Fachhochschule Nordwestschweiz

Dokumentation

Datenübertragung vom Pixhawk zum SBC

Auslesen der GPS-Daten vom Pixhawk über UART

17. Oktober 2016

Saner Kevin

Institut für Automation

1 Einleitung

Das folgenden Beispiel zeigt auf welche Schritte durchgeführt werden müssen um die Verbindung zum Pixhawk herzustellen, sowie was unternommen werden muss damit die Kommunikation mit Hilfe des Mavlink Protokolls hergestellt werden kann.

Voraussetzungen:

- Ubuntu Version 14.04 oder ähnlich
- QGroundControl unter Linux
- Pixhawk inkl. sämtlicher benötigter Peripherie
- 2 x FTDI-Kabel 3.3V
- Toolchain des Single Board Computer zum Cross-Kompilieren

2 QGroundControl

Um den Pixhawk in Betrieb zu nehmen muss auf dem Host-Computer ebenfalls die Applikation QGroundControl vorhanden sein. Die App kann als Imagedatei von QGroundControl heruntergeladen werden, was dessen Installation relativ einfach macht. Die heruntergeladene Datei sollte nun an sicheren Ort kopiert werden. Anschliessend müssen noch die Benutzerrechte geändert werden.

Command: $ chmod +x ./QGroundControl.AppImage

Dieser Befehl gibt dem momentanen Benutzer die Erlaubnis die Applikation auszuführen. Der folgende Befehl wiederum führt die Applikation schliesslich aus (oder doppelklicken).

Command: $ ./QGroundControl.AppImage

3 Inbetriebnahme des Pixhawk

3.1 Hardware

Bevor der Pixhawk ein erstes mal an der Computer angeschlossen werden kann, muss er zuerst entsprechend verkabelt werden. Denn einige Peripheriegerät sind zwingend notwendig für den Betrieb. Zur benötigten Peripherie gehören das GPS-Modul, der Taster sowie der Buzzer. Als Stromversorgung kann auch das mitgelieferte USB-Kabel verwendet werden, allerdings muss darauf geachtet werden, dass der verwendete USB-Port am Host-Computer die benötigte Spannung liefern kann. Für mehr Info: QUICK START GUIDE

