# „WiFi-Hack“ des TI-Rovers

Seit einigen Jahren nutze ich in der Schule schon den TI-Innovator™ Rover (im Weiteren kurz Rover), und seit jeher hat mich eine Sache „gestört“: Man kann den Rover nicht fernsteuern.

Im Rahmen meiner Seminarfacharbeit zum Thema „autonomes Fahren“ bin ich, nach langen Überlegungen, diese Problematik endlich angegangen. Ich habe zwei Rover zu Modellen von autonomen Fahrzeugen umgebaut, welche mittels WiFi kommunizieren und die Spurhaltung durch das Messen des Seitenabstandes realisieren.

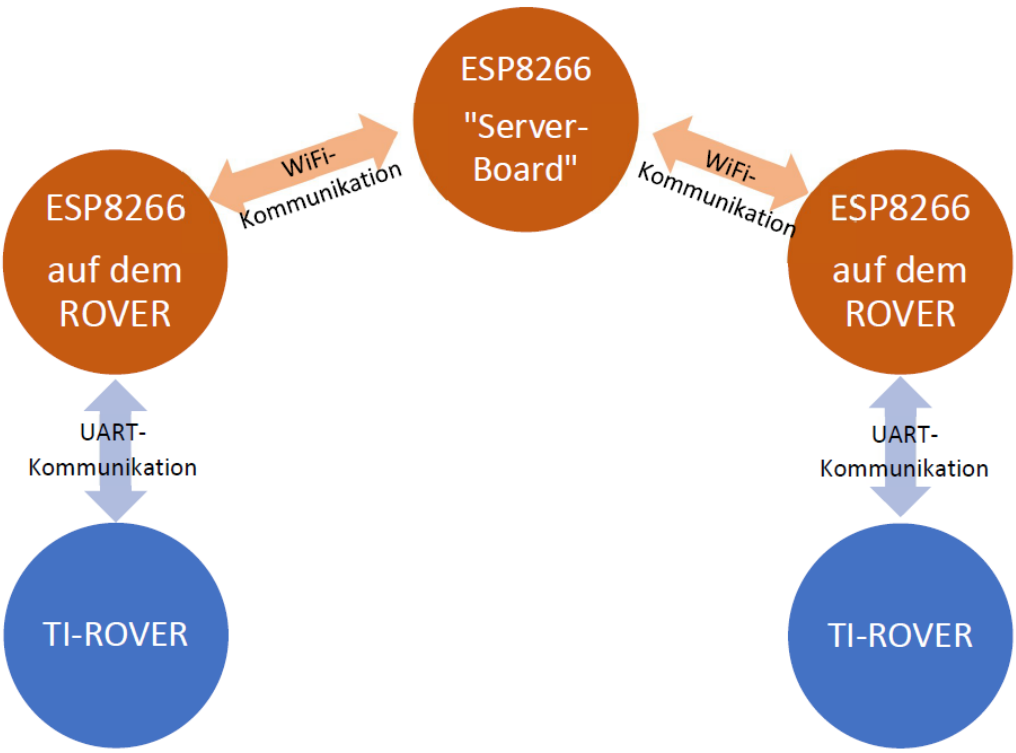


Abbildung 1: Verwendete Komponenten und deren Kommunikation

Für die WiFi-Kommunikation habe ich drei ESP8266-NodeMCU[1] Entwicklerboards verwendet, welche sich mit der gängigen Arduino-IDE sehr einfach programmieren lassen. Zwei der drei ESP8266s habe ich auf je einen der Rover platziert, damit diese mittels UART mit dem TI-System kommunizieren können. Der HUB hat eine entsprechende UART-Schnittstelle am OUT1-Anschluss implementiert, welche man mit den Befehlen „Set Serial *zuSendendeNachricht*“ und „Read Serial“ nutzen kann, auch wenn diese nicht im TI-Handbuch dokumentiert sind.

Das dritte ESP8266-Board habe ich an meinem Computer als „Server-Board“ angeschlossen, über das die gesamte WiFi-Kommunikation läuft. Man kann das Ganze auch ohne das „Server-Board“ umsetzen (also eine direkte Kommunikation zwischen den ESP8266 der Rover), aber ich finde es sehr praktisch, da man so die Kommunikation überwachen,

[1] https://www.mikrocontroller-elektronik.de/nodemcu-esp8266-tutorial-wlan-board-arduino-ide/

fernsteuern, bzw. in das Geschehen eingreifen kann, ohne den Rovern hinterherrennen zu müssen. Die reine WiFi-Kommunikation läuft auf dem UDP-Protokoll, da dies meiner Meinung nach am einfachsten umzusetzen ist. Dafür ist in jedem der drei ESP8266 ein lokaler UDP-Server eingerichtet, der eingehende und ausgehende Nachrichten bearbeitet. Dennoch bezeichne ich (wie auch in Abbildung 1) nur einen der drei ESP8266 als „Server-Board“, da dieser nämlich die zentrale Kommunikationsschnittstelle des ganzen Projektes ist, während die anderen beiden ESP8266 den lokalen UDP-Server nur aufgrund der technischen Funktionalität der verwendeten Library haben.

