Controlling Brushless DC-Motors via Raspberry Pi

With GUI (Graphical Userinterface), Brushless DC-motors, Motor-Controller

This tutorial explains the construction of a motor controller based on the Raspberry Pi (RPi). The speed of brushless DC-motors are normally controlled via PWM signal. Here is the relevant manual of the controller.

> https://www.krick-modell.de/shop_fachhandel/Texte/67055_en.pdf.
>
> *https://www.faschingbauer.me/trainings/material/soup/linux/hardware/brushless-motor/topic.html*

Hardware setup

This are the hardware components in our case:

- Raspberry Pi 4
- 2x brushless DC motor: krick MAX P285 Marine
- 2x Brushless electronic speed controller (ESC): QUICRUN 16BL30
- 2x Battery: Robitronic 5200mAh 40C, 2S (7,4V)
- Wires for Raspberry Pi's GPIO connection
- Monitor, Mouse and keyboard for Raspberry Pi (RPi)

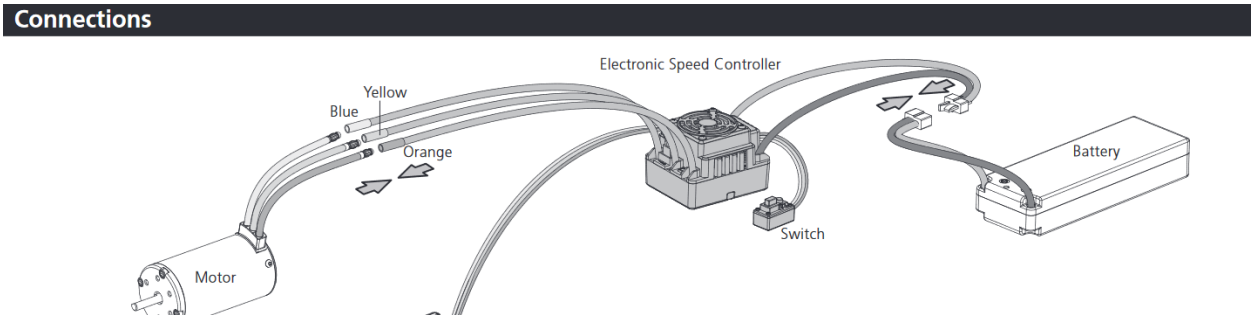
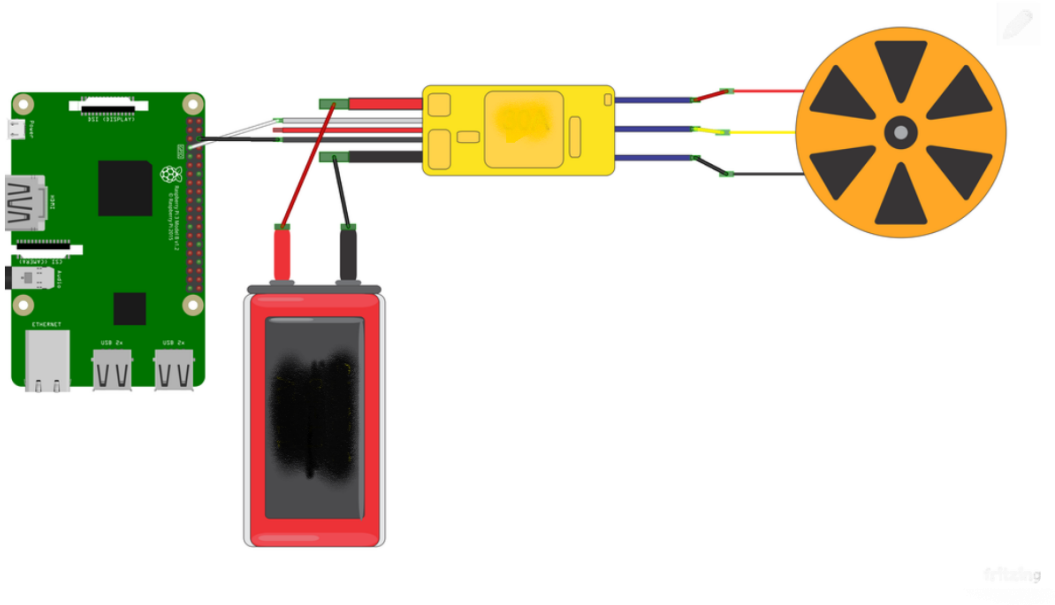


