**Bildupload zu Twitter (automatisch)**

Das automatische Hochladen eines aufgenommenen Fotos zu einem Twitteraccount, was zuerst als optionales Ziel für das Gesamtprojekt geplant war, entwickelte sich zum Ende der Projektphase zu einem entscheidenden Bestandteil des Gesamtkonzepts der Datenübertragung. Die Anforderungen des Projekts sahen lediglich die Speicherung der entstandenen Fotos vor, welche jeweils von einem Hindernis gemacht werden sollten.

Um nun die gespeicherten Bilder automatisch an den erstellten Twitteraccount der Mohne (Mohne\_2.0) zu pushen, muss über das Application Management von Twitter (apps.twitter.com) eine passende App erstellt werden. Hier hat, ähnlich wie bei ThingSpeak, die eindeutige Identifikation des Ziels für den Uploadvorgang höchste Priorität. Über Consumer Key, Consumer Secret und Access Token, die bei erfolgreicher Einrichtung der App automatisch generiert werden, kann der Twitteraccount aus dem Python Code heraus adressiert werden.

Wenn nun der Ultraschallsensor der Hinderniserkennung ein Hindernis erkennt, wird ein Foto gemacht und automatisch zu Twitter gepusht. Das Foto kann auch manuell ausgelöst werden, allerdings nur wenn sich die Mohne im Fernsteuerungsmodus befindet. Ein weiterer Vorteil dieser Methode ist die Auslagerung der Speicherung der Fotos auf Twitter, da immer nur das aktuelle Foto gespeichert und nach erfolgtem Upload überschrieben wird.

Um das Erscheinungsbild der Tweets der Mohne etwas zu polieren, wurden noch einige Statustexte erstellt, die mit einem Hashtag versehen und mit dem entsprechenden Bild zusammen hochgeladen werden. Dabei werden die Statustexte mit der Funktion choice() zufällig ausgewählt.

Letztendlich hat sich das zuerst optionale Ziel, die Bilder automatisch zu Twitter zu pushen, als ein sehr passendes Experiment im Bereich der Datenübertragung erwiesen. Der Aufbau des Twitteraccounts ist auf Abbildung X gut zu sehen. Außerdem ist die Funktion aus dem dem Bericht angefügten Code <FernFoto.py> ersichtlich.



Abbildung X (<https://twitter.com/0mohne?lang=de>)

**Verbindung zu ThingSpeak über mqtt-Protokoll**

Der zentrale Teil der Anforderungen für das Projekt war die Übertragung von Messdaten der Fahrzeuge über das Web hin zu einer Cloudplattform, hier ThingSpeak. ThingSpeak ist eine Open IoT Platform (Cloudservice), die im Zusammenspiel mit MATLAB eine Menge Funktionen bietet, darunter auch die Übertragung von Messdaten. Dabei waren gewisse Parameter vorgegeben, andere wiederum waren, in einem bestimmten Rahmen, frei wählbar. Für die Übertragung der Daten kam sowohl das HTTP/REST-Protokoll als auch das MQTT-Protokoll in Frage. Die Entscheidung fiel zugunsten des MQTT-Protokolls, das speziell für embedded devices mit geringer RAM- und CPU-Leistung. Darüber hinaus verbraucht MQTT sehr viel weniger Bandbreite und eignet sich daher optimal für die Verwendung mit Arduino und Raspberry Pi. Zusätzlich gibt es die Möglichkeit, Nachrichten mit unsecured TCP zu senden, womit nochmal die Performance verbessert wird.

Um das Protokoll für die Übertragung der Messdaten zu nutzen, ist es erforderlich, eine entsprechende Library auf dem Raspberry Pi zu installieren, in diesem Fall die Paho Library.

In dem dazugehörigen Code <mqtt\_Telemetrie\_ThingSpeak\_V3> sind einige Funktionen derzeit auf „False“ gesetzt, da sie einerseits die Übertragung der entsprechenden Nachricht auf anderen Wegen ermöglichen (statt UnsecuredTCP zum Beispiel UnsecuredWebsockets) oder das Ansprechen eines zweiten Channels auf ThingSpeak ermöglichen. Die Verbindung zu ThingSpeak ließ sich im Nachhinein relativ einrichten, vor allem da es auch in einer Übungsveranstaltung im Studienfach Datenübertragung vorgestellt wurde. Das eigentliche Übertragen der Telemetriedaten wird unter Punkt X.X näher beschrieben.

**Übertragung der Telemetriedaten und Umgebungsbedingungen zu ThingSpeak**

Nach erfolgreichem Herstellen einer Verbindung zu ThingSpeak sollten nun die geforderten Daten aufgenommen und automatisch an ThingSpeak übertragen werden. Die CPU-Temperatur, RAM-Auslastung, Data Up- und Download werden vom Raspberry Pi selbst erfasst, im Gegensatz zu Luftfeuchtigkeit und Umgebungstemperatur, die über einen Kombisensor (KY-015) ausgelesen werden.

Für diesen Kombisensor muss eine spezielle Library von Adafruit installiert werden, Adafruit\_DHT. Nach der Installation kann der Sensor ausgelesen werden, indem man einen entsprechenden GPIO-Pin definiert. Der Kombisensor gibt dann zuerst den Wert für die Luftfeuchte und dann den Wert für die Umgebungstemperatur aus. Die erfassten Werte für Luftfeuchte und Umgebungstemperatur können den Abbildungen X und X entnommen werden.

