[Deckblatt]

Berufsbildende Schule des Landkreises Ahrweiler

Höhere Berufsfachschule IT-Systeme

HBFIT15

2015

Entwicklungs eines mobilen Standortbestimmungsgerätes für WiFi-Netzwerke und Programmierung in C++

Klaus Müller

Erik Berreßem

22.02.2017

Inhaltsverzeichnis

1 Vorwort

„Was genau ist eigentlich eine WarKitty?“ – Diese Frage wurde mir während der Projektbearbeitung oft gestellt. Der Begriff „WarKitty“ ist ein Portmanteauwort, also ein Wort welches aus zwei Wörtern entstanden ist. Nämlich „War“ für Wardriving und „Kitty“ für Katze. Was eine Katze ist wissen die meisten dann auch, dies kann man nicht für Wardriving behaupten.

„Wardriving ist das systematische Suchen nach Wireless Local Area Networks mit Hilfe eines Fahrzeugs. [...] Einige Wardriver sehen die drei Anfangsbuchstaben dabei als Backronym für 'Wireless Access Revolution'.“[1]

Wardriving ist also ein Begriff der das suchen und dokumentieren von WLAN / WiFi Netzwerken beschreibt.

1.1 Projektidee

„Warum Katzen?“ – Das traditionelle Wardriving erfolgt entweder per Auto oder zu Fuß. Während man fährt / geht sammelt man Daten über WiFi Netzwerke in der Umgebung und verknüpft diese mit GPS Daten. Ich habe mir vorgenommen diesen Prozess noch weiter zu automatisieren. Was ich mir dachte war „Was wäre wenn man nur noch die Daten einsammeln müsste?“.

Für dieses Vorhaben überlegte ich mir verschiedene Realisierungsmöglichkeiten: Drohnen, Hunde, Katzen.

1 https://de.wikipedia.org/wiki/Wardriving, (Download: 16.02.2017).

Ich entschied mich letztendlich für die Katze da sie, im Gegensatz zum Hund, eine höhere Wahrscheinlichkeit hat zurückzukommen, wenn man sie frei laufen lässt und da das Projekt so leichter zu realisieren ist als bei einer Drohne (Kostenfaktor).

2 Projekt

2.1 Planung

Ich hatte nun also eine grobe Planung für mein Projekt. Was ich mir nun überlegen musste war wie ich es umsetzen wollte. Ich entschied mich dafür Arduino basierte Hardware, Git Versionsverwaltung durch GitHub und die PlatformIO IDE für Atom (im Gegensatz zur Arduino IDE) zu verwenden. Das sind nun viele neue Wörter, welche ich nun erklären werde.

Arduino – ist eine Open-Source Elektronik Platform die auf einfach zu benutzbarer Hardware und Software basiert. Es gibt viele Erweiterungen, sogenannte „Shields“, wie zum Beispiel SD-Karten Leser, GPS-Empfänger, usw. Arduino und Arduino ähnliche Produkte sind außerdem relativ günstig. Ich entschied mich für dass sogenannte „NodeMCU v1“, ein WiFi-Development Board mit einen mir bereits bekannten ESP8266 Mikrocontroller und den „NEO-6M/7M“ GPS-Empfänger von WaveShare.

Git – „[…] ist eine freie Software zur verteilten Versionsverwaltung von Dateien, die durch Linus Torvalds initiiert wurde.“[2] Git hat grob zusammengefasst zwei sehr Wichtige Funktionen: Das verwalten verschiedener Versionen einer Datei

2 https://de.wikipedia.org/wiki/Git, (Download: 16.02.2017).

und die Möglichkeit sogenannte „Branches“, also Abzweigungen eines Projektes, zu erstellen die auch wieder zusammengeführt werden können. Versionsverwaltung wird von mit später nochmal genauer Erklärt.

GitHub – ist ein Webbasierter Hosting-Dienst für Git-Projekte[3]

Atom – ist ein von GitHub entwickelter Open-Source Texteditor der sehr erweiterbar ist. So gibt es zum Beispiel Syntaxhervorhebung für so gut wie alle Programmiersprachen.

PlatformIO IDE – „[…] ist eine mächtige Alternative zu der recht spärlich ausgestatteten Arduino IDE und ist gerade für größere Projekte eine sinnvolle Ergänzung. Durch die zahlreich unterstützten Plattformen ist zudem kein „Umgewöhnen“ in der Entwicklungsumgebung mehr nötig, wenn man oft die Entwicklungsplattform wechselt.“[4]

2.2 Durchführung

Man kann meine Durchführung in grob drei Abschnitte aufteilen: Informationsbeschaffung, Coding und Testing, wobei diese Abschnitte nicht linear sondern anachronistisch abliefen.