Figure 1 Here we don't have the receiver

If we want to reverse the rotation of a motor we only switch the thick yellow and orange In our case there are no receiver. Instead we connect the three wires (black, red, white) to the GPIO

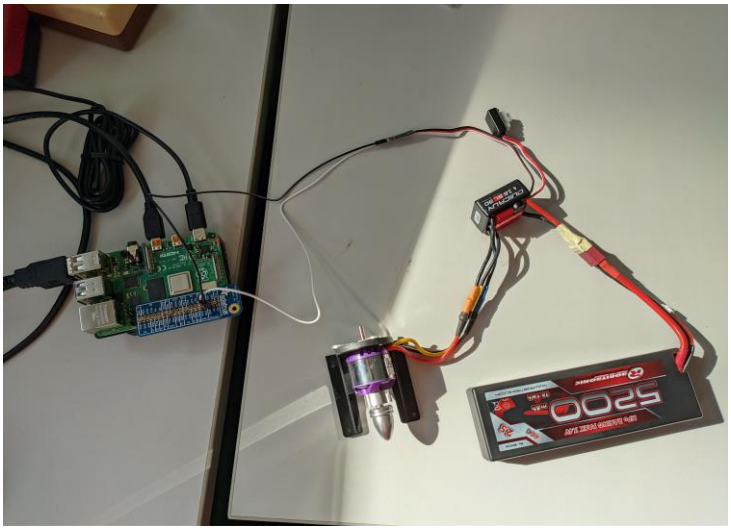


PINs of the RPI.

This image shows the GPIO PINS with the wires connected. To see the PIN layout: Turn on RPi, Open RPi Terminal and execute: `pinout`

Now we need to connect the three wires:

- Black wire goes to GND
- White wire goes to any GPIO PIN number (example GPIO PIN #4), here the PWM signal will be send
- Red wire is not connected to anything. Just ignore it.



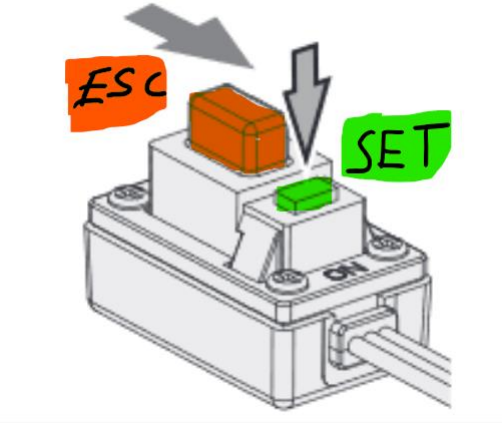
In our setup, the white wire from the first motor is connected to the GPIO PIN #4 and the white wire from the second motor is connected to the GPIO PIN #25.

Our additional Servo motor: orange wire to GPIO PIN #17, black wire to GND and red wire to 5V PIN.

The PIN correlation for motors are documented within the code: (GUI_ESC.py & GUI_ESC_server.py)

Set up the ESC hardware (QUICRUN 16BL30)

By default, the motor controller ESC let the motors only drive forward. We want to enable reverse speed. To do so, the switch has an ESC slider AND a set button.



STEP 1: Green LED flashes one mode

1. Turn off ESC, if it's not already off
2. Hold the SET button
3. Switch on ESC
4. Wait: Red LED flashes, and then green LED flashes once
5. Release SET button
6. Cycle through modes, pressing SET button (until red LED flashes 3times)
    a. Forward with brake (one red LED flash)

b. Forward/reverse with brake (two red LED flashes; this is the default mode)
c. Forward/reverse (three red LED flashes): *stop here*

