Workshop 11: GPIO & Motors

In this workshop, we'll learn about motor controller chips, and make a breadboard circuit to drive a motor.

Difficulty: Intermediate workshop

Contents

0 Introduction

The workshop shows how to control a motor with the Raspberry Pi, using a breadboard circuit and a standard motor controller chip. The motor is then programmed with the Python GPIO library.

This is an intermediate workshop, it will help to have some previous experience with Python.

These booklets were created using $\LaTeX$, an advanced typesetting system used for several sorts of books, academic reports and letters. The schematic diagrams were created using *Fritzing*, a free hardware prototyping and PCB design software.

What you'll need

The hardware requirements for this workshop are listed below.

- A solderless breadboard, preferably one with separate bus strips. (those labelled + & -)
- Male-to-female & male-to-male jumper cables.
- A 6V DC motor.
- A L293D or L293DNE motor controller.
- (Optional) a 4×AA Battery Pack

For the LED exercise in section 1, you will also require:

- An LED
- A $\sim 330\Omega$ Resistor

Code listings & asides

When you need to make changes to your code, they'll be presented in boxes like the following:

```
1  #This is a comment
2  while True:
3          print("Hello, World!")
```

You might not need to copy everything, so check the line numbers to make sure you're not copying something twice.

Questions?

If you get stuck, find errors or have feedback about these booklets, email: jam@jackjkelly.com

1 A Simple LED Circuit

1.1 Setting up the circuit

Let's start off with a simple circuit. Firstly, we'll always shut down and **unplug** our Pi before we begin connecting to the GPIO.

We will build our circuit using a breadboard. Breadboards have conductive wire underneath their circuits, which form 'lanes'. You can see these wires in figure 1.

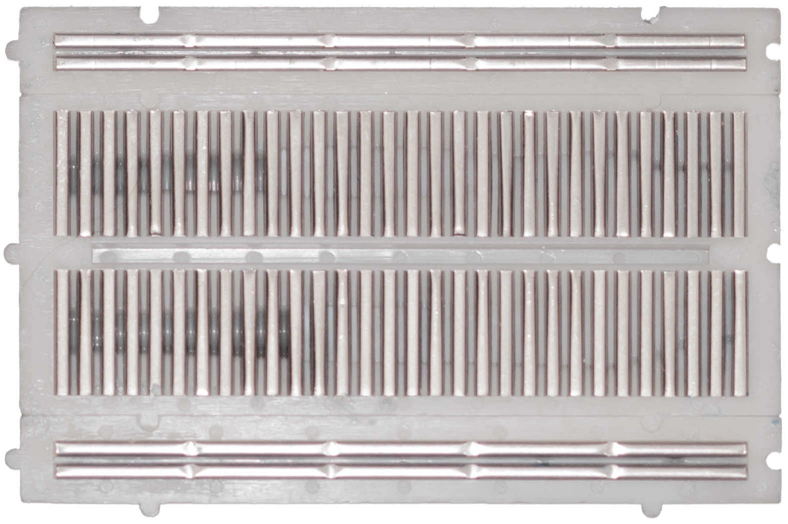


Figure 1: The underside of a breadboard, showing the direction of electrical connections.

We will use the positive and negative bus 'lanes' like the ends of a battery in a simple electronic circuit, current starts at the positive end, and travels through components to the negative end.

To make a complete circuit, we will connect these lanes to the Pi's GPIO.

1. **Connect** the positive bus to the 3V3 pin on the Raspberry Pi with a jumper cable.
2. **Connect** the negative bus to one of the GND pins with another jumper cable.

By convention, you should use a red cable for the positive terminal and black for the negative terminal.

3. **Place** the LED across the gap in the centre of the breadboard, a pin on each side.
4. Use a Jumper cable to **connect** the positive bus to the positive leg of the LED.
   - The LED is a Diode, so only lets current through one way.
   - the positive leg (anode) of the LED is the longer of the two legs.

We will now need to complete the circuit by connecting the negative pin of the LED.

However, the LED will try and take a lot of power. To prevent it from burning out the Raspberry Pi, we'll add a resistor to limit the current flowing through the LED.

5. **Insert** the resistor in the lane where the negative LED leg is placed, and bridge it to any other lane.
6. **Connect** this lane back to the negative bus with a jumper cable.

