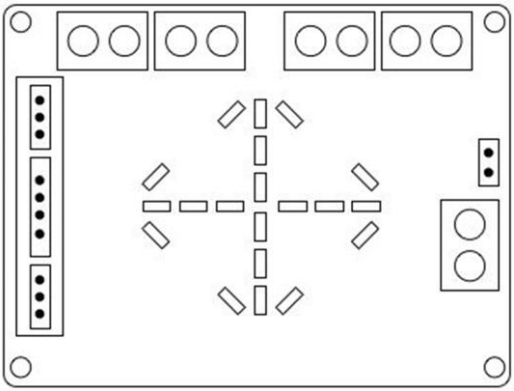
MAKER-SPHERE



Motor-shield v.1

WWW.MAKER-SPHERE.CO.UK

Motor-shield

Congratulations on buying the SB Components' Motor-shield.

Designed for the Raspberry Pi 2 and 3 this expansion board can control up to 4 motors, 2 IR sensors and a single ultrasonic sensor.

Combine it with one of the robot chassis from SB Components' shop to build your own autonomous Raspberry Pi powered robot.

This guide

This guide provides an introduction to the Motor-shield. It walks you through connecting it to your Raspberry Pi and using the provided Python library module to interact with sensors and motors.

To make the most of the Motor-Shield you will need some knowledge of the Python programming language. At the very least you will need to know how to start the Python editor ***Idle*** on your Raspberry Pi and execute Python code.

For more information see: https://wiki.python.org/moin/BeginnersGuide

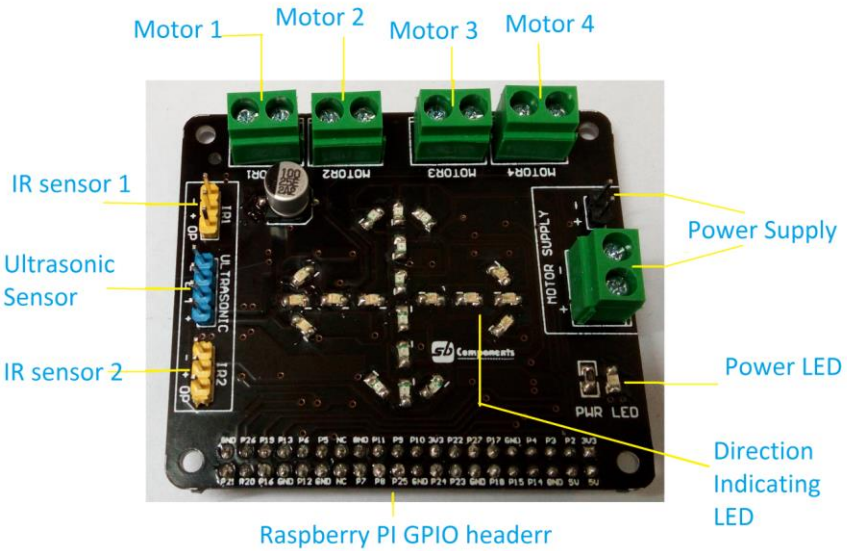
Alternatively refer to the ***Maker-Sphere Core*** project books from SB Components Ltd.

Specification

The Motor-shield is based on the L293D IC and is capable of driving 4 independent DC motors or 2 Stepper Motors. L293D is a dual full bridge driver that can output up to 1Amp per bridge with a voltage up to 24V.

Features:

- Dual H-Bridge IC L293D
- Motor Input Supply Range: 6V – 24V
- Single Motor Output Current: 600mA or 1A peak per channel
- On board Arrow Indicator for motor direction Indication
- 2 IR sensor connectors with 3.3V level output protection
- 1 Ultrasonic sensor connector with 3.3V level output protection



Mounting the Motor-shield

To attach the Motor-shield to the Raspberry Pi simply push the GPIO pin connectors on the underside of the Motor-shield onto the GPIO pins of the Raspberry Pi.

Be careful to ensure that all of the pins are correctly seated and the Motor-shield over-lays the Raspberry Pi.

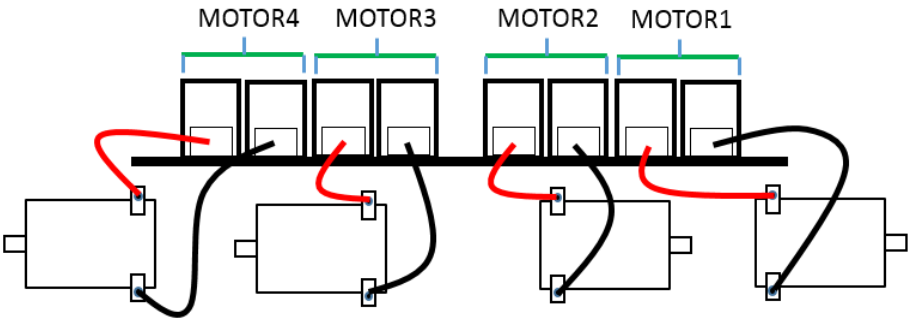
Powering your Motor-shield

The motor-shield can draw power from the Raspberry Pi but not enough to power motors when loaded with the chassis of your robot. For testing your robot program however the Pi can provide enough to power the LEDs.

