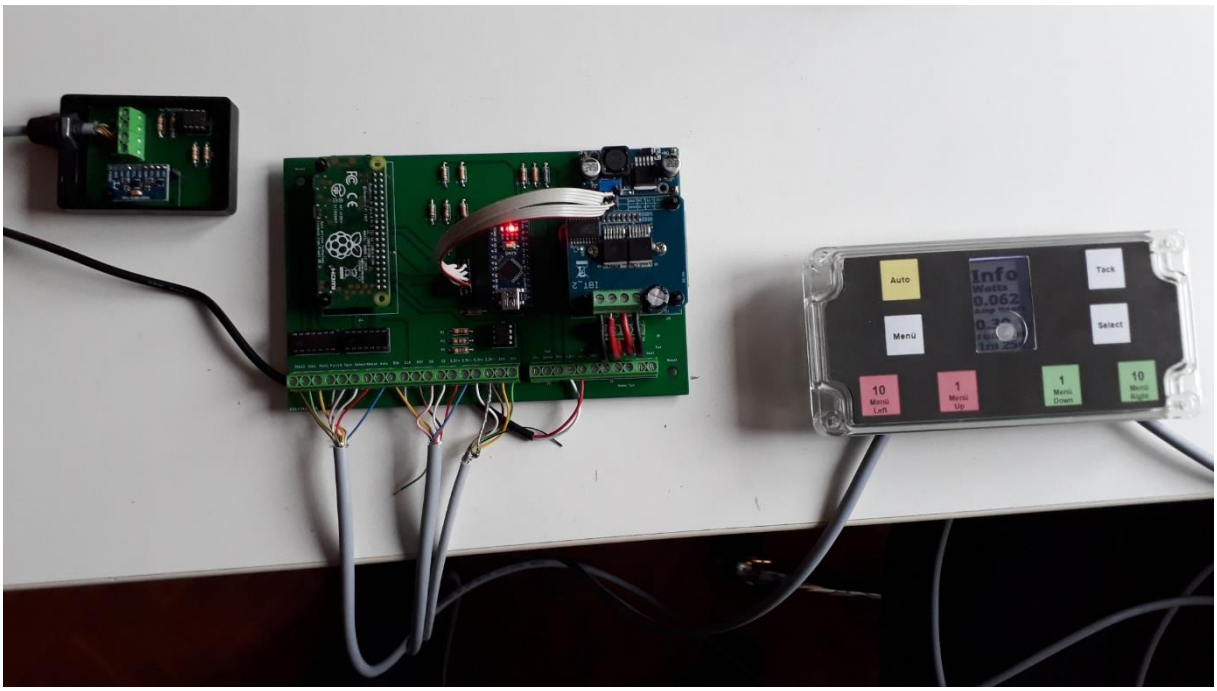
Pypilot- Self-made autopilot



Based on the developments of:

Sean D´Epagnier: https://pypilot.org/

and

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

I do not assume any liability for function and correctness of replica and operation is at your own risk

I use an automatic translation from German to English so sorry for the faults ☺

1 General

Autopilot relieve the crew on long and short distances. There are a variety of different electric models of different price ranges on the market.

Often so-called "pinnenpiloten" are used. These are relatively inexpensive and can be installed and integrated on board without any major effort.
There is a serious drawback to these models. The built-in Compass modules are not or only insufficiently decoupled from the boat movements. In addition, the user has limited access to the rule parameters. As a result, these autopilots will only function or fail to perform in harsher conditions with higher (especially subsequent waves coming from aftern).
If you want an autopilot that still works well even under the aforementioned conditions, you have to resort to a built-in pilot, which are significantly expensive and usually to be installed with some effort on board.
This is where Sean starts and has developed a solution based on a Raspberry Pi and the open-source chartplotter program OpenCPN that comes close to a good built-in pilot in control behavior, but costs significantly less than this one when building a home.
This autopilot called "pypilot" is very universal, the implementation is possible in many ways.

This guide describes a possible implementation. Rebuilders can replicate the solution described below or use it as orientation for their own solutions and implementations. The following guide describes the construction and integration into OpenCPN of a pypilot.

The autopilot is built on its own small computer system, which is then connected via Wifi to the computer with OpenCPN. Control and display is then carried out in OpenCPN via a corresponding plugin.
This solution was chosen because: The computer (Raspberry) with OpenCPN is not additionally loaded, thus running more stable and faster. In the event of a possible computer defect, the other computer will continue to run and be used. A computer defect does not result in a total failure.

No electronics knowledge is required to recreate the system described here.
Also not necessary are knowledge in programming Arduino or Raspberry.
Basically, dexterity, skill and joy in crafting are helpful.

All required software is set to github and can be downloaded from there.
In Chapter 7 the corresponding links are stored.

2 Basics

A pypilot system basically consists of various parallel programs/processes.

pypilot
The program evaluates the sensor data, includes the control algorithm and generates the control signals for the motor driver. One can say this program is the "server" or "master" of the autopilot.

Motorcontroler
This program is used to convert the control signals of the program pypilot into a control for an engine as an actuator for the control of the boat.
In addition, additional information (motor current consumption, temperature) is collected and reported back to pypilot.
In order to relieve the computer with the program pypilot, this processing is carried out with the help of an Arduinos (microprocessor) The communication between pypilot and the motor driver takes place via a serial interface.
A linear actuator can be used as a motor for tiller control.
With wheel control, the use of a corresponding motor is possible as with the original autopilots.

plugin OpenCPN
To control pypilot and display the information in OpenCPN, there is a corresponding plugin.
Communication between Pypilot and OpenCPN is carried out via network protocols. As a rule, the connection between the different computers is via Wifi (Wi-Fi).

3 Description of the system used

With the solution presented here, the autopilot is built on its own small computer system.
The computer system running on the OpenCPN is disconnected from this and is connected to the autopilot computer via Wifi(WLAN).

A RaspberryPi Zero W is used as the computer for the autopilot. In order not to overload the small Raspberry model, a special Linux variant is used as the operating system -Tinycore Linux. The variant of the program based on Tinycore Linux is called "Tinypilot".

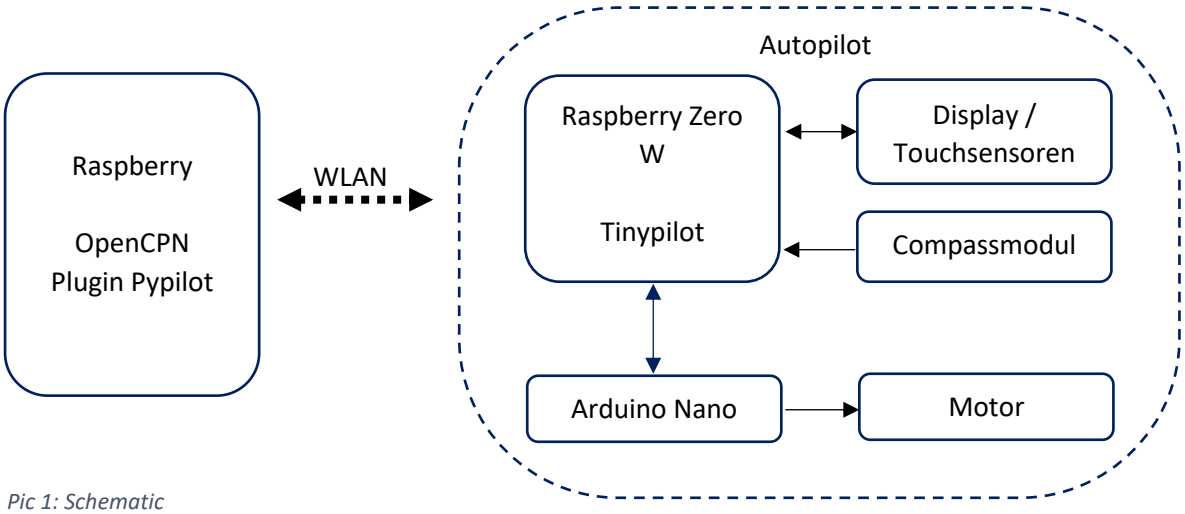
An Arduino-Nano is used as a motor driver. This is connected to the Raspberry via a serial interface. The motor control is via PWM signals and a so-called H-bridge (here as iBT-2 block). Simply, the high motor currents can be switched via the Arduino. The voltage, current consumption of the motor and temperature of the IBT-2 bridge are detected via the Arduino and transmitted to the Raspberry.

The Raspberry Zero and Ardunio, including the IBT-2 bridge, are placed on the motherboard. This board fits from the dimensions into a universal housing 120x200 mm.

A 9-axis gyroscope and compass module is connected to the Raspberry Zero via I2C. The compass module is housed on its own small board. Thus, it can be mounted separately from the motherboard (and magnetic interference). In order to bridge longer cable distances (>2m), a so-called I2C extender is provided.