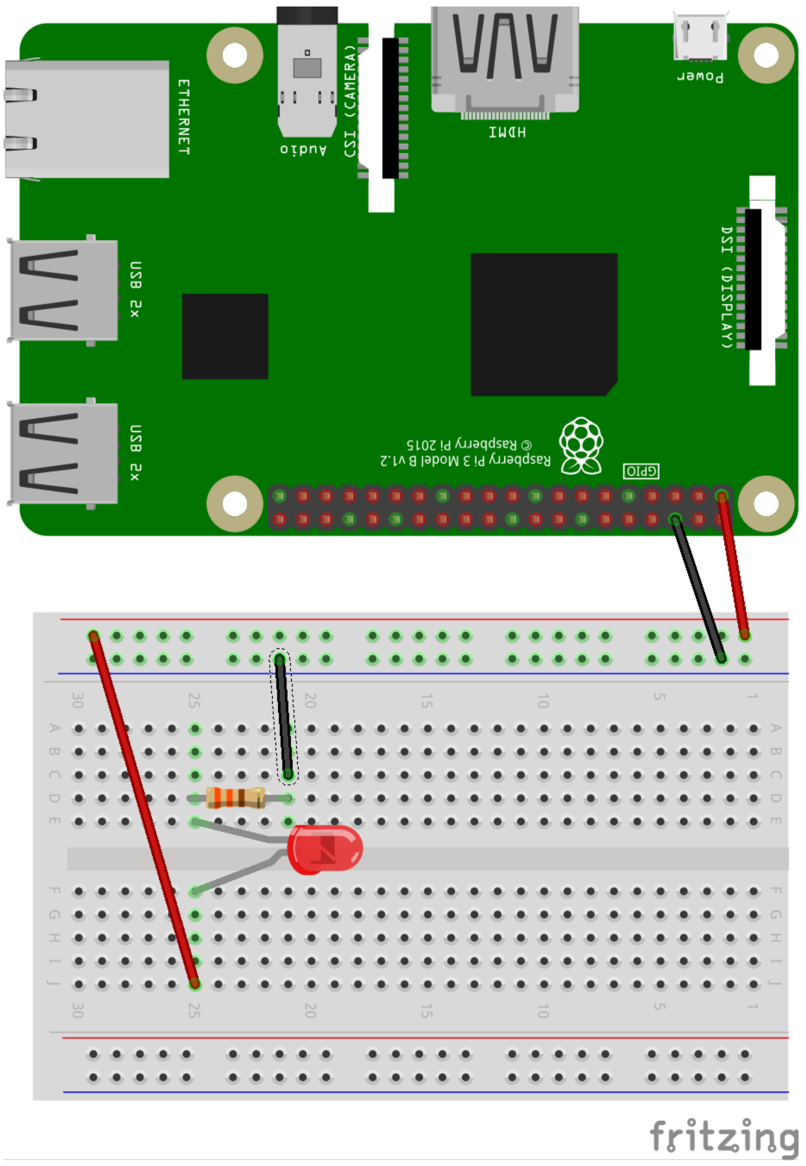


Figure 2: The complete LED circuit

Once you're happy your circuit matches figure 2, you can try turning on your Raspberry Pi. If everything is correctly connected, your LED will light up!

1.2 Modify for GPIO

Our LED is currently connected to a constant 3.3 volt supply, so we are unable to turn the LED off without unplugging the whole Pi. To control the LED through code, we need to use a GPIO pin.

1. Remember to shut down and **unplug** your Pi first.
2. **Remove** the two red cables that were supplying the 3V3 voltage.
3. Use a jumper cable to **connect** GPIO pin 23 to the positive pin of the LED.