2.2.1 Informationsbeschaffung

Da es bei meinen Projekt viel für mich neues Wissen zu erlangen hieß war das erste was ich während der Durchführung tat mich zu informieren. NMEA sentences,

3 Vgl. https://de.wikipedia.org/wiki/Git, (Download: 16.02.2017).
4 https://alexbloggt.com/platformio-vorgestellt, (Download: 16.02.2017).

Beacon frames, Buffer, Versionsverwaltung – All dies und noch mehr war neu für mich und es dauerte ein bisschen bis ich diese Begriffe, nicht nur durch Internetrecherchen sondern besonders auch durch trial-and-error, verstand. Wenn man nur eine relativ kurze Zeit von sechs Wochen für das Projekt hat ist es hilfreich gut zu Planen wie man sich informiert und über was man sich informiert, damit man nicht unnötig Zeit „Verschwendet“ dich über dinge zu informieren die man nicht wirklich in der Umsetzung des Projekts braucht.

„**NMEA 0183** (meist einfach NMEA genannt) ist ein Standard […] für die Kommunikation zwischen GPS-Empfänger und PCs sowie mobilen Endgeräten […]“[5] NMEA ist eine Abkürzung für **N**ational **M**arine **E**lectronics **A**ssociation, welche ihn entwickelt hat. So sieht ein NMEA-Datensatz aus:

$GPVTG,,T,,M,1.012,N,1.874,K,A*2B

$GPGGA,165719.00,50.562662,N,07.264540,E,1,09,0.88,76.3,M,46.9,M,,*67

$GPGSA,A,3,12,19,17,32,15,25,10,14,24,,,,1.50,0.88,1.21*09

$GPGSV,3,1,11,02,08,123,,06,16,088,,10,06,268,24,12,79,264,35*76

$GPGSV,3,2,11,14,15,321,28,15,14,179,18,17,20,041,32,19,35,053,26*78

$GPGSV,3,3,11,24,70,127,25,25,35,252,24,32,34,304,19*40

$GPGLL,5033.76060,N,00715.87459,E,165719.00,A,A*62

$GPRMC,165720.00,A,5033.76089,N,00715.87455,E,0.644,,180217,,,A*71

Ein NMEA-Datensatz besteht aus mehreren NMEA sentences, wobei das erste Wort eines Satzes den Datentyp angibt, der bestimmt wie der Rest des Satzes interpretiert werden soll. Diese sentences sind vollgepackt mit Informationen; so vieler, dass ich nicht auf alles eingehen kann. In folgenden werde ich zwei sentences dieses NMEA-Datensatzes auflösen.

5 https://de.wikipedia.org/wiki/NMEA_0183, (Download: 18.02.2017).

Jeder sentence beginnt mit einen „$“ um die sentences voneinander abzugrenzen. Das „GP“ gibt an von welchen Gerät die Informationen kommen. Anstelle von „GP“ für GPS-Empfänger könnte an dieser Stelle noch andere Kürzel stehen. „LC“ für Loran-C Empfänger (ein älteres Positionsbestimmungssystem), „OM“ für Omega Navigations Empfänger (ein älteres Radionavigationssystem; ausser Betrieb) und „II“ für Integrated Instrumantation (z.B. Autopiloten; AutoHelm Seatalk System) sind die gängigsten. Danach folgen die Informationen. Am Ende eines jeden sentence steht noch die Prüfsumme des Satzes, die sich aus den ASCII-Werten aller Zeichen berechnet und in hexadezimal dargestellt wird. Das Endgerät welches die Datensätze empfängt berechnet dann nochmal die Prüfsumme und vergleicht diese mit der vom GPS-Gerät berechneten. Unterscheiden diese sich, dann ist der Datensatz fehlerhaft.

$GPVTG,,T,,M,1.012,N,1.874,K,A*2B

VTG	**–**	**Geschwindigkeit und Richtung**	
,T	**–**	**Kurs**	**[NULL grad]**
,M	**–**	**Kurs**	**[NULL mag]**
1.012,N	**–**	**Geschwindigkeit**	**[1,012 knoten]**
1.874,K	**–**	**Geschwindigkeit**	**[1,874 km/h]**
A	**–**	**Art der Bestimmung**	**[A = Autonom]**

