

Aufbau eines drahtlosen Netzwerks

Benedikt Viebahn s0565139

June 26, 2019

Contents

1	Einleitung	2
2	Einrichtung	2
3	Kommunikation mit AT Befehlen	5
4	Protokoll	5
5	Implementierung	6
6	Quellen	7

1 Einleitung

Im Rahmen der Veranstaltung Technik Mobiler Systeme haben wir ein drahtloses Netzwerk mit Raspberry Pis aufgebaut. Die Aufgaben bestanden darin, den Raspberry Pi mit einem Microcontroller auszustatten, und ein Protokoll zu implementieren auf das wir uns vorher geeinigt hatten, und über das wir andere Netzwerkteilnehmer finden und mit ihnen kommunizieren können.

2 Einrichtung

Damit MCU und Raspberry Pi miteinander kommunizieren können müssen diese sowohl Hardware-, als auch Softwareseitig richtig konfiguriert werden. Die von uns verwendete MCU benötigt eine Verbindung zur Stromquelle, zu Ground, und jeweils eine Verbindung um Daten zu empfangen und zu senden. Die Pins müssen wie folgt miteinander verbunden werden (siehe Figure 1-3):

Raspberry Pi	MCU
Ground	GND
3.3 VDC Power	VIN
GPIO 15 TxD (UART)	RX
GPIO 16 RxD (UART)	TX





















Raspberry Pi 3 Model B (J8 Header)					
GPIO#	NAME			NAME	GPIO#
	3.3 VDC Power	1		2	5.0 VDC Power
8	GPIO 8 SDA1 (I2C)	3		4	5.0 VDC Power
9	GPIO 9 SCL1 (I2C)	5		6	Ground
7	GPIO 7 GPCLK0	7		8	GPIO 15 TxD (UART) 15
	Ground	9		10	GPIO 16 RxD (UART) 16
0	GPIO 0	11		12	GPIO 1 PCM_CLK/PWM0 1
2	GPIO 2	13		14	Ground
3	GPIO 3	15		16	GPIO 4 4
	3.3 VDC Power	17		18	GPIO 5 5
12	GPIO 12 MOSI (SPI)	19		20	Ground
13	GPIO 13 MISO (SPI)	21		22	GPIO 6 6
14	GPIO 14 SCLK (SPI)	23		24	GPIO 10 CE0 (SPI) 10
	Ground	25		26	GPIO 11 CE1 (SPI) 11
30	SDA0 (I2C ID EEPROM)	27		28	SCL0 (I2C ID EEPROM) 31
21	GPIO 21 GPCLK1	29		30	Ground
22	GPIO 22 GPCLK2	31		32	GPIO 26 PWM0 26
23	GPIO 23 PWM1	33		34	Ground
24	GPIO 24 PCM_FS/PWM1	35		36	GPIO 27 27
25	GPIO 25	37		38	GPIO 28 PCM_DIN 28
	Ground	39		40	GPIO 29 PCM_DOUT 29
<p>Attention! The GPIO pin numbering used in this diagram is intended for use with WiringPi / Pi4J. This pin numbering is not the raw Broadcom GPIO pin numbers.</p> <p>http://www.pi4j.com</p>					