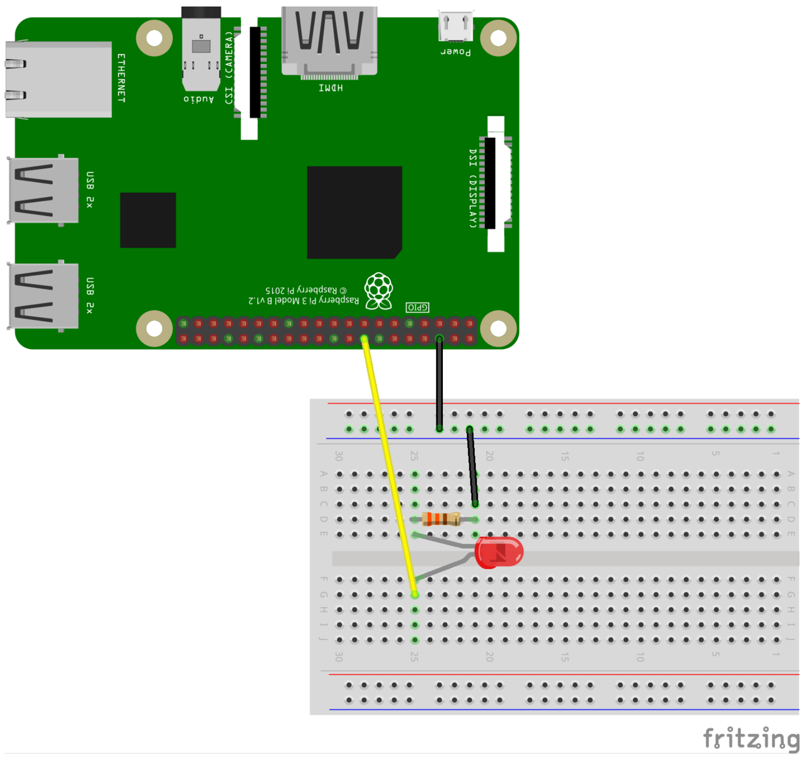


Figure 3: A GPIO controlled LED circuit

1.3 Programming the LED

Once again, if you're happy your circuit matches figure 3, you can start up your Raspberry Pi again. The LED will be off to begin with, so we will write a simple Python program to turn the LED on.

Once your Pi is started up, start *IDLE 3* and open the template python file for this workshop.

```python
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BOARD)

#Write your program below
```

Add the following lines into this file:

```python

#GPIO pins for LED
LED = 16
GPIO.setup(LED, GPIO.OUT)
```

Before we control something through GPIO, we need to setup the pin. We just stated that pin 16 (labelled 'LED') is an output device, i.e. we are outputting a signal to the light, we aren't reading a sensor input.

Note the pin number is 16, not 23. We plugged into GPIO pin 23, but some pins on this block have other functionalities, but are still numbered. (See Appendix A for full pinout)

Now it is set up, we can control the GPIO pin:

```

print("on")
GPIO.output(LED, GPIO.HIGH)
```

Once again, we are sending a command for pin number 16 ('LED'), this time telling it to output a 'high' signal. This will turn on our LED.

Let's have our LED on for a few seconds, then turn it off before the program ends. The function sleep() will cause the program to wait, then we will repeat the GPIO.output() function, this time using GPIO.LOW to turn off the LED.

```

sleep(5)

print("off")
GPIO.output(LED, GPIO.LOW)
GPIO.cleanup()
```

We place GPIO.cleanup() at the end of our program, to reset the pin setup for the next program.

You can now try running your program.

2 The Motor Circuit

2.1 Motor Controller Chips

<Written explanation to be added after the workshop>

The pinout for the L293D/L239DNE chip is shown in Appendix B.

2.2 Setting up a Motor Circuit

Firstly, we'll ensure we have the motors and chip powered.

1. Remember to shut down and **unplug** your Pi first.
2. **Place** the L293DNE chip across the central gap, noting the orientation. In the illustration, the semicircle cutout is facing to the right.
3. **Connect** both positive buses with a jumper cable to a 5V pin.
4. **Connect** the Motor voltage and Chip voltage pins with their closest positive bus.
5. **Connect** at least two ground pins on the chip to the motor's negative bus.
6. **Connect** at least two GND pins on the pi to the Motor negative bus.

Your breadboard should now look like figure 4. Now, we will add the GPIO inputs and motor.

7. **Connect** GPIO pin 23 to input 2A, and GPIO pin 24 to input 2B with a pair of jumper cables.
8. **Connect** GPIO pin 25 to Enable 2 with another cable.
9. **Connect** Connect the motor to the two motor pins (on the motor 2 side). *If you don't have crocodile clips, try bending the pins of two jumper cables to keep them in place.*
10. **Connect** at least two GND pins on the pi to the Motor negative bus.

