Rasperry pi Tutorial

Table of Contents

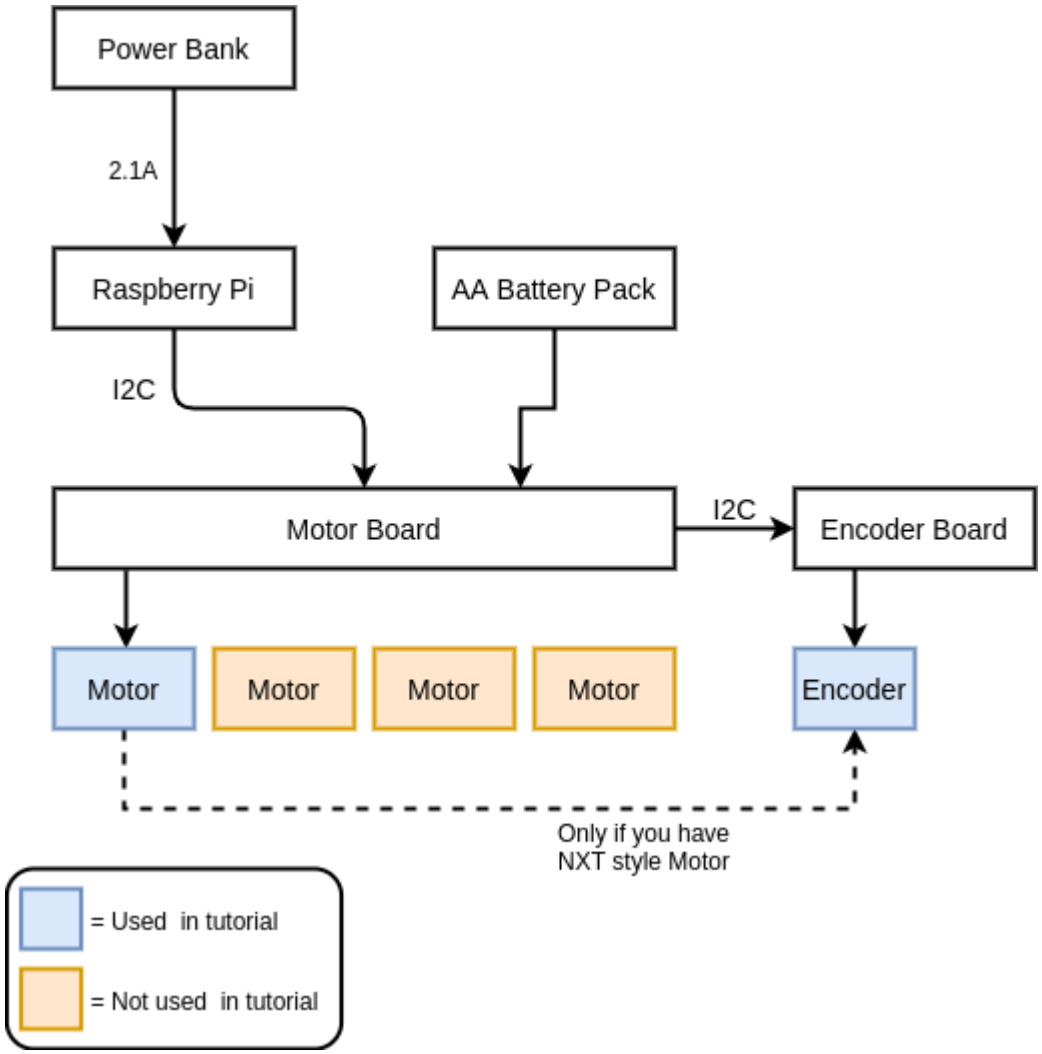
Introduction

In this tutorial, you will learn the basics of the SDP Raspberry pi (RPi). We will focus on wiring up the pi, connecting via SSH and controlling a motor.