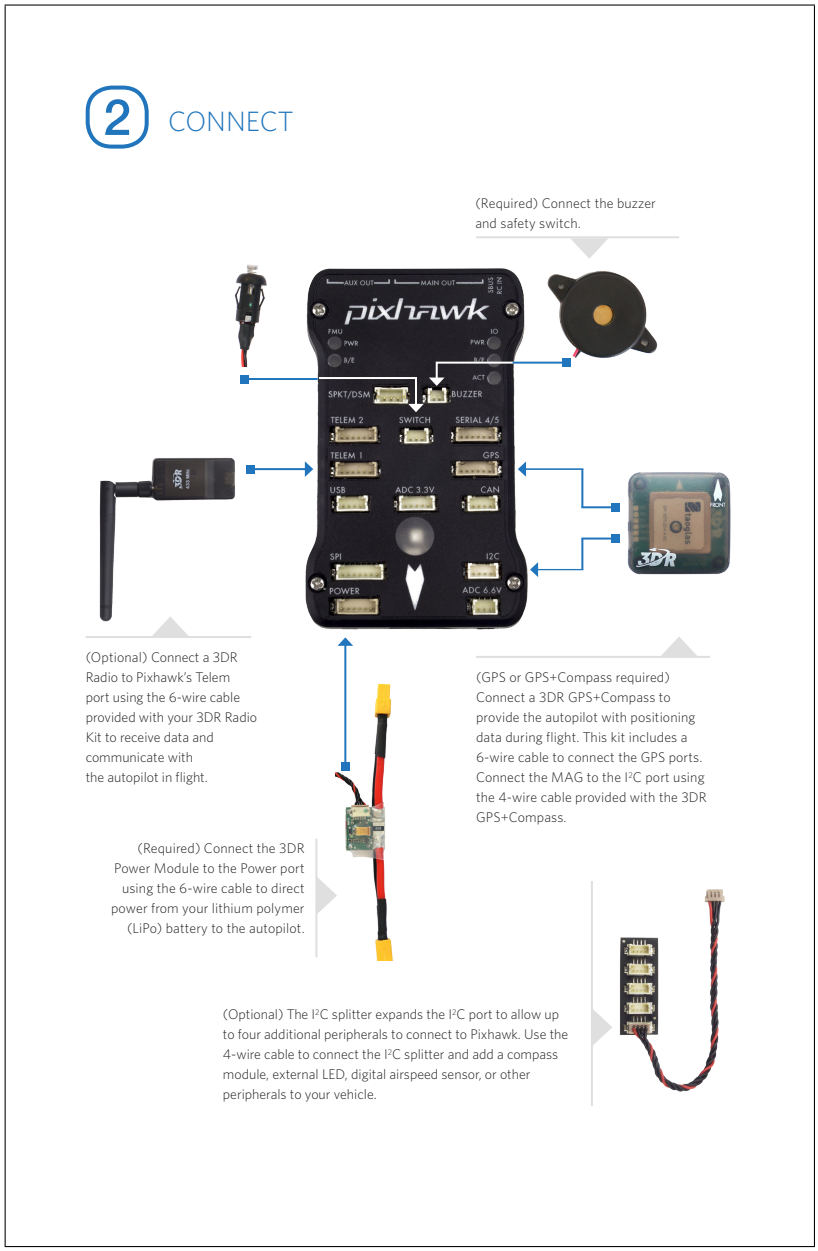


Abbildung 1: Verkabelung des Pixhawk

3.2 Software

Der Pixhawk ist nun Betriebsbereit. Um nun die Sensoren zu kalibrieren wird empfohlen zuerst ein Firmwareupdate durchzuführen. Dabei wird auch die später benötigte Mavlink-Anwendung installiert. Es ist darauf zu achten, dass die PX4-Firmware und nicht die ArduPilot-Firmware geladen wird. Nachdem die Firmware erfolgreich installiert wurde, müssen die Sensoren kalibriert sowie auch der Airframe ausgewählt werden.

QGroundControl führt dabei selbständig durch die Kalibration. Die Kalibration dient in erster Linie dazu die Maximalwerte der Beschleunigungssensoren zu bestimmen um die Lage

des Pixhawk zu definieren.

Um schliesslich überprüfen zu können ob alle Systeme einwandfrei funktionieren empfiehlt es sich den Pixhawk unter freiem Himmel in Betrieb zu nehmen, denn nur so kann garantiert werden, dass GPS und Kompass exakte Daten liefern. War die Inbetriebnahme erfolgreich wird nun die aktuell Position des Pixhawks sowie die Ausrichtung in der QGroundControl-Applikation angezeigt, die Status-LED pulsiert dabei grün.

4 Mavlink

Um Daten auf den Pixhawk zu senden bzw. zu empfangen wird das Kommunikationsprotokoll Mavlink (Micro Air Vehicle Communication Protocol) verwendet. Mavlink streamt dabei die Daten über eine serielle Schnittstelle zu einer Kontrollstation. Als Kontrollstation kann zum Beispiel die vorher schon erwähnte Software QGroundControl dienen.
Ziel der folgenden Arbeitsschritte ist es, die Mavlink-Messages mit einem C++-Programm interpretieren zu können. Diese Messages sollen dabei einerseits vom Host/Computer interpretiert werden können, in einem zweiten Arbeitsschritt soll das Programm aber für einen Phytec Single Board Computer kompiliert werden, damit dieser die Datenakquisition durchführen kann. Das verwendete C++-Programm basiert dabei grösstenteils auf dem UART-Beispiel auf Github.

4.1 Verkabelung

Um nun Daten vom Pixhawk auslesen zu können muss der Pixhawk zuerst entsprechend verkabelt werden. Erst durch diesen Schritt wird es möglich auf die Konsole des Pixhawk zuzugreifen. Wichtig ist dabei, dass ein 3.3V-FTDI-Kabel verwendet wird und kein 5V-FTDI-Kabel. Wahlweise kann der Pixhawk mit dem mitgelieferten USB-Kabel oder einer anderen Quelle mit Strom versorgt werden. Die Verkabelung muss nach folgendem Schema vorgenommen werden:

4.1.1 Verkabelung Konsole

Die Verkabelung der Konsole ist dabei nicht zwingend notwendig, kann aber hilfreich sein um sein um zu überprüfen ob das Mavlink-Protokoll auf dem verwendeten Port verfügbar ist.

