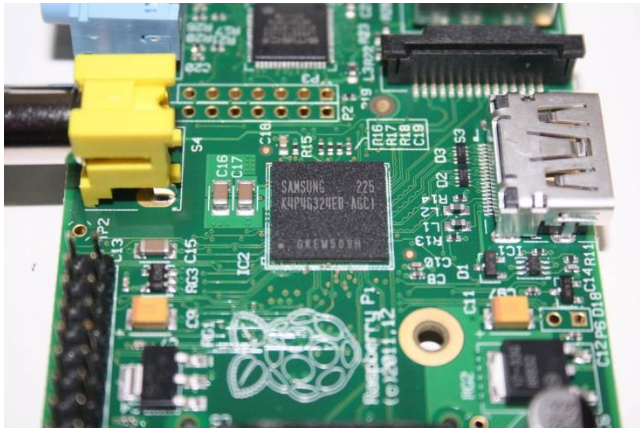
Raspberry pi interfacing

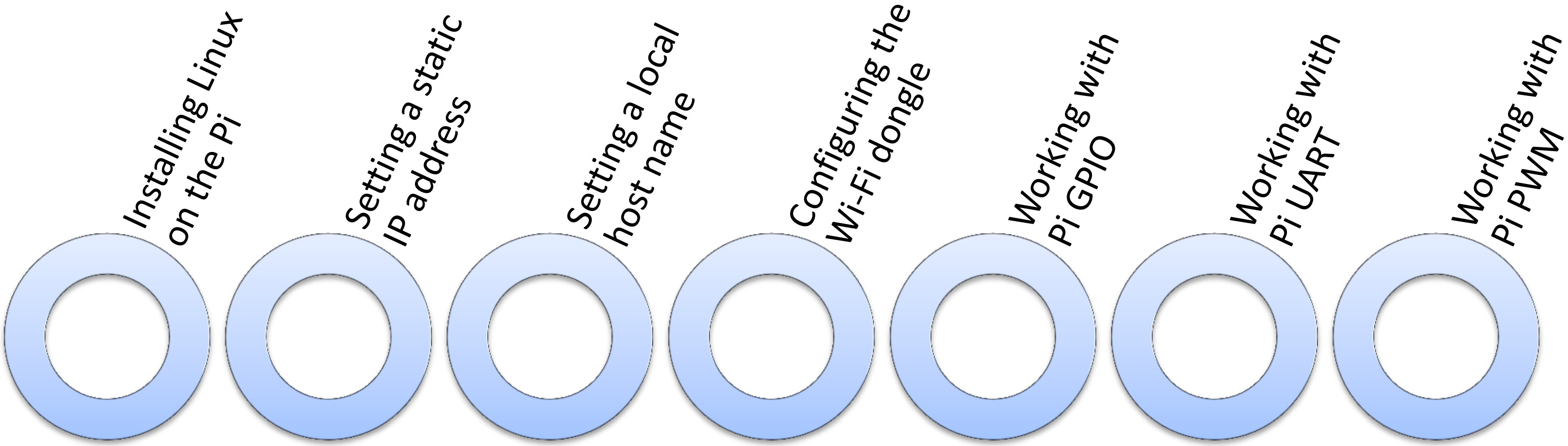
Lecture 6: Working on raspberry pi



Lecture Agenda

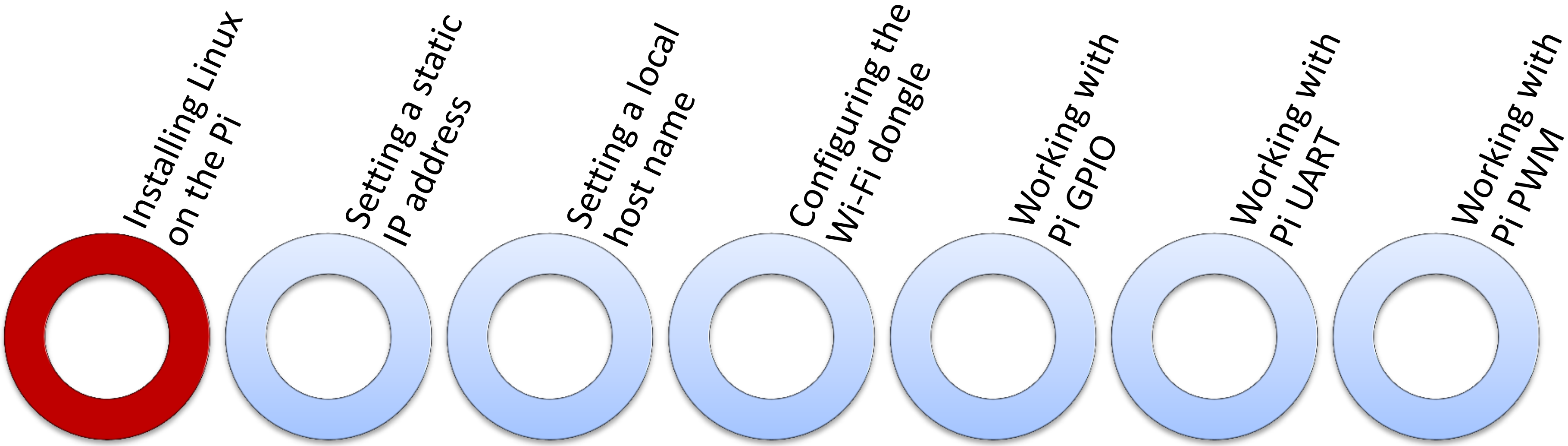
- Installing Linux on the Pi.
- Setting a static IP address.
- Setting a local host name.
- Configuring the Wi-Fi dongle.
- Working with Pi GPIO.
- Working with Pi UART.
- Working with Pi PWM.

Lecture Agenda



- Installing Linux on the Pi
- Setting a static IP address
- Setting a local host name
- Configuring the Wi-Fi dongle
- Working with Pi GPIO
- Working with Pi UART
- Working with Pi PWM

Installing Linux on the Pi



Installing Linux on the Pi

Setting a static IP address

Setting a local host name

Configuring the Wi-Fi dongle

Working with Pi GPIO

Working with Pi UART

Working with Pi PWM

Installing Linux on the Pi

➢ Downloading Linux image

- Download Linux image for raspberry pi, for example download the Raspbian image from the following website:

  http://raspberrypi.org/

➢ Image extracting

- Decompress the downloaded file, rename the decompressed file to any name for easier access using command line, for example let the file name be (assuming in home directory):

  ~/RaspberryImage.img

➢ Installing the image

- Put the SD Card in the computer and format it first (using disks utility for example, you can type “disks” in the already installed applications).

Installing Linux on the Pi

- Then type the following command in the terminal to know the SD Card name (Note the $ is just indication that this is a command, don't write it in your commands):

```
$ dmesg
```

- Then search in the last few lines for SD Card name, for example sdb (Note we will ignore sdb0 and sdb1 and so on as those are just the SD Card internal partitions if you didn't format the card).
- Type on the terminal and wait about 5 minutes to finish:

```
$ sudo dd bs=4M if=~/RaspberryImage.img of=/dev/sdb
```

- Now the Pi is ready to boot, connect your SD Card, keyboard, mouse, HDMI cable, LAN cable and WiFi dongle to the Pi and start working.
- Note that normally the HDMI cable will work automatically, but on some cases if it didn't work, we need to edit some configurations as we will illustrate next.

Installing Linux on the Pi

Configuring the HDMI

- In case that the HDMI didn’t work automatically, connect the SD Card to the computer and open the file called config.txt inside the SD Card boot partition.
- Note that you need to open it as root, in the command line type:

```
$ sudo gedit
```

And before hitting Enter key, drag the config.txt file from its place and drop it beside your command to automatically paste its path, then hit Enter.

