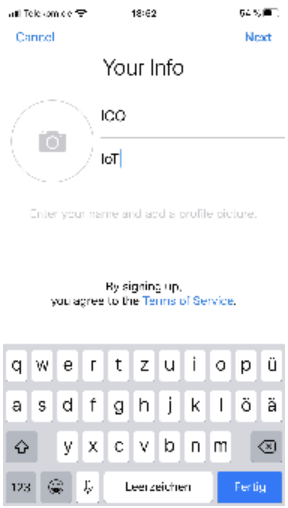
Part 2 - Alarm Nachrichten per Messenger versenden

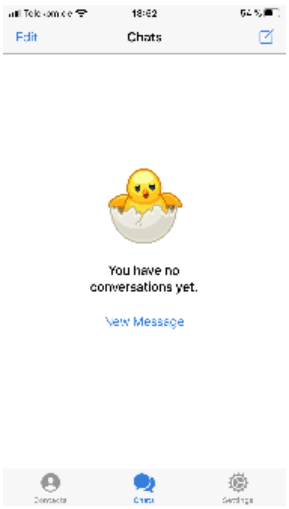
In unserem heutigen Tutorial wollen wir die Temperaturüberwachung aus dem ersten Tutorial - Sensoren auslesen weiter ausbauen. Dazu soll der Raspberry Pi bei Über- oder Unterschreitung eines einstellbaren Schwellwertes eine Warnung direkt auf das Handy versenden. Dazu nutzen wir den kostenfreien und bekannten Messenger Dienst Telegram der für alle gängigen Plattformen wie iOS, Android aber auch für Desktop Rechner verfügbar ist.

Inhaltsverzeichnis

Grundeinrichtung Telegram

Telegram kann aus dem jeweiligen App Store oder aber direkt von der Webseite von Telegram heruntergeladen und installiert werden. Die Grundeinrichtung gestaltet sich sehr einfach, nach Eingabe der Mobilfunknummer erhält man einen Authentifizierungscode per SMS, dieser muss dann eingegeben werden und anschließend werden nur noch die Kontaktinformationen benötigt.

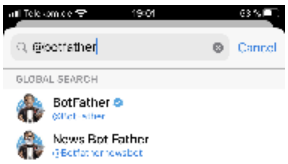




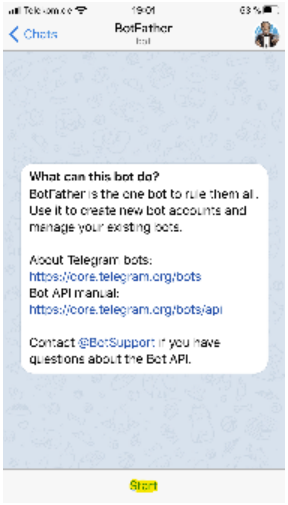
Telegram Bot einrichten

Für die spätere Kommunikation benötigen wir im ersten Schritt einen Telegram Bot. Dieser Bot ist eine Art Roboter über den automatisierte Aufgaben ausgeführt werden können. Über einen Bot kann ein Raspberry Pi auch gesteuert werden, so kann beispielsweise durch das Senden eines Kommandos in dem Chat der Raspberry Pi für eine bestimmte Aufgabe "getriggert" werden. Diese Funktionalität zeigen wir dann in einem späteren Tutorial.

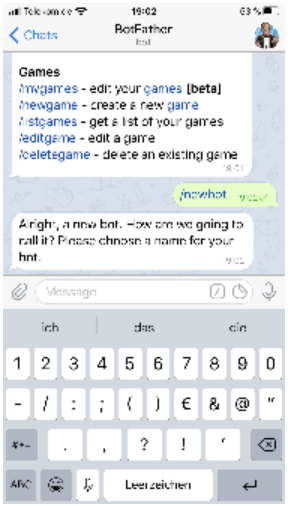
Um nun einen neuen Bot erstellen zu können müssen wir eine Kommunikation mit dem BotFather beginnen. Dazu starten wir oben rechts eine neue Kommunikation und geben im Suchfeld den Namen des BotFather an.

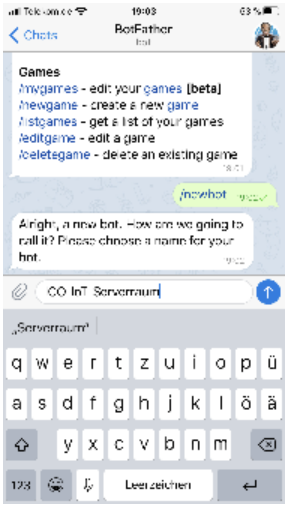


Nach der Begrüßung durch den BotFather können wir dann über die Schaltfläche "Start" einen Konversation mit ihm starten.