Pixhawk	(Serial 4/5)	FTDI	
1	+5V (red)		N/C
2	S4 Tx		N/C
3	S4 Rx		N/C
4	S5 Tx	5	FTDI RX (yellow)
5	S5 Rx	4	FTDI TX (orange)
6	GND	1	FTDI GND (black)

Im folgenden Bild wird dies auch noch grafisch dargestellt:

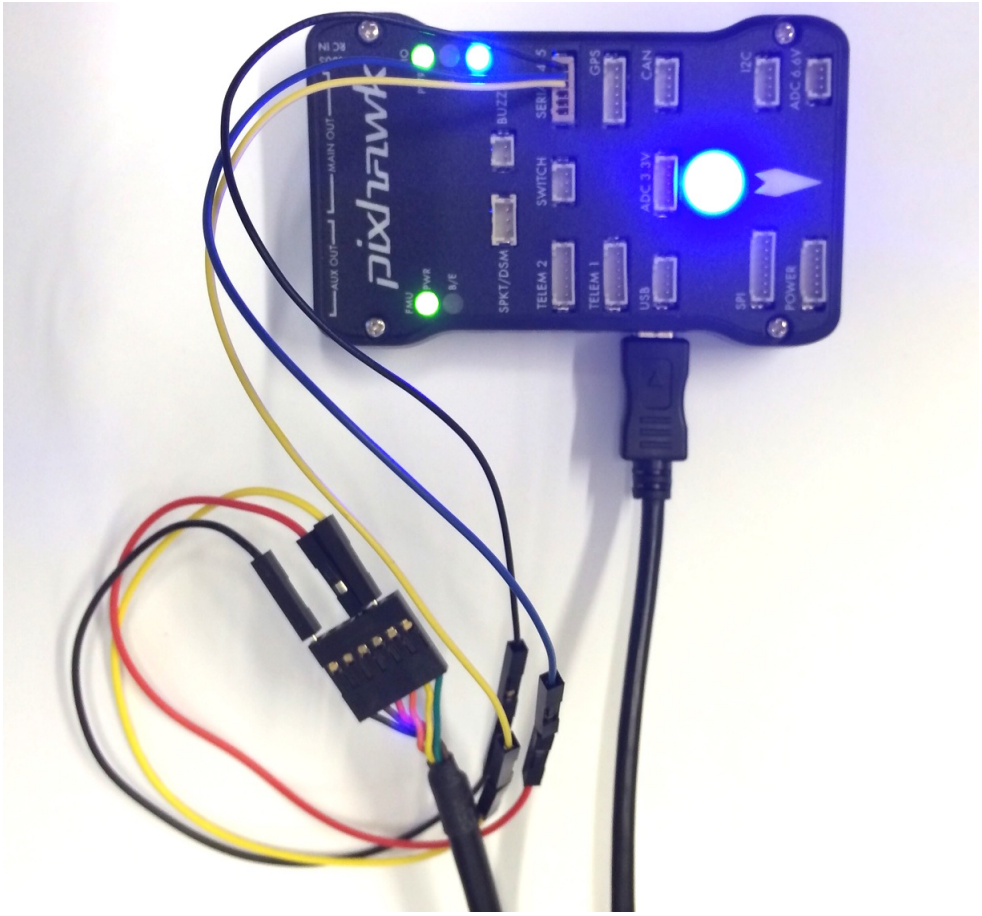


Abbildung 2: Verkabelung der Konsole

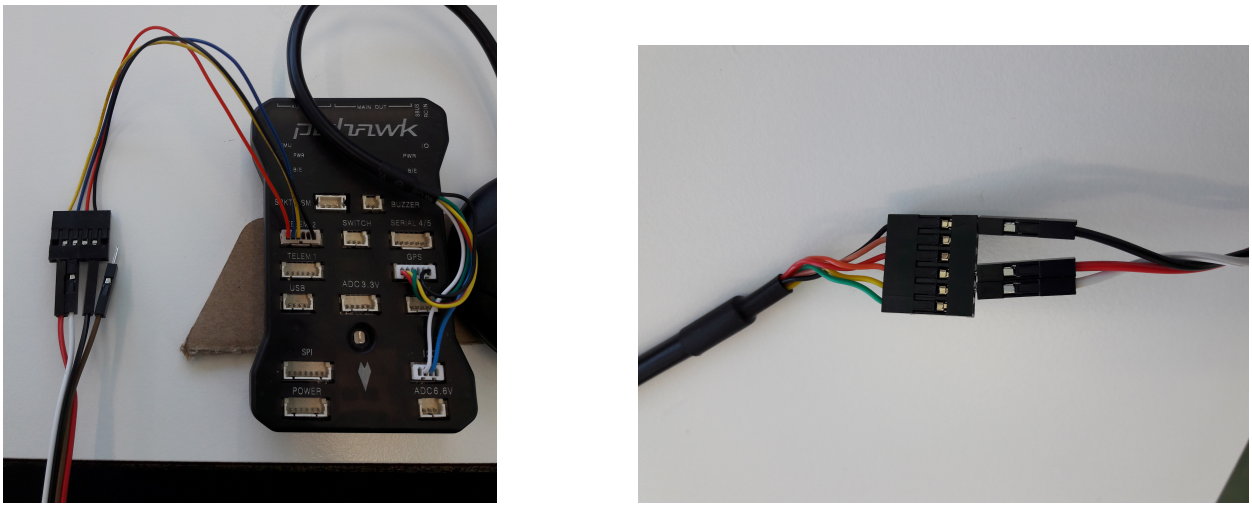
4.1.2 Verkabelung Mavlink-Kanal

Um schliesslich Mavlink-Meldungen übertragen zu können, muss ein Port, auf welchem Mavlink-Meldungen gestreamt werden verkabelt werden. Die geeigneten Port sind Telem 1 und 2 sowie der USB-Port.

TODO Wie die Verkabelung mit SBC und Pixhawk schliesslich aussieht muss noch ausgearbeitet werden. Dabei ist zu beachten, dass die Datenübertragung über die UART-Ports des Pixhawk (Telem 1 und 2) nicht immer reibungslos zu funktionieren scheint, wohingegen die Übertragung über das USB-Kabel praktisch immer funktioniert.

Pixhawk	Telem 1	FTDI	
1	+5V (red)		N/C
2	Tx (out)	5	FTDI RX (yellow)
3	Rx (in)	4	FTDI TX (orange)
4	CTS (in)		N/C
5	RTS (out)		N/C
6	GND	1	FTDI GND (black)

Im folgenden Bild wird dies auch noch grafisch dargestellt:



(a) (b)

Abbildung 3: Verkabelung der Mavlink-Schnittstelle

Kann der Telemetry-Port nicht verwendet werden, kann auch der serielle Port 4 verwendet werden. Dazu muss das FTDI-Kabel an die S4-Schnittstelle des Pixhawk angeschlossen werden (siehe Verkabelung Konsole). Diese könnte ausprobiert werden um die oben erwähnten Probleme mi der UART-Schnittstelle zu eliminieren.

4.2 Setup

Sind sämtliche Kabel angebracht entsteht das unten gezeigte Setup.