- The file after editing should look the same like this:

```
# For more options and information see
# http://www.raspberrypi.org/documentation/configuration/config-txt.md
# Some settings may impact device functionality. See link above for details

# uncomment if you get no picture on HDMI for a default "safe" mode
#hdmi_safe=1
```

Installing Linux on the Pi

➢ Configuring the HDMI

```
# uncomment this if your display has a black border of unused pixels visible
# and your display can output without overscan
disable_overscan=0

# uncomment the following to adjust overscan. Use positive numbers if console
# goes off screen, and negative if there is too much border
#overscan_left=16
#overscan_right=16
#overscan_top=16
#overscan_bottom=16

# uncomment to force a console size. By default it will be display's size minus
# overscan.
#framebuffer_width=1280
#framebuffer_height=720

# uncomment if hdmi display is not detected and composite is being output
hdmi_force_hotplug=1
```

Installing Linux on the Pi

- **Configuring the HDMI**

```
# uncomment to force a specific HDMI mode (this will force VGA)
hdmi_group=1
hdmi_mode=4

# uncomment to force a HDMI mode rather than DVI. This can make audio
work in
# DMT (computer monitor) modes
hdmi_drive=2

# uncomment to increase signal to HDMI, if you have interference, blanking, or
# no display
config_hdmi_boost=4

# uncomment for composite PAL
#sdtv_mode=2

#uncomment to overclock the arm. 700 MHz is the default.
#arm_freq=800
```

Installing Linux on the Pi

Configuring the HDMI

- Now remove the SD Card from your computer and connect it to the Pi and start it and it the HDMI now should work.

First steps on the Pi

- At the first time the Pi boots it shows a configuration screen, you can edit the configurations now or later after login by opening the same screen by typing the command

```
$ sudo raspi-config
```

  - Choose Expand File System option.
  - Choose on advanced options to enable SSH, I2C, SPI.
  - Choose update.

- After update, login by using “pi” as user name, and “raspberry” as password
- Type the following commands

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Installing Linux on the Pi

First steps on the Pi

- Now reboot the Pi

```
$ sudo reboot
```

- To connect remotely to the Pi using your PC inside the same network, we need first to know the Pi IP address, so on the Pi type the command

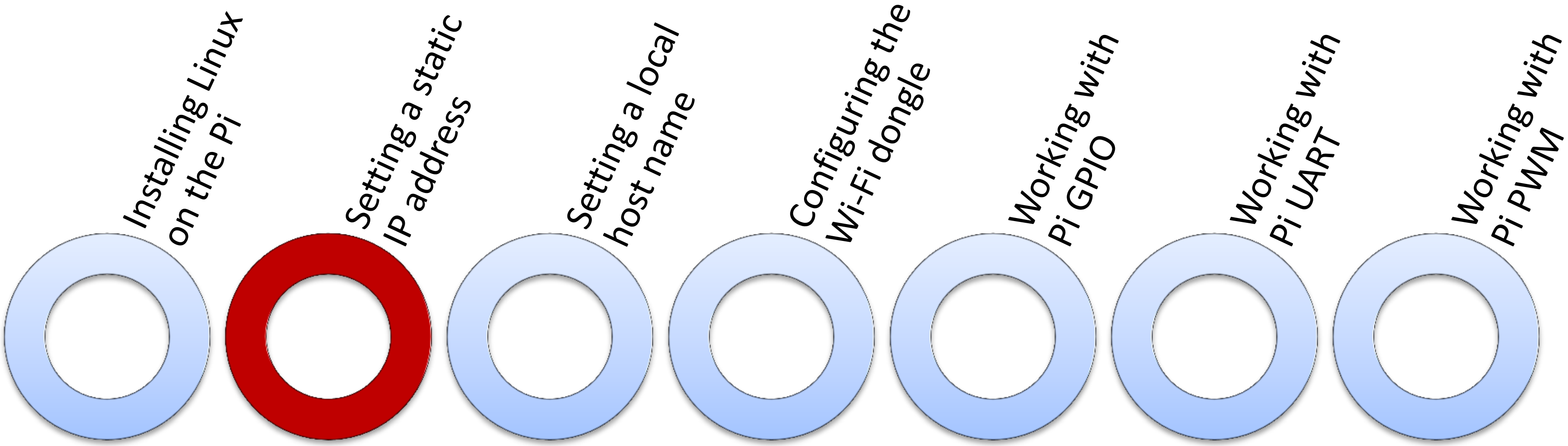
```
$ ifconfig
```

- You will find the Pi IP address beside the interface “eth0” as now we are connected to the network through LAN cable.
- Assuming that you found that the Pi IP is 192.168.1.2, now on your PC while you are connected on the same network, type the following

```
$ sudo ssh pi@192.168.1.2
```