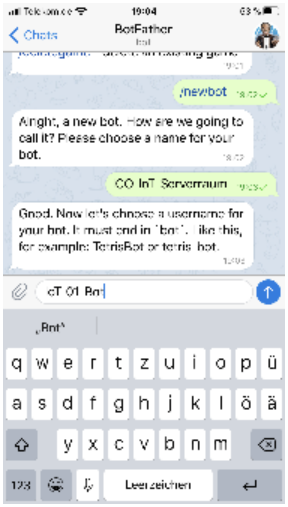


Über den Link "/newbot" oder durch Eingabe von "/newbot" können wir dann interaktiv einen neuen Bot erstellen. Der Name kann dabei frei vergeben werden.





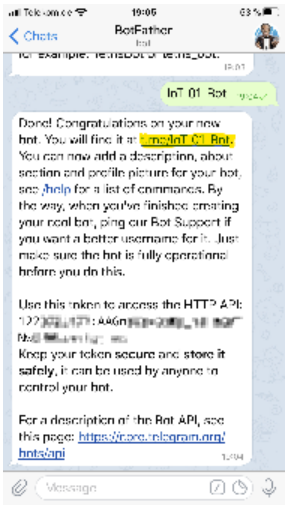
Nun benötigen wir noch einen Usernamen für diesen Bot. Dabei muss der Name allerdings mit Bot enden, also beispielsweise TetrisBot oder tetris_bot.

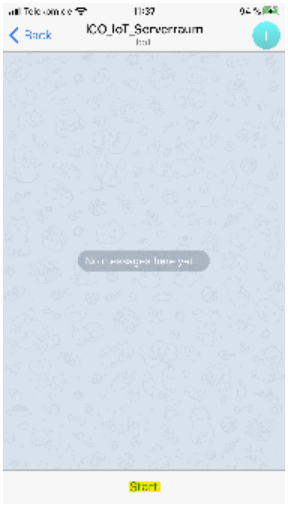


Nachdem der Benutzer angelegt ist erhalten wir eine Nachricht vom BotFather wo der API Token hinterlegt ist. Dieser ist im Format <BOT ID>:<Token> und sollte an einem sicheren Ort aufbewahrt werden.

Wer Zugriff auf diesen Token hat kann jederzeit Nachrichten an diesen Kanal senden!

Nach dem Klick auf den Link des Bots kann er dann über die Schaltfläche "Start" gestartet werden.





Danach ist der Bot einsatzbereit und wir können auf unserem Raspberry Pi direkt eine Kommunikation mit dem Bot starten.

Kommunikation mit dem Bot

Für die Kommunikation mit dem Bot nutzen wir für die ersten Tests die interaktive Python Shell und verwenden dafür die hervorragende Python Requests Bibliothek. Mit dieser Bibliothek lassen sich sehr leicht und intuitiv Befehle an HTTP oder HTTPS Seiten senden. Da die API des Telegram Bots ebenfalls auf einer HTTP API basiert vereinfachen wir damit den Zugriff auf den Bot. Python Requests ist in der Standardinstallation von Raspberry Pi OS bereits installiert, wir verwenden in diesem Fall Python 3.

Zuerst importieren wir die Requests sowie die JSON Bibliothek. Die JSON Bibliothek ist nicht unbedingt notwendig, da der Telegram Server aber mit JSON antwortet lässt sich damit die Ausgabe deutlich leichter formatieren und ist lesbarer.

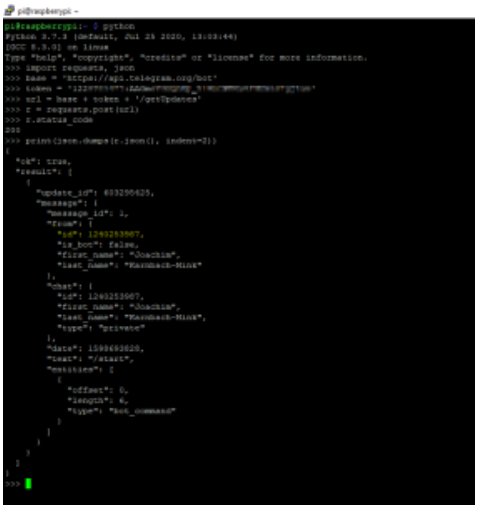
```
import requests, json
```

Nun definieren wir dir URL um später auf den Chat zugreifen zu können. Da wir für die Kommunikation die ID des Chats benötigen stellen wir dann gleich eine Anfrage an den Server nach vorhandenen Updates in diesem Chat, dabei wird uns auch die ID mitgeteilt. In der Variable "token" muss dann der vom BotFather mitgeteilte Token eingetragen werden.

```
base = 'https://api.telegram.org/bot'
token = '<Bot ID>:<Token>'
url = base + token + '/getUpdates'
```

