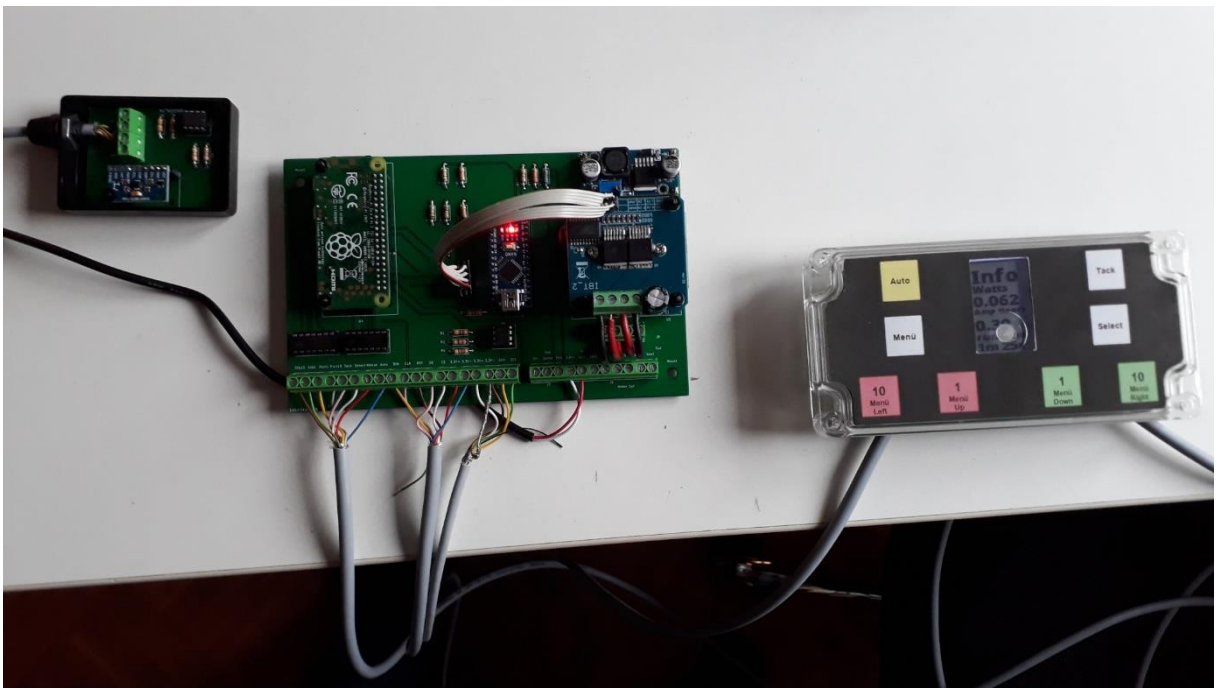
# Pypilot – Autopilot selbstgebaut




Basierend auf den Entwicklungen von:

Sean D´Epagnier: https://pypilot.org/

sowie

Timo Birnschein: https://hackaday.io/project/168592-opencpn-chart-plotter-w-autopilot-and-waypoints

Ich übernehme keinerlei Haftung für Funktion und Richtigkeit
Nachbau und Betrieb erfolgt auf eigene Gefahr

# 1 Einführung

Autopiloten entlasten auf langen wie auf kurzen Streckenabschnitten die Crew.
Es gibt eine Vielzahl an verschiedenen elektrischen Modellen unterschiedlichster Preisklassen auf dem Markt.

Oft werden sogenannte „Pinnenpiloten“ genutzt. Diese sind relativ preiswert und lassen sich ohne größeren Aufwand an Bord einbauen und integrieren.
Bei diesen Modellen gibt es einen gravierenden Nachteil.
Die eingebauten Compassmodule sind nicht oder nur ungenügend gegenüber den Bootsbewegungen entkoppelt. Dazu kommt, dass der Nutzer nur eingeschränkten Zugriff auf die Regelparameter hat. Das führt dazu, dass diese Autopiloten bei raueren Bedingungen mit höheren (insbesondere nachlaufenden von achtern kommenden) Wellen nur noch eingeschränkt funktionieren oder ganz versagen.

Will man einen Autopiloten der auch bei den zuvor genannten Bedingungen noch gut funktioniert muss man zu einem Einbaupiloten greifen, die deutlich teuer und meist mit etwas Aufwand an Bord zu installieren sind.

Hier setzt nun Sean an und hat auf Basis von einem Raspberry Pi und dem Open-Source-Kartenplotterprogramm OpenCPN eine Lösung entwickelt die im Steuerverhalten einem guten Einbaupiloten nahekommt aber bei Selbstbau deutlich weniger kostet als dieser.

Dieser Autopilot genannt „pypilot“ ist sehr universell gehalten, die Umsetzung ist vielfältig möglich.

Mit dieser Anleitung wird eine mögliche Umsetzung beschrieben.
Nachbauer können die folgend beschriebene Lösung nachbauen oder als Orientierung für eigene Lösungen und Umsetzungen nutzen.

Die folgende Anleitung beschreibt den Bau und die Einbindung in OpenCPN eines pypiloten.
Dabei wird der Autopilot auf einem eigenen kleinen Rechnersystem aufgebaut welches dann per Wifi an den Rechner mit OpenCPN angebunden ist. Kontrolle und Anzeige erfolgt dann in OpenCPN über ein entsprechendes plugin.

Diese Lösung wurde gewählt weil:
Der Rechner (Raspberry) mit OpenCPN nicht zusätzlich belastet wird, somit stabiler und schneller läuft.
Bei einem möglichen Rechnerdefekt der jeweils andere Rechner noch weiter läuft und genutzt werden kann. Ein Rechnerdefekt führt somit nicht zu einem Totalausfall.

Zum Nachbau des hier beschriebenen Systems sind keine Elektronik-Kenntnisse notwendig.
Ebenfalls nicht nötig sind Kenntnisse im Programmieren von Arduino oder Raspberry.

Grundsätzlich ist Fingerfertigkeit, Geschick und Freude am Basteln hilfreich.

Sämtliche benötigte Software ist auf github eingestellt und kann von dort heruntergeladen werden.
Im Kapitel 7 sind die entsprechenden Links hinterlegt.

# 2 Grundlagen

Ein pypilot-System besteht grundsätzlich aus verschiedenen parallel laufenden Programmen/Prozessen.

pypilot
Das Programm wertet die Sensordaten aus, beinhaltet den Regelalgorithmus und generiert die Steuersignale für den Motortreiber.
Man kann sagen dieses Programm ist der „Server“ oder „Master“ des Autopiloten.

Motortreiber
Dieses Programm dient dazu, die Steuersignale des Programms pypilot in eine Ansteuerung für einen Motor als Aktuator für die Steuerung des Bootes umzusetzen.
Darüber hinaus werden noch zusätzliche Informationen (Stromaufnahme Motor, Temperatur) erfasst und an pypilot zurückgemeldet.
Um den Rechner mit dem Programm pypilot zu entlasten erfolgt diese Abarbeitung mit Hilfe eines Arduinos (Mikroprozessor)
Die Kommunikation zwischen pypilot und dem Motortreiber erfolgt über eine serielle Schnittstelle.
Als Motor kann bei Pinnensteuerung ein Linearaktuator eingesetzt werden.
Bei Radsteuerung ist die Verwendung eines entsprechenden Motors wie bei den originalen Autopiloten möglich.

plugin OpenCPN
Um pypilot zu steuern und die Informationen in OpenCPN anzeigen zu lassen gibt es ein entsprechendes Plugin. Die Kommunikation zw. Pypilot und OpenCPN erfolgt über Netzwerkprotokolle. In der Regel erfolgt die Anbindung zwischen den verschiedenen Computern über Wifi (WLAN).

# 3 Beschreibung des verwendeten Systems

Bei der hier vorgestellten Lösung wird der Autopilot auf einem eigenen kleinen Rechnersystem aufgebaut.
Das Rechnersystem auf dem OpenCPN läuft ist davon getrennt und ist per Wifi(WLAN) mit dem Autopilotrechner verbunden.

Als Rechner für den Autopiloten wird ein RaspberryPi Zero W verwendet.
Um das kleine Raspberry Modell nicht zu überlasten wird eine spezielle Linux-Variante als Betriebssystem –Tinycore Linux- verwendet.
Die auf Tinycore Linux basierende Variante des Programms nennt sich „Tinypilot“.

Als Motortreiber kommt ein Arduino-Nano zum Einsatz. Dieser ist mittels serieller Schnittstelle an den Raspberry angebunden.
Die Motorsteuerung erfolgt über PWM-Signale und eine sogenannte H-Brücke (hier als IBT-2 Baustein). Vereinfacht gesagt können damit die hohen Motorströme über den Arduino geschaltet werden.
Über den Arduino werden Spannung, Stromaufnahme des Motors und Temperatur der IBT-2 Brücke erfasst und an den Raspberry übermittelt.

Der Raspberry Zero und der Ardunio einschließlich der IBT-2 Brücke werden auf der Hauptplatine angeordnet. Diese Platine passt von den Abmessungen in ein Universalgehäuse 120x200 mm.

An dem Raspberry Zero ist über I2C ein 9-Achsen Gyroskop und Kompassmodul angeschlossen.
Das Kompassmodul ist auf einer eigenen kleinen Platine untergebracht. So kann es getrennt von der Hauptplatine (und magnetischen Störungen) montiert werden.
Um längere Kabelstrecken (>2m) zu überbrücken ist ein sogenannter I2C Extender vorgesehen.

Zur Bedienung und Anzeige ist ebenfalls eine eigene Platine vorgesehen. Als Schalter kommen Touchmodule zum Einsatz, so kann die ganze Einheit in ein Gehäuse mit transparentem Deckel untergebracht werden und ist entsprechend wasserdicht.

Weiterhin ist eine Bedienung und Anzeige über das OpenCPN Plugin möglich.
Von dem OpenCPN-Rechner werden –sofern vorhanden-Wind- und GPS-Daten bereitgestellt und übermittelt.

