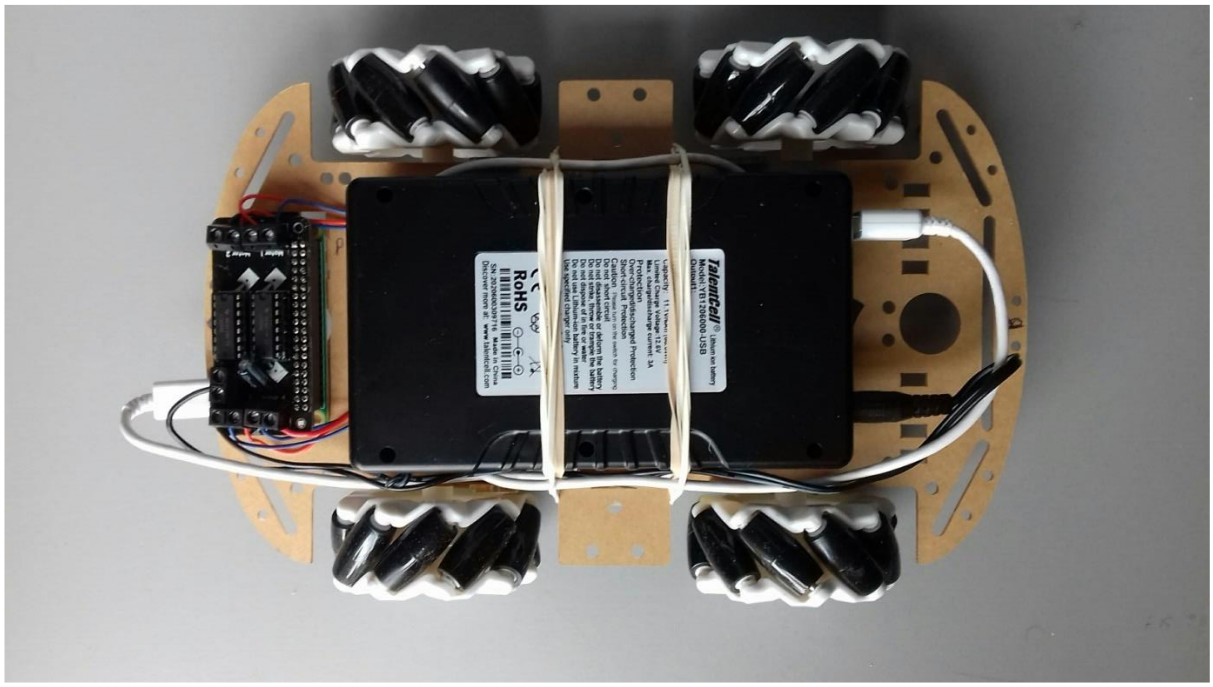
Raspberry Pi Zero Bluetooth Robot Car



An Elementary Guide to Building a Simple Robot Car

By

Peter Czaja

August 2021

Contents

1. Components required 3
2. Hardware/Software 3
   2a. Raspberry Pi Zero 3
   2b. MotoZero 3
   2c. Android 3
3. Circuit Diagram (Fritzing) 4
4. Project Assembly 4
5. Initial testing 6
6. Main Program 7
7. Setting up the Blue Dot on Android 9
8. Fitting the Mecanum Wheels 9
9. Sequence of Operation 10
10. Design notes 10

Raspberry Pi Zero Bluetooth Robot Car

1. Components required

- Raspberry Pi Zero with wifi (fully assembled)
- MotoZero motor control board
- 4 x 5 volt DC motors
- 4 x Mecanum wheels (2 x left, 2 x right)
- 4 x 6" lengths of single core wire – red (max 2 amps)
- 4 x 6" lengths of single core wire – blue (max 2 amps)
- 1 x 12" twin cable with DC5521 plug at one end (12v/3A max)
- 1 x 12" USB cable to micro USB (5V/2A max)
- 1 x Talentcell Lithium ion rechargeable battery Model YB1206000-USB
- 1 x Robot car chassis (or similar)
- 2 plastic mini screws and nuts

2. Hardware/Software

2a. Raspberry Pi Zero

It is assumed that the Raspberry Pi Zero (RPZ) has a fully up to date operating system installed on a 16GB (minimum) micro SD card.