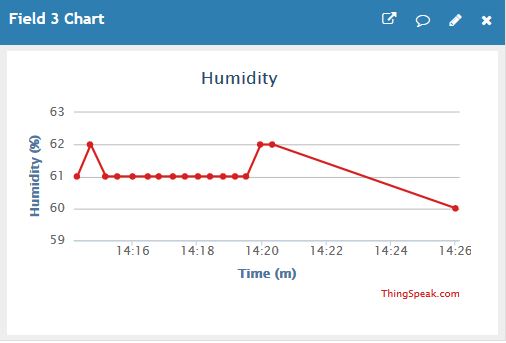


Abbildung X

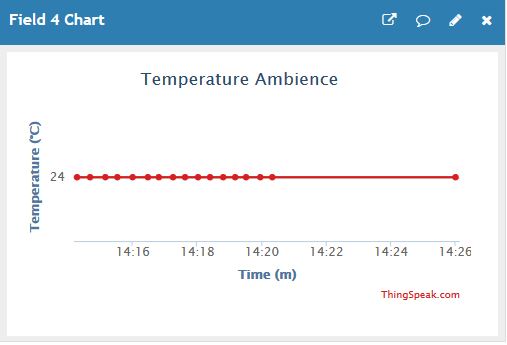


Abbildung X

Für die Übertragung der erfassten CPU-Temperatur und der RAM-Auslastung werden einfache, im Code <mqtt\_Telemetrie\_ThingSpeak\_V3> leicht ersichtliche Befehle verwendet. So ist zum Beispiel folgender Code ausreichend, um die RAM-Auslastung auszulesen:

ramPercent = psutil.virtual\_memory().percent

Bei der Übertragung der CPU-Temperatur musste der ausgelesene Datenstring etwas angepasst werden, damit lediglich ein Zahlenwert an ThingSpeak weitergegeben wird. Ohne diese Anpassung wäre es nicht möglich gewesen, die CPU-Temperatur auszulesen. Den Abbildungen X und X können die entsprechenden Werte für CPU und RAM entnommen werden.

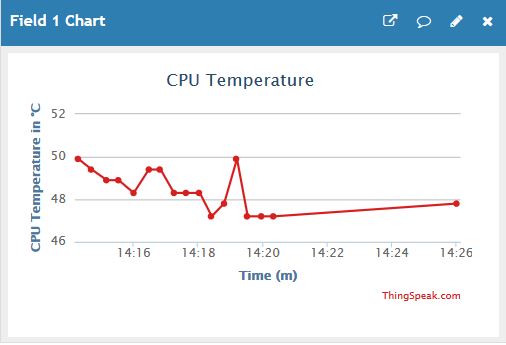


Abbildung X

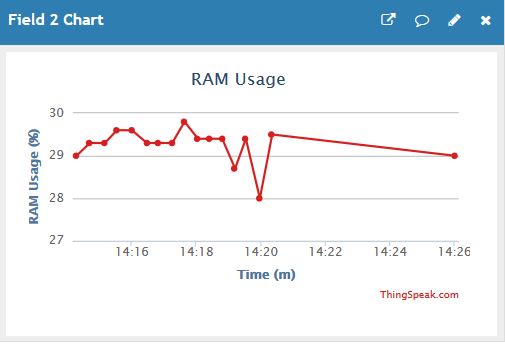
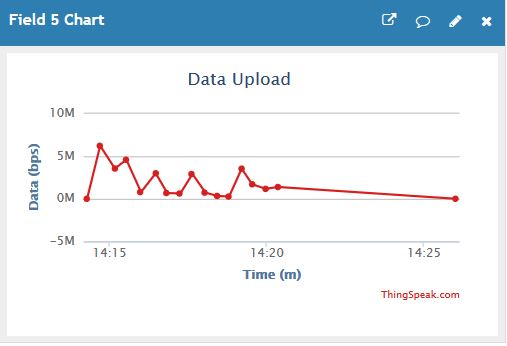


Abbildung X

Die Übergabe der Data Up- und Downloads war relativ kompliziert, da der aktuelle Wert durch einen Vergleich zwischen aktuellem Datenübertrag und vorherigen Datenübertrag ermittelt wird. Die erste Messung zeigt also zu Beginn bei beiden Feldern den Wert 0, und ab der nächsten Messung werden die aktuellen Werte angezeigt. Data Up- und Download können den Abbildungen X und X entnommen werden.



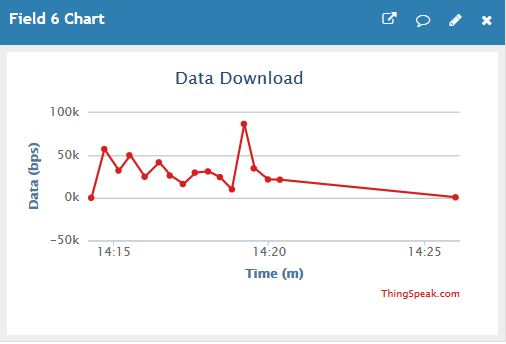
Abbildung X

Abbildung X