Nun können wir auch schon einen POST Request an die API von Telegram über die Requests Bibliothek senden. Die Ausgabe weisen wir dann der Variablen "r" zu um die Antwort auswerten zu können. Mit r.status_code bekommen wir den Status Code des Webservers zugesandt, mit dem Wert 200 haben wir eine erfolgreiche Antwort des Servers erhalten. Anschließend lassen wir uns die Antwort des Servers formatiert ausgeben um dann an die ID des Chats zu kommen.

```
r = requests.post(url)
r.status_code
200
print(json.dumps(r.json(), indent=2))
```

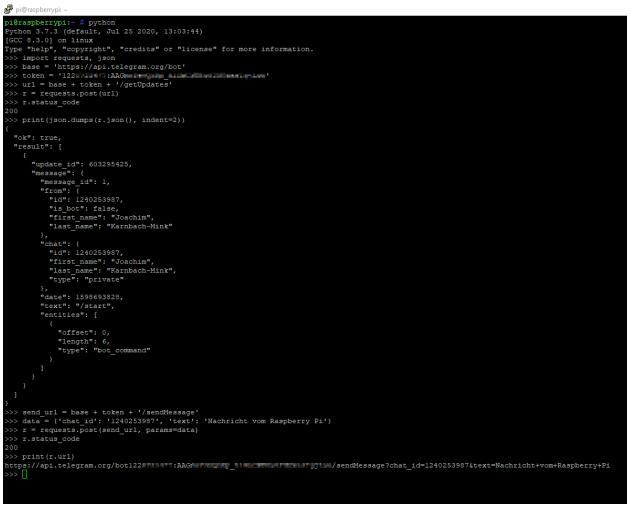


Da wir nun die Chat ID haben können wir nun eine erste Nachricht an den Bot schicken. Da Nachrichten an eine andere URL an Telegram gesendet werden müssen erzeugen wir im nächsten Schritt die URL send_url. Anschließend definieren wir die Chat ID und den Text der Nachricht und weisen es dem Python Dictionary data zu.

```
send_url = base + token + '/sendMessage'
data = {'chat_id': '1240253987', 'text': 'Nachricht vom Raspberry Pi'}
```

Nun können wir den Aufruf an die API wieder als POST Befehl starten und übergeben dabei das Dictionary als Parameter. Der Status Code 200 sagt dabei wieder aus das die Anfrage erfolgreich bearbeitet worden ist und in kürzester Zeit erhalten wir dann die Nachricht im Telegram Messenger. Die von der Requests Bibliothek erzeugte komplette URL lässt sich mit print(r.url) anzeigen.

```
r = requests.post(send_url, params=data)
r.status_code
200
print(r.url)
```



Erweiterung des Python Scripts

Da wir nun alle notwendigen Informationen wie Token und der Chat ID haben können wir nun unser Script aus dem vorherigen Tutorial erweitern. Damit nicht bei jedem Durchlauf des Scripts eine Nachricht bei einem erhöhtem oder zu niedrigem Temperaturwert in den Chat gesendet wird (Thema Spam 🙂) merken wir uns zusätzlich den Status ob eine Nachricht schon gesendet worden ist. Dadurch wird gewährleistet das ein erhöhter Wert nur einmal an den Chat gesendet wird.

temperature_v2.py

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

import requests
import time
import os
import logging
from logging.handlers import TimedRotatingFileHandler

logger = logging.getLogger('temperature')
logger.setLevel(logging.INFO)

formatter = logging.Formatter('%(asctime)s - %(message)s',
        datefmt='%d-%m-%Y %H:%M:%S')

loghandler = TimedRotatingFileHandler(
        '/home/pi/temperature.log', when='D', interval=1, backupCount=10)
loghandler.setLevel(logging.INFO)
loghandler.setFormatter(formatter)

logger.addHandler(loghandler)

logger.info('Programm started...')

sensors = {}
sensors['Serverraum Oben'] = {'id': '28-011927724003', 'sent_min': False, 'sent_max': False}
sensors['Serverraum Unten'] = {'id': '28-0119292513c2', 'sent_min': False, 'sent_max': False}

sensor_min = 20
sensor_max = 30

cpu_max = 60
cpu_sent = False

interval = 5

token = '<Bot ID>:<Token>'
url = 'https://api.telegram.org/bot' + token + '/sendMessage'
```