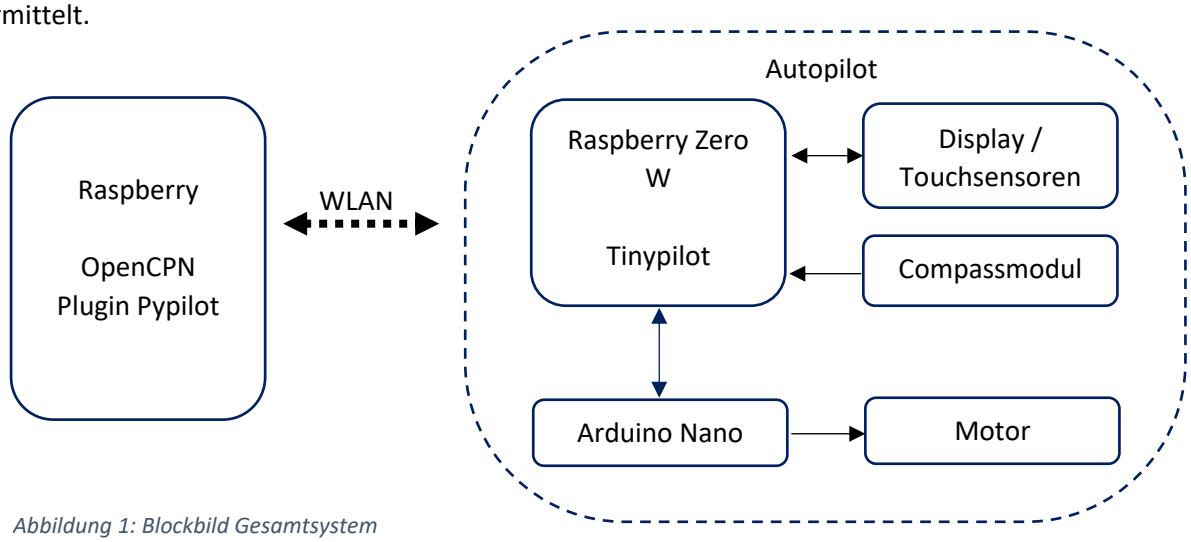



*Abbildung 1: Blockbild Gesamtsystem*

# 4 Aufbau Hardware

Allgemein:
Die vorgestellte Hardware lässt sich einfach nachbauen.
Durch Verwendung von Fassungen für ICs bzw. Steckverbinder für die Microcomputer sind die Komponenten thermisch relativ unkritisch.
Lötstellen sollten dennoch nicht länger als 10 sek erhitzt werden.
Für Lötanfänger gibt es diverse Tutorials im www, man kann mit einem Stück Lochrasterplatine und Drahtstückchen vorher üben.
Ich empfehle die Verwendung einer feinen Lötspitze <2mm Breite.

Falls man eine Lötstelle mit zu viel Lötzinn gelötet hat kann man mit Entlötlitze oder einer Entlötpumpe das überflüssige Lötzinn absaugen.
Notfalls geht auch ein Stück normales abisoliertes Kupferkabel (Litze), mit dem man unter Erhitzen überflüssiges Lötzinn „absaugt“.

## 4.1 Mainboard

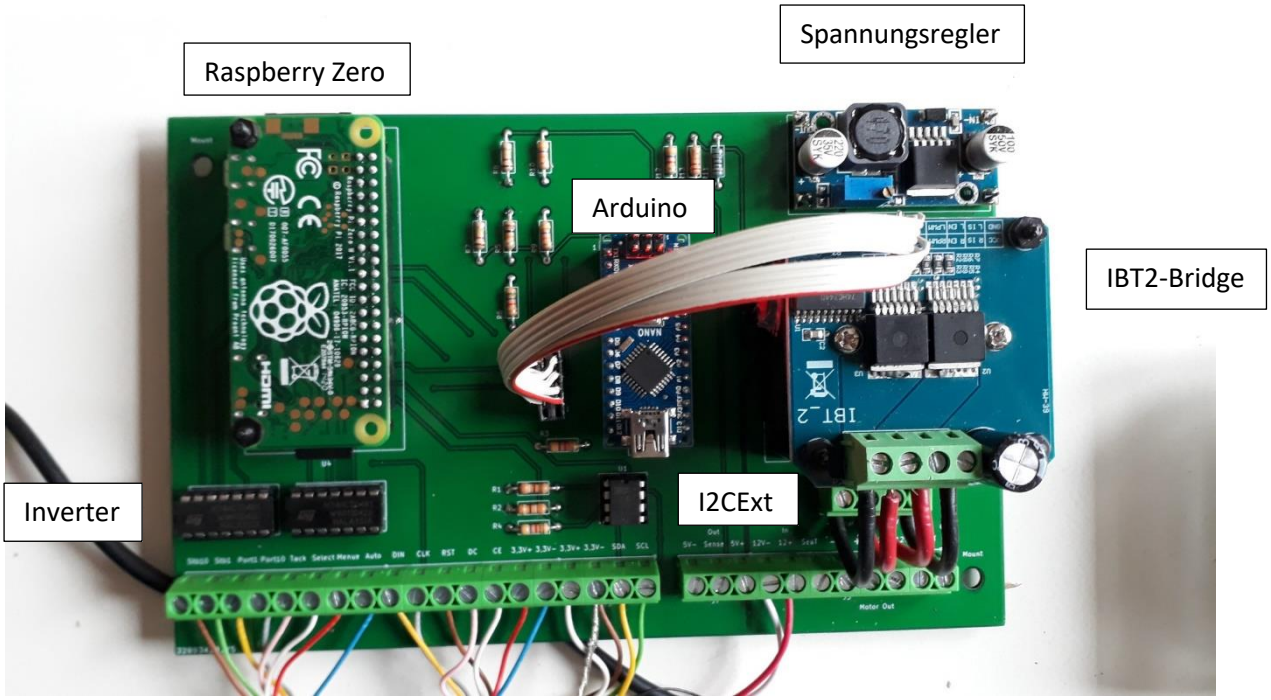



*Abbildung 2: Mainboard Hauptkomponenten*

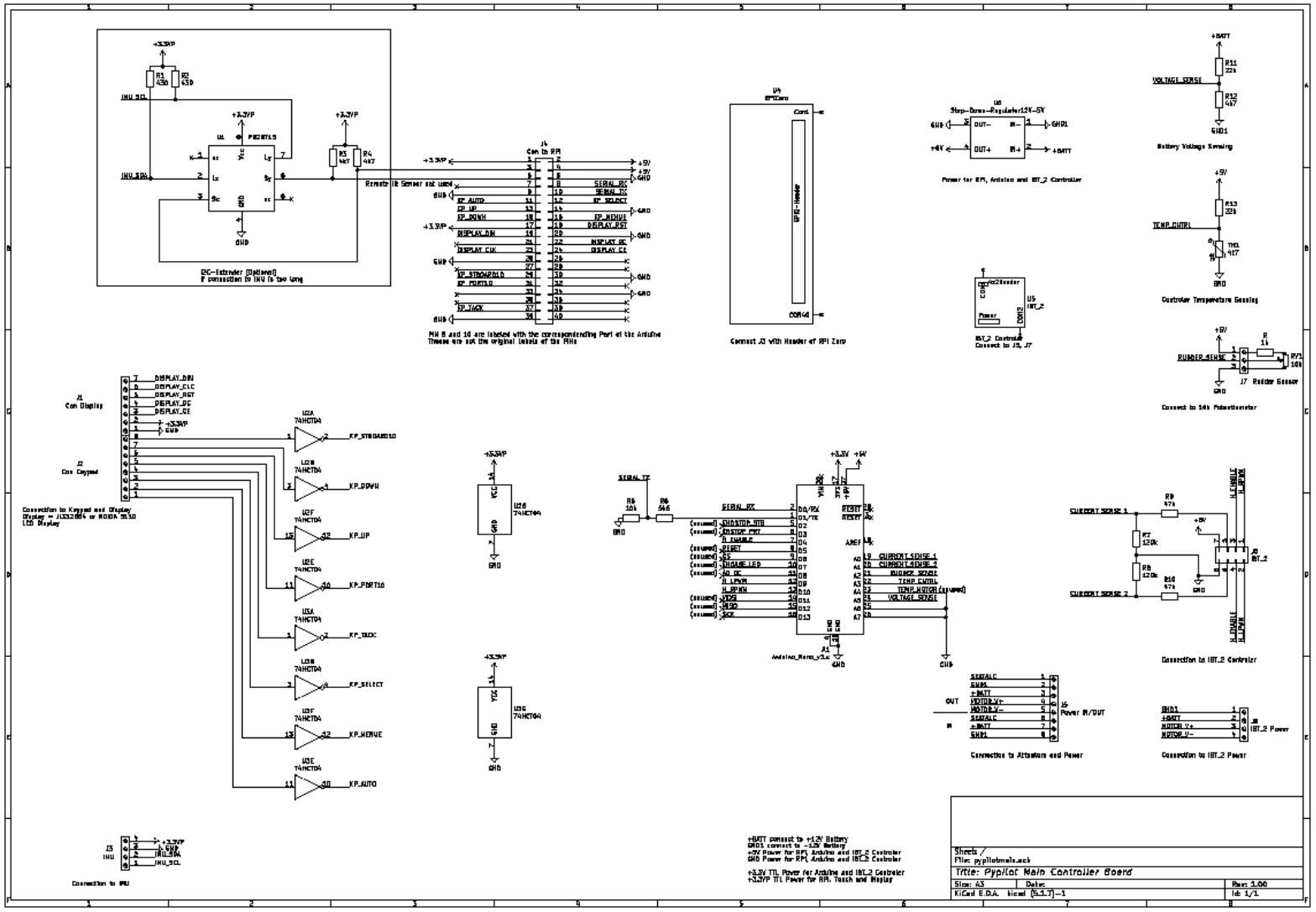



*Abbildung 3: Schaltplan Mainboard*

Teileliste

| Bezeichnung auf Platine | Bauteil | Wert / Bezeichnung |
|---|---|---|
| R1, R2 | Widerstand | 430 Ω |
| R3, R4, R12 | Widerstand | 4,7 kΩ |
| R5 | Widerstand | 10 kΩ |
| R6 | Widerstand | 5,6 kΩ |
| R7, R8 | Widerstand | 120 kΩ |
| R9, R10 | Widerstand | 47 kΩ |
| R11, R13 | Widerstand | 22 kΩ |
| TH1 | NTC- Widerstand | 4,7 kΩ |
| U1 | I2C Extender | P82B715 + Sockel |
| U2, U3 | Inverter | 74HCT04 + Sockel |
| U4 | Raspberry Pi Zero W | |
| U5 | IBT2 Bridge | |
| U6 | Step-Down Regler | Step-Down LM2596 4-35V<br>als fertiges Modul |
| A1 | Arduino Nano Vers. 3 | |

Zusätzlich Schraubklemmen zum Auflöten, Pfostenstecker und –Buchsen 1- und 2-reihig, Flachbandkabel, Wärmeleitpaste.
Spacer zur Montage der Komponenten.

*Abbildung 4: Pfostenstecker, -buchsen u. Schraubklemmen*

Hinweise:
Der I2C-Extender ist optional. Ich empfehle die Verwendung bei Leitungslängen zwischen Mainboard und Kompass ab 1,5m.
Will man keinen I2C-Extender, dann lässt man R1 bis R4 sowie U1 weg. Man setzt dann zwei kleine Drahtbrücken an der auf der Platine bei U1 markierten Stelle.