7. Turn off ESC to select that mode

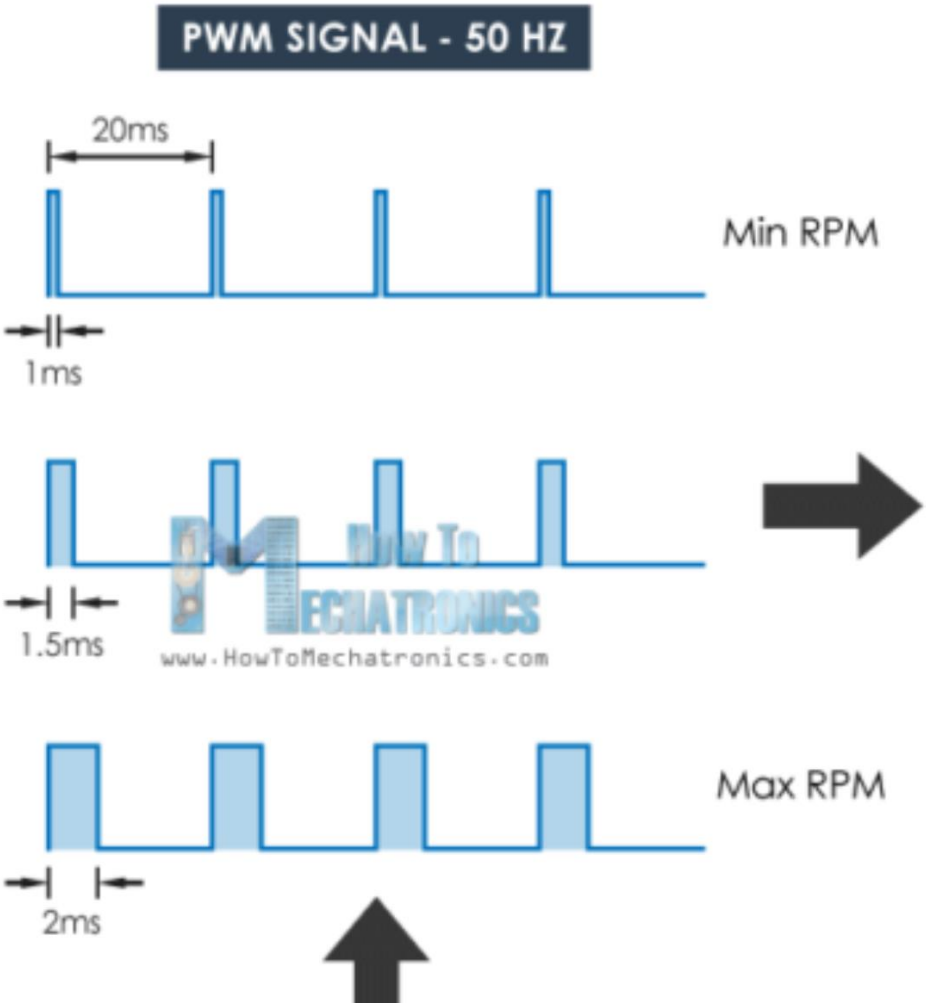
STEP 2: Green LED flashes twice mode

1. Turn off ESC, if its not already off
2. Hold the SET button
3. Switch on ESC
4. Wait: Red LED flashes, and then green LED flashes TWICE
5. Release SET button
6. Cycle through modes, pressing SET button. Choose a!! when one red LED flashes.
   a. Drake Brake Force 0% (one red LED flash): *stop here*
   b. Drake Brake Force 5% (two red LED flashes
   c. Drake Brake Force 10% (three red LED flashes)
7. Turn off ESC to select that mode

NOTE: **Every time you are using the motor you must TURN the ESC-button ON. After usage it should be turned off to save battery**.

PWM and CODE

The speed depends on the frequency on the PWM signal, which is sent through the white cable through RPi GPIO PIN.



So the ESC's "command protocol" requires us to output a PWM signal with a 50Hz (20ms) period. Within that period, the PWM signal's duty cycle is used to communicate the following commands to the ESC.

- *Neutral position*. The middle diagram in the picture, 1.5ms. Python Code: `p.set_servo_pulsewidth(GPIO, 1500)`
- *Backward rotation*. The top diagram; everything between 1ms and 1.5ms lets the motor run backwards. 1ms is maximum backwards speed. Python Code: `p.set_servo_pulsewidth(GPIO, 1000)`

- *Forward rotation*. The bottom diagram; everything between 1.5ms and 2ms runs the motor forward. 2ms is maximum forward speed. Python Code: `p.set_servo_pulsewidth(GPIO, 2000)`

Hardware: On our Raspberry Pi we connect the white wire of the first motor to GPIO PIN #4 and the second motor to GPIO PIN #25

Before running the code, open the Raspberry Pi OS Terminal and execute following commands:

- `pip3 install guizero`
- `pip3 install time`
- `pip3 install pigpio`
- `pip3 install RPi.GPIO`