A separate board is also provided for operation and display. Touch modules are used as switches, so the whole unit can be placed in a housing with a transparent lid and is correspondingly waterproof.

Furthermore, it is possible to operate and display via the OpenCPN plugin. The OpenCPN computer provides and transmits wind and GPS data, if available.



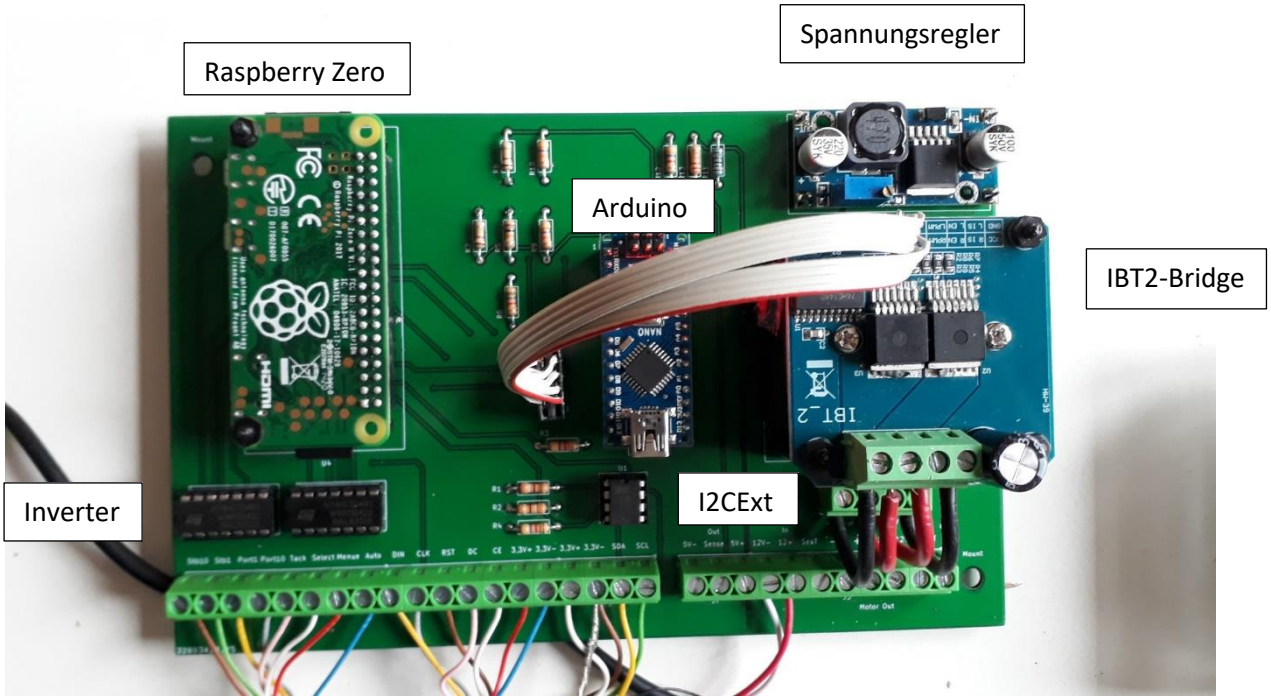
Pic 1: Schematic

4 Building the hardware

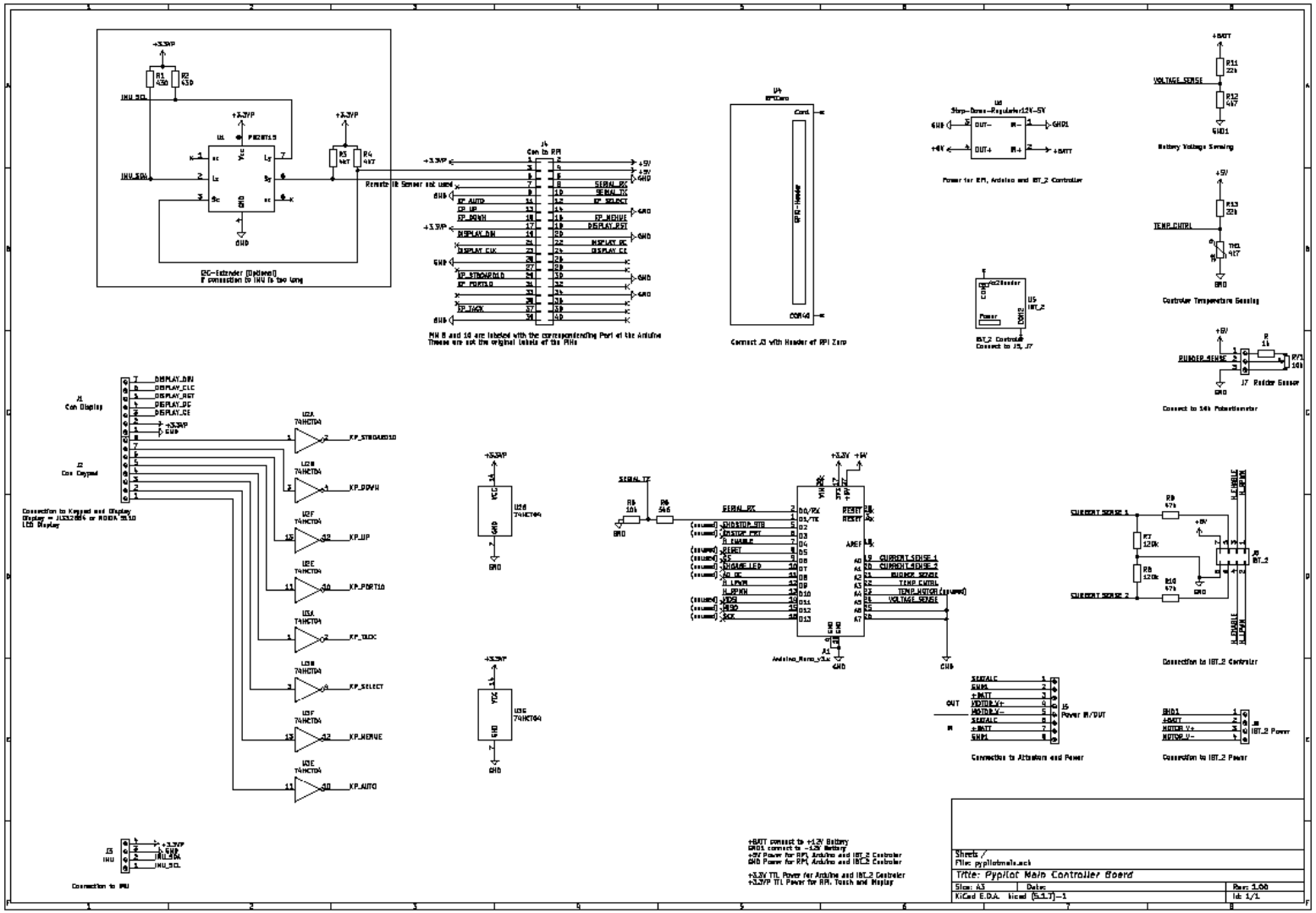
General:
The presented hardware is easy to recreate.
By using sockets for ICs or connectors for the microcomputers, the components are thermally relatively uncritical. Soldering points should not be heated for more than 10 sec.
For soldering beginners there are various tutorials in the www, you can practice with a piece of hole grid board and wire pieces beforehand.
I recommend the use of a fine soldering tip <2mm width.

If you have soldered a soldering joint with too much soldering tin, you can use soldering strands or a soldering pump to suck the superfluous soldering tin. If necessary, a piece of normal stripped copper cable (strand) is also possible, with which superfluous soldering tin is "extracted" under heating.