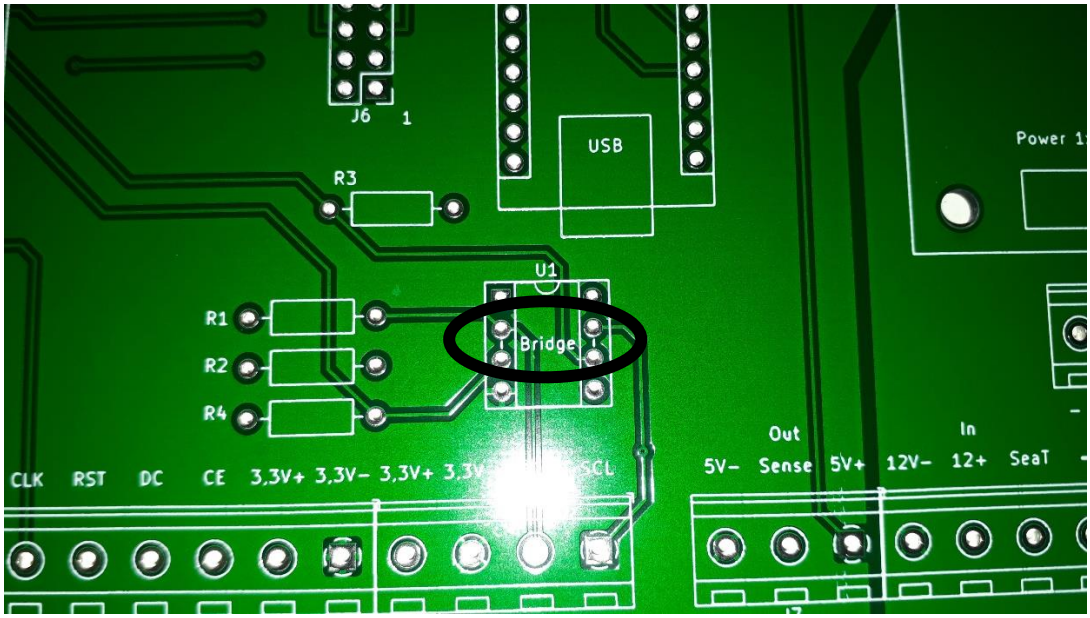



*Abbildung 5: Position Drahtbrücken ohne I2C Extender*

Will man keine Bedienmöglichkeit, lässt man die Inverter (U2, U3) weg.

Der Spannungsregler ist einstellbar. Am besten vor Montage wird der Regler an 12V angeschlossen und der Ausgang auf 5,05 bis 5,15V eingestellt.
Stellt man ihn später ein, dann ohne montierten Raspberry, Arduino und ICs

Tipps:
Der Spannungsregler ist ein fertiges Modul.
Zur Montage werden zunächst 4 einzelne Pfostenstecker in die Platine gesteckt, anschließend setzt man das Modul auf diese Stecker.
Dann verlötet man zunächst das Modul mit den Steckern, anschließend die Stecker mit der Platine.

Die 1-reihigen Steckerleisten können mit Seitenschneider gekürzt werden, die 2-reihigen mit einer kleinen Metallbügelsäge.
Beim Kürzen immer einen Kontakt weiter zählen und dort schneiden. Anschließend schneidet bzw. feilt man überstehenden Kunststoff weg.

Die passend abgelängten Steckerleisten werden an den Arduino gesteckt. Anschließend auf die Platine gesteckt und verlötet.

Man lötet bei Steckern und Fassungen zunächst nur einen Pin. Anschließend kann man das Bauteil ggf. auch durch Erhitzen der Lötstelle ausrichten.
Wenn alles passt werden die restlichen Anschlüsse verlötet.

Spacer zur Montage:
Für den Raspberry werden 2 Spacer 2,5x(10mm + Unterlegscheibe = 11mm) als Unterstützung auf der der Steckerleiste gegenüberliegenden Seite benötigt.
Die IBT2-Bridge wird mit 4 Spacern 3,0x(10mm + 15mm = 25mm) befestigt.

Der NTC-Widerstand dient zur Temperaturerfassung der IBT2-Bridge. Dazu wird dieser mit etwas Abstand zur Platine eingelötet und die Anschlüsse mittels Schrumpfschlauch o.ä. isoliert. Der Kopf wird mittels Wärmeleitpaste an den Kühlkörper thermisch angebunden.

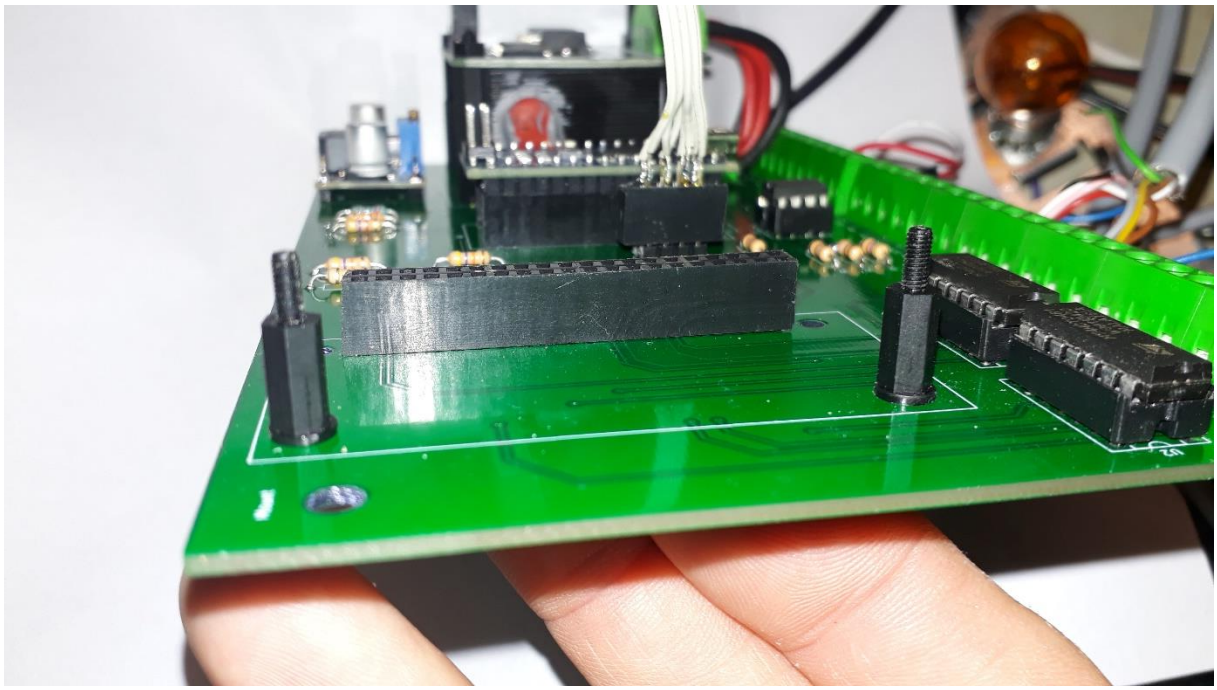


*Abbildung 6: Detail Spacer für Raspberry Zero*

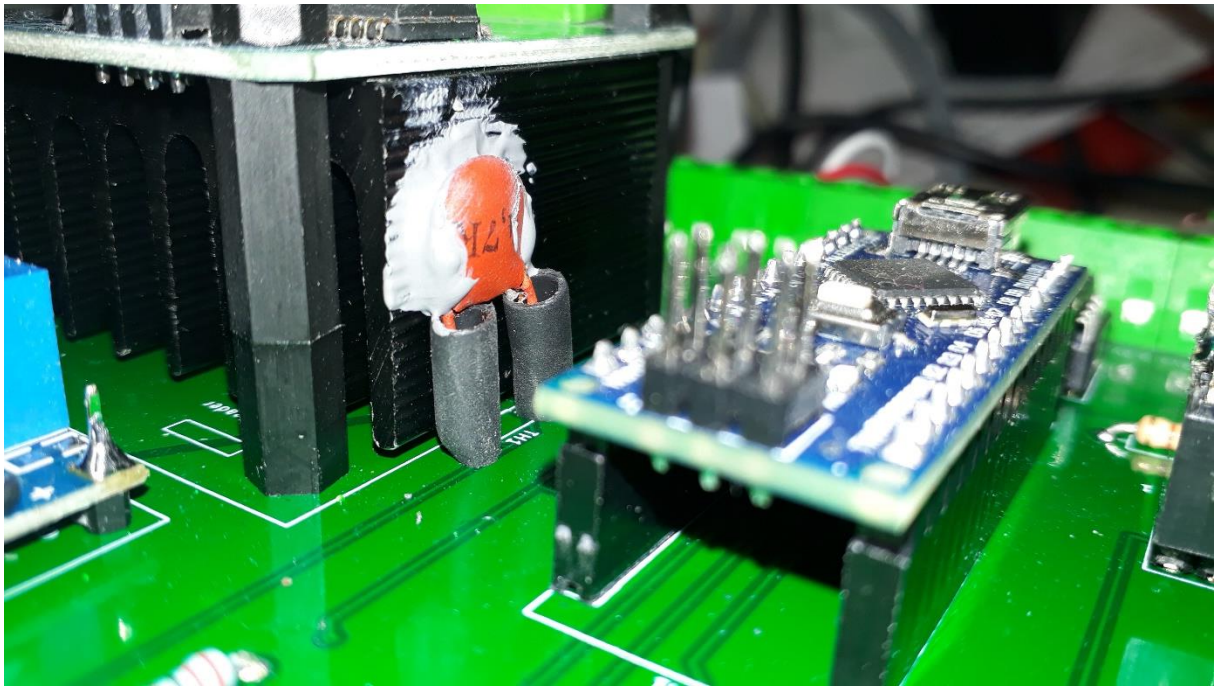


*Abbildung 7: Detail, links: Montage NTC an IBT2-Bridge, rechts: Arduino Nano auf Pfostenbuchsen*

Der Arduino wird mittels Pfostenbuchsen auswechselbar montiert.
Der Raspberry wird mit der Oberseite nach unten (face-down) in die Pfostenbuchsenreihe gesteckt.

Die Verbindung zwischen Steckerleiste J6 (neben dem Arduino) und der IBT2-Bridge wird mittels Flachbandkabel und Pfostenbuchsen erstellt.

IBT-2Bridge:
Dieser Baustein ist ein fertig so zu kaufendes Modul.
Wenn man die Bridge geliefert bekommt auf folgende Punkte achten:

Die Anschlussstecker können verbogen sein –vorsichtig ausrichten.