The RPZ comes pre-installed with Bluetooth, install if not.

The RPZ comes pre-installed with Python, versions 2 and 3. Version 2 is slowly being phased out. This project will be using Python 3.7.3 (there are later versions available).

Install bluedot software – refer to :- https://bluedot.readthedocs.io/en/latest/gettingstarted.html

2b. MotoZero

The MotoZero moto controller board arrives in a kit form and will need to be assembled. There is some very good documentation at this link to help you assemble the components :-

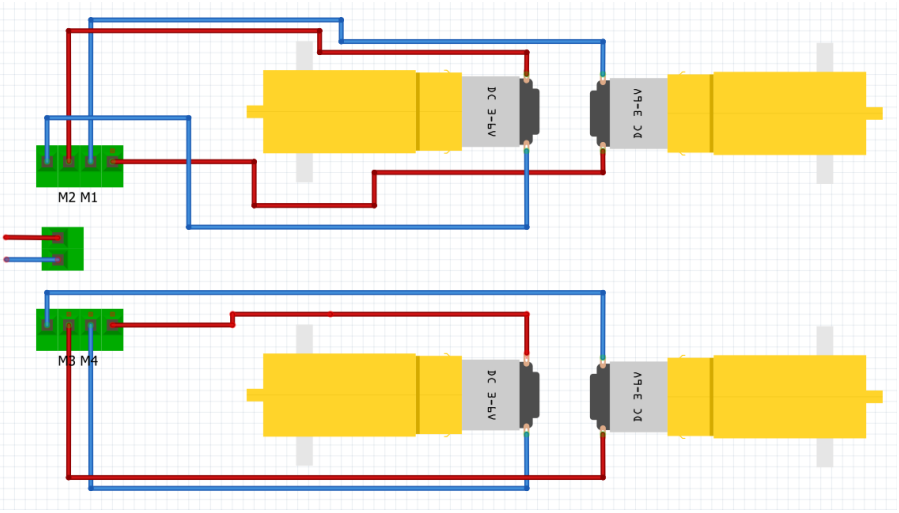
https://cdn.shopify.com/s/files/1/0176/3274/files/MotoZero_User_Guide_1.2.pdf

2c. Android

This project uses the Blue Dot Android app to control the RPZ motors. All information regarding its installation, use and application can be found at this link :-

https://bluedot.readthedocs.io/en/latest/gettingstarted.html

3. Circuit Diagram (Fritzing)



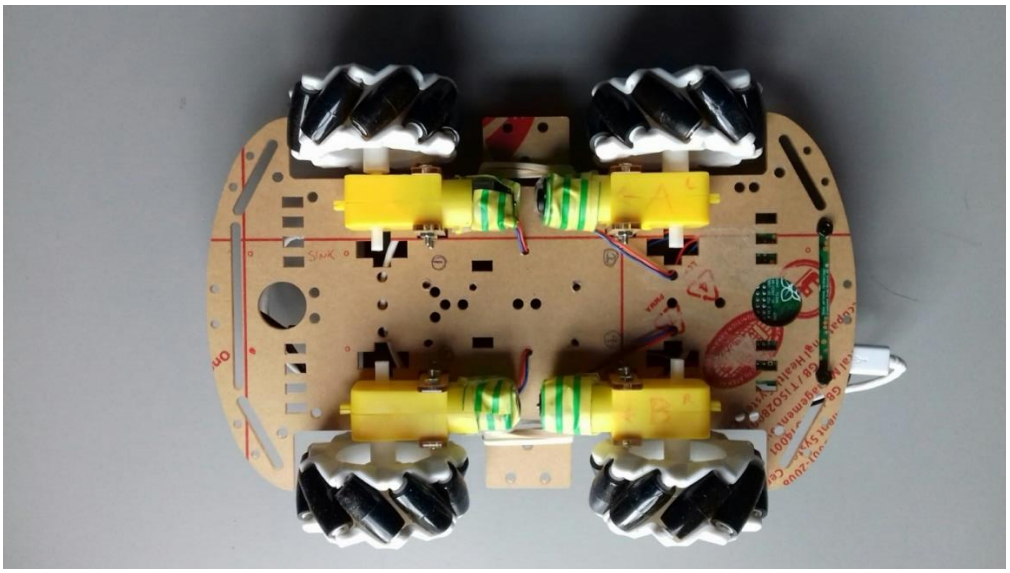
The diagram above is a simple representation of how the motors are wired to the MotoZero. The MotoZero sits on top of the RPZ, and the 40 MotoZero sockets fit exactly onto the 40 RPZ GPIO pins. The green terminals to the left hand side of the diagram represent the terminals that are present on the MotoZero motor controller.