Figure 1: Raspberry Pi Anschlüsse

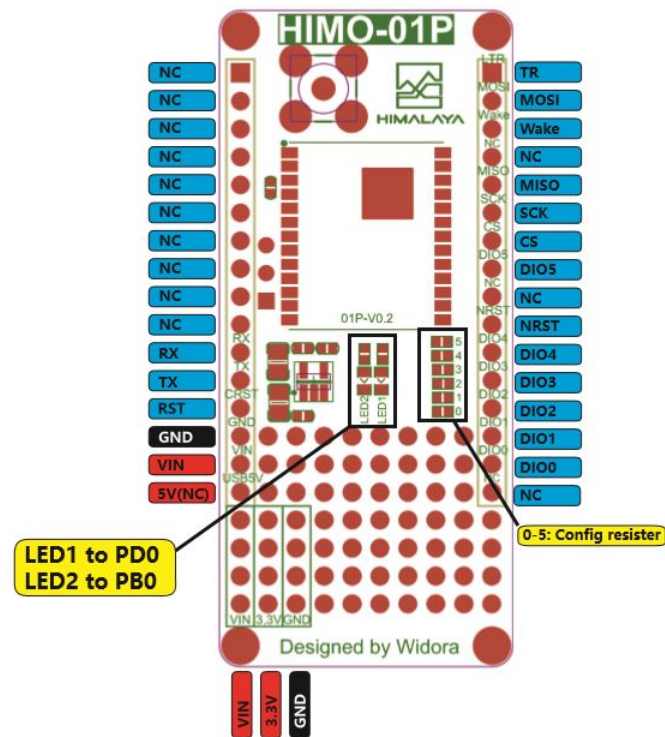


Figure 2: MCU Anschlüsse

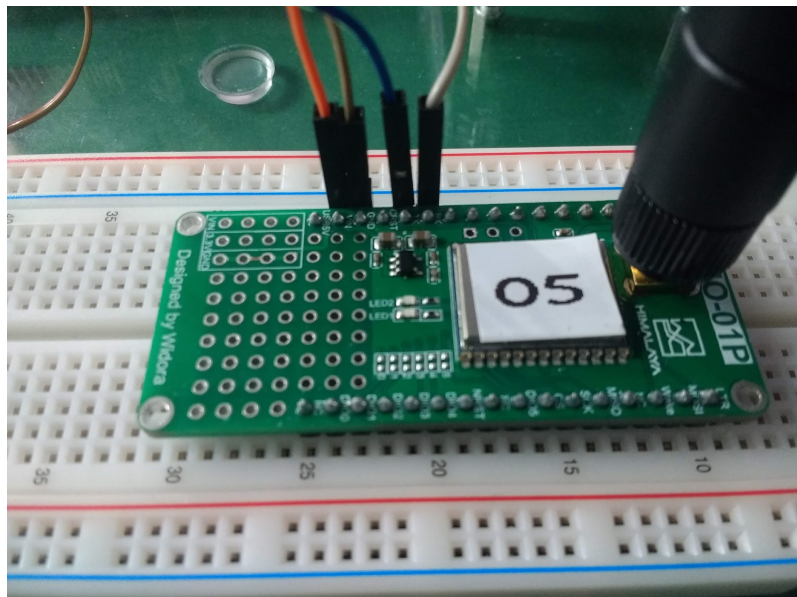


Figure 3: MCU fertig verbunden

Nachdem die Hardware richtig verbunden ist muss eventuell der Serial Port in den Einstellungen des Raspberry Pis unter Interfaces freigegeben werden. Mit dem Programm *cutecom* kann getestet werden ob die Kommunikation funktioniert. Das Programm sollte so konfiguriert werden:

Konfiguration	Wert
Baudrate	115200 Bits
Parity Bit	NONE
Stop bits	1
Data bits	8

3 Kommunikation mit AT Befehlen

Nun kann mit AT Befehlen getestet werden ob die Verbindung funktioniert. Wenn **AT\r\n AT,OK\r\n** zurück liefert funktioniert die Kommunikation. Mit **AT+ADDR=XXXX\r\n** kann die eigene Adresse festgelegt werden. XXXX ist dabei die Adresse und kann im Bereich von 0000 bis FFFF liegen, wobei FFFF die Broadcastadresse ist. Nachrichten an diese Adresse werden von allen empfangen. Um Daten zu senden muss zunächst mit **AT+DEST=XXXX\r\n** die Zieladresse festgelegt werden. Das Modul antwortet darauf wieder mit **AT,OK\r\n**. Dannach wird mit **AT+SEND=XX\r\n** die Größe der zu sendenden Daten angegeben. Dabei sollte beachtet werden, dass die maximale Länge 25 Byte beträgt. Daten die die festgelegte Länge überschreiten werden verworfen. Nachdem das Modul mit **AT,OK\r\n** antwortet, können nun Daten übertragen werden. Dabei antwortet das Modul erst mit **AT,SENDING\r\n** um mitzuteilen dass es sich im Sendezustand befindet, und nach erfolgreicher Übertragung mit **AT,SENDED\r\n**.

4 Protokoll

Zur Entdeckung der Netzwerkteilnehmer haben wir uns auf folgendes Network Discovery Protokoll geeinigt: Jeder Teilnehmer sendet mit zufälligen Abständen zwischen 30 und 60 Sekunden eine Nachricht mit dem Inhalt **RTI** an die Broadcastadresse **FFFF**. Jeder der diese Nachricht empfängt kann die Senderadresse speichern und weiß nun dass sie existiert und dass sie erreicht werden kann.

5 Implementierung

Das Script wurde in Python geschrieben und funktioniert wie folgt: Um auf die serielle Schnittstelle zuzugreifen wird das package *serial* verwendet.

```
import serial
ser = serial.Serial(
    port='/dev/ttyS0',
    baudrate=115200,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS
)
ser.isOpen()
```

Als nächstes legen wir unsere Adresse fest. Diese wird dem Programm beim Aufruf als Argument übergeben.

```
ser.write("AT+ADDR=" + sys.argv[1] + "\r\n")
```

Die momentan ausgewählte Zieladresse, die empfangenen Nachrichten und die bekannten Adressen werden gespeichert und immer aktuell gehalten. Außerdem wird gespeichert ob gerade auf eine **AT,OK** Nachricht gewartet wird, und ob eine empfangen wurde. Dadurch wird gewährleistet, dass Nachrichten erst gesendet werden nachdem alle vorher gesendeten Nachrichten angekommen sind. Das Programm nutzt drei Threads (+ main thread):

receive: Daten werden von der seriellen Schnittstelle empfangen und in einem Array für die spätere Verarbeitung gespeichert.

processReceived: Die empfangenen Nachrichten werden in der Reihenfolge in der sie empfangen wurden bearbeitet. Beim Empfangen einer **AT,OK** Nachricht wird ein globales Boolean *ok* auf *True* gesetzt. Ein anderer Thread der zuvor etwas gesendet hat und auf **AT,OK** wartet kann jetzt weiter machen und setzt *ok* und *waitingForOk* auf *False* wodurch anderen Teilen des Programms signalisiert wird, dass neue Nachrichten gesendet werden können. Beim Empfangen einer Nachricht mit dem Inhalt **RTI** wird die Adresse ausgelesen und gespeichert falls sie noch nicht bekannt ist.

sendRTI: Dieser Thread sendet RTI Nachrichten und schläft dann 30-60 Sekunden.

Im Main Thread werden Benutzereingaben ausgewertet. Mit *exit* kann das Programm beendet werden. Mit *printtable* werden alle bekannten Adressen ausgegeben. Mit *send;ADDR;MESSAGE* wird eine Nachricht gesendet.

6 Quellen

www.pi4j.com
www.widora.io