4.1 Mainboard



Pic 2: Mainboard

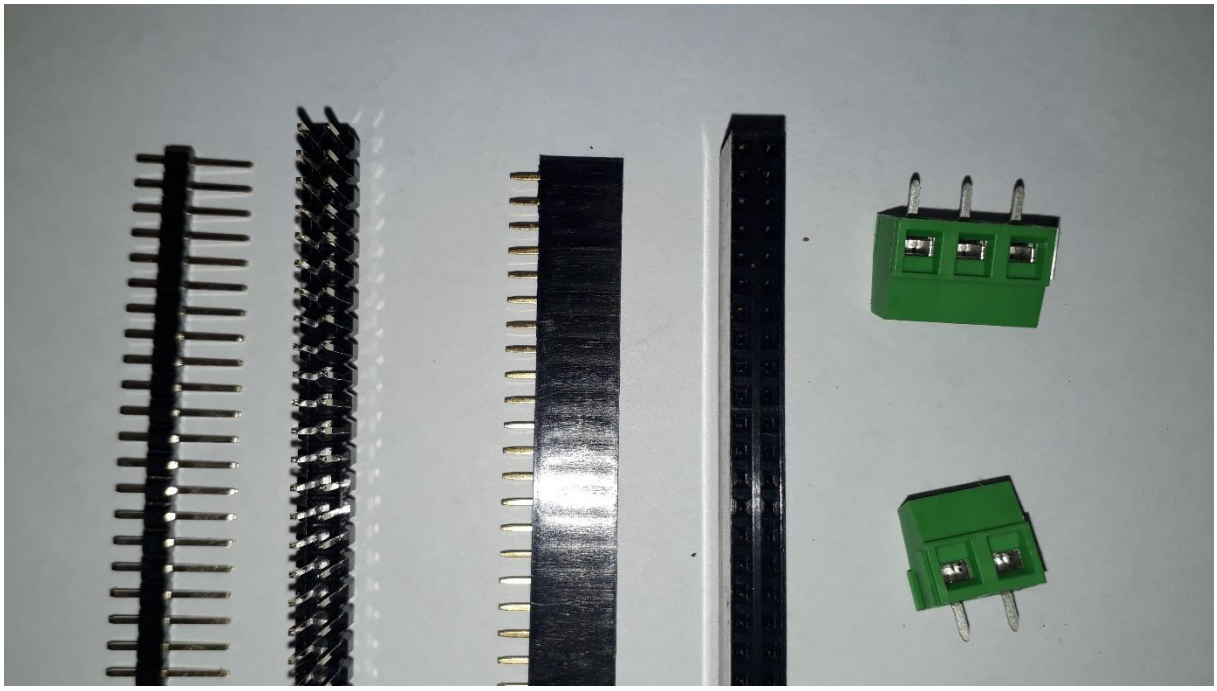


Pic 3: Schematic Mainboard

Partlist

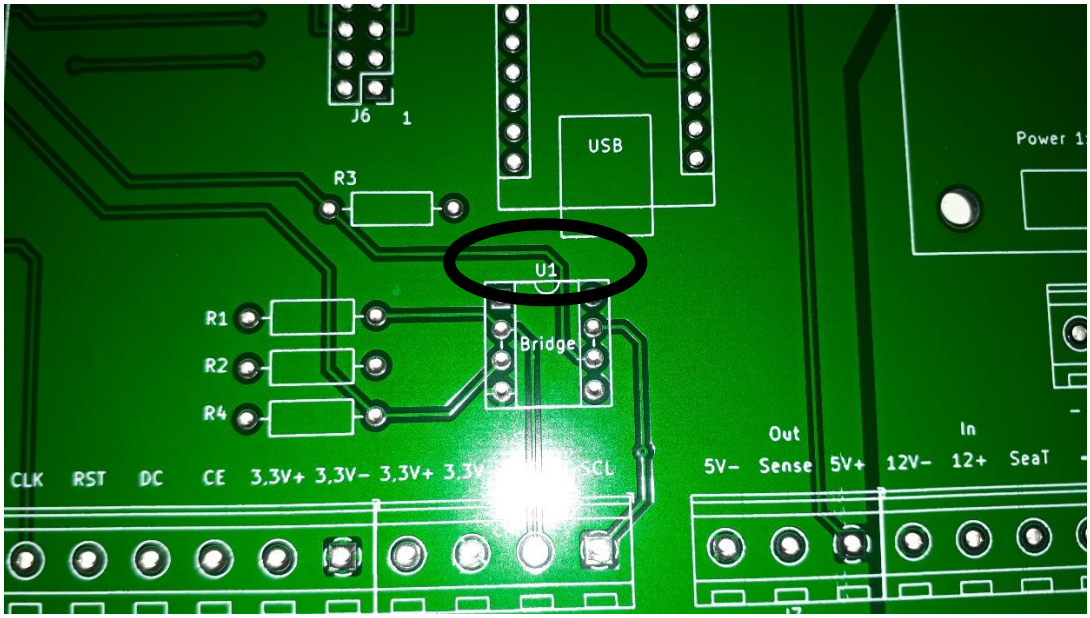
Bezeichnung auf Platine	Bauteil	Wert / Bezeichnung
R1, R2	Widerstand	430 Ω
R3, R4, R12	Widerstand	4,7 kΩ
R5	Widerstand	10 kΩ
R6	Widerstand	5,6 kΩ
R7, R8	Widerstand	120 kΩ
R9, R10	Widerstand	47 kΩ
R11, R13	Widerstand	22 kΩ
TH1	NTC- Widerstand	4,7 kΩ
U1	I2C Extender	P82B715 + Sockel
U2, U3	Inverter	74HCT04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

In addition, screw terminals for soldering, post plugs and sockets 1- and 2-row, ribbon cable, thermal conductive paste. Spacer for assembling the components.



Pic 4: Pfostenstecker, -buchsen u. Schraubklemmen

Notes:
The I2C extender is optional. I recommend the use for cable lengths between motherboard and compass from 1.5m. If you don't want an I2C extender, you owe R1 to R4 and U1. You then place two small wire bridges at the place marked on the board at U1.



Pic 5: Position Wire bridges without I2C extender

If you don't want to operate, you owe the inverders (U2, U3).

The voltage regulator is adjustable. The controller is best connected to 12V before installation and the output is set to 5.05 to 5.15V. If you set it later, then without mounted Raspberry, Arduino and ICs

Tips:
The voltage regulator is a finished module. For assembly, 4 individual post plugs are first inserted into the board, then the module is placed on these plugs. Then solder the module with the plugs, then the plugs with the board.

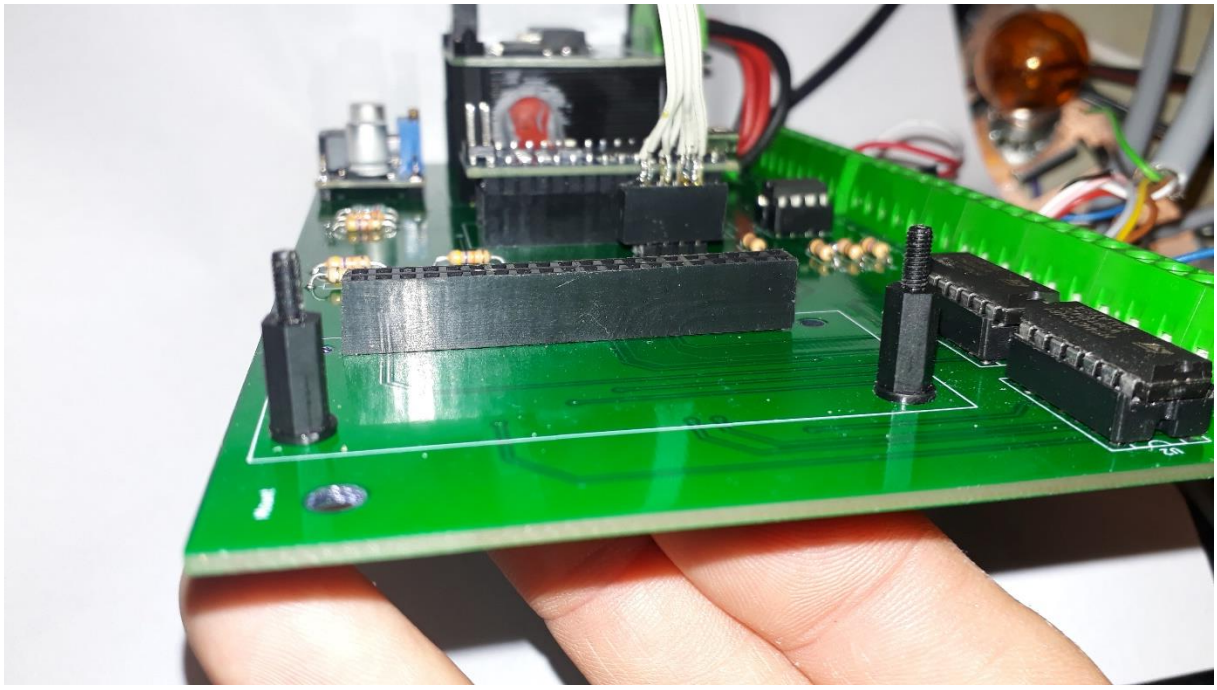
The 1-row connector strips can be shortened with side cutters, the 2-row with a small metal bar saw. When shortening, always count one contact further and cut there. Then cut or refine away the over-the-top plastic.

The appropriately gelled plug strips are plugged into the Arduino.
Then plugged onto the board and soldered.

