e-ale-lca2019

Embedded Apprentice Linux Engineer LCA2019

Version 1.0

e-ale

The E-ALE (Embedded Apprentice Linux Engineer) is a series of seminars held at existing conferences covering topics which are fundamental to a Linux professional in the field of Embedded Linux.

This seminar will spend equal time on lecture and hands on labs at the end of each seminar which allow you to practice the material you've learned.

> This material makes the assumption that you have minimal experience with using Linux in general, and a basic understanding of general industry terms. The assumption is also made that you have access to your own computers upon which to practice this material.

More information can be found at **https://e-ale.org**/

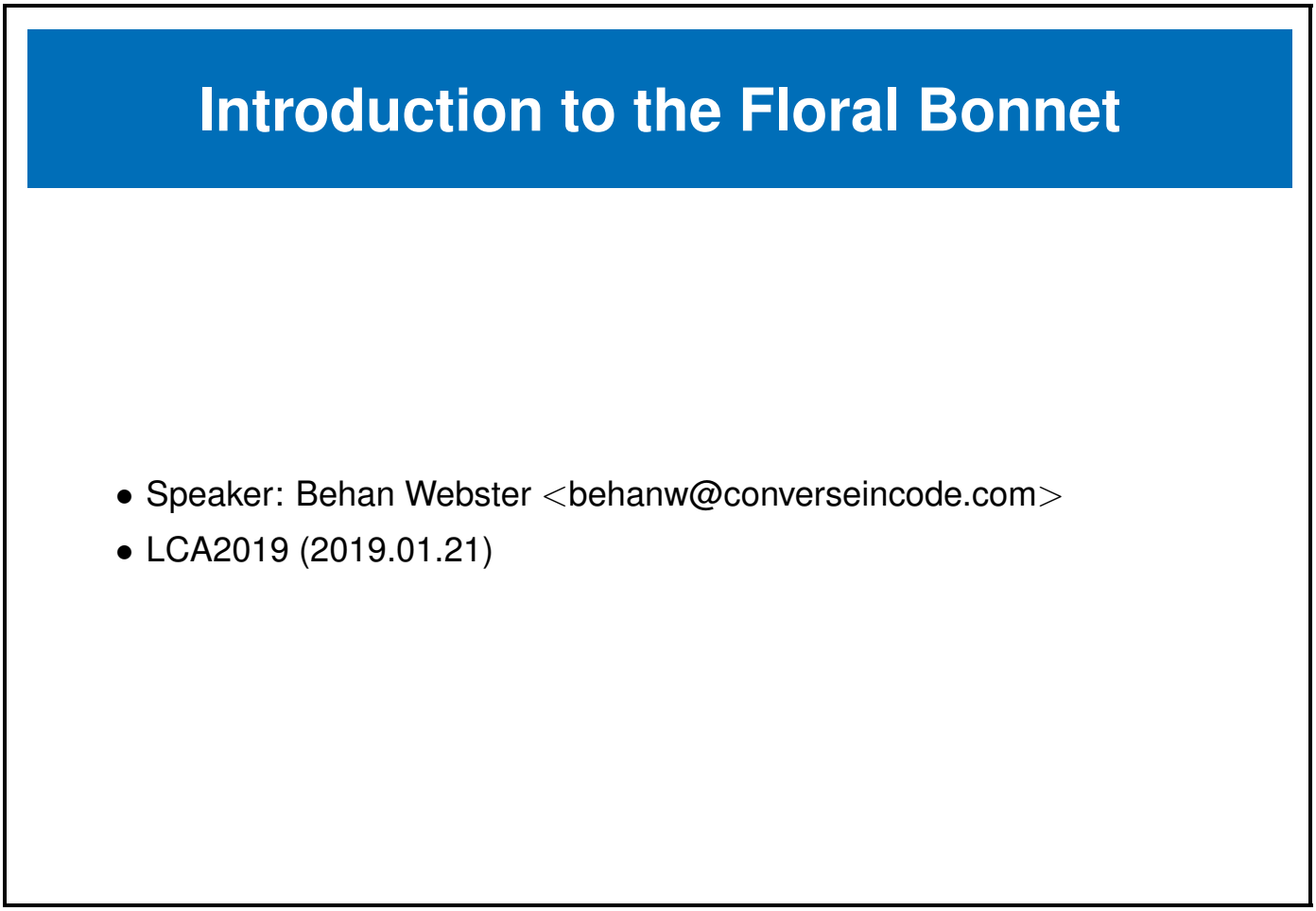
Contents

Chapter 1

The Floral Bonnet

e-ale

1.1 The Floral Bonnet



You can learn more about the **raspberry-pi zero W** at the following url.