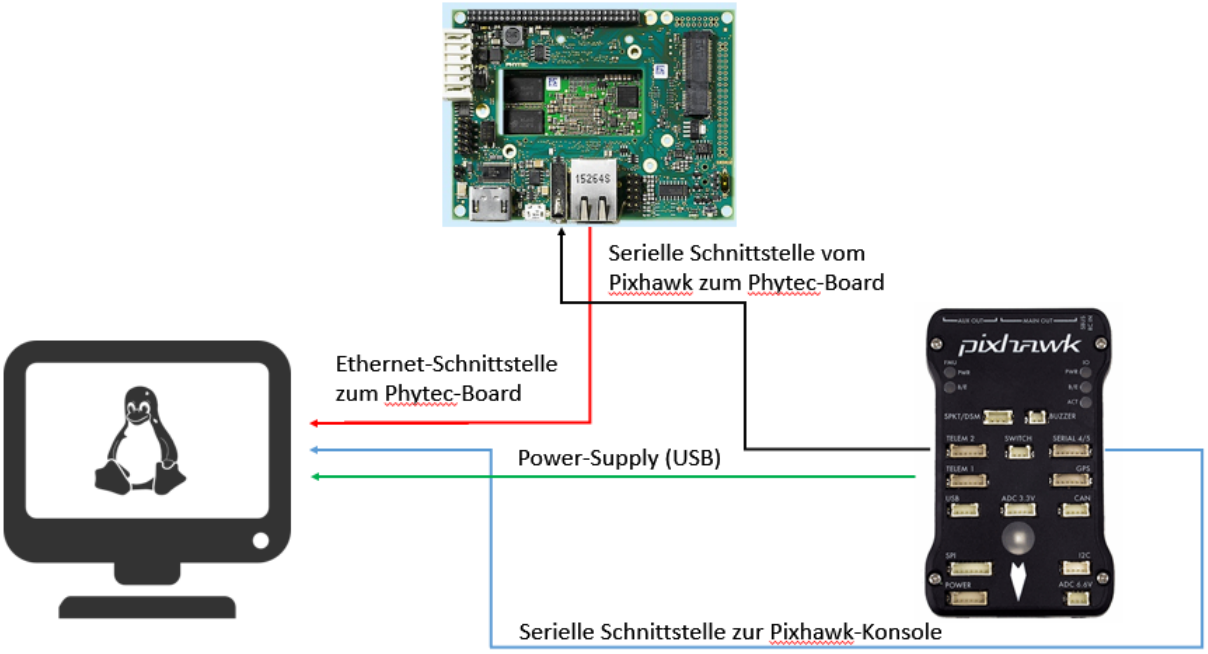


Abbildung 4: Verkabelung des Pixhawk

4.2.1 Einloggen auf Pixhawk-Kosole

Der Anwender sollte nun in der Lage sein, sich auf die Konsole des Pixhawks einzuloggen. Dazu muss folgender Befehl eingetippt werden:

Command: $ screen /dev/ttyUSB0 57600 8N1

Falls der Port-Name nicht gefunden wird, kann mit dem Befehl:

Command: $ ls /dev/ttyUSB*

nach allen verfügbaren Ports gesucht werden. Der Anwender ist nun über die serielle Schnittstelle auf dem Pixhawk eingeloggt.

Nun kann die Stromversorgung zum Pixhawk unterbrochen werden, damit dieser neu startet. Dabei werden unter anderen sämtliche Ports ausgegeben, auf welchen Mavlink empfangen werden kann. Ist der gewünschte Port nicht dabei, kann eine entsprechende Mavlink-Session über den jeweiligen Port wie folgt gestartet werden.

Command: $ nsh> mavlink start -d /dev/ttyACM0

Über welche Ports eine aktive Mavlink-Session läuft ist von Bedeutung wenn versucht wird die gesendeten Daten entgegen zu nehmen und zu verarbeiten. Das folgende Beispiel zeigt einen Ausschnitt des Start-up-Prozederes des Pixhawks, daraus ist ersichtlich über welche

Ports Mavlink-Messages gesendet bzw. empfangen werden.

```
INFO  [mavlink] mode: Normal, data rate: 1200 B/s on /dev/ttyS1 @ 57600B
INFO  [mavlink] mode: OSD, data rate: 1000 B/s on /dev/ttyS2 @ 57600B
INFO  [ver] match: PX4FMU\_V2
px4flow [165:100]
WARN  [px4flow] scanning I2C buses for device..
INFO  [mavlink] mode: Config, data rate: 800000 B/s on /dev/ttyACM0 @ 57600B
```

Abbildung 5: Programm Output

4.2.2 C++-UART-Mavlink-Schnittstelle

Wie schon erwähnt basiert das Programm auf dem UART-Interface-Beispiel verfügbar unter GitHub. Die entsprechenden Files können mit dem Befehl:

Command: $ git clone https://github.com/mavlink/c_uart_interface_example.git

auf dem Host-Computer kopiert werden. Die gesamten Definitionen des Mavlink-Protokoll können unter folgendem Link gefunden werden Mavlink. Da sind sämtliche Messages aufgeführt die vom Pixhawk gesendet werden können, und ebenso deren Definition.