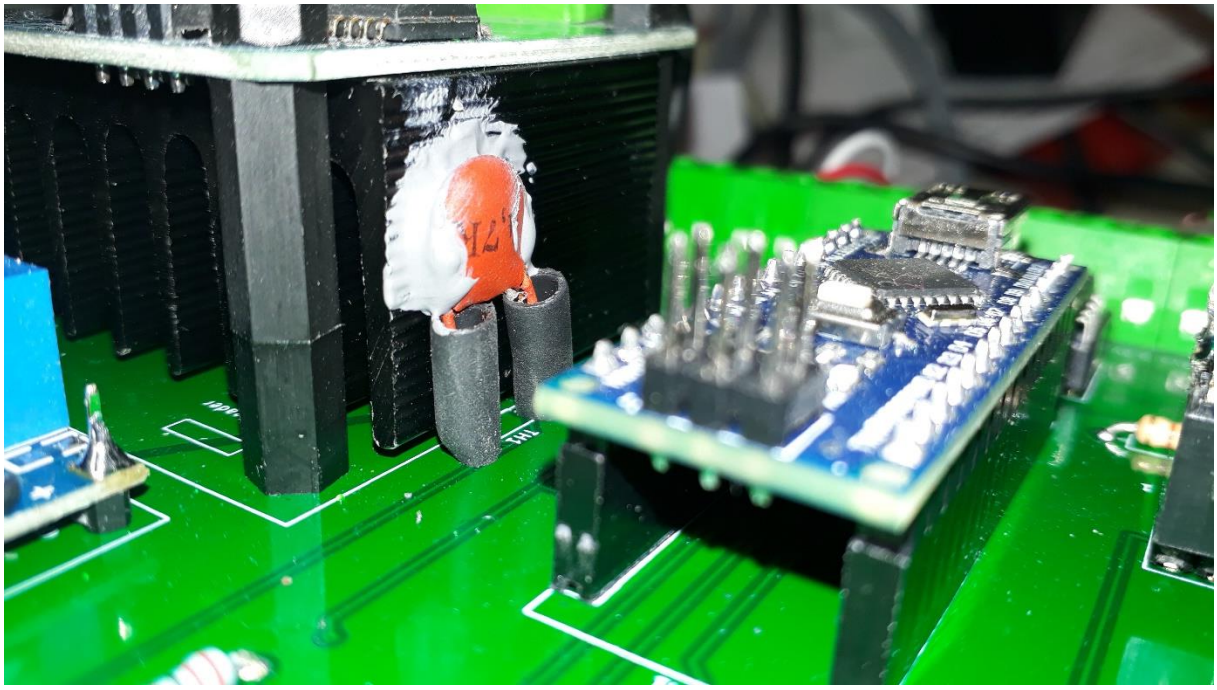
At first, only one pin is soldered with plugs and sockets. Subsequently, the component can also be aligned by heating the solderjoint if necessary. If everything fits, the remaining connections are soldered.

Spacer for assembly: For the Raspberry, 2 spacers 2.5x (10mm + washer = 11mm) are required as support on the opposite side of the connector strip. The IBT2 bridge is fastened with 4 spacers 3.0x (10mm + 15mm = 25mm).

The NTC resistor is used to detect the temperature of the IBT2 bridge. For this purpose, this is soldered with a little distance to the board and the connections are insulated by means of shrink hose or similar. The head is thermally connected to the heat sink by means of thermal paste.



Pic 6: Detail Spacer für Raspberry Zero



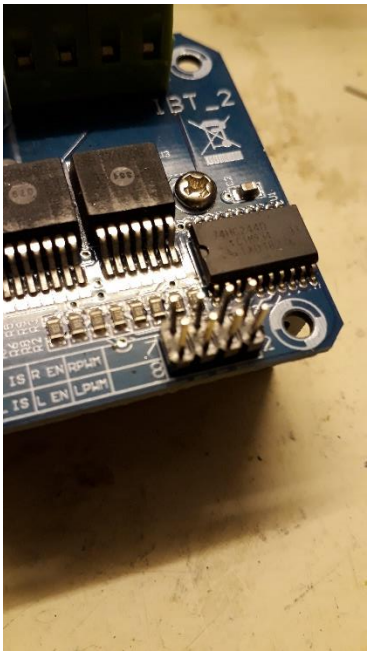
Pic 7: Detail, left: Mounting NTC on IBT2-Bridge, right: Arduino Nano on post bushings

The Arduino is mounted interchangeably by means of post bushings. The Raspberry is tucked into the post socket row with the top down.

The connection between the connector j6 (next to the Arduino) and the IBT2 bridge is created using ribbon cable and post bushings.

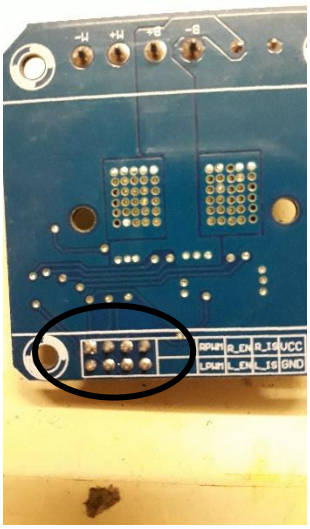
IBT-2Bridge: This module is a ready-to-buy module. When you get the bridge delivered pay attention to the following points:

The connectors can be bent - align carefully.



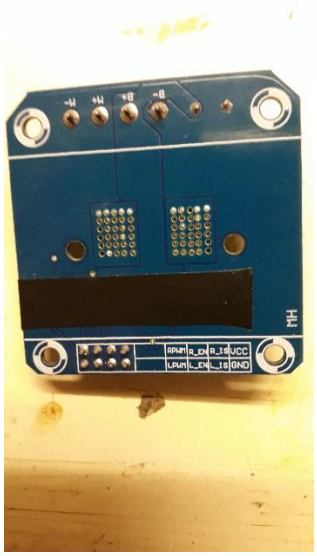
Pic 8: Check IBT2-Bridge post plugs

Check the soldering points for short circuits on the back

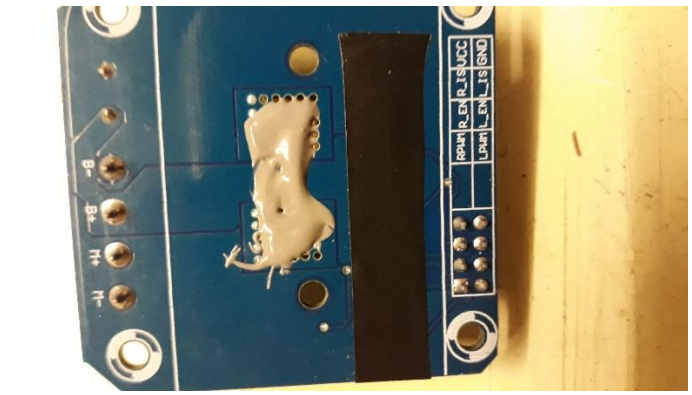


Pic 9: Investigate the soldering

When using commercially available linear or radial motors, the motor currents in ranges should be up to 5A. This should not overload the bridge thermally. To avoid problems, I recommend unscrewing the heat sink, applying some thermal paste to the back and reinstalling the heat sink. On this occasion, it is also possible to isolate the back contacts.



Pic 10: Insulation through-contacts (heat sink disassembled)