Make sure that the motors are wired in this fashion in order that they turn in the correct direction when power is applied, e.g., they all rotate in the same direction for a forward command (gpiozero motor.forward()) – more on this later.

DO NOT PUT THE MECANUM WHEELS ON AT THIS STAGE.

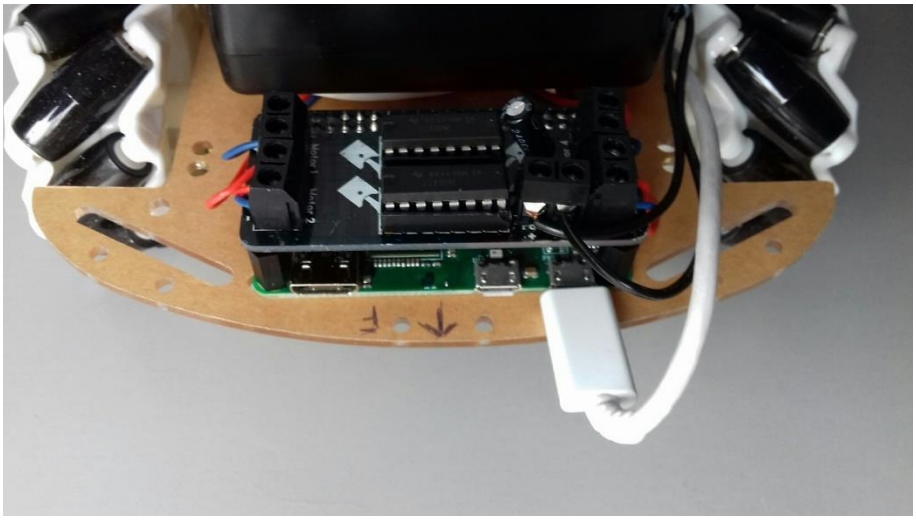
4. Project Assembly

This particular robot car project is of a very simple construction. This project has the DC motors on the underside (see below), and the RPZ, MotoZero and Talentcell battery on the top side (as seen on page 1).



Note that where the red and blue wires attach to the motors, there is green and yellow masking tape to prevent any shorting and to ensure that the wires are held firm against the motors. The terminals on this make of motor are quite delicate and care should be taken to avoid any unnecessary movement.

Fix your motors in this fashion to a chassis of your choice, this could be a blank piece of Perspex, or even an old wooden cigar box. Number your motors 1 to 4, as these will be connected to the respective MotoZero terminals 1,2, 3 & 4. In the photo above, the motors are numbered clockwise from top left 1, 2, 4 and 3.



In the photo above, you will just be able to make out the terminal numbers for motor 1 and motor 2 to the left of the MotoZero. These are in turn connected to motors 1 and 2 respectively on the right hand side of the robot car (facing forward).

This project uses the Talentcell rechargeable battery to power the RPZ and the MotoZero **separately**. The battery is fixed to the chassis by means of several rubber bands to allow quick removal, but a more permanent method could be used if required. The Talentcell battery has :-

- an on(-)/off(o) switch (centre)
- a DC5521 socket which provides the MotoZero with 9-12volts (left)
- a USB socket which provides the RPZ with 5volts (right)

A 12” wire (black in this project) is connected to the MotoZero power input and then plugged into the Talentcell battery, ensuring that polarity is correct. In a similar manner a 12” USB cable (white in this project) connects the Talentcell battery 5v USB socket to the RPZ main power mini USB socket.