Fahren die beiden Rover in einer Kolonne, so findet z.B. folgende Kommunikation statt: Die ESP8266 des vorderen Rovers meldet dem „Server-Board“ über WiFi, dass es losfahren will. Das „Server-Board“ wertet diese Nachricht aus und sendet anschließend den beiden ESP8266s (auf den beiden Rovern) über WiFi einen Zeitpunkt, an dem sie gemeinsam losfahren sollen. Die beiden ESP8266s warten nun bis zu dem Zeitpunkt und senden nun mittels UART-Protokolls über den OUT1 Anschluss des HUBs an das TI-System, dass das Fahrzeug nun losfahren soll.[2]

Da die Kommunikation über UART sehr zeitaufwändig ist, habe ich möglichst jede Berechnung auf den ESP8266 verlagert. Demnach berechnet der ESP8266 den TI-Befehl, z.B. „SET RV.MOTORS 255 -255“ und sendet diesen fertigen Befehl dann via UART über den OUT1 Anschluss an den HUB. Dann muss dieser diesen Befehl einfach nur ausführen. Bei der reinen Kommunikation, also der HUB-ESP8266 oder der WiFi-ESP8266-ESP8266 Kommunikation, habe ich ein Buffer-Konzept mit Handshake umgesetzt. Das heißt, der ESP8266 hat ein Array in dem alle ausgehenden Nachrichten gespeichert werden. Alle 200ms wird dieser Buffer dann bearbeitet und eine Nachricht gesendet. Der Empfänger sendet nach dem Empfangen der Nachricht unverzüglich eine „OK-Nachricht“ als Handshake zurück, damit der Sender weiß, dass alles problemlos angekommen ist. Dann rücken die Nachrichten in dem Sende-Array einen Platz vor, sodass in 200ms die nächste Nachricht gesendet wird. Wie viel Zeit man genau zwischen dem Senden von 2 Nachrichten vergehen lässt, muss man durch ein bisschen experimentieren herausfinden. Denn einerseits möchte man möglichst viele Nachrichten pro Sekunde verschicken können, andererseits darf aber nichts beim Senden verloren gehen.

Da das Ganze ein Modell für ein autonomes Fahrzeug sein sollte, habe ich zusätzlich ein neues Spurhaltekonzept entwickelt Dafür habe ich rechts und links je einen HC-SR04 Ultraschallsensor angebaut und an den ESP8266 angeschlossen (siehe Bild). Auch wenn ich im Endeffekt nur eine der beiden Seiten genutzt habe. Die Spurhaltung funktioniert wie folgt: Der linke Ultraschallsensor misst den Abstand zur Wand (oder zur Leitplanke o.ä.) und ist dieser Abstand kleiner oder größer als 20cm, so fährt der Rover von der Wand weg, bzw. zu der Wand hin. Dafür habe ich eine Funktion geschrieben, welche zuerst die Abweichung von den 20cm berechnet und dann in Abhängigkeit von dieser Abweichung die Geschwindigkeit der Motoren drosselt.

Zusätzlich wollte ich den bereits im Rover verbauten HC-SR04 Ultraschallsensor nutzen, damit der Rover anhält, wenn er sich einem Hindernis nähert. Da der HUB bereits mit der Kommunikation (mit dem ESP8266 über den OUT-1 Port) vollständig ausgelastet ist, habe

[2] Beispielvideo: https://www.youtube.com/watch?v=QA_S6bbzJpg

ich mich dazu entschieden, den bereits verbauten Ultraschallsensor vom Rover abzuklemmen und an den ESP8266 anzuschließen.

**Ausblick**

Dieser „WiFi-Hack“ hat enormes Potential. Ich habe als WiFi-Netzwerk zwar einen Hotspot meines Laptops genutzt, da ich so das Ganze am einfachsten einrichten konnte, ohne mich groß mit dem Aufbau von Netzwerken, DNS-Systemen, etc. beschäftigen zu müssen. Richtet man das WiFi hingegen „ordentlich“ ein, und nutzt man z.B. das Schul-WLAN, so kann man über das „Server-Board“ die Rover am anderen Ende des Hauses fernsteuern. Man kann den Code des ganzen Systems auch so abändern, dass nicht nur zwei spezifische Rover gesteuert werden, sondern beliebige viele, die sich mit dem „Server-Board“ verbinden.

Außerdem gibt es in der Arduino IDE vorgefertigte Libraries, die es ermöglichen den ESP8266 per WiFi zu programmieren, ganz ohne Kabelverbindung. Und da der HUB ja nur die vom ESP8266 vorgefertigten Befehle ausführt, kann man somit also den HUB per WiFi steuern!

**Danksagung**

Zu guter Letzt möchte ich mich noch bei Hubert Langlotz und Hans-Martin Hilbig bedanken, die dieses Projekt ermöglicht und mich bei der Umsetzung unterstützt haben.

**Autor:**
*Yannick Höffner*
yannick.hoeffner@t-online.de

**Info:**

*Yannick Höffner* ist Schüler des Elisabeth-Gymnasium Eisenach in Thüringen (Abschlussklasse A21M, Klassensufe 12.