Pic 11: Thermal paste

4.2 Compassmodul

Abbildung 12: Kompassmodul Hauptkomponenten

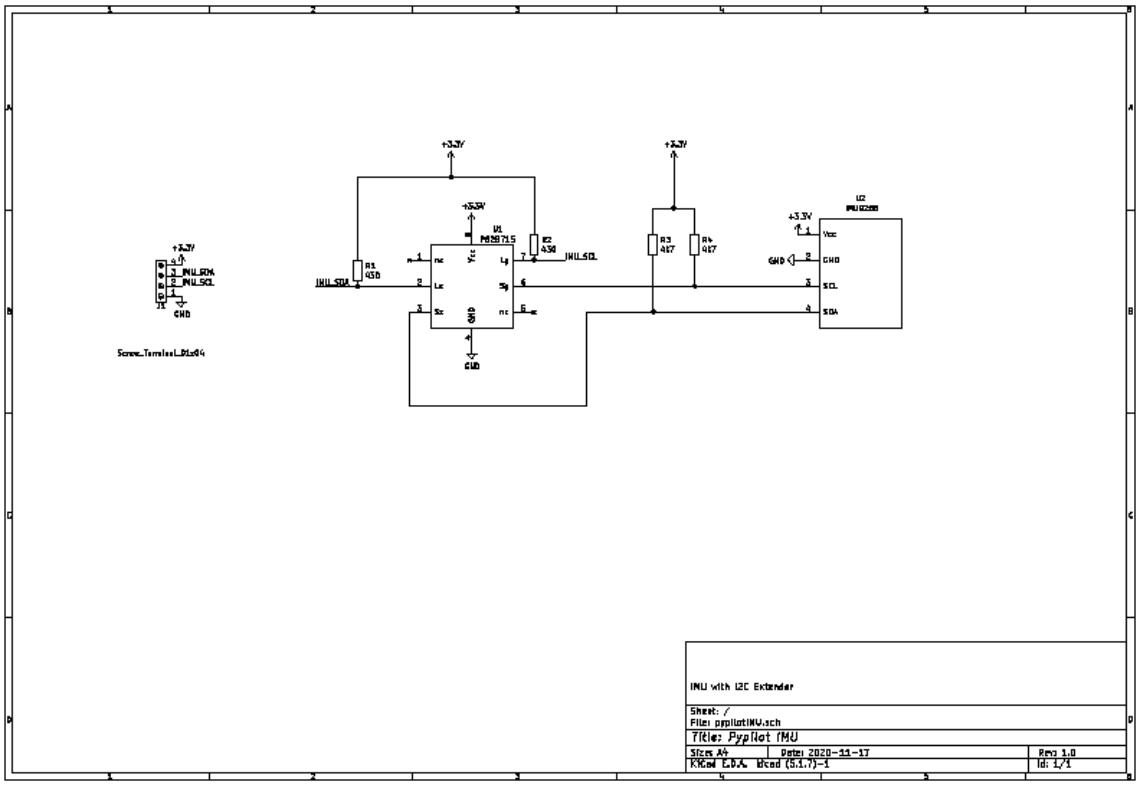


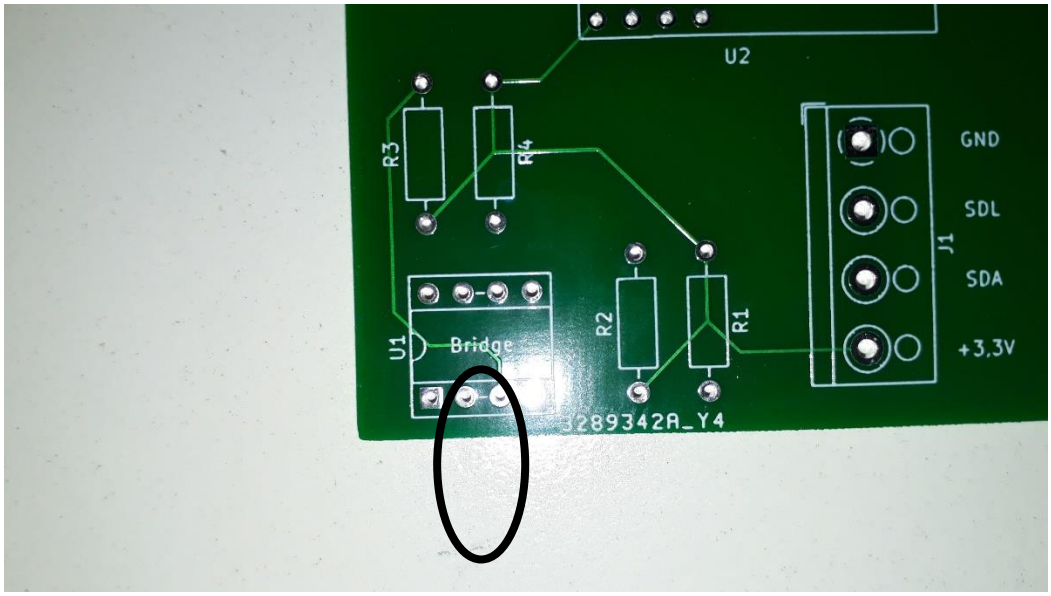
Abbildung 13: Schaltplan Kompassmodul

Partlist

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 Alternativbezeichnung IMU 9250 oder 9255

In addition, screw terminals for soldering, post bushings 1-row, spacer for mounting the components

Notes:
As with the motherboard, the I2C extender is optional. If you have decided to leave it away with the motherboard, it must also be left out with the compass module. The resistors R1 to R4 and the extender U1 are then eliminated. Wire bridges are set at the marked location instead.



Pic 14: Position Wire bridges without I2C extender

Tips:
The compass module is mounted by means of a spacer 2.5x (10mm + washer = 11mm) as support as well as a post socket strip interchangeable.

4.3 Display- / Controlmodul

The Pypilot can be operated via various interfaces:
plugin in OpenCPN, Webgui,
or via buttons and a small LCD display.

The latter is recommended in order to have a status display in the cockpit as well as a quick and easy way to operate basic functions without having to deal with a tablet or mobile phone for a long time.

With the presented solution, the buttons are realized by means of touch modules, so that the entire unit can be produced relatively easily waterproof.

Pic 15: Display and control unit under housing cover with label mask. Casting still visible

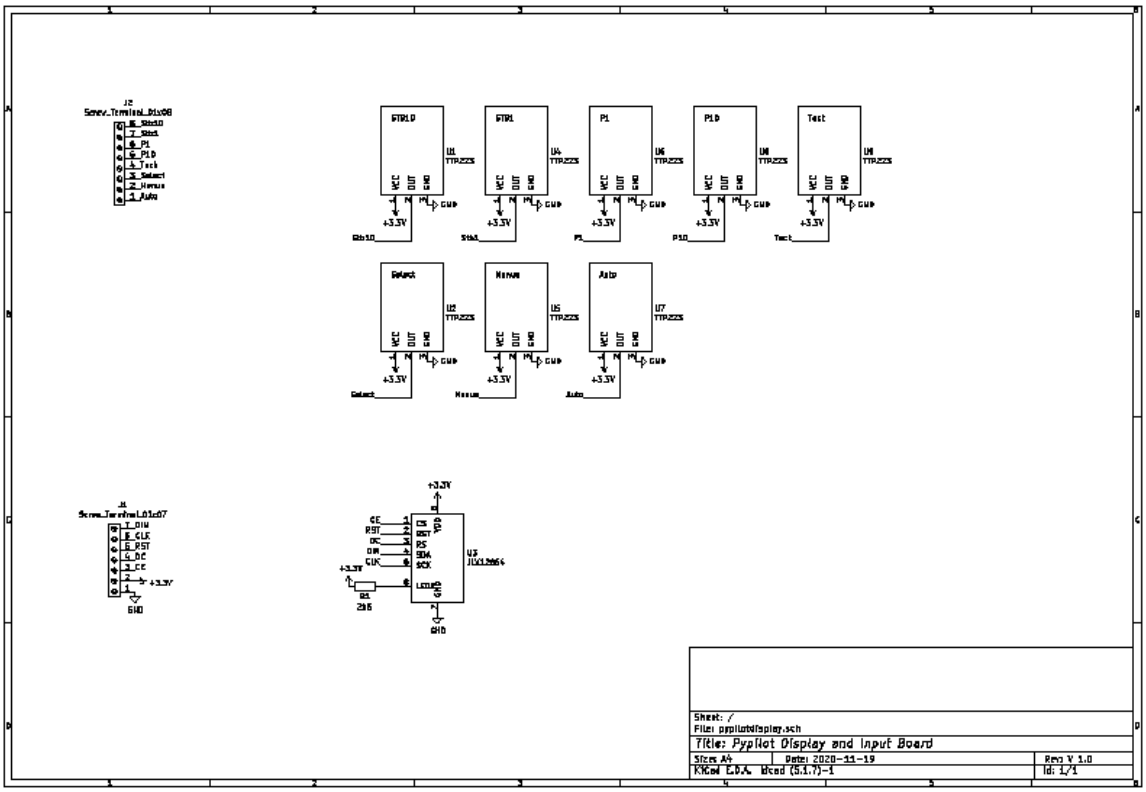


Abbildung 16: Schaltplan Display- und Bedieneinheit

Partlist

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

In addition, screw terminals for soldering, post bushings and plugs 1-row, spacer for mounting the components

Notes:
The display is mounted by spacers 2.5x5mm on the opposite side of the solder connections. A suitably lengthened post connector strip is soldered to the solder connections (pins down).

The screw terminals for connecting the cables to the motherboard are mounted on the bottom of the board.

The switching inputs for the buttons on the Raspberry switch to GND (meaning the input is switched to GND, the button is pressed).

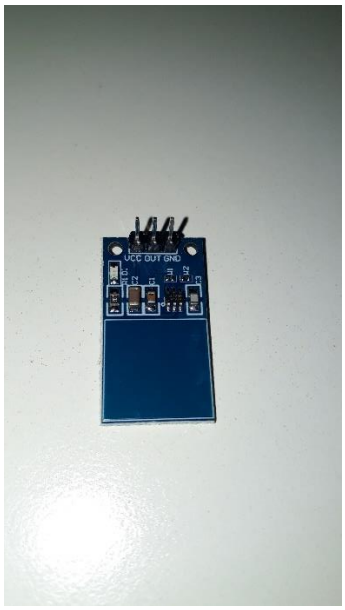
The touch sensors switch to +3.3V when actuated.
If not actuated, the sensors switch to GND. The inverter modules on the motherboard "rotate" the one suitable for the Raspberry, so that when a touch sensor is pressed, the relevant input on the Raspberry is also switched to GND.