To provide power for 4 motors you will need a supply of 7v to 12v with at least 1500mAH. For 2 motors power requirements are obviously less and you can use 4 x AA batteries, although these will drain quickly.

Connecting motors to the shield

Connect each motor to a single pair of labelled motor terminals on the Motor-shield (one motor terminal to one terminal on the Motor-shield).



Choosing the correct motor configuration

So which way round have you wired up your motors? What is forward and what is backward?

As each motor wired up to a chassis tends to be reversed in the direction it points it can be very confusing to get the wires connected in the correct order to make both go “forward”.

To help when putting your robot together and to minimise the amount of re-wiring, you can run a configuration test. For this to work you will need to download the Motor-Shield PiMotor library from https://sbcshop.github.io/motor-shield/

Then using *Idle* run the following code:

```
import PiMotor
import time

d = ["MOTOR1","MOTOR2","MOTOR3","MOTOR4"]
for motor in d:
    for config in range(1,3):
        print(motor + " config: " + str(config))
        m = PiMotor.Motor(motor,config)
        m.forward()
        time.sleep(2)
        m.stop()
```

This configuration test runs through the configurations available for each motor. Each motor can have its own definition of “forward” and “backward”. Note down the configuration number printed to the screen when your connected motors spin in the direction needed for forward movement. You will use this when writing your own robot code.

Controlling the motors

When you have wired up the motors and chosen which configuration each needs you can start writing code to control them.

The supplied PiMotor library module provides some helpful code to control the Motor-Shield. For details about the available code components go to https://sbcshop.github.io/motor-shield/ or at an **Idle** prompt import the PiMotor module and then type

```
help(PiMotor)
```

The following examples assume that you are using 2 motors; 1 connected to the MOTOR1 and 1 connected to the MOTOR2 terminals and that you have ***Idle*** open.

The following code will move your robot forward and backward:

```
from PiMotor import Motor
import time
m1 = Motor("MOTOR1",1)
m2 = Motor("MOTOR2",1)

##Then to make them rotate simply use:
m1.forward()
m2.forward()
time.sleep(5)
m1.stop()
m2.stop()
time.sleep(1)
m1.reverse()
m2.reverse()
time.sleep(5)
m1.stop()
m2.stop()
```

To save typing you can use the **LinkedMotors** class to combine multiple Motors into a single object. Here is the above code re-written using the LinkedMotors class:

```
from PiMotor import Motor
from PiMotor import LinkedMotors
import time
mset = LinkedMotors(Motor ("MOTOR1",1), Motor ("MOTOR2",1))
mset.forward()
time.sleep(5)
mset.stop()
time.sleep(1)
mset.reverse()
time.sleep(5)
mset.stop()
```

Using the Sensors

There are 3 sensor connectors on the Motor-Shield. Once you have correctly connected the relevant type of sensor you can reference and trigger the sensors in the following way:

```
from PiMotor import Sensor
import time
## create an ultrasonic sensor object
## set the trigger boundary to 10cm
us = Sensor("ULTRASONIC", 10)
us.trigger() ## Make the sensor take a reading
if us.Triggered: ## object was detected within 10cm of the sensor
    print("Boundary breached")

## create an IR sensor object
## set the trigger boundary to 0
ir1 = Sensor("IR1",0)
ir1.trigger()
if ir1.Triggered:
    print("IR detected object")
```

The Arrow LEDs

The Arrow LEDs are useful when programming your robot. Sometimes you want to test the correctness of your code without triggering a motor to turn (when for example the robot is on your desk). By calling the Motor.test() method with an argument of True an LED arrow will light up instead of the motor turning when the .forward method is called.

You can also control the arrow LEDs separately. The following code makes the LED arrows light up in turn:

```
from MotorShield import Arrow
import time
## create an ultrasonic sensor object
## set the trigger boundary to 10cm
a[0] = Arrow(1)
a[1] = Arrow(2)
a[2] = Arrow(3)
a[3] = Arrow(4)

for i in range(1,4):
    a[i].on()
    time.sleep(1)
    a[i].off(
```