Fachhochschule Nordwestschweiz

ZWISCHENBERICHT

Datenübertragung mit dem Pixhawk über die UART-Schnittstelle

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 Ubuntu

Als Host-Computer wird am besten ein Computer mit Ubuntu 14.04 oder einem ähnlichen Betriebssystem eingesetzt. So gestaltet sich auch die Herstellung der Verbindung zum Pixhawk am einfachsten. Wird ein dabei ein SBC von Phytec eingesetzt empfiehlt es sich alle Arbeitsschritte mit dem mitgelieferten Betriebssystem durchzuführen. Beim eingesetzten Betriebssystem muss sichergestellt werden, dass das Package „screen“ installiert ist. Ist dieses Package noch nicht vorhanden kann es mit dem untenstehenden Befehl nachinstalliert werden. Dieses Package wird benötigt um auf die Pixhawk-Konsole zugreifen zu können (siehe :

Command: $ sudo apt-get install screen

3 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

4 Inbetriebnahme des Pixhawk

4.1 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

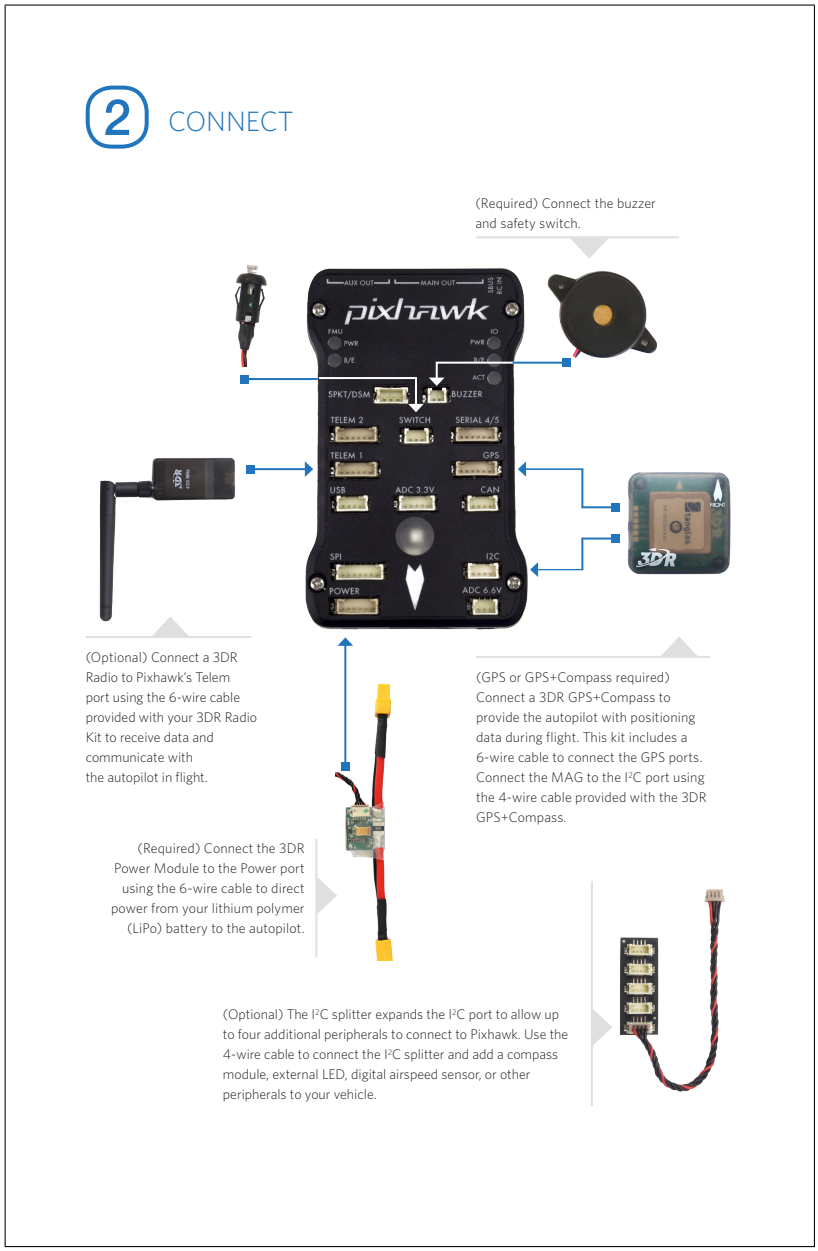


Abbildung 1: Verkabelung des Pixhawk

4.2 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.