- Then if prompted, type yes and hit enter to confirm the connection
- Then type “pi” as user name, and “raspberry” as password, and now you are one the Pi command line so any command you type is actually executed on the Pi not on your PC.

Setting a static IP address



Installing Linux on the Pi

Setting a static IP address

Setting a local host name

Configuring the Wi-Fi dongle

Working with Pi GPIO

Working with Pi UART

Working with Pi PWM

Setting a static IP address

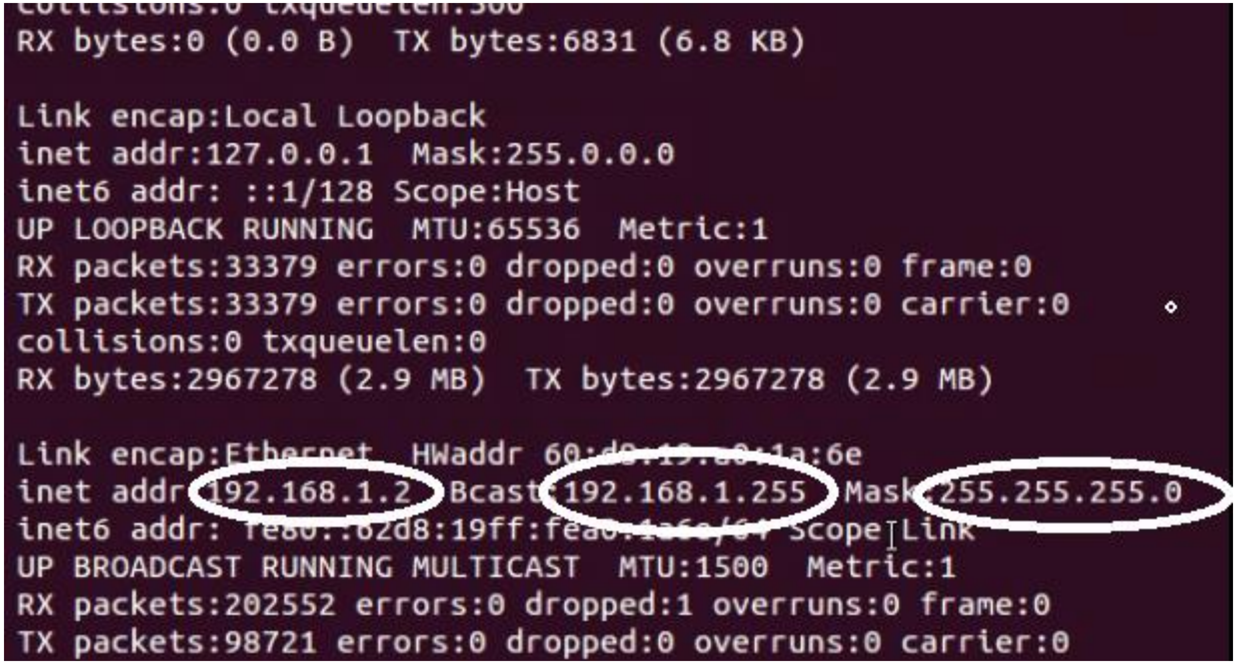
- The IP address we used to connect to the Pi may be changed by the network router, so to make it static edit the following file on the Pi :