$GPGGA,165719.00,50.562662,N,07.264540,E,1,09,0.88,76.3,M,46.9,M,,*67

GGA	**–**	**Zeit, Position und Qualität**	
165719.00	**–**	**Uhrzeit**	**[16:57:19 Uhr]**
50.562662,N	**–**	**Breitengrad**	**[50° 33' 45.5832'' N]**
07.264540,E	**–**	**Längengrad**	**[7° 15' 52.344'' E]**
1	**–**	**Qualität der Messung**	**[1 = GPS]**
09	**–**	**Anzahl der Satelliten**	
0.88	**–**	**HDOP (horizontale Genauigkeit)**	
76.3,M	**–**	**Höhe über den Meer**	**[76,3 meter]**
46.9,M	**–**	**Höhe Geoid minus Ellipsoid**	**[46,9 m.]**

Beacon Frames enthalten alle Informationen über ein Netzwerk welches es aussendet. Sie werden periodisch ausgesendet um die Präsenz eines WiFi-Netzwerkes anzukündigen. Beacon frames bestehen aus einen Header, welcher die Quell MAC-Adresse enthält, einem Body und einer Blockprüfzeichenfolge (BPF) die den Frame terminiert. Die BPF funktioniert genau wie die Prüfsummen von NMEA-Datensätzen. Der Body enthält einen Zeitstempel, der zur Synchronisation der lokalen Uhren der Empfänger dient, den Beacon Intervall, der angibt in welchen Intervall die Frames gesendet werden (meist 100ms), sowie Informationen zur Leistungsfähigkeit. In diesen Feld wird die Art des Netzwerkes, zum Beispiel Ad-hoc oder Infrastruktur Netzwerk, mitgeteilt. Abgesehen von diesen Informationen kündigt es die Unterstützung für Polling, sowie Verschlüsselungs Details an. Weitere Felder im Body sind die SSID, die Unterstützten Raten, eine „Traffic indication map“, sowie „Frequency-hopping“-, „Direct-Sequence“-, „Contention-Free“- und „IBSS“-Parameter.

„Eine **Versionsverwaltung** ist ein System, das zur Erfassung von Änderungen an Dokumenten oder Dateien verwendet wird.“[6] Die Hauptaufgaben einer Versionsverwaltung sind Protokollierungen der Änderungen, Wiederherstellung von alten Ständen einzelner Dateien, Archivierung der einzelnen Stände eines Projektes, Koordinierung des gemeinsamen Zugriffs von mehreren Entwicklern auf die Dateien und Gleichzeitige Entwicklung mehrerer Entwicklungszweige (engl. Branches) eines Projektes[6]. Versionsverwaltung ist besonders dann wichtig, wenn man an einen größeren Projekt und / oder mit mehreren Entwicklern arbeitet. Eines der bekanntesten Versionsverwaltungsprogramme ist Git, „[…] eine freie Software zur verteilten Versionsverwaltung von Dateien, die durch Linus Torvolds (den Initiator sowie die treibende Kraft hinter dem Linux-Kernel) initiiert wurde.“[7] „GitHub ist ein webbasierter Online-Dienst, der Software-Entwicklungsprojekte auf seinen Servern bereitstellt […]“[8], welchen ich verwende. Wenn man ein neues Git Repository initialisieren möchte kann man das im Terminal machen indem man zum Ordner navigiert, den man initialisieren möchte und dann mithilfe von „git init“ das Git Repository erstellt. Änderungen lassen sich mit „git add <dateiname>“ oder mit

6 Vgl. https://de.wikipedia.org/wiki/Versionsverwaltung, (Download: 21.02.2017).
7 https://de.wikipedia.org/wiki/Git, (Download: 21.02.2017).
8 https://de.wikipedia.org/wiki/GitHub, (Download: 21.02.2017).

Wildcards wie „git add *“ dem Index hinzufügen und mit „git commit -m “<Commit-Nachricht>““, wobei man <Commit-Nachricht> mit einer Nachricht über die Änderung ersetzt, bestätigen. Nun sind die Änderungen im Index der noch verändert werden kann. Um die Änderung festzumachen muss man dann noch „git push origin <Branch>“, wobei <Branch> ersetzt wird durch den Zweig auf den man pushen möchte (z.B. master, Dev), ausführen. All dies lässt sich auch mit der GitHub GUI erledigen die von GitHub entwickelt wurde und eine grafische Oberfläche für Git bietet.

3 Problemstellung

4 Zielsetzung

5 Kritische Reflexion

5.1 [Wie verlief das insgesamt?]

5.2 Was war gut?

5.3 Was hätte anders sein können oder müssen?

5.4 Sind die Ziele erreicht worden?

6 Quellenangaben

7 Anhang

7.1 Projektantrag

7.2 Projektstrukturplan

7.3 Projektablaufplan