5 Mavlink

Um Daten auf den Pixhawk zu senden bzw. zu empfangen wird das Kommunikationsprotokoll Mavlink (Micro Air Vehicle Communication Protocol) verwendet. Mavlink sendet dabei C-structs 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.

5.1 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-Kabel verwendet wird und kein 5V-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:

Verkabelung Konsole:

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:

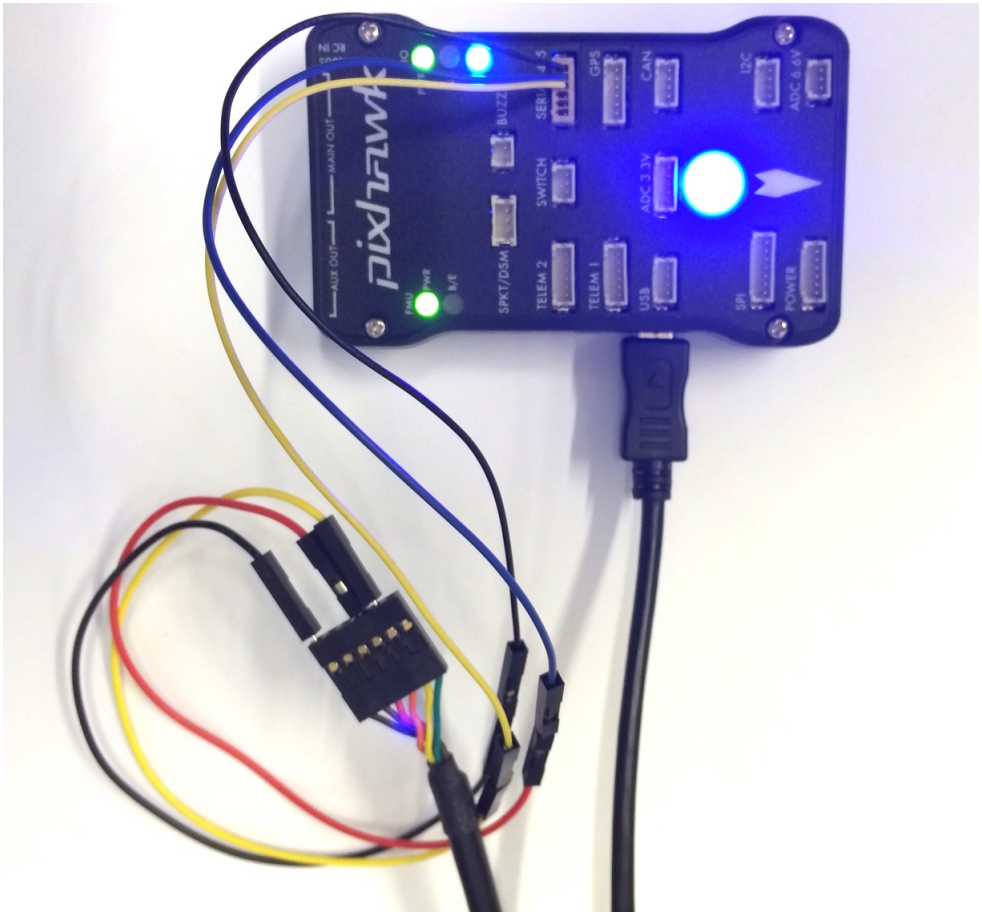
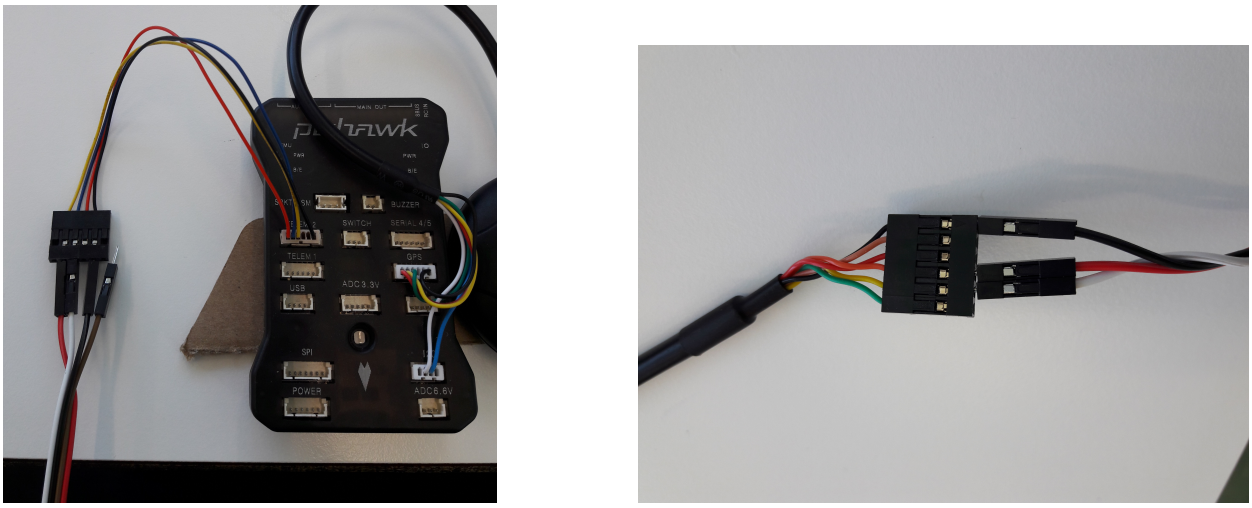