https://github.com/unreproducible/bonnet_floral

Brought to you by

- Linux Foundation Training has provided speaker funding
- ARM is subsidizing the manufacturing of the floral bonnet for LCA2019

There are many raspberry-pi zero bonnets available from such sources as Adafruit.

https://www.adafruit.com/category/929

Raspberry-pi Zero

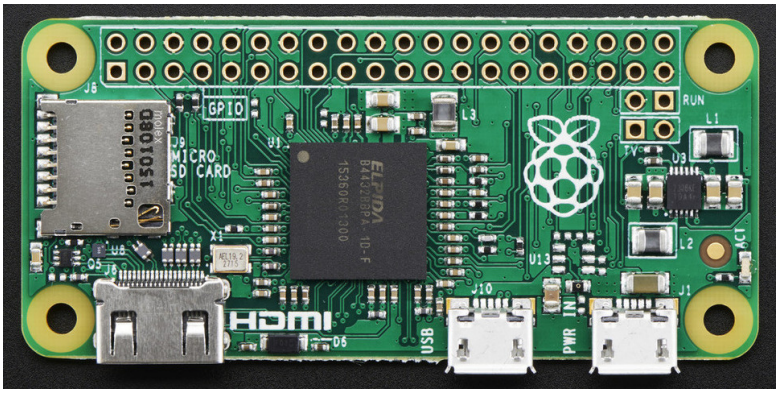


Figure 1.2: **Raspberry-pi Zero**

- Effectively a 65 mm by 30 mm version of the raspberry-pi 2
- 32-bit 1GHz single-core ARM CPU, 512MB RAM, etc

You can learn more about the **raspberry-pi zero** at the following url.

https://www.raspberrypi.org/products/raspberry-pi-zero/

Raspberry-pi Zero Wireless

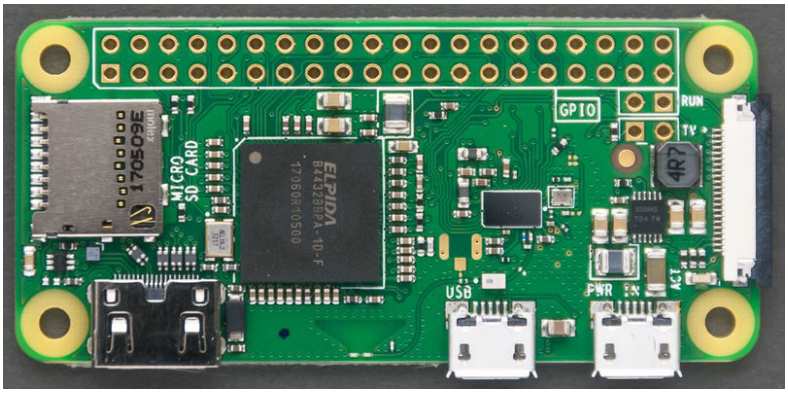


Figure 1.3: **Raspberry-pi Zero Wireless**

- The wireless version of the Zero with wifi and bluetooth

You can learn more about the **raspberry-pi zero W** at the following url.