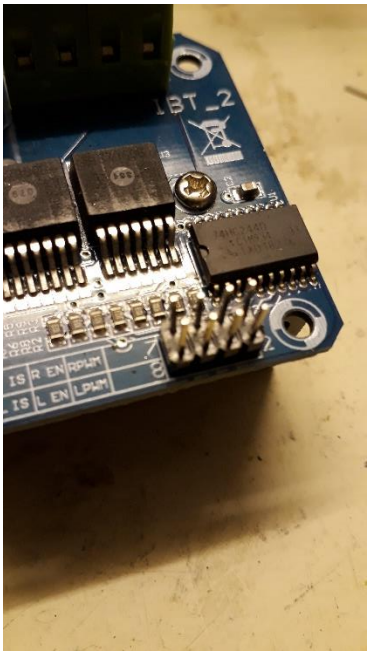



*Abbildung 8: IBT2-Bridge Pfostenstecker prüfen*

Auf der Rückseite die Lötstellen auf Kurzschlüsse prüfen

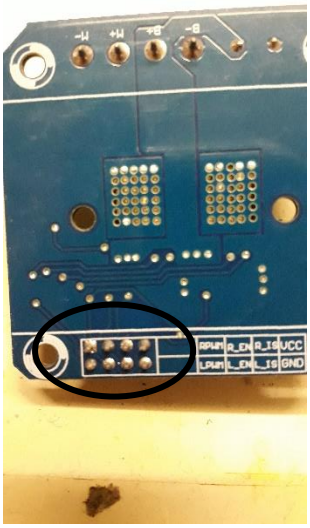



*Abbildung 9: Lötstellen prüfen*

Bei Verwendung von handelsüblichen Linear- oder Radialmotoren dürften sich die Motorströme in Bereichen um maximal 5A bewegen. Das sollte die Bridge thermisch nicht überlasten. Um Probleme auszuschließen empfehle ich, den Kühlkörper abzuschrauben, etwas Wärmeleitpaste auf die Rückseite aufzutragen und den Kühlkörper wieder zu montieren. Bei der Gelegenheit kann man auch die rückseitigen Durchkontaktierungen isolieren.

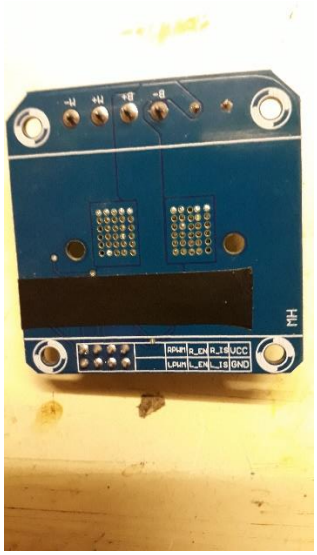



*Abbildung 10: Isolierung Durchkontaktierungen (Kühlkörper demontiert)*

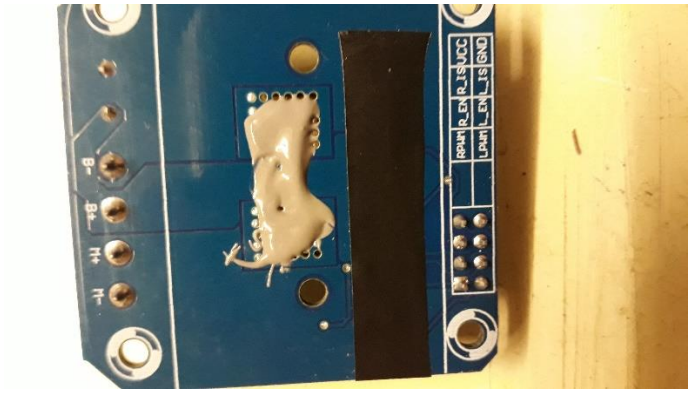



*Abbildung 11: Wärmeleitpaste*

## 4.2 Kompassmodul


*Abbildung 12: Kompassmodul Hauptkomponenten*

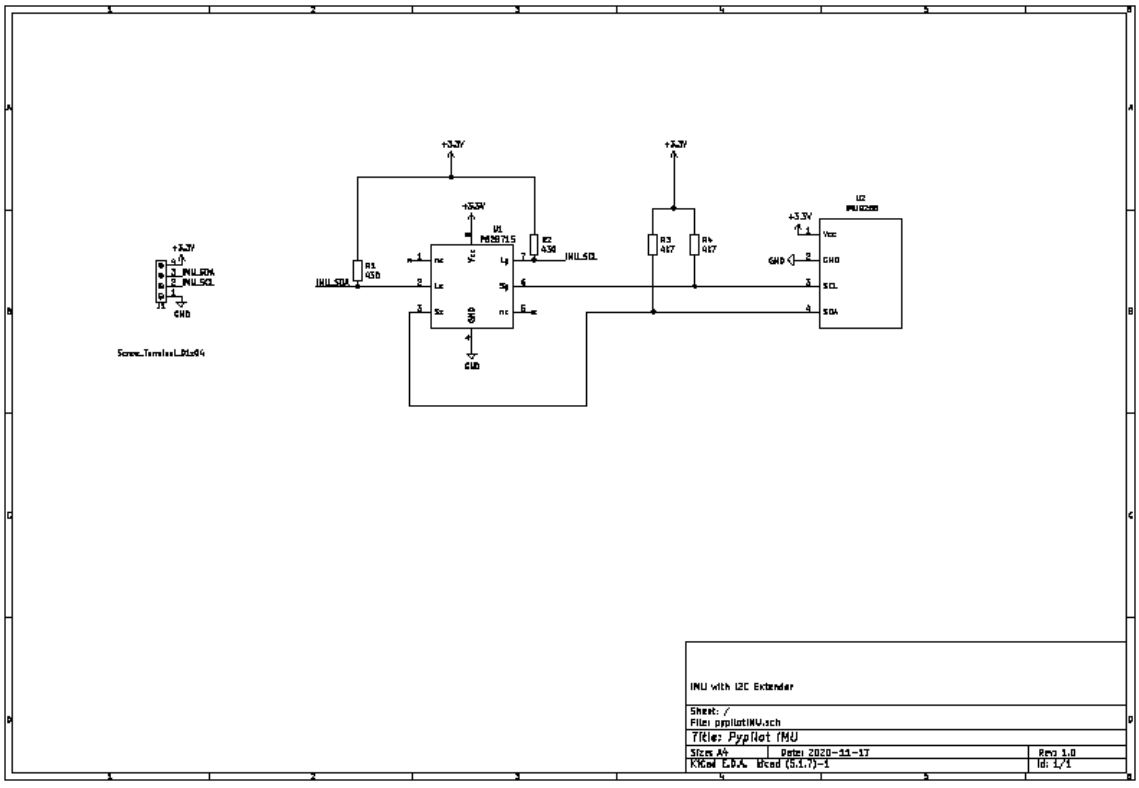



*Abbildung 13: Schaltplan Kompassmodul*

Teileliste

| Bezeichnung auf Platine | Bauteil | Wert / Bezeichnung |
|---|---|---|
| R1, R2 | Widerstand | 430 Ω |
| R3, R4 | Widerstand | 4,7 kΩ |
| U1 | I2C Extender | P82B715 + Sockel |
| U2 | Kompassmodul | MPU 9250 oder 9255<br>Alternativbezeichnung<br>IMU 9250 oder 9255 |

Zusätzlich Schraubklemmen zum Auflöten, Pfostenbuchsen 1- reihig,
Spacer zur Montage der Komponenten

Hinweise:
Wie bei dem Mainboard ist der I2C-Extender optional.
Hat man sich entschlossen diesen bei dem Mainboard weg zu lassen, so muss er auch bei dem Kompassmodul weg gelassen werden.
Es entfallen dann die Widerstände R1 bis R4 sowie der Extender U1.
An markierter Stelle werden stattdessen Drahtbrücken gesetzt.

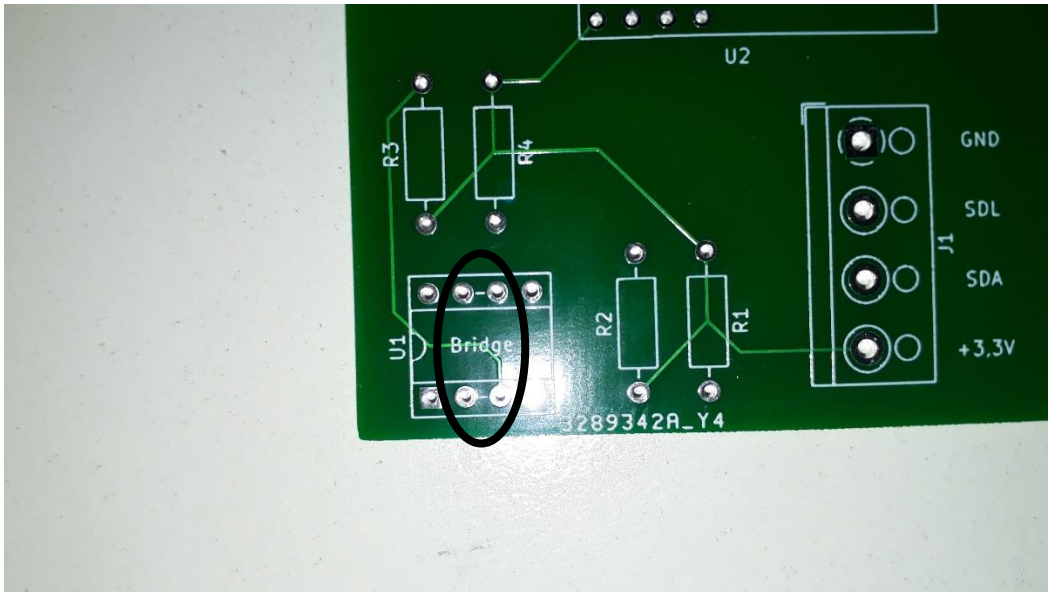



*Abbildung 14: Position Drahtbrücken ohne I2C-Extender*

Tipps:
Das Kompassmodul wird mittels einem Spacer 2,5x(10mm + Unterlegscheibe = 11mm) als Unterstützung sowie einer Pfostenbuchsenleiste auswechselbar montiert.

## 4.3 Display- / Bedieneinheit

Die Bedienung des Pypiloten kann über verschiedene Schnittstellen erfolgen:
Plugin in OpenCPN,
Webgui,
oder über Tasten und ein kleines LCD-Display.

Letzteres empfiehlt sich um im Cockpit eine Statusanzeige sowie eine schnelle und einfache Möglichkeit der Bedienung von Grundfunktionen zu haben ohne erst lange mit Tablet oder Handy zu hantieren.

Bei der vorgestellten Lösung sind die Taster mittels Touchmodulen realisiert, dadurch lässt sich die gesamte Einheit relativ einfach wasserdicht herstellen.