Abbildung 2: Verkabelung der Konsole

Verkabelung Mavlink-Kanal:

Pixhawk	Telem 2	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).

5.2 Setup

Sind sämtliche Kabel angebracht entsteht folgendes Setup:

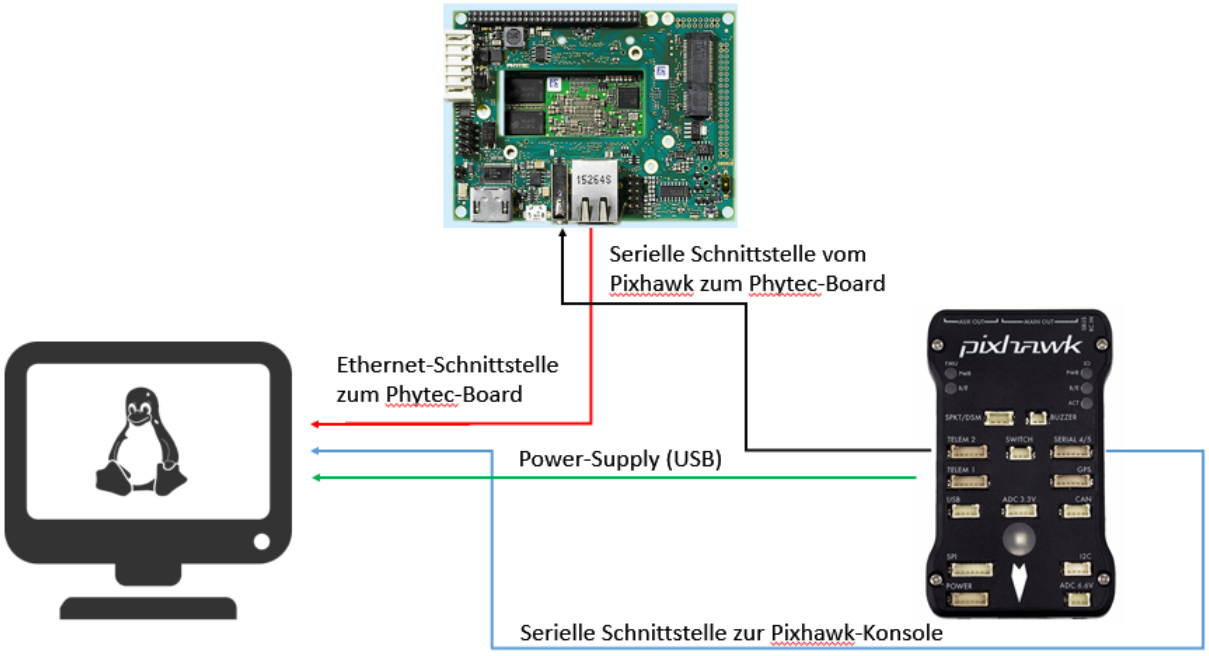


Abbildung 4: Verkabelung des Pixhawk

5.3 Test des Setups

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

5.3.1 C++-Mavlink-Schnittstelle

Das Programm welches verwendet wird ist auf GitHub frei verfügbar. 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. Nun muss das Programm erst einmal kompiliert

werden. Um das Programm sowie die Verbindungen zum Pixhawk zu testen sollte das Programm zuerst lokal kompiliert werden (ohne Cross-Compiler). Der folgende Befehl kompiliert das Programm zu einem Binary:

Command: $ cd /path/to/directory/c_uart_interface_example
Command: $ make

Der Befehl „make“ hat nun mithilfe der, im Verzeichnis vorhandenen „Makefiles“ ein Executable erstellt. mit welchen bereits getestet werden kann ob, die Verkabelung funktioniert und ob der Pixhawk korrekt initialisiert wurde.
Um das eben erstellt Programm zu starten muss der folgende Befehl ausgeführt werden:

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

Sieht der Output dabei wie folgt aus konnte die Verbindung erfolgreich hergestellt werden.

```
OPEN PORT
Connected to /dev/ttyUSB0 with 57600 baud, 8 data bits, no parity, 1 stop bit
(8N1)

START READ THREAD

CHECK FOR HEARTBEAT
Found

GOT VEHICLE SYSTEM ID: 1
GOT AUTOPILOT COMPONENT ID: 50