https://www.raspberrypi.org/products/raspberry-pi-zero-w/

Raspberry-pi Zero Wireless with Headers

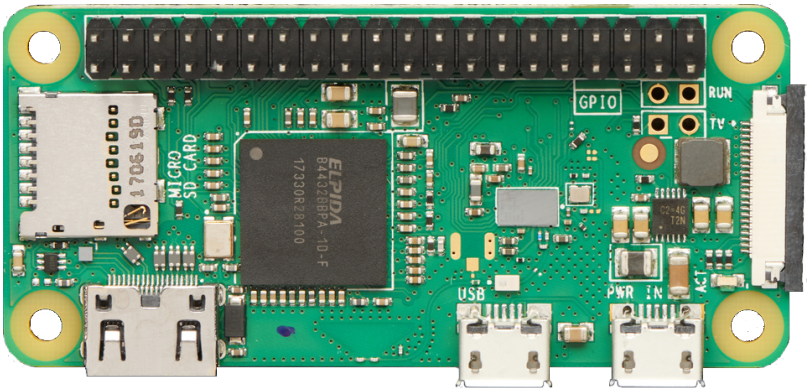


Figure 1.4: **Raspberry-pi Zero WH**

- We need the version with the header already installed for a bonnet

You can learn more about the **raspberry-pi zero W** at the following url.

https://www.raspberrypi.org/products/raspberry-pi-zero-w/

Raspberry-pi bonnet

- Expansion boards for the raspberry-pi are called **hats**
- Since the rpi-zero is smaller, expansion boards for the zero are called **bonnets**

There are many raspberry-pi zero bonnets available from such sources as Adafruit.

https://www.adafruit.com/category/929

The Floral Bonnet

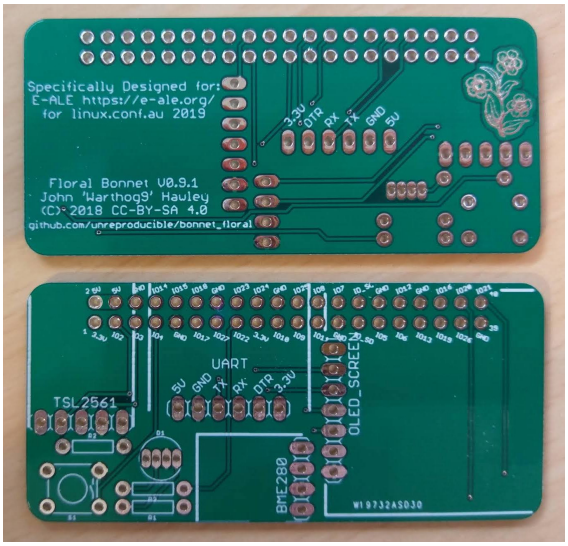


Figure 1.5: **The E-ALE Floral Bonnet for LCA**

- E-ALE has produced the floral bonnet specially for LCA2019

You can learn more about the **raspberry-pi zero W** at the following url.

https://github.com/unreproducible/bonnet_floral

Raspberry-pi Zero Pinouts

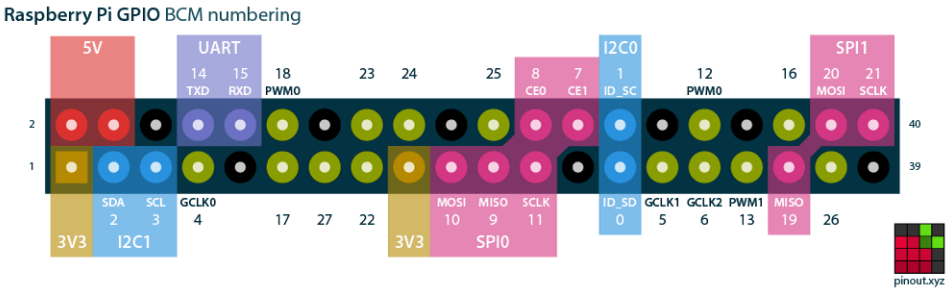


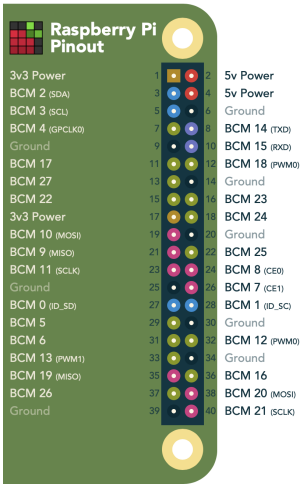
Figure 1.6: **Raspberry-pi Zero Pinouts**

- The pinouts for all recent 20-pin raspberry-pi boards are the same.