*Abbildung 15: Display- und Bedieneinheit unter Gehäusedeckel mit Beschriftungsmaske. Gussansatz noch sichtbar*

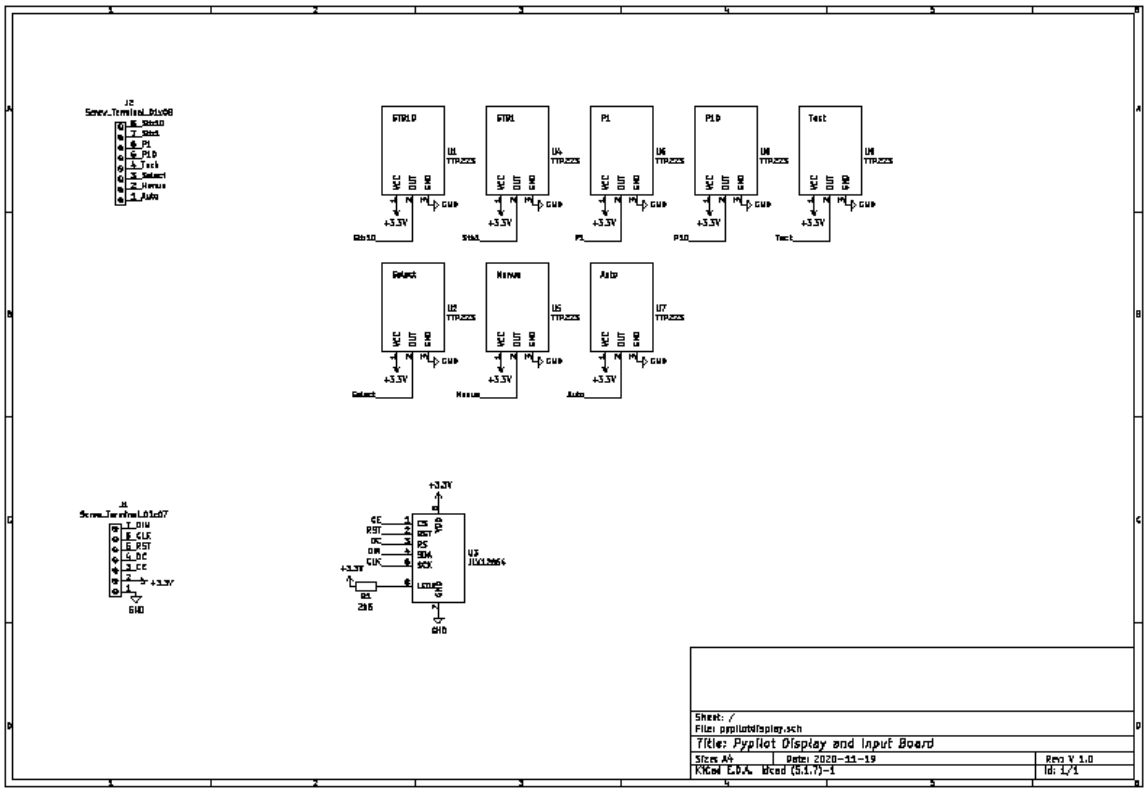



*Abbildung 16: Schaltplan Display- und Bedieneinheit*

Teileliste

| Bezeichnung auf Platine | Bauteil | Wert / Bezeichnung |
|---|---|---|
| R1 | Widerstand | 2,5 kΩ |
| U3 | LCD-Display | Jlx12864 |
| U1-U2,U4-U8 | Touchmodul | TP223 Touchmodul |

Zusätzlich Schraubklemmen zum Auflöten, Pfostenbuchsen und -Stecker 1- reihig,
Spacer zur Montage der Komponenten

Hinweise:
Das Display wird mittels Spacern 2,5x5mm auf der den Lötanschlüssen gegenüberliegenden Seite montiert.
An die Lötanschlüsse wird eine passend abgelängte Pfostensteckerleiste gelötet (Pins nach unten).

Die Schraubklemmen zum Anschluss der Leitungen zum Mainboard werden auf die Unterseite der Platine montiert.

Die Schalteingänge für die Tasten bei dem Raspberry schalten auf GND (das bedeutet wird der Eingang auf GND geschaltet ist die Taste betätigt).

Die Touchsensoren schalten –wenn betätigt- auf +3,3V.
Wenn nicht betätigt schalten die Sensoren auf GND.
Durch die Inverterbausteine auf dem Mainboard wird das für den Raspberry passend „gedreht“, so dass, wenn ein Touchsensor betätigt wird, der betreffende Eingang am Raspberry auch auf GND geschaltet wird.

Man kann statt den Touchsensoren auch Taster benutzen. Diese müssen dann wie die Touchsensoren geschaltet werden, also im geschalteten Zustand auf +3,3V und unbetätigt auf GND. Werden die Schalter unbetätigt nicht auf GND geschaltet, fallen die Inverter auf einen undefinierten Schaltzustand (per Zufall auf GND oder +3,3V) so dass die betreffenden Eingänge am Raspberry dann per Zufall geschaltet sind oder nicht.

Das Display ist optional und kann einfach weg gelassen werden

Die Touchsensoren (oder Taster) sind optional und können einfach weg gelassen werden.
In diesem Fall werden die Inverterbausteine auf dem Mainboard nicht eingesetzt.

Die Sensoren werden mit den Pins nach unten eingebaut.
Die Oberseite muss möglichst an dem Deckel anliegen. Das klappt am besten und sichersten wenn man auf der Oberseite die überstehenden Pinstücke abschneidet und vorsichtig wegfeilt. Die Pins sind dann immer noch eingelötet. Dann liegt die Sensorfläche großflächig am Deckel an und kann durch den Deckel hindurch detektieren.

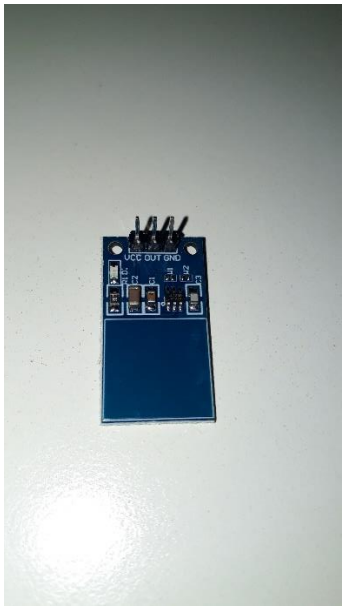



*Abbildung 17: Touchmodul*

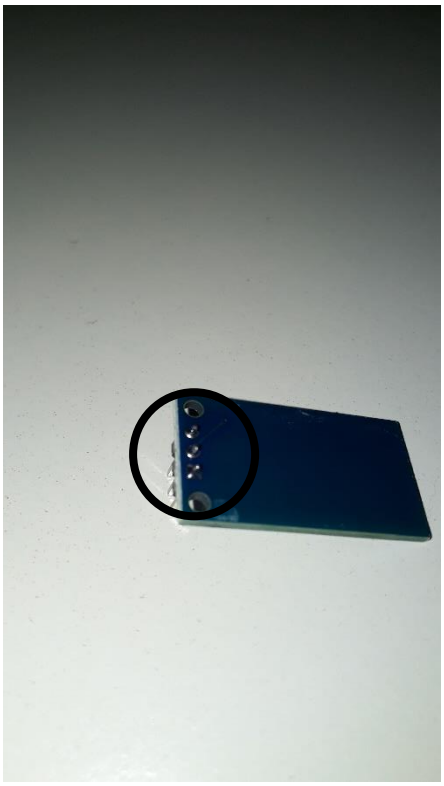


*Abbildung 18: Touchmodul Oberseite. Pins abfeilen / abschneiden*

An den vorgesehenen Stellen auf der Platine werden Pfostenbuchsen für die Touchmodule eingelötet, so können die Module einfach eingesteckt werden.
Um die Module zu stützen werden zw. Platine und Modul ca. 11mm hohe Kunststoff- oder Holzscheibchen geklebt.

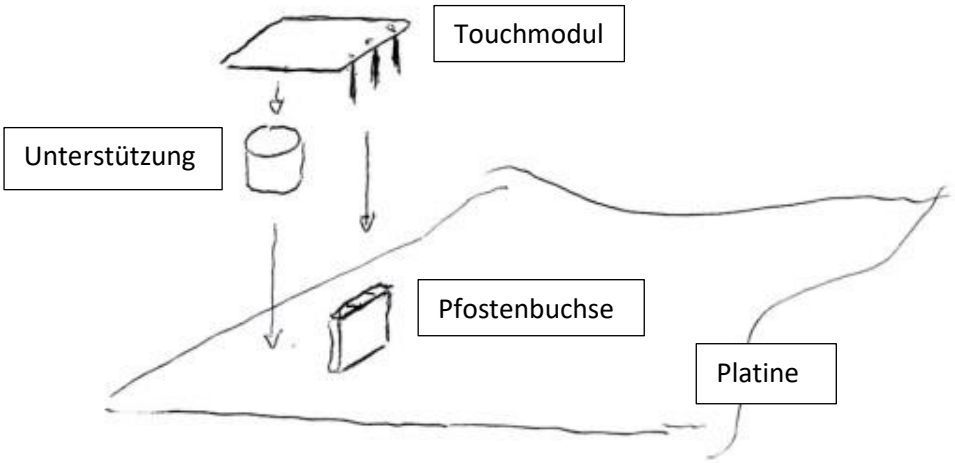



*Abbildung 19: Skizze Montage Touchmodul*


*Abbildung 20: Montierte Display- / Bedieneinheit*


*Abbildung 21: Detail Montage Touchmodule*

Die gesamte Platine passt in ein Gehäuse 160 x 80 mm.
Die Platine muss direkt an den Deckel anliegend in das Gehäuse montiert werden. Dazu eignen sich z.B. Aluröhrchen (oder Spacer) als Abstandshalter.

Die Maske mit der Beschriftung wird ausgedruckt und ausgeschnitten in den Deckel eingelegt.

Ein im Deckel vorhandener Gussansatz kann wegpoliert werden (Filzscheibe, Polierpaste mit Öl)

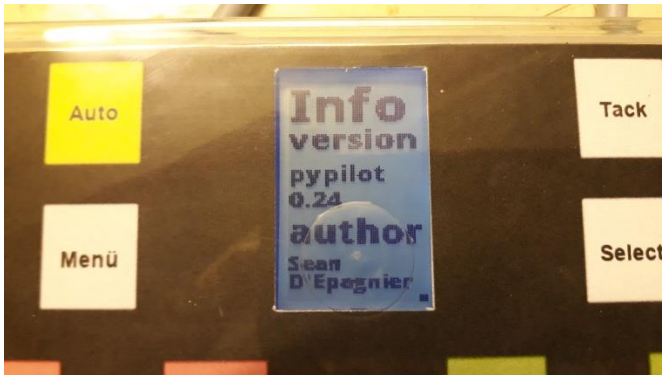



*Abbildung 22: Poliertes Displayfeld*

Wenn man noch ein Päckchen Silica gel in dem Gehäuse befestigt, wird Kondensfeuchte reduziert bzw. ganz verhindert.

# 5 Software

Sämtliche benötigte Software ist auf github eingestellt und kann von dort heruntergeladen werden.
Im Kapitel 7 sind die entsprechenden Links hinterlegt.
Kenntnisse in Programmierung sind nicht nötig.
Vorausgesetzt wird, dass man weiß wie man Images auf SD-Karten brennt und wie man sein OpenCPN-System mit einem Wifi-Access-point einrichtet und den dann verwendeten IP-Adressbereich kennt.
Installiere und aktiviere das pypilot Plugin auf Deinem OpenCPN-System.
In Openplotter wird nur das Plugin benötigt, nicht das Programm pypilot (das läuft ja auf dem Raspberry Zero).

## 5.1 Raspberry Pi Zero

Der Raspberry dient als Rechnerbasis für das Programm pypilot.
Es liegt als fertiges Image vor und muss nur auf eine SD-Karte geflasht werden.