The RPZ and MotoZero are fixed to the chassis by means of 2 plastic mini screws and nuts.

5. Initial testing

An initial test is required to ensure that all the motors are connected correctly, and more importantly that they all turn in the correct direction as dictated by the program.

Before using the battery, ensure that it is fully charged. It is possible to use other batteries as a power source, but this particular battery has 2 convenient power outlets to power the RPZ and motors separately.

Making sure that the MotoZero power lead (DC5521 9-12v) is plugged into the Talentcell battery, and that the Mecanum wheels are **not** fitted to each motor, place the robot car on to a flat surface with the motors downwards i.e. battery and RPZ upwards.

Connect the RPZ micro power socket to the Talentcell battery (USB 5v).

Turn the battery switch to the on position "1".

Wait for a few minutes while the RPZ starts up and goes through its initialisation process.

From a laptop, desktop or Android connect to the RPZ using VNC viewer. VNC viewer can be downloaded and installed from here - https://www.realvnc.com/en/connect/download/viewer/. There are several android based apps that can help find the IP address of the RPZ if not known prior to using VNC viewer.

Once connection is established, logon in the usual manner (this will depend on the original setup of the RPZ), store the following python3 program onto an area of the RPZ, and run the program.

```
from gpiozero import Motor, OutputDevice
from time import sleep

m1 = Motor(24,27)                          # Assign motors
m2 = Motor(6, 22)
m3 = Motor(23, 16)
m4 = Motor(13, 18)

m1_enable = OutputDevice(5, initial_value=1)    # Enable all motors
m2_enable = OutputDevice(17, initial_value=1)
m3_enable = OutputDevice(12, initial_value=1)
m4_enable = OutputDevice(25, initial_value=1)

speed = 0.2                                # Set motor speed (0.1 = slow, 1.0 = fast – max)

print("Forward")                           # Run test
m1.forward(speed)
m2.forward(speed)
m3.forward(speed)
m4.forward(speed)

sleep(3)
```

This program should run the motors in a forward direction for 3 seconds. Whilst the program runs, check each motor axle in turn to ensure that they are all turning in the same direction. Rerun the program as required.

If any motor is not turning in the correct direction, wait for the program to end, shutdown the RPZ, power off, and swap the wires of the terminals on the MotoZero that are not turning in the required direction. Turn on power and repeat above process until satisfied that wheels are in sync.

All motors must be turning in unison and in the correct direction before proceeding.