Here's the code: (just copy&pase):

```
#########################################################################
######
#REQUIRED HARDWARE: 2x BRUSHLESS DC-MOTORS with controllers, 1x
Servomotor
#########################################################################
######
#To run this code on Raspberry Pi OS you have to open terminal
#and run this commands: sudo pip3 install guizero

from guizero import App, PushButton, Text, Slider, info
import time
import RPi.GPIO as GPIO
GPIO.setwarnings(False)
#GPIO.setmode(GPIO.BOARD)
GPIO.setmode(GPIO.BCM)
import os
os.system ("sudo pigpiod")
import pigpio

#Connect the orange wire of Servo Motor to #PIN 17
GPIO1=4  #Motor1 signal from GPIO PIN #4.
GPIO2=25  #Motor2 signal from GPIO PIN #25.
p1 = pigpio.pi();
```

```
p2 = pigpio.pi();
"""
#THIS CODE IS FOR MOTOR CONTROL VIA "import RPi.GPIO as GPIO""
#Dutycycle value range: m1.ChangeDutyCycle(min 7 - max 12)
GPIO.setup(GPIO1, GPIO.OUT)
m1 = GPIO.PWM(GPIO1, 50)
m1.start(0)

GPIO.setup(GPIO2, GPIO.OUT)
m2 = GPIO.PWM(GPIO2, 50)
m2.start(0)
"""

#p.set_servo_pulsewidth(GPIO, 1000)  ->  max.backwards speed
#p.set_servo_pulsewidth(GPIO, 1500)  ->  no speed
#p.set_servo_pulsewidth(GPIO, 2000)  ->  max.forward speed


def motor1(slidervalue):
    while True:
        value = int(slidervalue)*5 + 1500
        p1.set_servo_pulsewidth(GPIO1, value)
        print("pwm in ns:", value)#ns =nanosecond
        break

        """
        value = int(slidervalue)/20 + 7
        if value ==7:
            m1.ChangeDutyCycle(0)
            print("stopped")
        else:
            m1.ChangeDutyCycle(value)
            print("Dutycycle:", value)
```

```
        break
       """


def motor2(slidervalue):
    while True:
        value = int(slidervalue)*5 + 1500
        p2.set_servo_pulsewidth(GPIO2, value)
        print("pwm in ns:", value)#ns =nanosecond
        break


        """
        value = int(slidervalue)/20 + 7
        if value ==7:
            m2.ChangeDutyCycle(0)
            print("stopped")
        else:
            m2.ChangeDutyCycle(value)
            print("Dutycycle:", value)
        break
"""
#Servo-motor control via PWM directly from Raspberry Pi GPIO
#Connect the orange servo wire to PIN17
servoPIN = 17  #Motor signal from GPIO PIN 17.

GPIO.setup(servoPIN, GPIO.OUT)
p = GPIO.PWM(servoPIN, 50) # GPIO 17 for PWM with 50Hz
p.start(7.5) # Initialization

def servo(slidervalue):
    value = int(slidervalue)/18 + 7.5
    p.ChangeDutyCycle(value)
```

```
"""
With p.ChangeDutyCycle(value) we can change the servo position by
entering the DUTY Cycle value
p.ChangeDutyCycle(2.5)   #duty cycle of 2.5% means -90degree angle
p.ChangeDutyCycle(7.5)   #duty cycle of 7.5% means 0degree angle
p.ChangeDutyCycle(12.5)   #duty cycle of 12.5% means +90degree angle
"""
def stop1():
    """Stop all movement."""
    p1.set_servo_pulsewidth(GPIO1, 0)
    print("Movement stopped")
def stop2():
    """Stop all movement."""
    p2.set_servo_pulsewidth(GPIO2, 0)
    print("Movement stopped")



def close_window():
    stop1()
    stop2()
    app._close_window()



#GUI window (internal name: app)
app = App(title="Motor Touch Control", width=480, height=320,
layout="grid")
#infotext = Text(app, text="Control the speed and direction of the
boat", grid=[0,0])
#infotext.text_size=10
#testbutton = PushButton(app, command=testfunc, text="Testbutton",
grid=[0,2])
motor1text = Text(app, text="motor1", grid=[0,3])
motor2text = Text(app, text="motor2", grid=[0,5])
```

```
servotext = Text(app, text="servo(angle)", grid=[0,7])
motor1_slider = Slider(app,height=35 , width=200, command=motor1,
start=-100, end=100, grid=[0,4])
motor2_slider = Slider(app,height=35 ,width=200, command=motor2,
start=-100, end=100, grid=[0,6])
servo_slider = Slider(app,height=35 ,width=350, command=servo, start=-
90, end=90, grid=[0,8])
motor1stop = PushButton(app, command=stop1, text="Stop motor1",
grid=[1,4])
motor2stop = PushButton(app, command=stop2, text="Stop motor2",
grid=[1,6])
close = PushButton(app, command=close_window, text="EXIT", grid=[1,8])

app.display()
```

Create a python file (name.py) on the RPi and paste the code from above.

Run it.