If you're happy that your circuit is equivalent to the one shown in figure 5, you can try turning on your Pi. If it makes it to the desktop, you probably haven't done anything too disastrous!

2.3 Programming the Motor

Start with a copy of the template program from before:

```python
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BOARD)

#Write your program below
```

This time, we will set up all three of the GPIO pins that we connected:

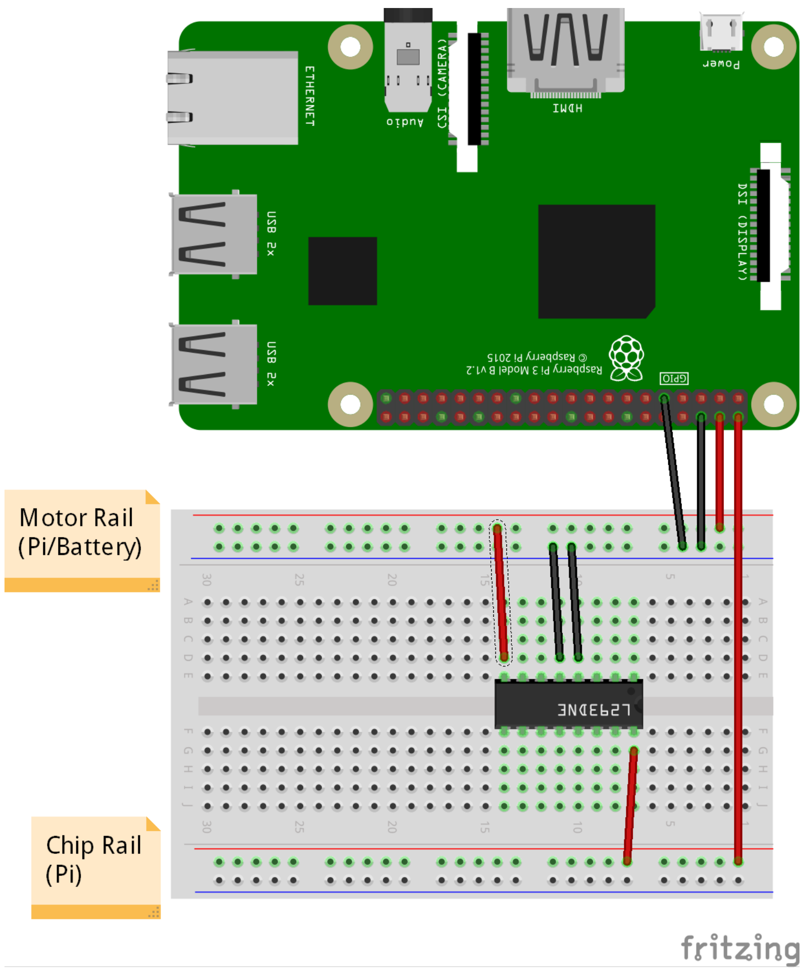


Figure 4: Powering the L293DNE.

```python

#GPIO pins for controller
Enable = 22
InputA = 16
InputB = 18

GPIO.setup(Enable, GPIO.OUT)
GPIO.setup(InputA, GPIO.OUT)
GPIO.setup(InputB, GPIO.OUT)
```

Remember that to turn on a motor, we need to set two opposing signals for A and B, then set enable to high. Therefore, to turn the motor on:

```python

print("on")
GPIO.output(InputA, GPIO.LOW)
GPIO.output(InputB, GPIO.HIGH)
GPIO.output(Enable, GPIO.HIGH)
```

As with our LED, we will leave the motor running for a few seconds before turning it off. We do not need to change the A or B input, we can just change 'Enable' to turn off the motor.

```

sleep(3)

print("off")
GPIO.output(Enable, GPIO.LOW)
GPIO.cleanup()
```

After adding the GPIO.cleanup() function, you can now try running your program.

Make sure your motor is connected, and hold onto the body of the motor whilst running the program, as the motor likes to wiggle free of it's wires.

Reverse Gear

One final thing to check. Our motor controller should allow us to run the motor in reverse.