Anleitung:

- Lade das Image für den Raspberry von github herunter
- Entpacke die Image-Datei (7zip)
- Kopiere die Image-Datei auf eine SD-Karte (z.B. mit Windiskimager)
  Mindestgröße der SD-Karte 2 GB. Bei größeren Karten wird nicht mehr Speicherplatz genutzt.
  Es reichen Class 4-Karten.
  Nicht jede SD-Karte funktioniert mit dem Raspberry, wenn er nicht bootet eine andere probieren.
- Stecke den Raspberry noch nicht auf die Platine, stecke die SD-Karte mit dem Image in den SD-Halter des Raspberry, verbinde ihn mit einem USB-Ladegerät und versorge ihn mit Spannung. Der Raspberry bootet (flackernde grüne LED).
  Blinkt diese regelmäßig oder leuchtet gar nicht, dann eine andere SD-Karte probieren.
- Wenn der Raspberry bootet warte ca. 1min.
- Nehme Dein Smartphone oder Tablet, schalte WiFi ein, es sollte dann ein WLAN „pypilot“ zu finden sein. Verbinde Dich mit dem WLAN (kein Password nötig).

- Öffne den Browser auf Deinem Gerät und gebe in die Addresszeile diese IP-Addresse ein: 192.168.14.1
  Es wird die Startseite der WebGui (Browserbasierte Bedienoberfläche) geöffnet.
  Der Raspberry braucht bis zu zwei Minuten nach dem Booten bis diese Oberfläche aufgerufen werden kann. Wenn nach Aufruf im Browser die Meldung „Seite existiert nicht“ o.ä. etwas warten, dann nochmal versuchen.

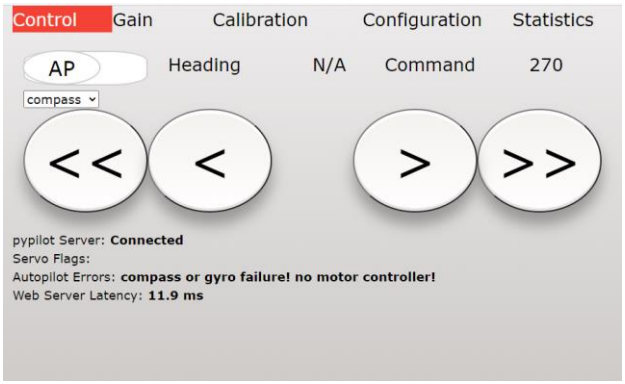



*Abbildung 23: Startseite WebGui*

- Navigiere zu „Configuration“, dann zu „Configure Wifi“

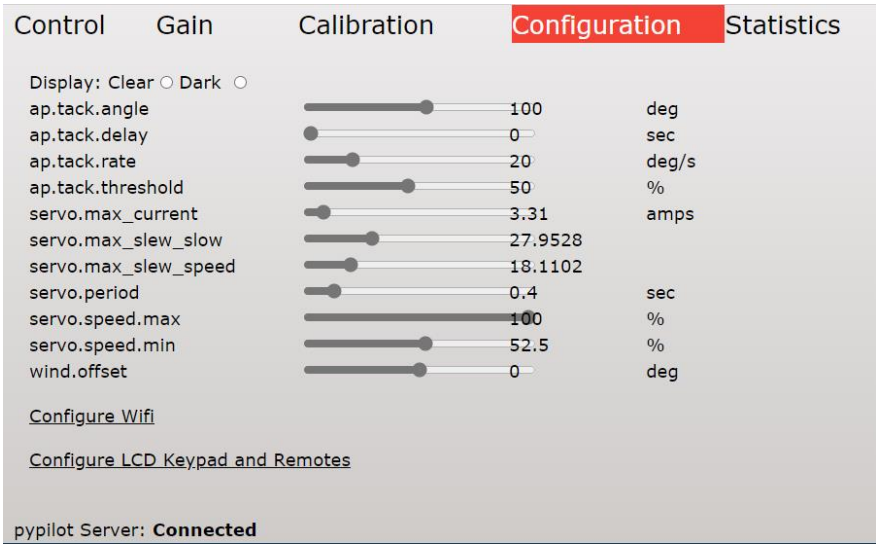



*Abbildung 24: WebGui Seite "Configuration"*

- Wähle bei Modus „Managed (AP-Client)“ aus.
  Gebe SSID und Password von dem Wifi von dem OpenCPN-System (Openplotter-System) ein.
  Gebe eine IP-Adresse aus dem IP-Bereich des Accesspoints (der im OpenCPN- oder Openplotter-System eingerichtet wurde) bei dem Feld „Client Mode Address“ ein.
  Für Openplotter ist hier 10.10.10.60 voreingestellt.
  Bestätige durch click auf „Submit“

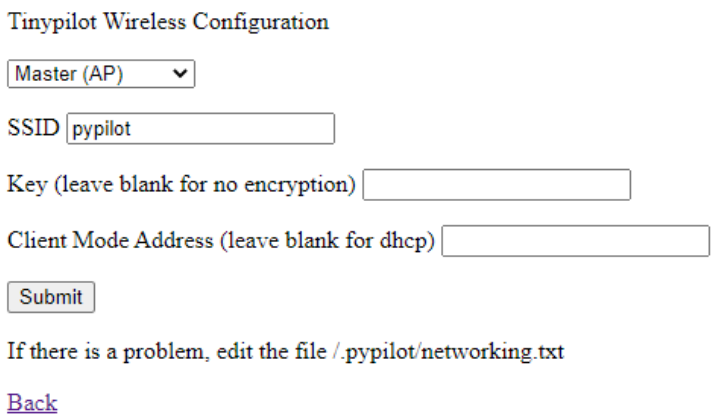



*Abbildung 25: WebGui Netzwerkkonfiguration*

Die Vorgabe einer IP-Adresse für den Client an dieser Stelle vereinfacht im weiteren Verlauf die Konfiguration und Anbindung in OpenCPN / Openplotter.

- Reboote den Raspberry (kurz von der Stromversorgung trennen)
- Starte Dein OpenCPN-System mit Wifi-Access-Point
- In OpenCPN starte das pypilot plugin (Aktivere den AP einmal durch klick auf das Feld „AP“, dann ist das Fenster richtig dargestellt, deaktiviere den AP wieder durch klick auf „AP“).
- Im Plugin Navigiere zu Einstellungen
- In der Adresszeile für den Host gebe die vorher vergebene IP-Adresse für den Raspberry Zero ein und bestätige mit „ok“
- Nach einer kurzen Zeit sollte in der obersten Zeile im pypilot-Fenster die Nachricht connected with… stehen und die Farbe für das Symbol des plugins in der Menueleiste wechselt von grau zu rot
- Trenne den Raspberry Zero von der Stromversorgung, das Symbol wird wieder grau und in der Zeile oben im Fenster steht nun disconnected
- Stecke den Raspberry auf den vorgesehenen Steckplatz auf dem Mainboard

- Zur Übertragung weiterer Daten (Wind, GPS, Routeninformationen) an den Autopiloten richte in OpenCPN eine neue Verbindung ein (unter Einstellungen) mit folgender Konfiguration:

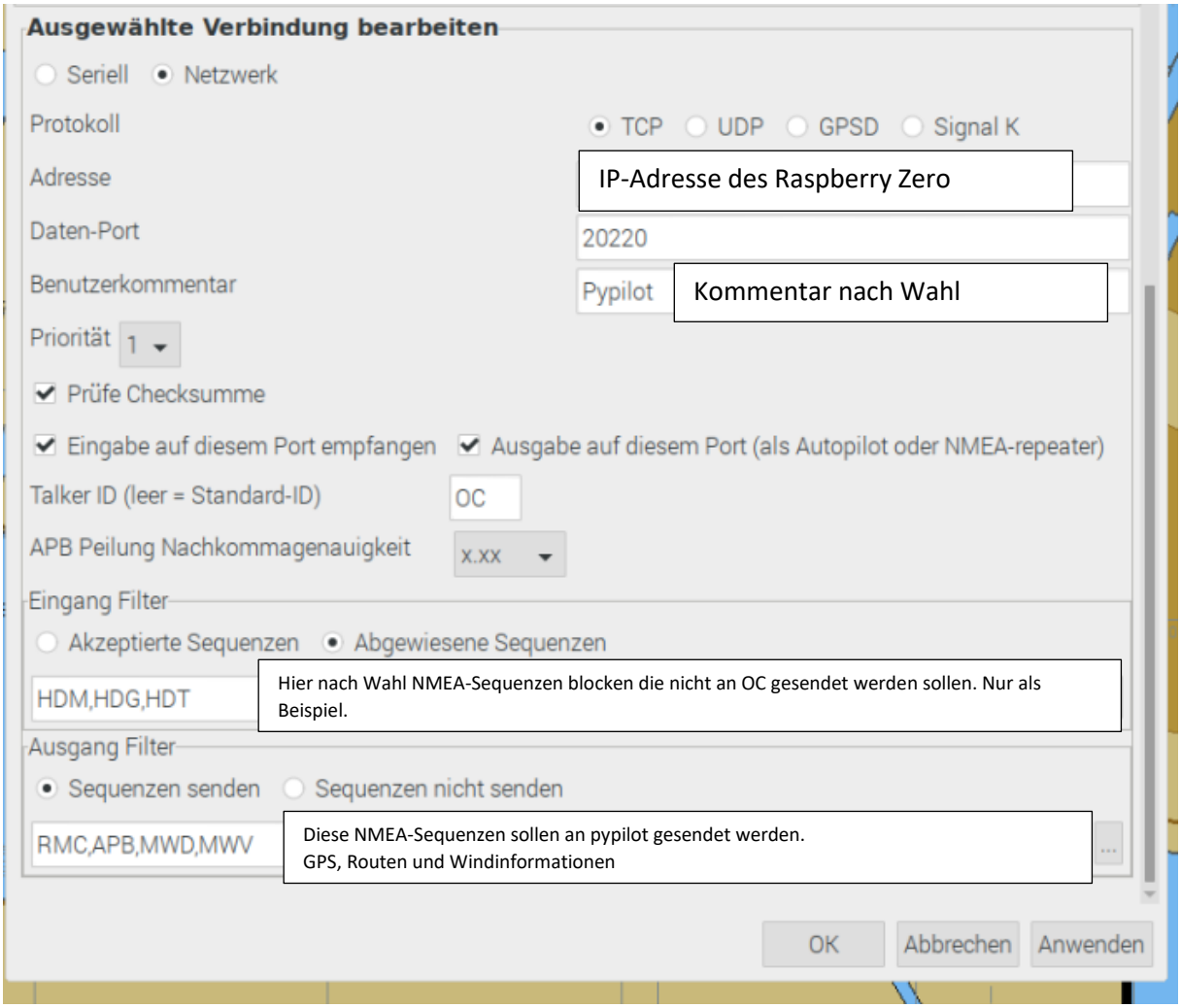



*Abbildung 26: Verbindungseinstellung in OpenCPN*

Damit sind die Vorbereitungen in OpenCPN abgeschlossen

## 5.2 Arduino Nano

Die Drehzahlsteuerung und Drehrichtung des Motors erfolgt über einen Arduino Nano als Motorcontroler mittels entsprechendem Programm.

