Q-Bot: Configuring a Raspberry Pi Zero W

To run headless, optionally with Cloud9

This guide assumes you have a Raspberry Pi Zero W and want to set it up to run it headless (without a monitor and keyboard) on your wireless network. In my particular case, I'm using the Pi as the "brain" for the Q-bot robot; I want to connect to it using SSH wirelessly to control it, and I want to connect to a programming environment running on the Pi with a web browser.

Getting the Pi Zero operating on your network

1. Download the latest image of Raspbian Stretch Lite: https://www.raspberrypi.org/downloads/raspbian/
2. Using Etcher (or a similar image writing application), burn the image to your SD card
3. Mount the resulting burned SD image; you should see a boot volume with a number of files inside.
4. Create a new file called wpa_supplicant.conf in the root / directory of the **boot partition** (the smaller of the two partitions), with the following contents:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=US

network={
    ssid="YOURSSID"
    psk="YOURPASSWORD"
    scan_ssid=1
    key_mgmt=WPA-PSK
}
```

5. Create an empty file called "ssh" in the root / directory of the boot partition of the SD card. The empty file will enable ssh, and will be deleted upon the next startup. Here are the notes from: https://www.raspberrypi.org/documentation/remote-access/ssh/

```
Enable SSH on a headless Raspberry Pi (add file to SD card on another
machine)

For headless setup, SSH can be enabled by placing a file named ssh,
without any extension, onto the boot partition of the SD card from
another computer. When the Pi boots, it looks for the ssh file. If it is
found, SSH is enabled and the file is deleted. The content of the file
does not matter; it could contain text, or nothing at all.
```

```
If you have loaded Raspbian onto a blank SD card, you will have two
partitions. The first one, which is the smaller one, is the boot
partition. Place the file into this one.
```

6. Also in the root / directory of the boot partition, edit the config.txt file, and add the following lines at the end:

```
# Enable UART
enable_uart=1
```

7. Unmount the image and put the SD in the raspberry pi. Power it up and wait a couple of minutes.
8. If all goes well, the device will register it on your local network using mDNS and appear as “raspberrypi.local”. You can ping and ssh into the device now.
9. Alternatively: log into the Raspberry Pi using SSH:

user = pi, password = raspberry, ip address = [address on local subnet], example:
> **pi@192.168.0.107**
pi@192.168.0.107's password: **raspberry**

If the device doesn’t show up, then you need to go into troubleshooting mode. For example, if you’re working on a small local network where you don’t have a network admin to yell at you for running scans, you can power off the device and run an nmap scan (“sudo nmap -sn 192.168.0.0/24”) to identify devices on your local subnet. Boot the Pi Zero and scan again, and chances are that any new device is your Pi.

If THAT doesn’t work, try the following:

1. Try rebooting the Pi and see if that fixes it. Be patient on the first boot, it might take a few minutes.
2. Your wireless network might use a different authentication type. In the wpa_supplicant.conf file above, I was connecting to a network that used WPA Personal authentication, so I used the line “key_mgmt=WPA-PSK”. If you’re using WPA Enterprise, it needs to change to “key_mgmt=WPA-EAP”. If you’re using something else, google it.

Updating the OS and installing Cloud9

Cloud9 is a web-based development environment. It’s much more convenient when writing code for a project like the Q-bot compared to using something like a native editor like nano or vi in an SSH session. Setting it up takes some time, and consumes resources on a computer that isn’t exactly a powerhouse to begin with, so if you don’t want it or need it, feel free to skip ahead.

1. Reboot the Pi

2. Log into the Pi and run:

```
sudo apt-get update && sudo apt-get upgrade
```

   Be patient, it might take a while.

3. From your home directory, grab the NodeJS ARM package and extract it to /usr/local:

```
cd ~

wget http://nodejs.org/dist/v0.10.28/node-v0.10.28-linux-arm-pi.tar.gz

tar -xzf ~/node-v0.10.28-linux-arm-pi.tar.gz

cd node-v0.10.28-linux-arm-pi/

sudo cp -rf * /usr/local

export NODE_PATH="/usr/local/lib/node_modules"
```

4. Grab a copy of Cloud9 from Git and build it:

```
git clone git://github.com/c9/core.git c9sdk
cd c9sdk
scripts/install-sdk.sh
```

5. Once that's done, you can launch Cloud9 any time you want to use it by running the following command from the prompt:

```
~/c9sdk/server.js -l 0.0.0.0 -a : -w $MY_PROJECT_PATH  &
```

Once all of those installations are complete, you should be able to use Cloud9 via web browser, and use the Adafruit library and example code to control servos and ESCs from the Pi.

Other Required Libraries

Update Python:

```
sudo apt-get update
sudo apt-get install -y python3 python3-pip python-dev
sudo pip3 install rpi.gpio
```

Install support for i2c (a 2-wire serial protocol):

```
sudo apt-get install python-smbus
sudo apt-get install i2c-tools
```

Python library for the vl53l1x laser distance sensor:

```
sudo pip3 install smbus2

sudo pip install vl53l1x
```

Install numpy:

```
sudo pip3 install numpy
```

Enjoy the project files, and please reach out to us with questions, comments, and stories about your work with this project! We can't wait to hear from you.

For hardware: jeff@loislab.org
For software: michael@loislab.org

Reference:

This guide leans heavily on the information provided in the articles below. Thank you to the authors who shared their knowledge and made this guide (and the project overall) possible.

Adafruit's guide Pi Zero W headless config:
https://learn.adafruit.com/raspberry-pi-zero-creation/overview

Chintan Pathak's guide to installing Cloud9 on a Raspberry Pi:

https://medium.com/@chintanp/using-cloud9-3-0-ide-on-raspberry-pi-954cf2d6ab8e