You can learn more about the **raspberry-pi zero pinouts** at the following url.

`https://pinout.xyz/`

Raspberry-pi Zero Pins

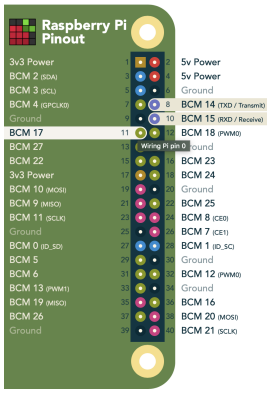


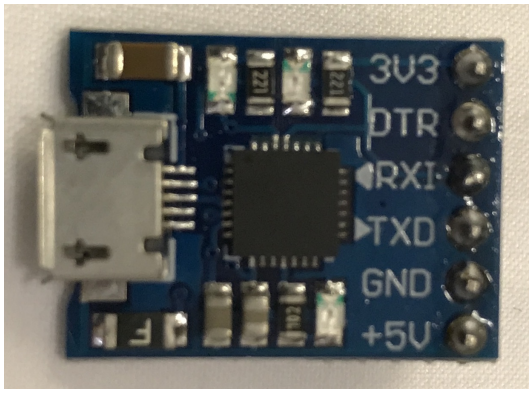
- Pins are shared amongst multiple peripherals
- A pin multiplexer is used to choose the configuration of the pins in use.

You can learn more about the **raspberry-pi serial pins** at pinout.xyz.

https://pinout.xyz/

Raspberry-pi Zero Serial



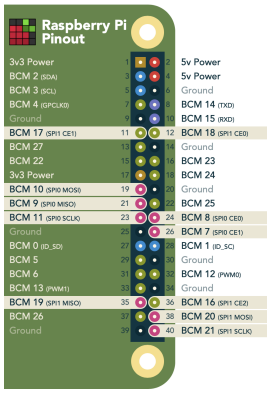


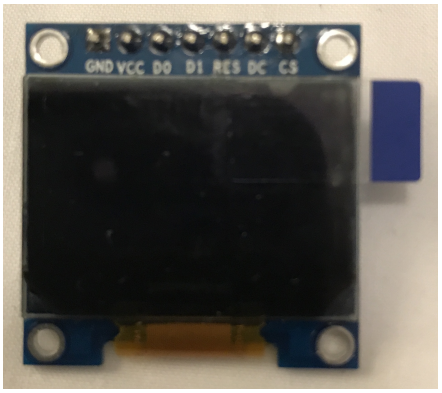
- We have access to the console serial port through the UART pins using a UART-to-USB converter like the CP2102
- We can use this serial port to configure the bootloader and debug kernel issues

You can learn more about the **raspberry-pi serial pins** at pinout.xyz.

https://pinout.xyz/pinout/uart

Raspberry-pi Zero SPI



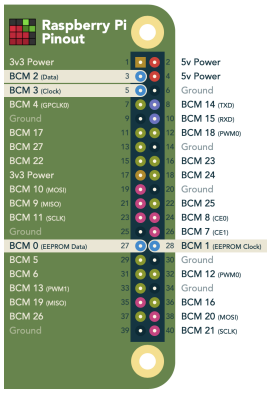


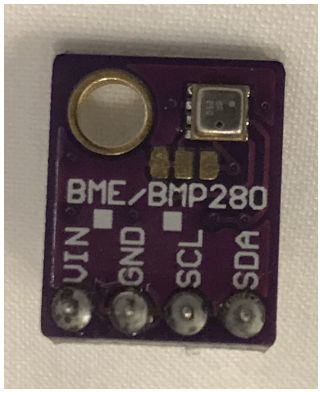
- We can access 2 SPI ports with these pins
- We will use one of the SPI channels to control the **SSD1306 OLED screen** on the floral bonnet

You can learn more about the **raspberry-pi SPI pins** at pinout.xyz.