Programmablauf

Das Programm wurde dann noch mit ein paar Funktionen ergänzt, damit das Programm im FindMine-Projekt verwendet werden kann. Der Programmablauf sieht aktuell wie folgt aus: Zuerst versucht das Programm den ausgewählten Port zu initialisieren. Der Standard-Port ist dabei USB0 mit einer Baudrate von 57600. Um einen anderen Port zu verwenden können dem Programm Parameter übergeben werden, muss nicht extra der Code geändert werden. Nun wird versucht die System-IDs, des Pixhawk zu erfahren. Als nächstes wird vom Pixhawk eine Home-Position erwartet. Auf diese wird solange gewartet bis der Pixhawk ein GPS-Signal findet. Damit man das Programm auch ohne GPS-Log debuggen kann, kann dieser Schritt übersprungen werden. Allerdings erhält man so nur Dummy-Daten. Es ist zu beachten, dass das GPS-Signal nur empfangen wird wenn man sich unter freiem Himmel befindet.
Als nächstes sendet der Host-Computer, eine Zielposition an den Pixhawk, die dieser dann zu erreichen versucht. Diese Funktion (enable/disable offboard control) wird wahrscheinlich zu einen später Zeitpunkt nicht mehr verwendet und kann entfernt werden.
Nun versucht der Host-PC Daten vom Pixhawk zu lesen. Da das Lesen der Daten, insbesondere der Positionsdaten, die Hauptaufgabe des Programms ist wird dies nun in einer Endlosschlaufe ausgeführt. Der Main-Loop wird momentan maximal im Sekundentakt, also mit 1 Hz ausgeführt. Diese Frequenz muss noch optimiert werden. Der Pixhawk liefert die Daten mit einer Maximalfrequenz von 10 Hz, dies ist also das Maximum, dass erreicht werden kann. Dabei werden ähnlich dem SBGC-Interface, die Daten in ein CSV-File geschrieben. Die Positionsdaten, können dabei als relative wie auch als globale Werte empfangen werden. Die

Positionsdaten werden zudem mit dem Zeitstempel des Pixhawk wie auch des SBCs ergänzt. Der Zeitstempel des Pixhawk entspricht dabei i.d.R. der seit Systemstart vergangenen Zeit in Mikrosekunden (siehe HIGHRES_IMU). Die Art wie der Zeitstempel des Pixhawk kreiert wird entspricht der des SBGC-Interface.
Zur Zeit werden auch noch die restlichen Daten die vom Pixhawk gesendet werden in ein File abgespeichert. Dies geschieht bei jedem 50 Durchlauf des Main-Loop. Stellt sich heraus, dass diese Daten ebenfalls nie verwendet werden, kann dieser Programmteil ebenfalls entfernt werden.

Kompilation
Das Programm kann unter Ubuntu mit den im Projekt enthaltenen Make-Files problemlos kompiliert werden. Um das Programm für den SBC kompilieren zu können, müssen die Anweisungen im Cross-Compilation-Guide befolgt werden.

Um das eben erstellte Programm zu starten muss der folgende Befehl ausgeführt werden:

Command: $ ./mavlink_control -d /dev/ttyACM0

4.2.3 Ergebnis

Damit das Programm nun auf dem Phytec SBC ausgeführt werden kann, muss das entsprechende Setup erstellt werden (siehe 4.2). Zudem muss der entsprechende Port ausgewählt werden, ist der Port nicht verfügbar bricht das Programm mit einer Exception ab. Ist alles korrekt konfiguriert können der Pixhawk und der Phytec SBC nun über Mavlink kommunizieren, dass heisst es können Kommandos gesendet wie auch empfangen werden.

Das Programm kann mit demselben Befehl wie beim Starten vom Host-Computer ausgeführt werden (siehe 8). Unter der Bedingung, dass der Pixhawk ebenfalls korrekt initialisiert und über GPS-Signal verfügt, wird nun ein File mit den erhaltenen GPS-Daten sowie den IMU-Daten erstellt, dass in das Post-Processing miteinbezogen werden kann.

Literaturverzeichnis

[1] Pixhawk Autopilot: *Quick Start Guide* [online] Available at: https://3dr.com/wp-content/uploads/2014/03/pixhawk-manual-rev7.pdf [Zugriff am 28.02.2017]

[2] Lorenz Meier: *C-UART Interface Example* [online] Available at: https://github.com/mavlink/c_uart_interface_example [Zugriff am 28.02.2017]

[3] ETHZ: *Mavlink Common Messages Set* [online] Available at: https://pixhawk.ethz.ch/mavlink/e [Zugriff am 28.02.2017]