You can also use buttons instead of the touch sensors. These must then be switched like the touch sensors, i.e. in the switched state to +3.3V and unactivated on GND. If the switches are not switched to GND unactivated, the inverders fall to an undefined switching state (by chance on GND or +3.3V) so that the respective inputs on the Raspberry are then switched by chance or not.

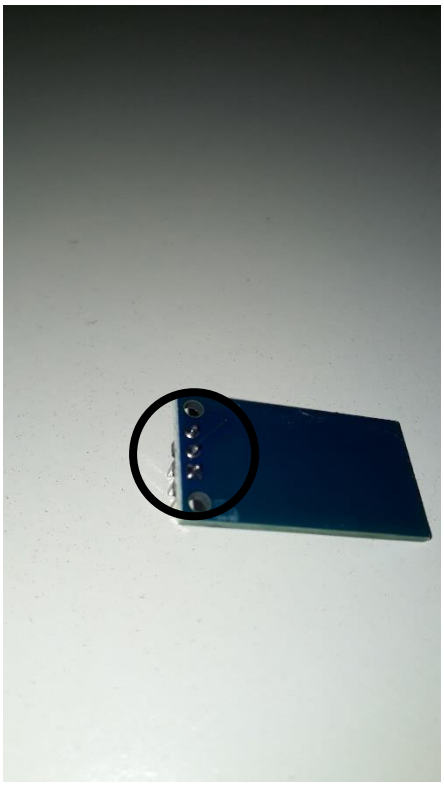
The display is optional and can be easily omitted

The touch sensors (or buttons) are optional and can be easily omitted. In this case, the inverter modules are not used on the motherboard.

The sensors are installed with the pins down. The top must be attached to the lid if possible. This works best and safest if you cut off the protruding pin pieces on the top and gently refine them. The pins are then still soldered. Then the sensor surface is attached to the lid over a large area and can detect through the lid.

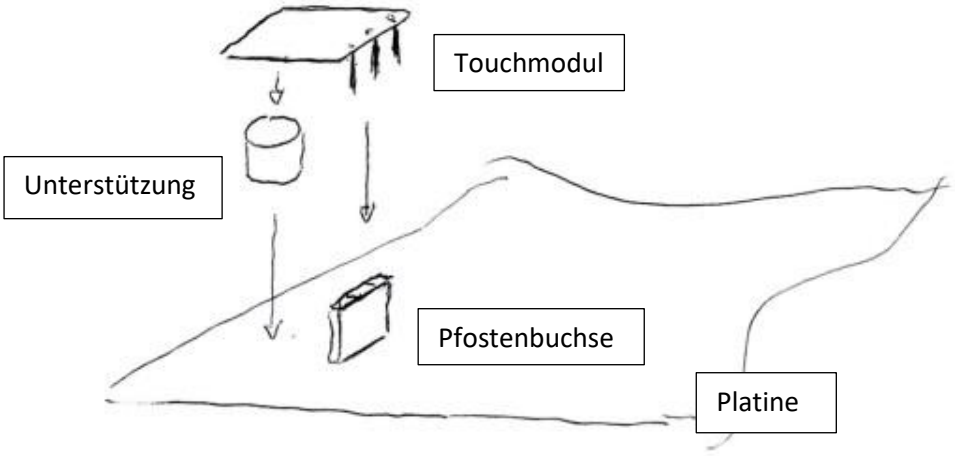


Pic 17: Touchmodul



Pic 18: Touchmodul Top. Refine /cut off pins

At the designated points on the board, post sockets for the touch modules are soldered, so the modules can be easily plugged in. In order to support the modules, approx. 11mm high plastic or wooden discs are glued between the board and module.



Pic 19: Skizze mounting Touchmodul

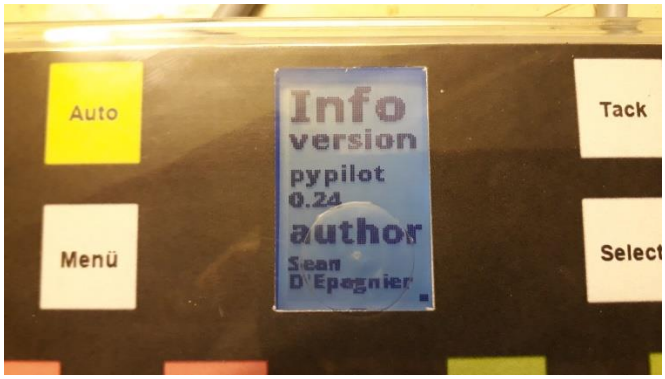
Abbildung 20: Montierte Display- / Bedieneinheit

Abbildung 21: Detail Montage Touchmodule

The entire board fits in a 160 x 80 mm housing. The board must be mounted directly on the lid in the housing. For this purpose, e.B. aluminium tubes (or spacers) are suitable as spacers.

The mask with the label is printed out and cut out into the lid.

A casting base present in the lid can be polished away (felt disc, polishing paste with oil)



Pic 22: Polished display area

If you attach a packet of silica gel to the housing, condensation moisture is reduced or completely prevented.

5 Software

All required software is set to github and can be downloaded from there.
In Chapter 7 the corresponding links are stored.
Knowledge of programming is not necessary.
Assuming that you know how to burn images on SD cards and how to set up your OpenCPN system with a Wifi access point and know the IP address range used.
Install and activate the pypilot plugin on your OpenCPN system. In Openplotter only the plugin is needed, not the program pypilot (this runs on the Raspberry Zero).

5.1 Raspberry Pi Zero

The Raspberry serves as a computer base for the program pypilot.
It is available as a finished image and only needs to be flashed to an SD card.

guide:

- Download the image for the Raspberry from github
- Unpack the image file (7zip)
- Copy the image file to an SD card (e.B. with Windiskimager)
  minimum size of the SD card 2 GB. Larger maps do not use more storage space.
  Class 4 cards are enough.
  Not every SD card works with the Raspberry if it doesn't boot another try.
- Do not plug the Raspberry onto the board yet,
  plug the SD card with the image into the SD holder of the Raspberry, connect it to a USB charger and supply it with voltage.
  The Raspberry boots (flickering green LED).
  If it flashes regularly or does not light up at all, then try another SD card.
- When the Raspberry boots wait about 1min.
- Take your smartphone or tablet, turn on WiFi, and you should find a Wi-Fi "pypilot". Connect to the Wi-Fi network (no password required).

- Open the browser on your device and enter this IP address in the address line: 192.168.14.1
  The home page of the WebGui (browser-based user interface) is opened.
  The Raspberry takes up to two minutes after booting until this interface can be called. If after calling in the browser the message "Page does not exist" or similar wait a little, then try again.

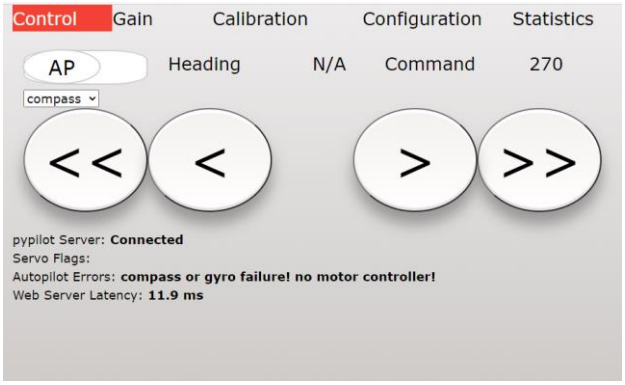


Abbildung 23: Startseite WebGui

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

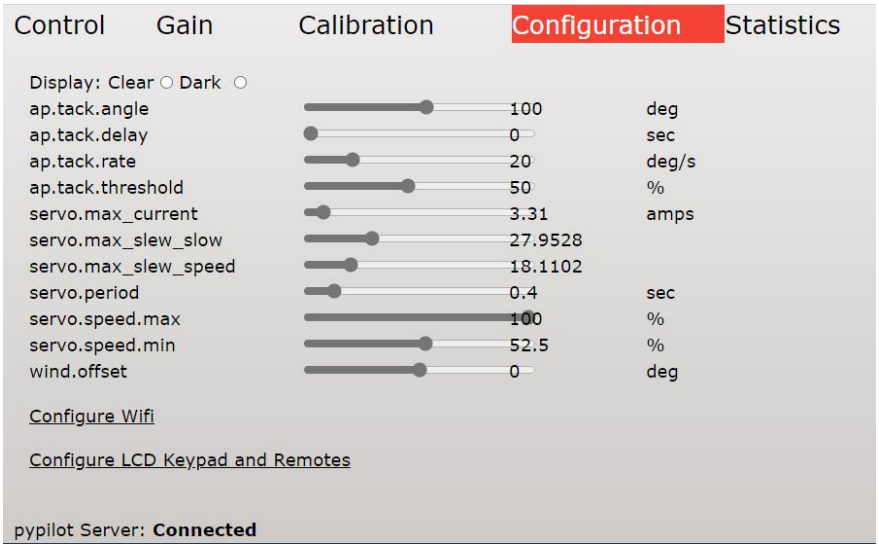


Abbildung 24: WebGui Seite "Configuration"

