M.A.R.S. Rover on Raspberry Pi Zero 2

Contents

NOTE: Use Raspberry Pi OS Setup.

Use **sudo raspi-config** to enable SPI, I2C and legacy camera support.

Installing the M.A.R.S. Rover Python Software

First you will need to prepare your Pi for the Fireleds (fully compatible with neopixels).
Install the rpi_wpython3 dr s281x package:

```
sudo pip3 install rpi_ws281x
```

~~Alternatively you can run Pimoroni's curl script for their Unicorn HATs.~~
~~With your Pi connected to the internet, run (you don't need to install the resources and examples):~~

~~curl -sS https://get.pimoroni.com/unicornhat | bash~~

Download the Python library module and example software for M.A.R.S. Rover with

```
wget https://4tronix.co.uk/rover.sh -O rover.sh
sudo chmod 777 rover.sh
./rover.sh
```

This installs the following, which should be run from a terminal.

Creates a folder home/pi/marsrover for all the example files and library module rover.py (the main library module)

- **calibrateServos.py** should be the first program you run. This ensures that the wheels are all pointing in the correct direction. You must use the terminal to run this and you will require sudo because it uses the LEDs as indicators.
    - **sudo python3 calibrateServos.py**
    - Select each servo in turn using '1' to '5' keys and adjust the servo using the left and right arrow keys until it is straight. Press 's' to save the calibration and exit the program.
- **motorTest.py** Dsudoemonstrate driving the motors. Must be run in a terminal.
- **servoTest.py** Demonstrate controlling the servos. Must be run in a terminal, not from an IDE.
- **ledTest.py** Flashes all LEDs through Red, Green, Blue and White. This must be run using sudo. ie: sudo python ledTest.py
- **sonarTest.py** shows the distance in cm for an obstacle using the ultrasonic distance sensor mounted on the mast head.
- **keypad.py** shows the numeric value of each key pressed on the optional keypad
- **driveRover.py** is a basic driving program using the arrow keys to steer and move

Using the rover.py Library Module

To use this module, you must first import it into your program using

import rover

Before using any other functions, you should call the init () function to initialize all the variables used by the library. The init () function has two optional parameters.

1. Default brightness of the LEDs. If omitted, this defaults to 40. The following call will initialize the library and set the default brightness to 100:

rover.init (100)

If you set the brightness to zero, the LEDs are not initialized. As this would require use of the sudo command, it can be beneficial to call **init (0)** so that your program can be run without using sudo.

When your program has finished, it is good practice to close everything tidily by calling:

rover.cleanup()

Motor Functions

- **stop():** Stops all motors – coast slowly to a halt
- **brake():** Stops all motors – brakes quickly
- **forward(speed):** Sets all motors to move forward at speed. 0 <= speed <= 100
- **reverse(speed):** Sets all motors to reverse at speed. 0 <= speed <= 100
- **spinLeft(speed):** Sets left and right motors to turn opposite directions at speed. 0 <= speed <= 100
- **spinRight(speed):** Sets left and right motors to turn opposite directions at speed. 0 <= speed <= 100

Fireled Functions

- **setColor(color):** Sets all LEDs to color – requires show()
- **setPixel(ID, color):** Sets pixel ID to color – requires show()
- **show():** Updates the LEDs with state of LED array
- **clear():** Clears all LEDs to off – requires show()
- **rainbow():** Sets the LEDs to rainbow colors – requires show()
- **fromRGB(red, green, blue):** Creates a color value from R, G and B values
- **toRGB(color):** Converts a color value to separate R, G and B
- **wheel(pos):** Generates rainbow colors across 0-255 positions

UltraSonic Function

getDistance(): Returns the distance in cm to the nearest reflecting object. 0 == no object.

EEROM Functions

The onboard EEROM can store 1024 bytes of information. The first 16 bytes are used to store the servo calibration offsets, so "only" 1008 bytes are available to the user. The servo offset data is hidden, so the user data is addressed from 0 to 1007. Values return are

signed bytes with values from -128 to +127 inclusive. This could be used for example to store a sequence of commands from the Keypad, creating a set of remote instructions, etc.

readEEROM(Address): Returns the data byte stored at user address Address

writeEEROM(Address, Data): Saves the data byte Data at the user address Address

Web Streaming

Install VLC Media Player on Client Computer.

Run the following on the Pi.

```
sudo modprobe bcm2835-v4l2
cvlc v4l2:///dev/video0 --v4l2-width 640 --v4l2-height 480 --v4l2-chroma
h264 --sout '#standard{access=http,mux=ts,dst=0.0.0.0:8081}'
```

On the Client in VLC Player, Media → Open network stream → http://192.168.9.126:8081

Using `motion`

Motion includes a systemd service which makes it easy to autostart at boot, stop or restart.

To use it in IP webcam mode the following settings should be set in `/etc/motion/motion.conf`:

```
daemon on
stream_localhost off # when 'on' others hosts will NOT see the server
output_pictures off
ffmpeg_output_movies off
stream_maxrate 24
framerate 24
width 640
height 480
```

Next, in `/etc/default/motion` set:

```
start_motion_daemon=yes
```

To make the service autostart on boot:

```
systemctl enable motion
```

Use `systemctl start/stop/restart motion` for the corresponding action.

Default streaming port is `8081`. The port is set under `stream_port` in `/etc/motion/motion.conf`.

https://elinux.org/RPi-Cam-Web-Interface

https://projects-raspberry.com/live-camera-streaming-raspberry-pi-motion-setup/

https://motion-project.github.io/motion_build.html

https://raspberry-valley.azurewebsites.net/Streaming-Video-with-Motion/

https://blog.miguelgrinberg.com/post/stream-video-from-the-raspberry-pi-camera-to-web-browsers-even-on-ios-and-android

https://www.makeuseof.com/tag/live-stream-youtube-raspberry-pi/

https://github.com/jacksonliam/mjpg-streamer

https://blog.alexellis.io/live-stream-with-docker/