INITIAL POSITION XYZ = [ 8.2935 , -1.1447 , -0.7609 ]
INITIAL POSITION YAW = 2.1539

START WRITE THREAD

ENABLE OFFBOARD MODE

SEND OFFBOARD COMMANDS
POSITION SETPOINT XYZ = [ 3.2935 , -6.1447 , -0.7609 ]
POSITION SETPOINT YAW = 2.1539
0 CURRENT POSITION XYZ = [  8.2935 , -1.1447 , -0.7609 ]
1 CURRENT POSITION XYZ = [  8.2935 , -1.1447 , -0.7609 ]
2 CURRENT POSITION XYZ = [  8.2524 , -1.1444 , -0.7667 ]
3 CURRENT POSITION XYZ = [  8.2205 , -1.1431 , -0.7747 ]
4 CURRENT POSITION XYZ = [  8.1920 , -1.1421 , -0.7737 ]
5 CURRENT POSITION XYZ = [  8.1920 , -1.1421 , -0.7737 ]
6 CURRENT POSITION XYZ = [  8.1539 , -1.1414 , -0.7847 ]
7 CURRENT POSITION XYZ = [  8.1522 , -1.1417 , -0.7820 ]

DISABLE OFFBOARD MODE

READ SOME MESSAGES
Got message LOCAL_POSITION_NED (spec:
https://pixhawk.ethz.ch/mavlink/#LOCAL_POSITION_NED)
    pos  (NED):  8.152975 -1.141093 -0.784075 (m)
Got message HIGHRES_IMU (spec: https://pixhawk.ethz.ch/mavlink/#HIGHRES_IMU)
    ap time:     3611390110
    acc  (NED):   0.005503  0.044659 -9.740363 (m/s^2)
    gyro (NED):  -0.003064  0.003857  0.000005 (rad/s)
    mag  (NED):  -0.117767 -0.335362 -0.253204 (Ga)
    baro:        1020.519958 (mBar)
    altitude:    -60.341393 (m)
    temperature: 46.779999 C

CLOSE THREADS

CLOSE POR
```

Abbildung 6: Programm Output

Der Port „ttyACM0“ entspricht dabei dem USB-Port des Pixhawk, um nun auch die serielle Verbindung über das FTDI-Kabel zu testen muss der Mavlink-Port (Device: -d) zu „ttyUSB*“ geändert werden. Ist der Output derselbe funktioniert auch diese Verbindung.