- Select Managed (AP Client) mode. Enter SSID and Password from the Wifi from the OpenCPN (Openplotter System). Enter an IP address from the IP range of the accesspoint (which was set up in the OpenCPN or Openplotter system) in the Client Mode Address field. For openplotters 10.10.10.60 is preset here. Confirm by clicking on "Submit"

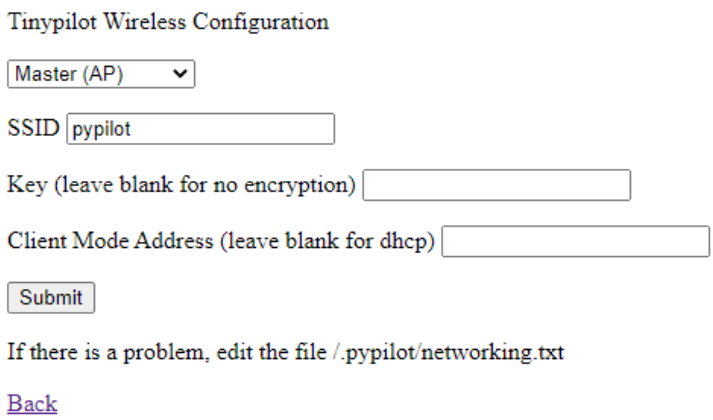


Abbildung 25: WebGui Netzwerkkonfiguration

The specification of an IP address for the client at this point simplifies the configuration and connection in OpenCPN / Openplotter.

- Reboot the Raspberry (shortly disconnect from the power supply)
- Starte Dein OpenCPN-System mit Wifi-Access-Point
- In OpenCPN, start the pypilot plugin (activate the AP once by clicking on the "AP" field, then the window is displayed correctly, deactivate the AP again by clicking on "AP").
- In the plugin Navigiere to Settings
- In the address bar for the host, enter the previously assigned IP address for the Raspberry Zero and confirm with "ok"
- After a short time, the message in the top row of the pypilot window should be connected with... and the color for the plugin icon in the menu bar changes from gray to red
- Disconnect the Raspberry Zero from the power supply, the icon turns grey again and in the line at the top of the window is now disconnected
- Plug the Raspberry onto the designated slot on the motherboard

- To transfer further data (wind, GPS, route information) to the autopilot, set up a new connection in OpenCPN (under Settings) with the following configuration:

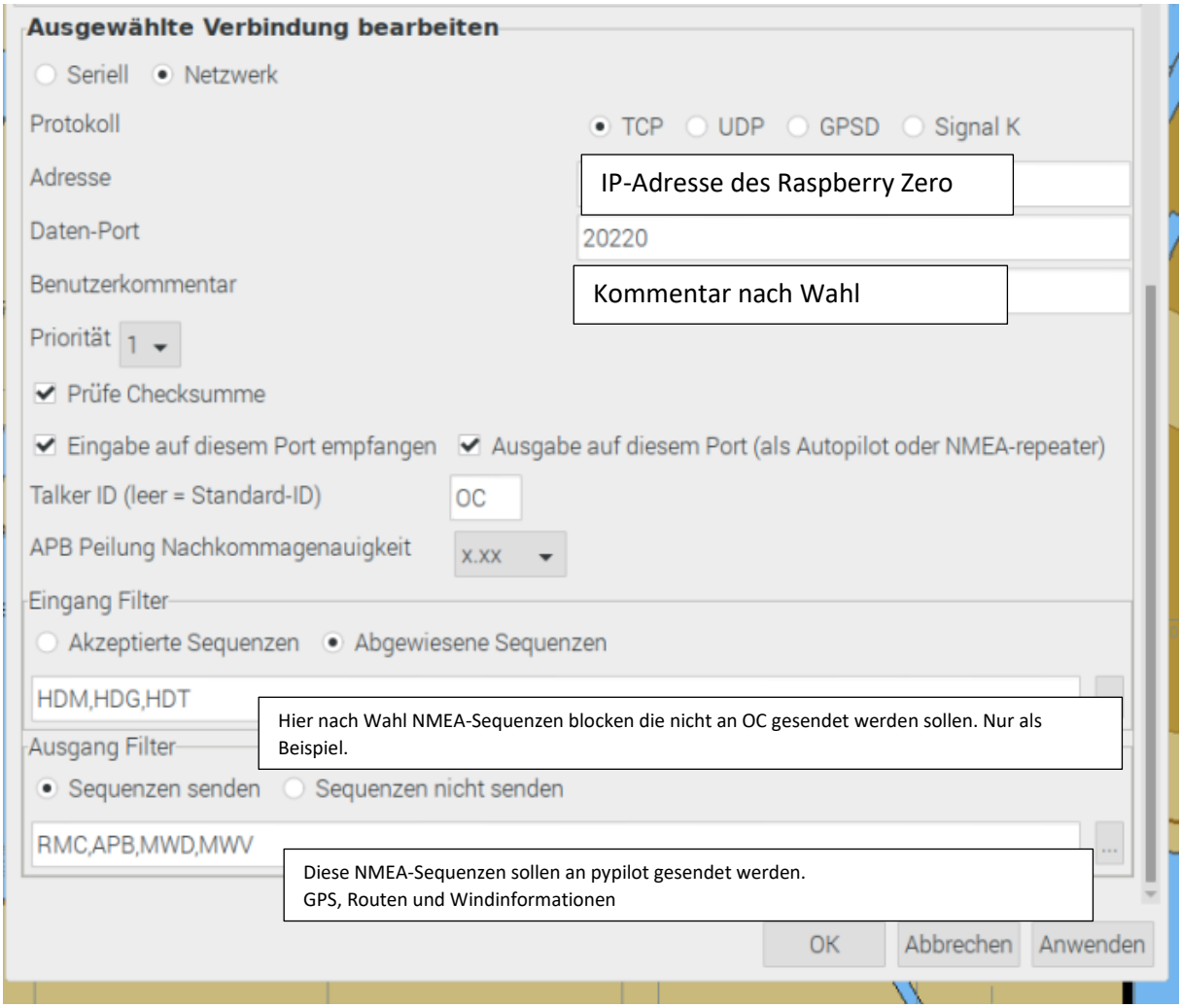


Abbildung 26: Verbindungseinstellung in OpenCPN

This completes the preparations in OpenCPN

5.2 Arduino Nano

The speed control and direction of rotation of the motor is carried out via an Arduino Nano as motor controller by means of the corresponding program.

The program must be "flashed" on the Arduino.

On github both the binaries (which can be easily flashed to the Arduino without further programming work) and the Arduino Sketch are set for own program developments.

There are two binaries on github

pypilotmotorcontrollerwithoutrudder.**** - Binäry file if no ruddersensor is used
The function rudder angle sensor is disabled

Pypilotmotorcontrollerwithrudder.**** - Binäry file if a ruddersensor is used
This file can also be used without a rudder angle sensor.
Then you have to connect "Sense" with -5V.

Instructions for flashing the binaries (Windows Calculator):

- Download and install the usb driver for the Arduino Nano from github
  (the driver is for the Arduino models CH340G or CH341SER USB chipset)
- Download the zip folder xloader from github and unpack it
- Download the appropriate binary for the Arduino from github
- Connect the Arduino to your PC via USB
- Start the program Xloader

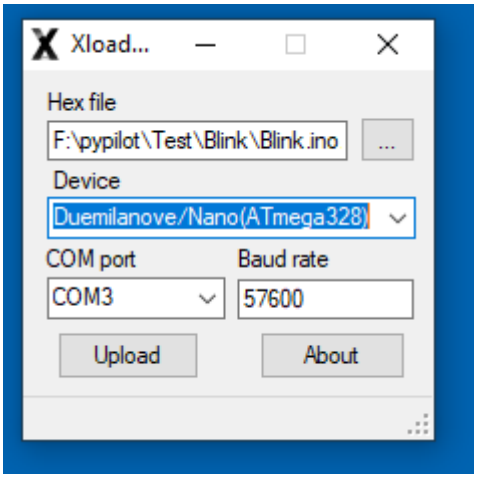


Abbildung 27: Programm XLoader

- Open with click on ... the binary file, select "Duemilanove/Nano(ATmega328) as the device, select the correct COM port. The baud rate is set automatically.
  Start flashing with "upload" on the Arduino flashing the LEDs.
  If the LEDs stop flashing, flashing is finished.
- Disconnect the Arduino from the PC and plug it onto the slot on the motherboard

Notes for programmers:

Image for Raspberry:

The image tinypilot for the Raspberry is modified for the use of the LCD display "jlx12864".
To do this, the entry "nokia5110" has been changed to "jlx12864" in the file ~/.pypilot/hat.conf.
You can basically equip the display and control unit with the "Nokia 5110" display. Then the entry in the file must be changed accordingly.
However, the currently available displays have qualitative problems, and the display on the "jlx" displays is more pleasant.