```
chat_id = '1240253987'

def send_telegram(message):
    data = {'chat_id': chat_id, 'text': message}
    r = requests.post(url, params=data)
    if r.status_code == 200:
        return True
    else:
        return False

def main():
    global cpu_sent
    while True:
        for name, sensor in sensors.items():
            file = open('/sys/bus/w1/devices/' + str(sensor['id']) + '/temperature')
            filecontent = file.read()
            file.close()

            temp = float(filecontent) / 1000

            if temp < sensor_min:
                logger.info("ERROR: temperature too low for %s: %.2f °C", name, temp)

                if sensor['sent_min'] == False:
                    # sent Telegram message
                    message = "Temperatur zu niedrig für " + name + ": " + str(temp) + "°C"
                    send_telegram(message)
                    logger.info("!!! Message sent to Telegram !!!")
                    sensor['sent_min'] = True
            else:
                if sensor['sent_min'] == True:
                    sensor['sent_min'] = False
                    logger.info("%s: Operating in normal mode", name)
                    message = "Temperatur für " + name + " wieder im normalen Bereich"
                    send_telegram(message)

            if temp > sensor_max:
                logger.info("ERROR: temperature too high for %s: %.2f °C", name, temp)

                if sensor['sent_max'] == False:
                    # sent Telegram message
                    message = "Temperatur zu hoch für " + name + ": " + str(temp) + "°C"
                    send_telegram(message)
                    logger.info("!!! Message sent to Telegram !!!")
                    sensor['sent_max'] = True
            else:
                if sensor['sent_max'] == True:
                    sensor['sent_max'] = False
                    logger.info("%s: Operating in normal mode", name)
                    message = "Temperatur für " + name + " wieder im normalen Bereich"
                    send_telegram(message)

            logger.info("%s: %.2f °C", name, temp)

        cpu_raw = os.popen("vcgencmd measure_temp").readline()
        cpu_temp = float(cpu_raw[5:][:4])
        if cpu_temp > cpu_max:
            logger.info("ERROR: CPU temperature too high: %.2f °C", cpu_temp)
            if cpu_sent == False:
                # sent Telegram message
                message = "CPU Temperatur zu hoch: " + str(cpu_temp) + "°C"
                send_telegram(message)
                logger.info("!!! Message sent to Telegram !!!")
                cpu_sent = True
        else:
            if cpu_sent == True:
                cpu_sent = False
                message = "CPU Temperatur wieder im normalen Bereich"
                send_telegram(message)
                logger.info("CPU temperature operating in normal mode")
```

```
        logger.info("CPU: %s °C", cpu_temp)
        time.sleep(interval)

if __name__ == '__main__':
    main()
```

Dabei wurde folgende Änderungen oder Erweiterungen gegenüber der Version 1 vorgenommen:

Zeile	Beschreibung
4	Importieren der Requests Bibliothek
26, 27	Hinzufügen von sent_min und sent_max als Variable ob eine Nachricht bereits versendet wurde. Diese werden dem jeweiligen Sensor zusätzlich hinzugefügt. Standardmäßig sind die Werte auf False, wenn eine Nachricht gesendet wurde ändert sich der Wert auf True.
29, 30	Schwellwerte für Minimal- und Maximaltemperatur. Bei Über- oder Unterschreitung erfolgt eine Benachrichtigung
32, 33	Schwellwert für die CPU Temperatur, Variable für gesendete Nachricht
37 - 39	Parameter für den API Zugriff von Telegram
41 - 47	Funktion zum Versenden einer Telegram Nachricht. Diese Funktion nimmt eine Nachricht in der Variable message entgegen und versendet diese dann zu Telegram
50	cpu_sent als globale Variable definiert damit in der Funktion darauf zugegriffen werden kann
53	Anpassung der ID des Sensors an das neue Dictionary von Zeile 26 und 27
59 - 67	Testen ob der Minimal Wert erreicht ist Sollte sent_min auf False stehen wird eine Nachricht versendet und die Variable auf True gesetzt
68 - 73	Sollte der Minimal Wert nicht erreicht sein wird geprüft ob sent_min auf True ist (Nachricht wurde gesendet, Temperatur ist wieder erhöht). Es erfolgte eine erneute Benachrichtigung und der Wert wird wieder auf False gesetzt
75 - 89	Gleiches Schema für erhöhte Temperatur wie in den Zeilen 59-73
93, 94	Umwandlung der CPU Temperatur in eine Fließkommazahl
95 - 108	Gleiches Schema für die Benachrichtigung bei erhöhter CPU Temperatur

Das Script bietet noch weitere Ausbaumöglichkeiten an, so könnte beispielsweise noch vorab geprüft werden ob eine Internetverbindung steht, bei fehlerhaftem Zugriff auf Telegram nicht die Werte für gesendet zu ändern oder aber teilweise auf das Logging zu verzichten da dadurch ein permanenter Zugriff auf die SD Karte notwendig ist.