https://pinout.xyz/pinout/spi

Raspberry-pi Zero I2C



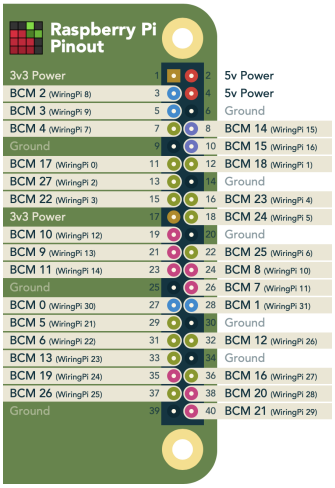


- We can access I2C devices via the I2C pins
- We will use the I2C bus to talk to the **BME280 Environmental sensor** and the **TSL2561 light sensor** on the floral bonnet

You can learn more about the **raspberry-pi i2c pins** at pinout.xyz.

`https://pinout.xyz/pinout/i2c`

Raspberry-pi Zero GPIO pins

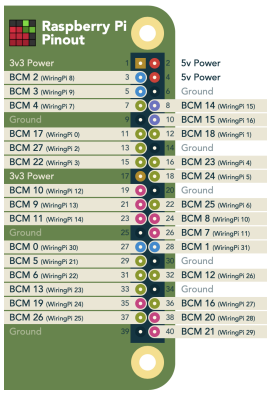


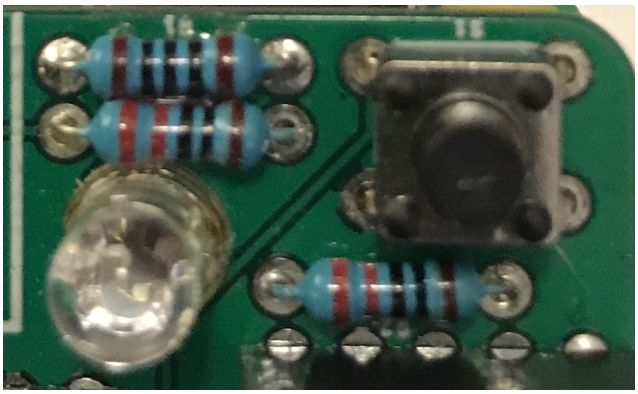
- We have 28 possible GPIOS, though many of these are shared with the previously mentioned buses
- Some of the GPIOs also have PWM capability

You can learn more about the **raspberry-pi serial pins** at pinout.xyz.

`https://pinout.xyz/pinout/uart`

Raspberry-pi Zero GPIO pins



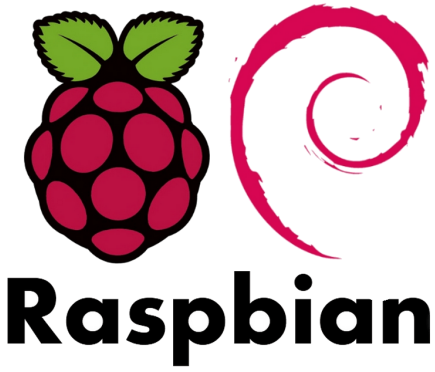


- We will use these pins to interface to a push-button (input) GPIO on the floral bonnet
- We will use 3 of these as output pins (including the 2 PWMs) to drive a tri-color LED

You can learn more about the **raspberry-pi serial pins** at pinout.xyz.

https://pinout.xyz/pinout/uart

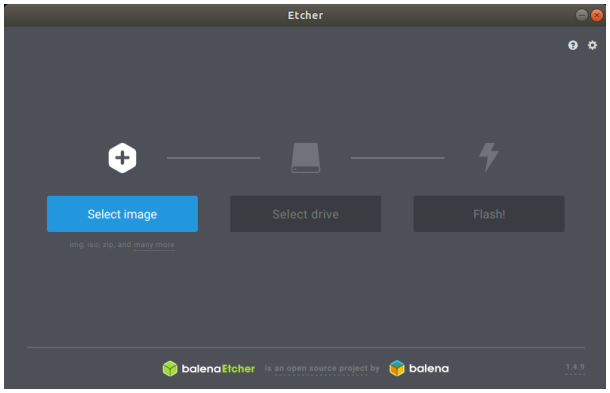
Using Raspian as the Operating System



- We need to put a Linux based operating system on our board
- Raspian is one of the many options for OS on the raspberry-pi
- We will be using Raspian for the remaining labs in these seminars

You can learn more about the **Raspbian** at `https://www.raspbian.org`

uSD card



- We will install a Raspbian image directly (instead of using NOOBS) to the uSD card with **Balena Etcher**
- This will allow us to setup our board over a USB cable

Once the burn is complete you will need to pull the sd card out then reinsert it into the card reader to have it automounted under `/media/$USER`

Once mounted we have to make a few changes to the files on the next page.

