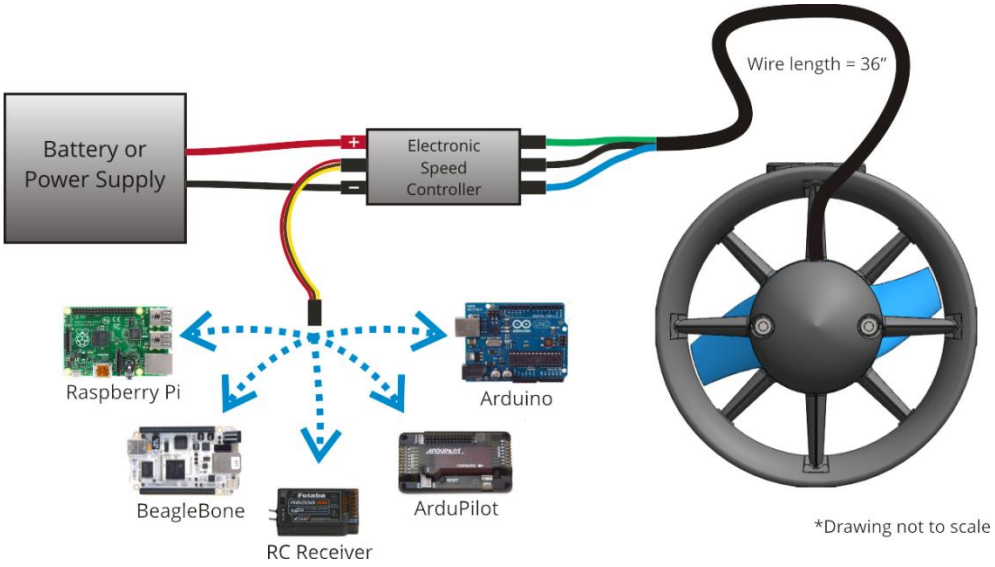
2016.6.27 Let Raspberry Pi Drive Thruster

Part 1: Assembly of Devices



Connect ESC with other devices

1. Connect the three motor wires to the motor. The order of connections does not matter, however, switching any two wires will change the direction of the motor. The output phases A, B, and C are completely interchangeable
2. Connect the red power wire and black ground wire to a power source like a battery. You will hear a few beeps from the ESC.
3. Connect the signal cable to your signal source like an RC radio receiver or microcontroller board. The yellow wire is the signal wire. The red wire is the battery eliminator circuit (BEC) output, which supplies 5V at 500mA to power a control system. The brown wire is ground.
4. Send a stopped signal (1500 microseconds) for a few seconds to initialize the ESC. You will hear a long tone.

Reference for Assembly

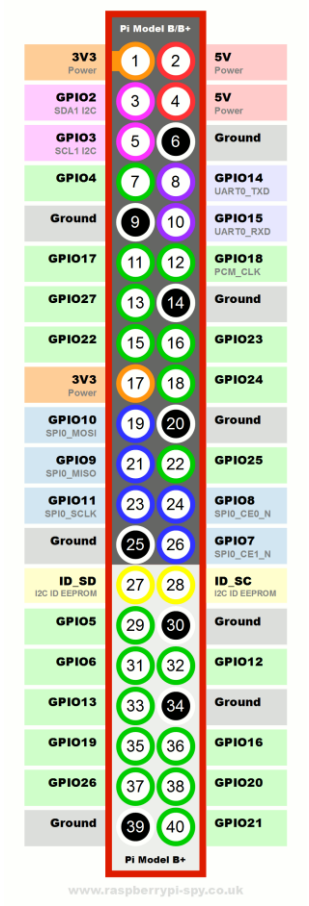
T200 Thruster Documentation http://docs.bluerobotics.com/thrusters/t200/

Basic ESC Documentation http://docs.bluerobotics.com/besc/

Raspberry Pi GPIO Layout of Different Pi Versions

http://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/

Raspberry Pi GPIO Layout – Pi 3 Model B



Part 2: Drive Thruster

Extended Library for Controlling GPIO on Raspberry Pi

Pigpio library: works on all versions of the Pi. It is written in C but may be used by other languages.

Library Documentation http://abyz.co.uk/rpi/pigpio/index.html

Install http://abyz.co.uk/rpi/pigpio/download.html

There are other popular extended libraries but don't work well in our project.

GPIO library: This library uses "software PWM", meaning that the timing for the PWM signal is managed by the main processor instead of a dedicated timer. Thus, the generated PWM is relatively inaccurate.

RPIO library: This library only works on Pi 1, because the library has stopped updating since 2013.

Complete Guide about how to control RPi GPIO with different programming language

http://elinux.org/RPi_GPIO_Code_Samples#pigpio

Python code to drive thruster

```
# This program first initializes ESC and then sends PWM pulses to servo unit on ESC. After receiving PWM pulses, ESC sends signals to the thruster and the thruster will start spinning.

#!/usr/bin/env python

# To use API in Pigpio library, the pigpio daemon must be running
sudo pigpiod # start the pigpio daemon

import pigpio
import time

SERVO = 11 # pin number connected to signal cable of ESC

pi = pigpio.pi() # Connect to local Pi.

# Initialize (calibrate) ESC by setting its max and min throttle (for BlueROV ESC, this step can be skipped.)
pi.set_servo_pulsewidth(SERVO, 1000) # Minimum throttle.
time.sleep(1)

pi.set_servo_pulsewidth(SERVO, 2000) # Maximum throttle.
time.sleep(1)

# send "stop" signal to initialize BlueROV ESC
pi.set_servo_pulsewidth(SERVO, 1500)
time.sleep(1)

# Send PWM pulses to ESC
# If a throttle of 1100 fails to drive thruster, increase by 100 each time until the thruster starts spinning (Note: the thruster stops spinning at a throttle of 1500)
pi.set_servo_pulsewidth(SERVO, 1400) # Slightly open throttle.
time.sleep(1)

pi.set_servo_pulsewidth(SERVO, 0) # Stop servo pulses.
pi.stop() # Disconnect from local Raspberry Pi.
```

BlueESC Initialization

There's actually no calibration routine in the firmware, they are set to hard limits of:

1100 µs: full reverse
1500 µs: stopped (and initialize)
1900 µs: full forward

Send a “stopped” signal at 1500 µs for a few seconds (one or two seconds) to allow the ESC to initialize. It will beep and the lights will flash briefly.

No need to set max & min signal.

Send a PWM signal at 0 µs is an invalid command and will produce unusual results because such signal is outbound. A 0 µs pulse is the same as no pulse at all, which means that the ESC is not receiving any signal and after a short time it will require re-initialization to work.

ESC deinitializes if it has not received any signal for 3~10s.

Specification of BlueESC

Pulse Width Signal	
Signal Voltage	3.3-5 volts
Update Rate	50-400 Hz
Stopped	1500 microseconds
Max forward	1900 microseconds
Max reverse	1100 microseconds
Signal Deadband	+/- 25 microseconds (centered around 1500 microseconds)

Reference

BlueESC Documentation

Execute C program via terminal

- If <pigpio.h> is imported

Compile & link

gcc -o program_name program_name.c -lpigpio -lrt –lpthread

Run

sudo ./program_name

- if <ncurses.h> is imported

Compile & link

gcc program_name -lncurses

Note: file name should not have space

Reference for Programming

RPi = Raspberry Pi

Original source of the program above

How to start and stop the pigpio daemon

Virtual oscilloscope provided by Pigpio library

Why need to calibrate ESC

Introduction to Pulse Width Modulation