5.4 Cross-Compiling

Da der Phytec Single Board Computer über eine andere Architektur verfügt als ein normaler Desktop-Computer muss das Programm speziell dafür kompiliert werden. Der Phytec verfügt über die, für embedded Systeme, typische ARM-Architektur. Alle benötigten Tools für das Cross-Compiling sind dabei auf dem mitgelieferten USB-Stick.

5.4.1 Eclipse

Da die Entwicklungsumgebung Eclipse bereits auf dem Phytec USB-Stick vorhanden ist, empfiehlt es diese auch zu verwenden. Eine Anleitung wie diese für das Entwickeln von C-Programmen konfiguriert werden muss, befindet sich ebenfalls auf dem USB-Stick (Application Guide). Um zu überprüfen, ob die Toolchain und der Compiler korrekt funktionieren sollte zuerst ein einfaches C-Programm erstellt werden. Eine komplette Anleitung um ein „Hello World“ in C zu entwickeln findet man dabei im „Application Guide“ (S. 10 - 23) von Phytec.

5.4.2 C++-Compiling

Das zu kompilierende Mavlink-Interface wurde dabei in C++ geschrieben, was es um einiges komplizierter macht dieses zu kompilieren. Die folgende Anleitung soll Schritt für Schritt aufzeigen was unternommen werden muss, um ein auf dem Phytec lauffähiges Executable des Mavlink-Interfaces zu erhalten.

Projekt erstellen: Für die Entwicklung eines C++-Programms muss in Eclipse ein neues C++-Projekt erstellt und benannt werden. Ein leeres Projekt ist dabei am besten geeignet.

Build-Pfade: Damit für das Mavlink-Interface sämtliche Header-Files zur Verfügung stehen, muss das entsprechende Verzeichnis im Projekt miteinbezogen werden. Mit rechtsklickenäuf das Projekt können unter „Properties -> C/C++Build -> Settings -> Includes“ Build-Pfade zum Projekt hinzugefügt werden. Für das Mavlink-Interface muss das Verzeichnis „/path/to/dir/mavlink/include/mavlink/v1.0“ hinzugefügt werden.

C++-Files: Als nächster Schritt können alle C++-Files zum Projekt hinzugefügt werden. Die, für das Mavlink-Interface, benötigten Files sind „autopilot_interface.cpp“, „mavlink_control.cpp“ und „serial_port.cpp“ und deren Header-Files. Treten dabei keine Fehler in

den C++-Files auf, ist dies ein Zeichen dafür, dass bis zu diesem Schritt alles richtig gemacht wurde.

Konfiguration: Da das Mavlink-Interface-Programm ein sogenanntes „multithreaded“ Programm ist, muss die Bibliothek „pthread“ ins Programm eingebunden werden. Dazu muss unter „Properties -> C/C++ Build -> Settings -> GCC C++ Linker -> Libraries“ „pthread“ hinzugefügt werden.

Die Konfiguration des „GCC C Compiler“ erfolgt gleich wie zur Entwicklung von C-Programmen. Als „Command“ muss lediglich ${CC} eingegeben werden. Auf die gleiche Art kann auch der „GCC C++ Compiler“ konfiguriert werden.