Make the following changes to your program, and have a go at filling in the missing parameters for the GPIO.output function calls:

```

sleep(3)

print("off")
GPIO.output(Enable, GPIO.LOW)

sleep(1)

print('reverse')
GPIO.output(    ,       )
GPIO.output(    ,       )
GPIO.output(    ,       )

sleep(3)

print("off")
GPIO.output(Enable, GPIO.LOW)
GPIO.cleanup()
```

It will be difficult to tell which way the motor is spinning with nothing attached, but it tends to make a slightly different noise when running in the other direction.

To double check, you can try placing some Blu Tack on the motor shaft and shaping it to make the spin more obvious.

Congratulations on getting your motor working!

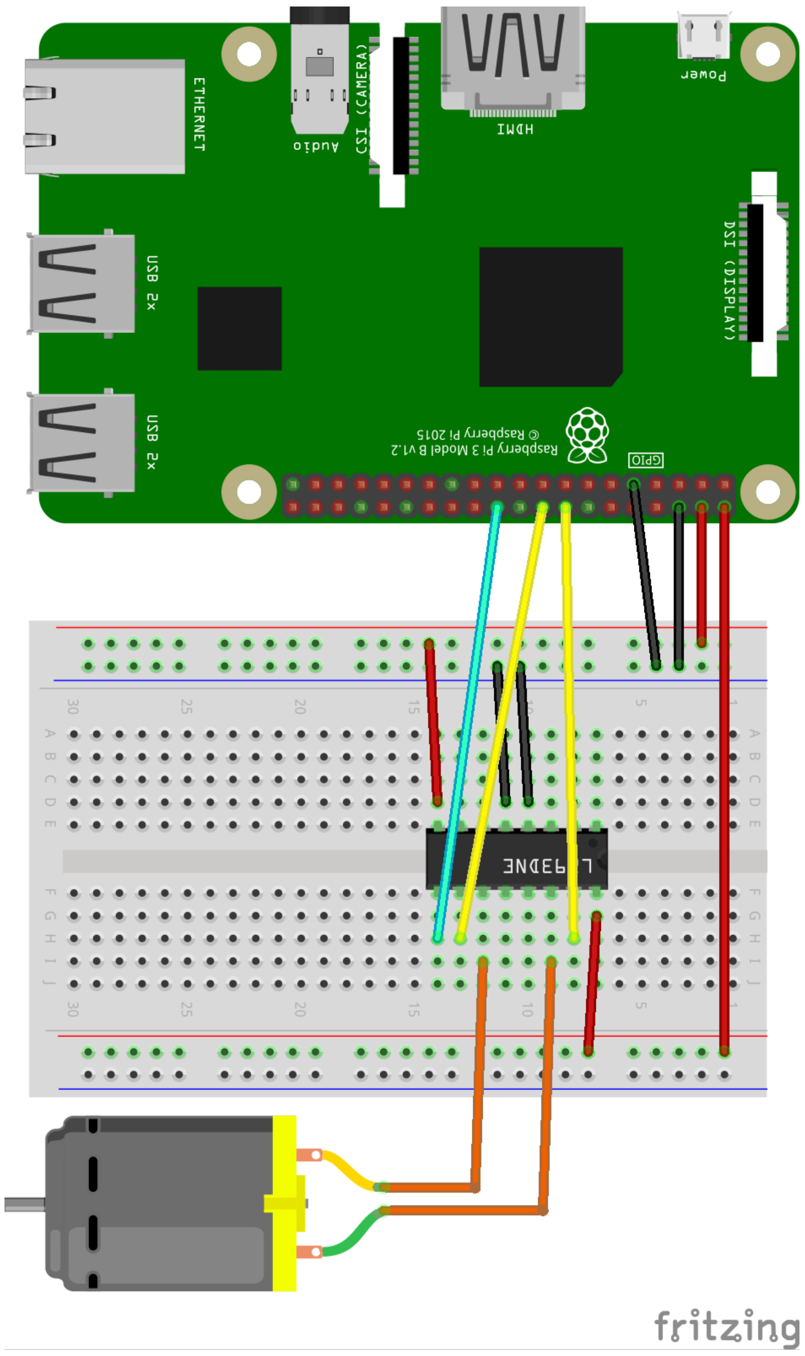
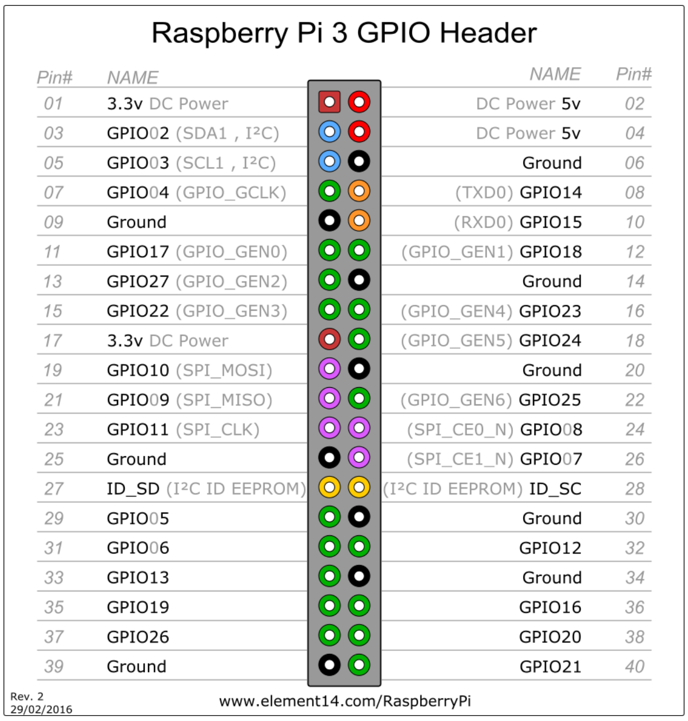