6. Main Program

The following is a copy of the original main program. It can be changed to suit the individual's requirements :-

```
from gpiozero import Motor, OutputDevice
from bluedot import BlueDot
from time import sleep
import os

bd = BlueDot(cols=5, rows=3)    # Setup android control buttons
bd.color = "red"                # Motion buttons in red, except...
bd.border = True
bd[2,1].color = "blue"          # Quit button in blue
bd[0,0].color = "green"         # Shift buttons green
bd[0,1].color = "yellow"        # Slide buttons yellow
bd[4,0].color = "green"
bd[4,1].color = "yellow"
bd[0,2].color = "black"         # Pause buttons black
bd[4,2].color = "black"

bd[1,0].visible = False         # Blank out other squares in block
bd[3,0].visible = False
bd[1,2].visible = False
bd[3,2].visible = False

m1 = Motor(24,27)               # Assign motors
m2 = Motor(6, 22)
m3 = Motor(23, 16)
m4 = Motor(13, 18)

m1_enable = OutputDevice(5, initial_value=1)   # Enable all motors
m2_enable = OutputDevice(17, initial_value=1)
m3_enable = OutputDevice(12, initial_value=1)
m4_enable = OutputDevice(25, initial_value=1)

speed = 0.2                     # Set motor speed NB: use at slow speed to start

def forward():
    print("Forward")
    m1.forward(speed)
    m2.forward(speed)
    m3.forward(speed)
    m4.forward(speed)

def backward():
    print("Backward")
    m1.backward(speed)
    m2.backward(speed)
    m3.backward(speed)
    m4.backward(speed)

def left():
    print("Left")
    m1.forward(speed)
    m2.forward(speed)

def right():
    print("Right")
    m3.forward(speed)
    m4.forward(speed)
```

```
def shiftl():                   # Crab left
    print("Shift Left")
    m1.backward(speed)
    m2.forward(speed)
    m3.forward(speed)
    m4.backward(speed)

def shiftr():                   # Crab right
    print("Shift Right")
    m1.forward(speed)
    m2.backward(speed)
    m3.backward(speed)
    m4.forward(speed)

def slidel():                   # Forward diagonally left
    print("Slide Left")
    m2.forward(speed)
    m3.forward(speed)

def slider():                   # Forward diagonally right
    print("Slide Right")
    m1.forward(speed)
    m4.forward(speed)

def pause():
    print("PAUSE")
    m1.stop()
    m2.stop()
    m3.stop()
    m4.stop()

def quit():
    print("STOP")
    m1.stop()
    m2.stop()
    m3.stop()
    m4.stop()
    sleep(2)
    os._exit(0)                 # Exit program

while True:                     # Continuous loop

    bd[2,0].when_pressed = forward    # Check which button pressed and call subroutine
    bd[2,2].when_pressed = backward
    bd[1,1].when_pressed = left
    bd[3,1].when_pressed = right
    bd[0,0].when_pressed = slidel
    bd[4,0].when_pressed = slider
    bd[0,1].when_pressed = shiftl
    bd[4,1].when_pressed = shiftr
    bd[0,2].when_pressed = pause
    bd[4,2].when_pressed = pause
    bd[2,1].when_pressed = quit
```

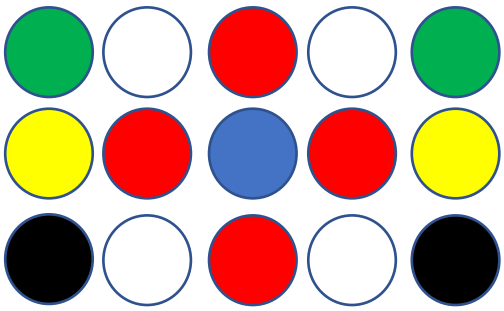
Once saved, this program can be run, and it will wait until a “button” is pressed on the android. Finishing reading this document BEFORE attempting to run the robot.

7. Setting up the Blue Dot on Android

Download and install Blue Dot onto the Android device.

The android will **not connect** with the RPZ **until** the main program is run. This can either be done via a Mobile VNC or SSH app, or VNC/SSH from a laptop or desktop PC.

When running, the Blue Dot controls will look like this on the android screen :-



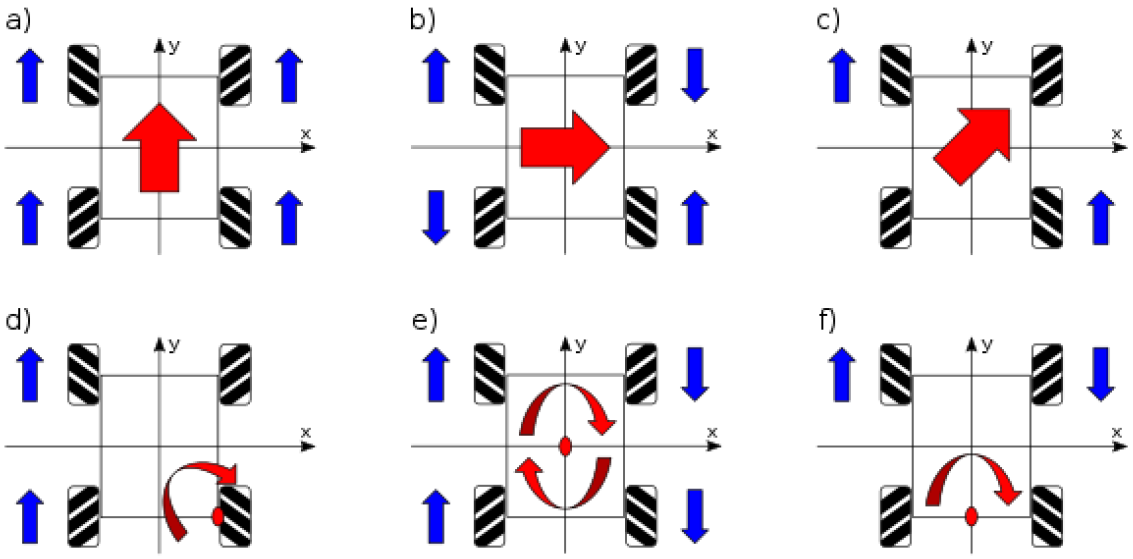
RED	=	Forward, Backward, Left or Right
YELLOW	=	Shift Left or Right (crab L/R)
GREEN	=	Slide Left or Right (diagonal forward L/R)
BLACK	=	Pause
BLUE	=	Quit
WHITE	=	Not activated