Als Linker kann jedoch nicht der GCC-Linker verwendet werden. Dazu muss ein G++-Linker verwendet werden. Dazu muss der komplette Pfad des Linker eingebunden werden. Somit muss zuerst nach dem G++-Compiler gesucht werden. Das gesucht File heisst „arm-phytec-linux-gnueabi-g++“ und befindet sich normalerweise im Ordner „/opt/yogurt/iMX6-PD15.2.0/sysroot/x86_64-yogurtsdk-linux/usr/bin/arm-phytec-gnueabi“. Wie im „Application Guide“ beschrieben muss auch hier in den Linker-Settings ${LDFLAGS} ergänzt werden.

Als letztes muss auch noch der „GCC Assembler“ konfiguriert werden. Dazu muss auch lediglich, wie im „Application Guide“ beschrieben, unter „Command“ ${AS} eingegeben werden.

Nun sollte durch „Build Project“ erfolgreich ein Binary (Executable) hergestellt werden können.

Post-build: Um das Programm sogleich auf dem Phytec ausführen zu können, können die Post-build steps so konfiguriert werden, dass das erstellt Executable sogleich auf den Phytec geladen und ausgeführt wird. Dazu muss folgendes Command ergänzt werden: „scp ./NameOfExec root@192.168.3.11:/. ;ssh root@192.168.3.11 /NameOfExec“. Das Executable wird so im Root-Verzeichnis des SBCs ausgeführt.

Achtung: Die Netzwerkumgebung muss natürlich zuerst richtig konfiguriert werden, damit die Post-build steps korrekt ausgeführt werden.

Konfiguration Phytec: Wird zum ersten Mal versucht das erstellte Executable auszuführen, wird wahrscheinlich eine Fehlermeldung angezeigt. Nämlich dass das File nicht gefunden wird. Diese Fehlermeldung wird ausgegeben da der zum Ausführen des Programms benötigte Interpreter nicht gefunden wird. Der Name des benötigten Interpreters erfährt man durch die folgende Befehlszeileneingabe:

Command: $ readelf -l NameOfExec

Damit das Programm ausgeführt werden kann, muss das in der Zeile „Requesting program interpreter“ File im richtigen Verzeichnis vorhanden sein. Sucht man dieses File auf dem Phytec SBC wird man feststellen, dass dieses nicht vorhanden ist. Allerdings existiert ein File mit einem ähnlichen Namen, wie zum Beispiel „ld-linux-armhf.so.3“. Dies ist der auf dem Phytec vorhandene Interpreter oder zumindest zeigt dieses File auf den Interpreter. Um nun den Interpreter dem erstellten Programm zugänglich zu machen muss ein symbolischer Link erstellt werden, der auf das File „ld-linux-armhf.so.3“ zeigt. Der Link kann wie folgt erstellt werden:

Command: $ ln -s /lib/ld-linux-armhf.so.3 /lib/ld-linux.so.3

Nun sollte das erstellte Programm ausgeführt werden können.

5.4.3 Ergebnis

Nach dem Post-build wird das erstellte Programm auf dem Phytec ausgeführt, jedoch, in der Regel, sogleich wieder durch eine Exception abgebrochen. Die passiert weil das Standard-Device (/dev/ttyUSB0) nicht verfügbar ist. Ist jedoch die erste Programmzeile „OPEN PORT“ sichtbar, ist das Programm lauffähig. Nun muss lediglich der Pixhawk wie in Kapitel 5.2 erklärt an den SBC angeschlossen werden. Somit können der Pixhawk und der Phytec SBC nun über Mavlink kommunizieren, dass heisst es können Kommandos gesendet wie auch empfangen werden.

Das Programm kann nach der korrekten Verkabelung auf dem Phytec SBC ausgeführt werden, dazu kann derselbe Befehl wie beim Starten vom Host-Computer verwendet werden (siehe 7). Ist der Programmoutput ebenfalls derselbe wie unter bei Abb. 6 wurde das Setup richtig erstellt so wie sämtliche Schritte zur Kompilation des Programms korrekt ausgeführt.

Literaturverzeichnis

[1] Prosys OPC UA Java SDK: *Preisliste* [online] Available at: https://downloads.prosysopc.com/opcua/Prosys_OPC_UA_Java_SDK_Price_List.pdf [Zugriff am 21.06.2016]