Figure 5: The complete motor circuit.

Appendices

A GPIO Pinout (GPIO.BOARD)



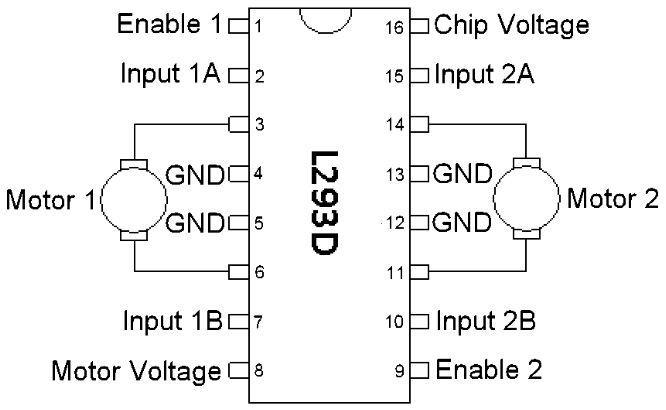
Raspberry Pi 3 GPIO Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I²C)	DC Power 5v	04
05	GPIO03 (SCL1 , I²C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)	(I²C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Rev. 2
29/02/2016

www.element14.com/RaspberryPi

B L293D/L293DNE Pinout



Source: https://www.rs-online.com/designspark/raspberry-pi-controlling-a-motor-via-gpio

C Completed Code Listings

0_LED.py

```python
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BOARD)

#Write your program below

#GPIO pins for LED
LED = 16
GPIO.setup(LED, GPIO.OUT)

print("on")
GPIO.output(LED, GPIO.HIGH)

sleep(5)

print("off")
GPIO.output(LED, GPIO.LOW)
GPIO.cleanup()
```

1_Motor.py

```python
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BOARD)

#Write your program below

#GPIO pins for controller
Enable = 22
InputA = 16
InputB = 18

GPIO.setup(Enable, GPIO.OUT)
GPIO.setup(InputA, GPIO.OUT)
GPIO.setup(InputB, GPIO.OUT)

print("on")
GPIO.output(InputA, GPIO.LOW)
GPIO.output(InputB, GPIO.HIGH)
GPIO.output(Enable, GPIO.HIGH)

sleep(3)

print("off")
GPIO.output(Enable, GPIO.LOW)

```

```python
sleep(1)

print('reverse')
GPIO.output(    ,        )
GPIO.output(    ,        )
GPIO.output(    ,        )

sleep(3)

print("off")
GPIO.output(Enable, GPIO.LOW)
GPIO.cleanup()
```