Wiring up the pi

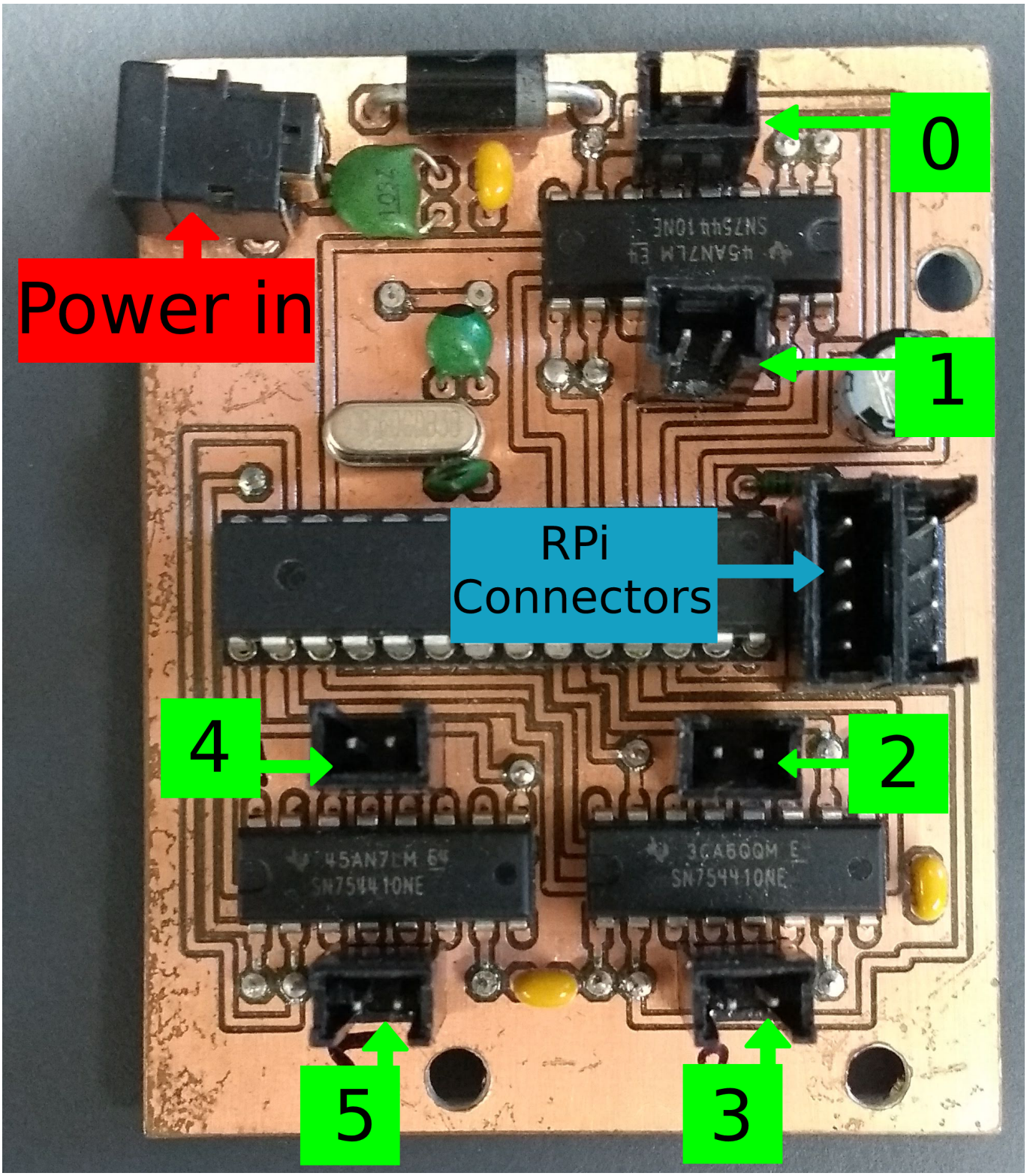
For this tutorial, we will use a subset of the full kit you received on kit hand-out day.

Follow the diagram below to set up the hardware we need for this tutorial.



Make sure the pi is plugged into the 2.1A line from the power bank

Motor Board ports are numbered as follows:



Connecting to the pi

Once everything is plugged in as above, the pi will be powered up.

If you can't see the red light on the pi, remove the micro-usb cable between the power bank and the raspberry pi. Your pi should now power up.

After powerup, wait a few seconds for the boot sequence to complete

Using SSH

The case around your RPi contains a label with the name of your board. You will need this name to start using the RPi.

If you are using a non-DICE machine, make sure you are on the same network as the robot (usually SDProbots).

Once this is setup up, open a terminal window.

Use the SSH command to connect to your robot:

```
ssh -XC pi@<robot_name> -t tmux a -t tmux-main
```

where the `<robot_name>` is replaced with the name of the robot. For example if your robot is called `Panda`, you would run: `ssh -XC pi@Panda`.

Remember to add the `-XC` option so that you can later start graphical text editor and debug your image processing remotely.

When you are prompted, enter the password:

```
r00t
```

When these commands have been run, you should be connected to your robot and can start writing your code.

Ensure i2c is Enabled

The i2c bus should be activated on the pi. If not, follow the instructions in this link

Moving Motors and Reading Encoders

```
from motors import Motors
from time import time, sleep

# Create an instance of the Motors class used in SDP
mc = Motors()

motor_id = 0                            # The port that your motor is
plugged in to
speed = 100                             # forward = positive, backwards =
negative
run_time = 2                            # amount of seconds to run motors


# Move motor with the given ID at your set speed
mc.move_motor(motor_id,speed)

start_time = time()
```

```
# Encoder board can be fragile - always use a try/except loop

while time() < start_time + run_time:
  mc.print_encoder_data()
  sleep(0.2)            # Use a sleep of at least 0.1, to avoid errors

mc.stop_motors()
```

The code explained

```
motor_id
```

is the number of the port into which you plugged your motor into according to the diagram above

```
speed
```

gives you limited control over the velocity of the motor, though it doesn't work will. With the older motors, you can set this between 80 and 100.

Set positive to move the motor forwards and negative to move backwards.

```
mc.print_encoder_data()
```

prints out the values of the encoders on the encoder boards. These are not translated into exact rotations for the encoder in the tutorial, as any encoder you use during teh course will need to be calibrated.

Instead, these numbers tell you the amount of clicks taken since the last reading. You can use this to control how much your wheel turns in a given direction instead of relying on time.

The format of the encoder readings are as follows:

- If the number starts starts from 1 and increases, the encoder is being turned forward
- If the number starts from 255 and decreases, the encoder is being turned backwards

Now try editing the values and moving the encoders to change the values on the encoder output.

Moving with Conditions

You can also try making a sequence of actions for your motors.

Try making the motor make the following motions to get more familiar with the equipment:

- Move forwards for 100 clicks
- stop motors 2 seconds
- Move backwards for 100 clicks

- stop motors

Read a specific encoder

Instead of printing the value of all encoders, what will be more useful to you during your project is to read the value of a given encoder port. To do this, instead of using

```
mc.print_encoder_data()
```

instead, use:

```
mc.read_encoder(id)
```

where id is the port number on the encoder board that you wish to read