  /etc/network/interfaces

- First we need to prepare the configurations, on the Pi type

  $ ifconfig

You will se something like this, store those numbers as we will need them



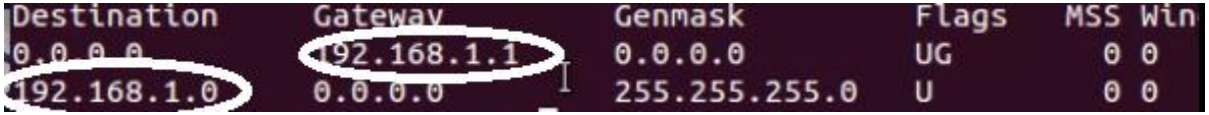
```
collisions:0 txqueuelen:500
RX bytes:0 (0.0 B)  TX bytes:6831 (6.8 KB)

Link encap:Local Loopback
inet addr:127.0.0.1  Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING  MTU:65536  Metric:1
RX packets:33379 errors:0 dropped:0 overruns:0 frame:0
TX packets:33379 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:2967278 (2.9 MB)  TX bytes:2967278 (2.9 MB)

Link encap:Ethernet  HWaddr 60:d8:19:a0:1a:6e
inet addr:192.168.1.2  Bcast:192.168.1.255  Mask:255.255.255.0
inet6 addr: fe80::62d8:19ff:fea0:1a6e/64 Scope:Link
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:202552 errors:0 dropped:1 overruns:0 frame:0
TX packets:98721 errors:0 dropped:0 overruns:0 carrier:0
```

Setting a static IP address

- Then type

```
$ netstat -nr
```

You will se something like this, store those numbers as we will need them



```
Destination     Gateway         Genmask         Flags   MSS Win
0.0.0.0         192.168.1.1     0.0.0.0         UG        0 0
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0
```

- Now type:

```
$ sudo nano /etc/network/interfaces
```

- Uncomment the following line to look like this:

```
#iface eth0 inet dhcp
```

- Add after that line the following lines

```
iface eth0 inet static
address 192.168.1.2
netmask 255.255.255.0
network 192.168.1.0
broadcast 192.168.1.255
gateway 192.168.1.1
```