Das Programm muss dazu auf den Arduino geladen „geflasht“ werden.

Auf github sind sowohl die Binärdateien (die ohne weitere Programmierarbeit einfach auf den Arduino geflasht werden können) als auch der Arduino Sketch für eigene Programmentwicklungen eingestellt.

Auf github sind zwei Binärdateien vorhanden

pypilotmotorcontrollerwithoutrudder.**** - Binärdatei wenn kein Ruderlagesensor verwendet wird
Die Funktion Ruderlagesensor ist deaktiviert.

Pypilotmotorcontrollerwithrudder.**** - Binärdatei wenn ein Ruderlagesensor verwendet wird.
Diese Datei kann auch ohne Ruderlagesensor verwendet werden. Dann ist der Anschluss „Sense“ mit -5V zu verbinden.

Anleitung zum flashen der Binärdateien (Windows Rechner):

- Lade den usbtreiber für den Arduino Nano von github herunter und installiere ihn
  (Der Treiber ist für die Arduino-Modelle CH340G bzw. CH341SER USB Chipsatz)
- Lade den zip-Ordner xloader von github herunter und entpacke ihn
- Lade die entsprechende Binärdatei für den Arduino von github herunter
- Verbinde den Arduino über USB mit deinem PC
- Starte das Programm Xloader

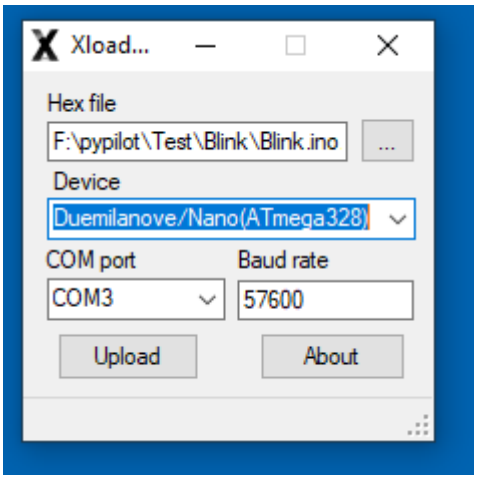



*Abbildung 27: Programm XLoader*

- Öffne mit click auf … das Binärfile, wähle als Device „Duemilanove/Nano(ATmega328), wähle den richtigen COM-Port. Die Baudrate stellt sich automatisch ein.
  Starte das flashen mit „upload“, auf dem Arduino blinken die LEDs.
  Wenn die LEDs nicht mehr blinken ist das flashen beendet.
- Trenne den Arduino vom PC und stecke ihn auf den Steckplatz auf dem Mainboard

Hinweise für Programmierer:

Image für Raspberry:

Das Image tinypilot für den Raspberry ist für die Verwendung des LCD-Displays „jlx12864“ modifiziert.
Dazu wurde in der Datei ~/.pypilot/hat.conf der Eintrag „nokia5110“ in „jlx12864“ geändert.
Man kann grundsätzlich die Display- und Bedieneinheit mit dem „Nokia 5110“-Display ausrüsten.
Dann muss der Eintrag in der Datei wieder entsprechend geändert werden.
Allerdings haben die derzeit erhältlichen Displays qualitative Probleme, zudem ist die Darstellung auf dem „jlx“-Displays angenehmer.

Programm für den Arduino:

Auf Github ist auch der Sketch (Programm) für den Arduino eingestellt und kann heruntergeladen werden um das Programm an eigene Bedürfnisse anzupassen.

Wenn man den Ordner mit dem Arduino Sketch herunterläd und in der Arduino IDE bearbeitet müssen evt. weitere Bibliotheken heruntergeladen und in der IDE eingebunden werden.

Grundsätzlich werden die wichtigen Konfigurationseinstellungen in der config.h und pins.h vorgegeben.
In pins.h werden hauptsächlich die Ein- und Ausgänge definiert.
In config.h werden die Grunddefinitionen vorgenommen. Dort wird zum Beispiel festgelegt, welche H-Brücke benutzt wird, welche Werte die verwendeten Widerstände zur Strom-, Spannungs- und Temperaturmessung haben und welche Funktionen (Endschalter, Ruderlagesensor und anderes) aktiviert oder deaktiviert sind.

Der Motor wird mittels PWM angesteuert.
Die PWM-Frequenz ist von mir auf 31 kHz vorgegeben worden und liegt damit außerhalb des für Menschen hörbaren Bereiches. Man nimmt so nur das „normale“ Motorengeräusch war.
Bei niedrigeren PWM-Frequenzen ist das Motorengeräusch deutlich hörbar von der PWM-Frequenz überlagert.

Die Festlegung der PWM-Frequenz erfolgt in der Zeile 96 des Hauptprogramms und könnte dort angepasst werden.
Informationen dazu findet man im www, zum Beispiel hier:
http://www.scynd.de/tutorials/arduino-tutorials/3-luefter-steuern/3-1-pwm-ohne-pfeifen.html

# 6 Inbetriebnahme

Das Mainboard hat eine Reihe von Anschlüssen.
Die folgende Auflistung geht von links nach rechts

| Stecker | Bezeichnung | Anschluss |
|---|---|---|
| J2 Bedientasten | Stb10 | Taste Steuerbord +10 |
| J2 | Stb1 | Taste Steuerbord +1 |
| J2 | Port1 | Taste Backbord +1 |
| J2 | Port10 | Taste Backbord +10 |
| J2 | Tack | Taste Tack (Wende) |
| J2 | Select | Taste Select |
| J2 | Menue | Taste Menü |
| J2 | Auto | Taste Auto |
| | | |
| J1 Display | DIN | DIN für LCD-Display |
| J1 | CLK | CLK für LCD-Display |
| J1 | RST | RST für LCD-Display |
| J1 | DC | DC für LCD-Display |
| J1 | CE | CE für LCD-Display |
| J1 | 3,3V+ | +3,3V für Display- und Bedieneinheit |
| J1 | 3,3V- | -3,3V für Display- und Bedieneinheit |
| | | |
| J3 Kompassmodul | 3,3V+ | +3,3V für Kompassmodul |
| J3 | 3,3V- | -3,3V für Kompassmodul |
| J3 | SDA | SDA Kompassmodul |
| J3 | SCL | SCL Kompassmodul |
| | | |
| J7 Ruderlagesensor | -5V | -5V für Ruderlagesensor |
| J7 | Sense | Mittenabgriff Poti Ruderlage |
| J7 | +5V | +5V für Ruderlagesensor |
| | | |
| J5 Motor und Spannung | In 12V- | Anschluss 12V- |
| J5 | In 12+ | Anschluss 12V+ |
| J5 | In Seat | Seatalkanschluss, wird nicht von pypilot genutzt |
| J5 | - | Anschluss Steuermotor - |
| J5 | + | Anschluss Steuermotor + |
| J5 | Out 12+ | Ausgang 12V+ |
| J5 | Out 12V- | Ausgang 12V- |
| J5 | Out Seat | Ausgang Seatalk, wird nicht von pypilot genutzt |
| J8 | | Verbindung zur IBT2-Bridge Leistung |

Hinweis zu J5:
Die Anschlüsse In 12V-, In12+ und In Seat sind intern auf der Platine direkt verbunden mit den jeweiligen Out Anschlüssen Out 12V-, Out12+ und Out Seat.

Hintergrund ist, dass ich derzeit einen Raymarine Pinnenpilot mit Seatalk-Anschluss benutze.
Pypilot wird zukünftig über die 12V-Versorgung des Pinnenpiloten versorgt.
Als Backup werden die drei Leitungen zum Pinnenpilotanschluss weitergeführt, so dass jederzeit bei Ausfall des pypilot auf den alten Pinnenpilot zurückgegriffen werden kann.

Vorbereitung Inbetriebnahme:

Falls der Spannungsregler auf dem Mainboard noch nicht eingestellt wurde:
Entferne den Raspberry, den Arduino und alle ICs aus den Halterungen vom Mainboard.
Ziehe den Stecker des Flachbandkabels an der IBT2-Bridge ab.
Verbinde In 12V- und 12+ mit einer 12V Spannungsquelle.
Stelle den Spannungsregler so ein, dass Du am Ausgang 5,05 bis 5,15V misst.
Trenne die Stromversorgung.
Stecke den Raspberry, Arduino und die ICs in die Halterungen auf dem Mainboard.
Stecke das Flachbandkabel wieder auf den Steckplatz der IBT2-Bridge.

Verbinde das Display und die Taster sowie das Kompassmodul mit dem Mainboard.
Verbinde den Aktuator mit dem Mainboard.
Bei Verwendung eines Ruderlagesensors: Die beiden äußeren Anschlüsse des Potentiometers werden an + / - 5V angeschlossen, der Mittenabgriff an Sense.
Zum Schutz des Potentiometers gegen Falschanschluss empfehle ich einen zusätzlichen Widerstand 1kΩ in Reihe zu schalten (siehe Schaltplan).

Verbinde J8 mit dem Leistungsanschluss (Schraubklemmen) der IBT2-Bridge, die Anschlüsse sind an beiden Schraubterminals gleich angeordnet (siehe auch Abbildung 2).

Raspberry und Arduino sind entsprechend Kapitel 5 vorbereitet.

Starte Dein Opencpn-System mit Wifi-Accesspoint, starte das pypilot-Plugin.

## 6.1 Inbetriebnahme Pypilot

Verbinde In 12V- und In 12+ mit der 12V Spannungsversorgung und schalte sie ein.

Eine flackernde grüne LED an der (jetzt Unterseite) des Raspberrys zeigt, dass er bootet.
Das Hintergrundlicht des Displays leuchtet.
Auf dem Arduino leuchtet eine rote LED. Nach kurzer Zeit beginnen diese zu blinken,
der Raspberry kommuniziert mit dem Arduino.
Nach ca. 20 bis 50 sek. wird auf dem Display eine Headinginformation und weitere Infos angezeigt.
Nach ca. 60 bis 120 sek. hat OpenCPN eine Verbindung zum pypilot.
Prüfe ob durch Tastendruck auf die Richtungstasten im Plugin der Aktuator entsprechend bewegt wird.
Durch Umpolen des Anschlusses kann die Drehrichtung des Aktuators ggf. angepasst werden.

Probleme und mögliche Lösungen:

| | |
|---|---|
| Es tut sich gar nichts, keine LEDS leichten, nichts passiert | Spannung vorhanden?<br>Richtig herum angeschlossen?<br>Spannungsregler richtig eingestellt?<br>Keine Anschlüsse vertauscht?<br>Keinen Anschluss vergessen, alle richtig angeschlossen?<br>SD-Karte im Raspberry?<br>SD-Karte mit Programm geflasht?<br>Wenn ja, andere SD-Karte probieren<br>Arduino geflasht? |
| Der Raspberry bootet, keine LEDs beim Arduino | Arduino bekommt keine Spannung oder ist defekt. |
| Der Raspberry bootet, LEDs flackern beim Arduino keine Verbindung zu OpenCPN | Einstellungen Wifi in pypilot und OpenCPN prüfen |
| Pypilot läuft, Verbindung zu OpenCPN ist vorhanden Aktuator läuft nicht | Wenn der Aktuator gar nicht läuft trenne den Ruderlagesonsor vom Mainboard und prüfe erneut.<br>Wenn immer noch kein Erfolg dann ist möglicherweise die IBT2-Bridge defekt. |

## 6.2 Einstellungen in OpenCPN

In OpenCPN müssen einige Grundeinstellungen vorgenommen werden.

