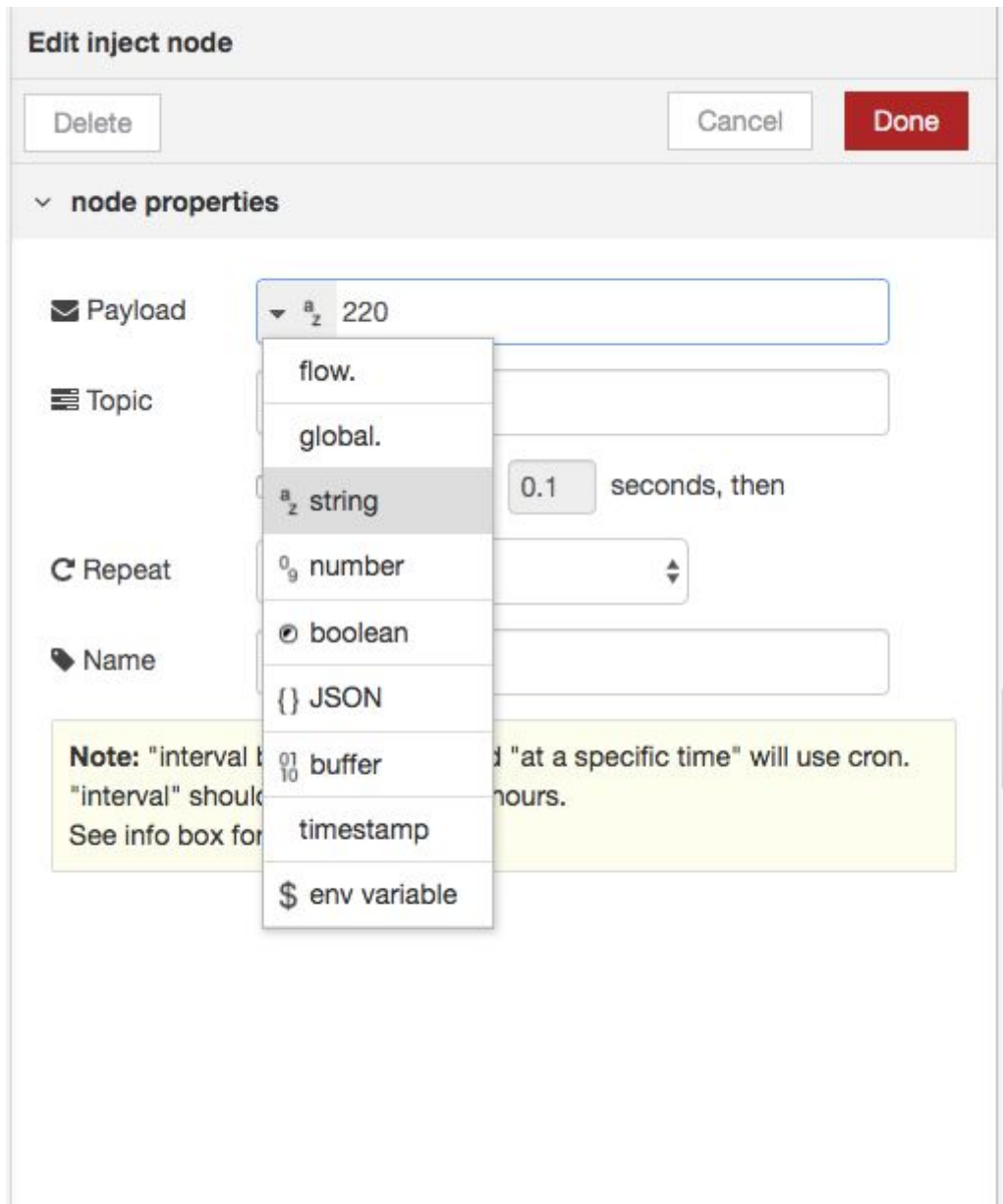


Abbildung 13

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

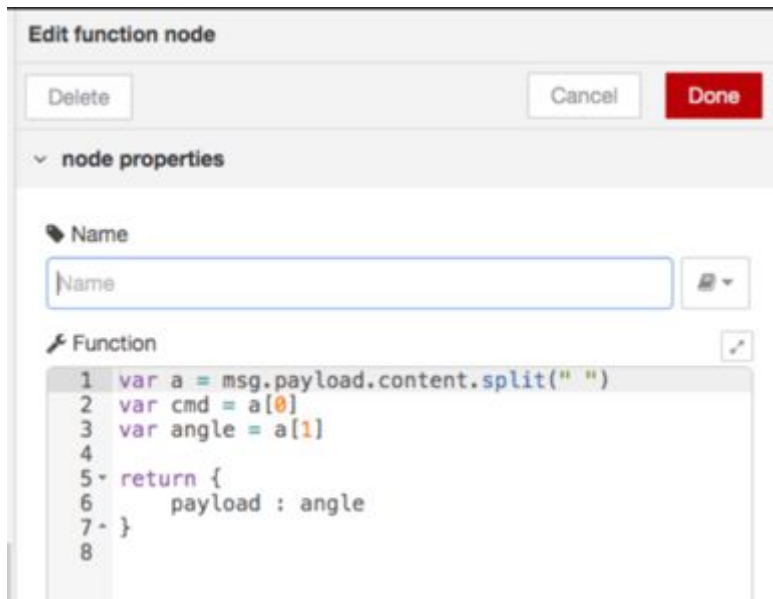


Abbildung 14

In der Funktion wird eine Variable “a” erzeugt, die aus dem Inhalt der Telegramm Nachricht besteht. “angle” ist der zweite Teil von der Nachricht (der Grad vom Winkel) , in unserem Beispiel ist es 90. Es wird der Wert, welchen der der Servo als Grad nimmt zurückgegeben.

Quellenverzeichnis

<http://raspberrypiguide.de/>

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

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