These controls and colours can be changed within the python program.

(NB: At time of writing, it was not possible to place text into the various circles)

8. Fitting the Mecanum Wheels

Having made sure that all the motors run in the correct direction, as per section 5, place the Mecanum wheels onto the motors as per the following diagram (source : Wikipedia) :-



Movements to any directions:
blue: wheel drive direction; *red:* vehicle moving direction
a) Moving straight ahead, **b)** Moving sideways, **c)** Moving diagonally, **d)** Moving around a bend, **e)** Rotation, **f)** Rotation around the central point of one axle

Using diagram a) as a guide, and ensuring that the forward motion of the motors is in the direction of ”y”, (as if looking down on the robot car), place the wheels onto each motor so that the angle of the rollers is the same as indicated in the diagram.

The remaining diagrams b) to f) show the effect of robot car movement depending on the direction of the individual motor indicated by the blue arrow (NB: not all are used in this project).

9. Sequence of Operation

- Plug in power to RPZ and MotoZero and turn on Robot Car
- Place Robot Car on level surface away from obstacles.
- Turn on Android/Laptop/Desktop
- Turn on Bluetooth on Android/Laptop/Desktop
- VNC into Robot Car RPZ (may require several attempts if RPZ not ready)
- Turn on RPZ Bluetooth via VNC
- Locate and run main python program (program will wait for event)
- Activate Blue Dot app on Android
- Pair Android with RPZ (see section 2c, and search for “Pair a Raspberry Pi and Android phone”)
- Select connection of Android to RPZ.
- (Coloured controls will now appear – as per Section 7)
- Practise to perfect your technique !

During operation, the L293D chips on the MotoZero can get quite hot. **DO NOT TOUCH THEM** and **DO NOT run the program at full speed for any length of time** and, avoid obstacles and rough terrain which would otherwise cause stress to the motors, and hence cause the L293D chips to overheat. Speed = 0.2 is ample !

The Mecanum wheels used in this project are plastic and can tend to slide on a wooden laminate floor, and may not turn at all on carpet, depending on the thickness of the pile. Rubber wheeled Mecanum wheels are available, and these are more effective in gripping laminate flooring (and consequently more expensive !)

Lastly, Remember to Power down the RPZ and switch off the battery when finished.

10. Design notes

This project is of course very simple, and literally held together by elastic bands ! However, with some ingenuity (and patience), a more elaborate chassis can be constructed from using a suitable sized (firm) plastic box with the appropriate holes for affixing the motors and for passing the wires through. Alternatively, basic chassis can be purchased online, and an example is shown here .

When running the robot forward in a straight line, the robot may tend to veer slightly to one side. This is because not all the motors rotate at the same frequency, despite being powered by the same voltage. To improve this, the speed can be altered for the offending motor(s), by setting up speeds for each individual motor – it can be a little bit trial and error until it becomes consistent.

The Motozero maybe slightly underpowered for this project, and this can be changed so that two, dual H-bridges are used instead. However, this will necessitate a change in how the circuit is constructed, but the software should stay the same.