Configure Raspian from Host OS

- We need to make some changes to be able to setup Raspbian headless over the USB cables (without a monitor and keyboard)
- With your uSD card still in the SD card reader...
- Use your favourite text editor to modify the following 2 files on the SD card
- Add the following to the end of `boot/config.txt`

```
dtoverlay=dwc2
enable_uart=1
```

- Add the following after `rootwait` in `boot/cmdline.txt`

```
rootwait modules-load=dwc2,g_ether
```

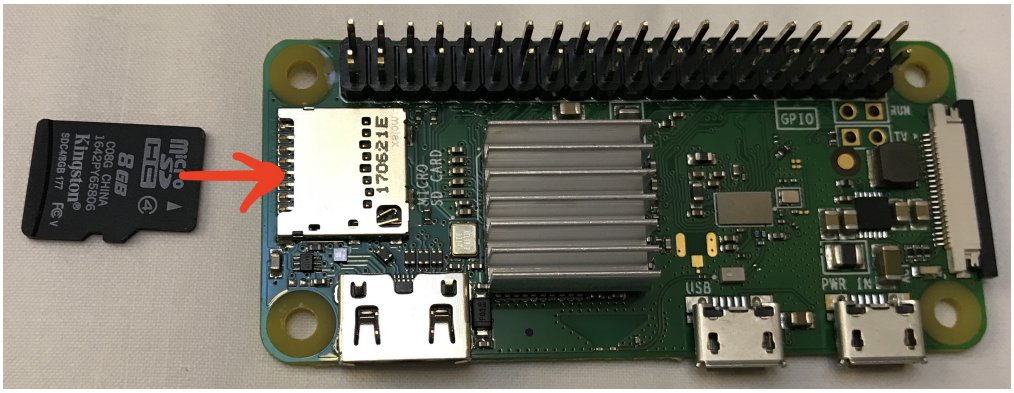
- (it needs to be immediately after `rootwait` before anything else)
- We also need to start ssh by creating `boot/ssh`

```
$ touch /media/$USER/boot/ssh
```

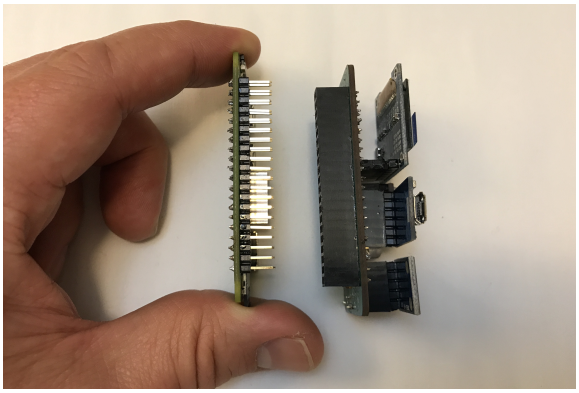
We can make the changes above in many ways, but to do it from the command line we can do something like this:

```
$ echo "dtoverlay=dwc2" >> /media/$USER/boot/config.txt
$ echo "enable_uart=1" >> /media/$USER/boot/config.txt
$ sed -ie 's/rootwait/rootwait modules-load=dwc2,g_ether/' /media/$USER/boot/cmdline.txt
$ touch /media/$USER/boot/ssh
$ sync
$ umount /media/$USER/*oot*
```