Program for the Arduino:

On Github the Sketch (program) for the Arduino is also set and can be downloaded to adapt the program to your own needs.
If you download the folder with the Arduino Sketch and edit it in the Arduino IDE you maybe have to download more libraries and include them in the IDE.
Basically, the important configuration settings are given in the config.h and pins.h.
In pins.h, the inputs and outputs are mainly defined.
In config.h, the basic definitions are made. It determines, for example, which H-bridge is used, which values have the resistors used for current, voltage and temperature measurement, and which functions (limit switch, rudder angle sensor and others) are activated or deactivated.

The motor is controlled by PWM.
The PWM frequency has been set by me to 31 kHz and is therefore outside the audible range for humans. One only takes the "normal" engine noise. At lower PWM frequencies, the motor noise is clearly audibly overlaid by the PWM frequency.

The PWM frequency is set in line 96 of the main program and could be adjusted there.
Information can be found in the www, for example here (german):
http://www.scynd.de/tutorials/arduino-tutorials/3-luefter-steuern/3-1-pwm-ohne-pfeifen.html

6 Commissioning

The motherboard has a number of connectors.
The following collection goes from left to right

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

Note on J5:
The connections in 12V, In12+ and In Seat are internally connected on the board directly to the respective Out connectors Out 12V-, Out12+ and Out Seat.

The background is that I am currently using a Raymarine pin pilot with Seatalk connection. In the future, Pypilot will be supplied via the 12V supply of the pinnenpilot. As a backup, the three lines to the pin pilot connection are continued, so that the old pin pilot can be used at any time in the event of a failure of the pypilot.

Preparation Commissioning:
If the voltage regulator on the motherboard has not yet been set:
Remove the Raspberry, The Arduino and all ICs from the mounts from the motherboard. Unplug the ribbon cable plug on the IBT2 bridge. Connect in 12V and 12+ to a 12V voltage source.
Set the voltage regulator so that you measure 5.05 to 5.15V at the output. Disconnect the power supply. Insert the Raspberry, Arduino and ICs into the mounts on the motherboard.
Reconnect the ribbon cable to the iBT2 bridge slot.

Connect the display and buttons as well as the compass module to the motherboard. Connect the actuator to the motherboard.
When using a rudder position sensor: The two outer connections of the potentiometer are connected to + / - 5V, the center handle to sense. To protect the potentiometer against false connection, I recommend switching an additional resistor 1k in series (see schematic).

Connect J8 to the power connector (screw terminals) of the IBT2 bridge, the connections are arranged the same at both screw terminals (see also Pic 2).

Raspberry and Arduino are prepared according to Chapter 5.

Start your Opencpn system with Wifi accesspoint, start the pypilot plugin.

6.1 Commissioning Pypilot

Connect in 12V and In 12+ to the 12V power supply and turn it on.

A flickering green LED on the (now bottom) of the Raspberry indicates that it is booting.
The background light of the display lights up. A red LED lights up on the Arduino. After a short time they start flashing, the Raspberry communicates with the Arduino.
After approx. 20 to 50 sec. a heading information and further information is displayed on the display.
After approx. 60 to 120 sec. OpenCPN has a connection to the pypilot.
Check whether the actuator is moved accordingly by pressing a button on the directional keys in the plug-in.
By reversing the connection, the direction of rotation of the actuator can be adjusted if necessary.

Problems and possible solutions:

Nothing happens, no LEDS light, nothing happens	Tension present? Properly connected around? Voltage regulator set correctly? No connections swapped? Don't forget a connection, all connected correctly? SD card in Raspberry? SD card flashed with program? If so, try other SD card Arduino flashed?
The Raspberry boots, no LEDs on the Arduino	Arduino does not get any voltage or is defective.
The Raspberry boots, LEDs flicker at the Arduino no connection to OpenCPN	Check settings of Wifi in pypilot and OpenCPN
Pypilot is running, connection to OpenCPN is present Aktuator don´t move	If the actuator is not running at all disconnect the rudderangle sensor from the mainboard and check. If still no success then the IBT2 bridge may be defective.

6.2 Settings in OpenCPN

OpenCPN requires some basic settings.

When using a rudder position sensor:
In the plugin navigate to Calibration, then Rudder.
Press "Reset calibration" once drive the actuator to center position.
Then press "Centered". In the field "Range" enter the rudder angle at maximum output actuator at for example 20 for 20° (When mounting a Raymarine-Pinnenpilot according to instructions, it is 18° at the final positions of the pinnenpilot)
Drive the actuator entirely to the starboard. Then press "Starboard Range".
Drive the actuator all the way to the port. Then press "Port Range". Confirm with Ok. T
he actuator should then drive halfway in the middle position when pressing "C" in the plugin.

The start and follow-up, as well as the speed of the motor, is set and adjusted in the plugin under "Calibration" and then "Settings" with the points: servo.max_slew_slow, servo.max_slew_speed, servo.speed.max and servo.speed.min.

Compass calibration:

It is not guaranteed that the position and accelerometer sensors of the compass module are already calibrated from factory.
I therefore recommend calibration in OpenCPN.

To do this, you first call "Calibration" in the plugin, then the item "Aligment".
The compass module is placed on a flat surface and pressed "Level".
When the bar has passed, select the "Accelerometers" tab.

Then you turn the compass module to all 6 sides (imagine that the module is in a rectangular housing, place the compass module on all 6 sides of the housing), on each side you hold the module approx. 10 sec. without moving it.
In the status window you get a corresponding message if the calibration was successful and the "age" of the calibration should be recounted. Repeat this until you see a new calibration. If you are not successful, you may have to change the compass module because the position sensors do not work correctly or are defect.

After calibrating the position/accelerometer sensors, select the tab "Aligment" again, place the compass module on a flat surface again and press "Level". If the compass module has been installed later on board, this step will be repeated.

The actual calibration of the compass is carried out by driving full circles with the boat, on the table the compass module is rotated 360 degrees (quietly a larger circle radius and as little magnetic interference as possible). The compass course should then be displayed in the plugin halfway appropriately.

You can now test whether the compass module is moving to display corresponding compass courses. It is also possible to test which settings on the control parameters lead to corresponding control actions on the actuator.

But you will only be able to get a rough impression, you can test it properly only on board.

The calibration steps for leveling and calibrating the compass (360-degree circle) must be repeated in any case after installation on board.

6.3 Settings control parameters

The control parameters must and can be adjusted to the respective wind and wave situation. I copy here the original instructions / notes about it from the wiki page of pypilot

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 and sources

https://pypilot.org/
All over pypilot, Seite von Sean D´Epagnier
https://github.com/McNugget6750/pypilot/tree/master/arduino
Modified Arduino Sketch for the motor controller, 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/en/pypilot/
pypilot auf Open-boat-projekts
https://open-boat-projects.org/en/
Much more great projekts

Software for the solution described here:
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

Pictures and schematics:
https://github.com/AndreasW29/pypilot-tinypilot-mysolution-photos-
Pictures
https://github.com/AndreasW29/pypilot-tinypilot-mysolution-infos
Kicad schematics and guide

Components and modules (all examples, I do not receive a commission):
In the local electronics store or
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-Regulator

https://pcnautic.nl/nl/autopilot
Linear actuators with limit switches, rudder position sensor and Raymarine mounts,
very friendly online retailer

The circuitboards can be ordered from me (andreasw1402@gmail.com) or with the Gerber files in the Github folder from a board manufacturer.

8 Thanks

At this point I would like to thank Sean D'Epagnier for his development and deployment of pypilot.
On the Website https://pypilot.org/ you have the opportunity to make a donation under "Contact". If you replicate and successfully operate a pypilot, I think a donation is appropriate and fair.

9 Known issues

Tacking function does not work.
You can set up additional directional change keys +100 as "Tacking" button in the plugin.
You can also prove the buttons of the control unit with different direction changes via the menu on the pypilot.

In the plugin, the settings and gains are temporarily not displayed - solution compile the plugin from source.