Bei Verwendung eines Ruderlagesensors:
Im Plugin navigiere zu Calibration, dann Rudder.
Drücke einmal „Reset calibration“ fahre den Aktuator auf Mittenstellung.
Drücke dann „Centered“.
Gebe in das Feld „Range“ den Ruderwinkel bei Maximalauslage Aktuator an zum Beispiel 20 für 20°
(Bei Montage eines Raymarine-Pinnenpiloten nach Anleitung sind es 18° bei den Endstellungen des Pinnenpiloten)
Fahre den Aktuator ganz nach Steuerbord. Drücke dann „Starboard Range“.
Fahre den Aktuator ganz nach Backbord. Drücke dann „Port Range“.
Bestätige mit Ok.
Der Aktuator sollte dann bei Druck auf „C“ im Plugin halbwegs in Mittenstellung fahren.

Den An- und Nachlauf, sowie Geschwindigkeit des Motors wird im Plugin unter „Calibration“ und dann „Settings“
mit den Punkten:
servo.max_slew_slow, servo.max_slew_speed, servo.speed.max und servo.speed.min eingestellt und angepasst.

Kompasskalibrierung:

Es ist nicht sichergestellt, dass die Lagesensoren des Kompassmoduls bereits ab Werk kalibriert sind. Ich empfehle daher die Kalibrierung in OpenCPN.

Dazu ruft man im Plugin zunächst wieder „Calibration“ auf, dann einmal den Punkt „Aligment“.
Das Kompassmodul legt man auf eine ebene Fläche und drückt „Level“. Wenn der Balken durchgelaufen ist wählt man den Tab „Accelerometers“.
Anschließend dreht man das Kompassmodul auf alle 6 Seiten (stelle Dir vor, das Modul ist in einem rechteckigen Gehäuse, stelle das Kompassmodul auf alle 6 Seiten des Gehäuses), auf jeder Seite hält man das Modul ca. 10 sek. ruhig.
Im Statusfenster erhält man dann, wenn die Kalibrierung erfolgreich war, eine entsprechende Meldung und das „Alter“ der Kalibrierung sollte neu gezählt werden.
Man wiederholt dies solange bis man eine neue Kalibrierung angezeigt bekommt.
Falls man keinen Erfolg hat, muss man gegebenenfalls das Kompassmodul tauschen, weil die Lagesensoren nicht (richtig) funktionieren.

Nach Kalibrierung der Lagesensoren wählt man wieder den Tab „Aligment“, legt das Kompassmodul nochmal auf eine ebene Fläche und drückt „Level“.
Hat man das Kompassmodul später an Bord eingebaut wird dieser Schritt wiederholt.

Die eigentliche Kalibrierung des Kompasses erfolgt durch fahren von Vollkreisen mit dem Boot, auf dem Tisch wird das Kompassmodul um 360 Grad gedreht (ruhig einen größeren Kreisradius und möglichst wenig magnetischen Störeinfluss).
Es sollte dann im Plugin halbwegs passend der Kompasskurs angezeigt werden.

Man kann nun testen, ob bei Bewegung des Kompassmoduls entsprechende Kompasskurse angezeigt werden.
Ebenfalls kann man testen, welche Einstellungen an den Regelparametern zu entsprechenden Regelverhalten am Aktuator führen.

Man wird aber nur einen groben Eindruck erhalten können, richtig testen kann man das erst an Bord.

Die Kalibrierungsschritte zum Nivellieren (Level) und zur Kalibrierung des Kompass (360-Grad Kreis) sind auf jeden Fall nach Einbau an Bord zu wiederholen.

## 6.3 Einstellungen Regelparameter

Die Regelparameter müssen und können auf die jeweilige Wind- und Wellensituation eingestellt werden.

Ich kopiere hier die originale Anleitung / Hinweise dazu von der wiki-Seite von pypilot (englisch)

The basic autopilot uses an enhanced PID filter to form a feedback loop. Various gains can be adjusted to improve performance and vary depending on the boat, seastate, and rudder drive motor

The gains are as follows:
P - proportional - heading error
I - integral - based on the accumulated error
D - derivative - rate of turn
DD - derivative' - rate of rate of turn
PR - proportional root - square root of heading error
R - reactive gain - reverse of command delayed
FF - feed forward - change in heading command

It is recommended to use the opencpn plugin, or openplotter control for tuning the gains because visual feedback is provided.

To get started retuning from scratch (or on a new boat) set all of the gains to zero, except the P and D gains. It is possible to have a fully usable (but less efficient) autopilot using only these two gains.

Set the P gain to a low value (say .003) and the D gain to .01. Typically on larger boats, you will need higher values, but it really depends on how fast the drive motor turns the rudder.

The hard over time is how long it takes to turn the rudder from end stop to end stop. This is typically 30 degrees for each side. If a smaller motor is geared down more, and takes, say 16 seconds, then these gains should be doubled to P=.006 and D = .02 as a starting point.

If the boat takes too long to correct the course and spends a long time to one side of the correct heading, increase these two gains. If the motor is working too hard, and frequently crosses the correct heading, decrease these gains.

P - proportional gain This value should normally be set low. If it is set too high, the boat will constantly turn across the desired heading. If it is too low, the boat may fail to maintain course. As it is increased a higher D gain is needed to compensate (prevent overshoot)

D - derivative gain This is the gyro gain, and the main driving gain of the autopilot. Most of the corrections should be as a result of this gain. Once the best value is found it can typically work in a range of conditions, however, in light air, it can be reduced (along with reducing other gains) to significantly reduce power consumption especially if the boat is well balanced.

PR - proportional root gain This gain can be really useful preventing oscillation especially upwind. To use it, increase it until it takes effect, and gradually back off on the P gain. You will still need some P gain, but it may be less than half of before if a sufficient PR gain is used.

DD - derivative' gain This gain is useful to improve reaction time. It can allow for corrections sooner than they would occur from the D gain alone. To use it, gradually increase this value up to 1.5x the D gain value without changing other gains, and compare the results.

FF - feed forward gain This gain is only useful when making course changes. For holding heading it has no effect. Following a route can cause course changes. It can be very useful in improving the response time since a low P value is normally desirable, this gain is the main contributor when the course is adjusted.

I - integral gain This gain does not need to be used to hold a course, however it can compensate if the actual course held is different from the commanded course. If following routes, and the boat tends to follow along a line parallel to the route, this will compensate for that error. It is best to start at zero, and very carefully increase it until the results are improved. If the value is too high, it will simply increase power consumption.

D2 - derivative squared gain This gain is not very well proven, but the intention is to compensate for large yaw rates from wave action. Typically set it to zero unless you want to experiment.

Hints:

upwind - less D gain, more P (or PR) gain downwind - more D gain, and possibly add DD gain light wind - less gains - save power strong wind - more gains - needed to operate correctly

For sailing in protected waters, steering a less straight course is a tuning error, and will only increase power consumption.

If you can tolerate less straight steering it may save power in waves. Generally you just want to keep the sails pulling, and the average course that you desire. This was always the goal with a wind vane anyway, and can save power consumption as well as wear on the autopilot drive motor.

# 7 Links und Quellen

https://pypilot.org/
Alles über pypilot, Seite von Sean D´Epagnier
https://github.com/McNugget6750/pypilot/tree/master/arduino
Modifizierter Arduino-Sketch für den Motorkontroller, Seite von Timo Birnschein
https://forum.openmarine.net/forumdisplay.php?fid=17
pypilot Forum
https://www.segeln-forum.de/board194-boot-technik/board35-elektrik-und-elektronik/board195-open-boat-projects-org/68916-pypilot/
pypilot Diskussion im Segeln-Forum
https://open-boat-projects.org/de/pypilot/
pypilot auf Open-boat-projekts
https://open-boat-projects.org/de/
Noch mehr tolle Projekte

Software für die hier beschriebene Lösung:
https://github.com/AndreasW29/pypilot-tinypilot-mysolution-imagerpizero
Image für den Raspberry Zero modifiziert für LCD jlx12864
https://github.com/AndreasW29/pypilot-tinypilot-mysolution-motorcontroller
Arduino-Sketch sowie Binärdateien, USB-Treiber, XLoader

Bilder und Schaltpläne:
https://github.com/AndreasW29/pypilot-tinypilot-mysolution-photos-
Bilder
https://github.com/AndreasW29/pypilot-tinypilot-mysolution-infos
Kicad Schaltpläne und Anleitung

Bauteile und Module (alles Beispiele, ich bekomme keine Provision):
Im örtlichen Elektronikladen oder
https://www.conrad.de
Onlineversand
https://www.reichelt.de
Onlineversand
https://www.christians-shop.de/
https://www.makershop.de/
https://www.christians-shop.de/DC-DC-Wandler-LM2596S-step-down-Modul-3A-Eingang-32-40V?curr=EUR&gclid=Cj0KCQiAoab_BRCxARIsANMx4S68lCFaYa-yIJPeo61ckyHV4JwbGiFKceG_IVig1G9Boi91khLzdDYaAqlOEALw_wcB
https://de.aliexpress.com/item/32213228644.html?spm=a2g0s.9042311.0.0.212a4c4doqMEkt
https://de.aliexpress.com/item/32827461392.html?spm=a2g0s.9042311.0.0.212a4c4doqMEkt
Kompassmodul, Arduino, Touchmodule, LCD,Step-Down Regler
https://pcnautic.nl/nl/autopilot
Linearaktuatoren mit Endschaltern, Ruderlagesensor und Raymarineaufnahmen,
sehr freundlicher Onlinehändler

Die Platinen kann man bei mir bestellen (andreasw1402@gmail.com) oder mit den Gerber-Dateien m Github-Ordner bei einem Platinenhersteller ordern.

# 8 Danksagung

An dieser Stelle möchte ich Sean D´Epagnier für seine Entwicklung und Bereitstellung von pypilot danken.
Auf der Website https://pypilot.org/ hat man die Möglichkeit unter „Contact“ eine Spende zu tätigen.
Wenn man einen pypiloten nachgebaut und erfolgreich betreibt finde ich eine Spende angebracht und fair.

# 9 Bekannte Probleme

Tacking Funktion geht derzeit nicht
Man kann hilfsweise im Plugin zusätzliche Richtungsänderungstasten +100 als „Tacking“-Taste einrichten
Man kann die Tasten der Bedieneinheit über das Menü am Pypiloten ebenfalls mit anderen Richtungsänderungen belegen.

Im Plugin werden zeitweise die Einstellungen und Gains nicht angezeigt
– Lösung Plugin selber compilieren