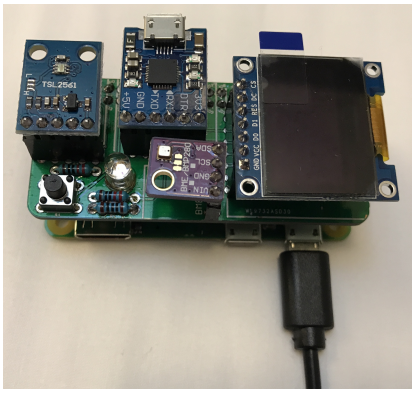
Move the uSD card into the rpi



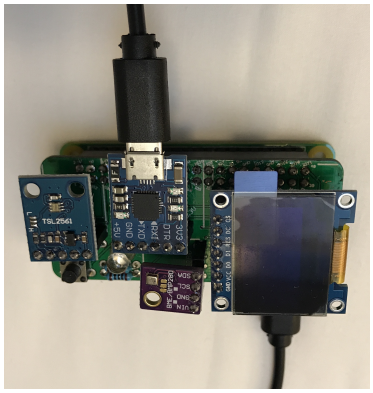
Attach the bonnet to the rpi-0 wh



Attach power to the rpi-0 wh



Attach the serial USB cable to the rpi-0 wh



Connect to the serial port with a terminal program like screen

- There are many terminal programs like **screen**, **minicom** or **putty**
- We will use **screen** in our examples

```
$ screen /dev/ttyUSB0 115200
```

- (You can eventually leave screen with **Ctrl-a, k, y**)

Log into the board

```
[  OK  ] Stopped LSB: Autogenerate and use a swap file.
[  OK  ] Started Show Plymouth Reboot Screen.
[  OK  ] Stopped target Remote File Systems.
[  OK  ] Stopped target Remote File Systems (Pre).
<<<SNIP>>>
[  OK  ] Started /etc/rc.local Compatibility.
[  OK  ] Started Permit User Sessions.
         Starting Hold until boot process finishes up...
         Starting Terminate Plymouth Boot Screen...

Raspbian GNU/Linux 9 raspberrypi ttyS0
raspberrypi login: pi
```

Login as **pi** with password **raspberry**.

Change your password

You probably want to change your password

```
pi@raspberrypi:~$ passwd
Changing password for pi.
(current) UNIX password: raspberry
Enter new UNIX password: <your password>
Retype new UNIX password: <your password again>
passwd: password updated successfully
pi@raspberrypi:~$
```

Don’t forget this password

Now connect with ssh

- Get your rpi IP address

```
pi@raspberrypi:~$ ifconfig usb0
usb0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 169.254.12.238  netmask 255.255.0.0  broadcast 169.254.255.255
```

- With the RNDIS device configured with a link-local address on your host you can now ssh to the target

```
host$ ifconfig ens160u4u1
ens160u4u1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 169.254.42.138  netmask 255.255.0.0  broadcast 169.254.255.255
host$ ssh pi@169.254.12.238
pi@169.254.12.238's password: *******
```

Now we test the peripherals

- Copy `testkit.tar.xz` over to the rpi

```
host$ rsync -aP testkit.tar.xz pi@169.254.12.238:~/
host$ ssh pi@169.254.12.238
pi@raspberrypi:~ $ tar xvf testkit.tar.xz
pi@raspberrypi:~ $ ls testkit
BME280  RGBLed.py  TSL2561  button.py  commands.txt
:luma.examples  py-rpi-ssd1306
```

Now test GPIO

```
pi@raspberrypi:~/testkit $ cd
pi@raspberrypi:~ $ cd testkit
pi@raspberrypi:~/testkit $ python RGBLed.py
```

See the LED light up, then stop it with a `Ctrl-C`

```
pi@raspberrypi:~/testkit $ python button.py
Button Pressed
Button Pressed
^C
```

By pressing the button we see messages on button up and down.

Now test GPIO

```
pi@raspberrypi:~/testkit $ cd
pi@raspberrypi:~ $ cd testkit
pi@raspberrypi:~/testkit $ python RGBLed.py
```

See the LED light up, then stop it with a `Ctrl-C`

```
pi@raspberrypi:~/testkit $ python button.py
Button Pressed
Button Pressed
^C
```

By pressing the button we see messages on button up and down.