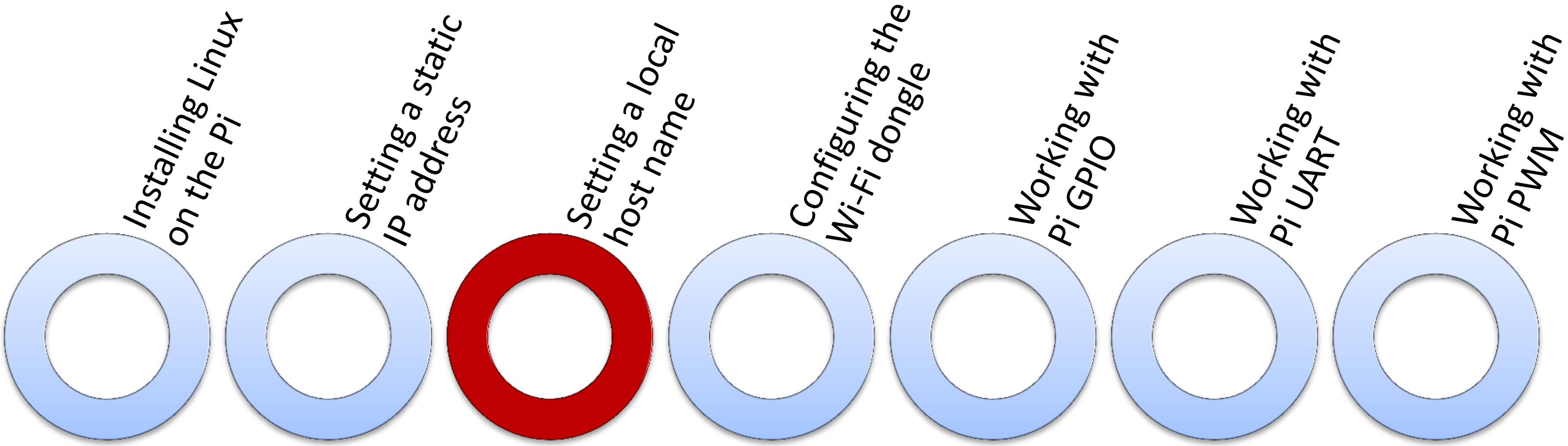
Setting a static IP address

- The file after editing should look like this, now close the file by hitting CTRL+X and hit Enter:

```
auto lo
iface lo inet loopback
#iface eth0 inet dhcp
iface eth0 inet static
address 192.168.1.2
netmask 255.255.255.0
network 192.168.1.0
broadcast 192.168.1.255
gateway 192.168.1.1
allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

- Now whenever the Pi boots, it will automatically take the IP 192.168.1.2

Setting a local host name



Installing Linux on the Pi

Setting a static IP address

Setting a local host name

Configuring the Wi-Fi dongle

Working with Pi GPIO

Working with Pi UART

Working with Pi PWM

Setting a local host name

- The problem of the static IP address is the when we move to another network with other configurations it will not work, so we will give the Pi a local host name that we can deal with the Pi using that name whatever its IP.
- First type the following command to set password for root user as we will need it

```
$ sudo passwd
```

- Then we need to undo the configurations of the static IP, open the same file again /etc/network/interfaces and undo all changes again

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp
allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

Setting a local host name

- Then type

```
$ sudo apt-get install avahi-daemon
```

- Now inside any network you can connect to the Pi using the name “raspberrypi.local”, and the root password

```
$ sudo ssh raspberrypi.local
```

*Note: You can switch to “pi” user by typing `$ su pi`

- To change the name, change “raspberrypi” in the following 2 files to the name that you want (ex. “mynewname” or anything)

```
$ sudo nano /etc/hosts
$ sudo nano /etc/hostname
```

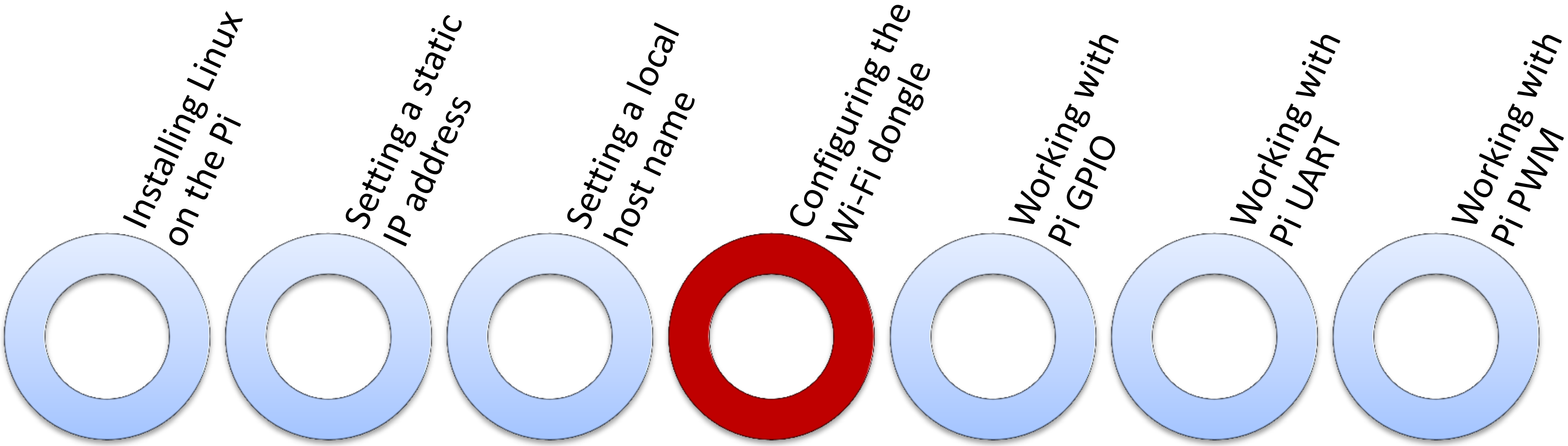
- Then to commit, run the command multiple times until it run successfully without errors.

```
$ sudo /etc/init.d/hostname.sh
```

- Then reboot so you can use your new name

```
$ sudo ssh mynewname.local
```

Configuring the Wi-Fi dongle



Installing Linux on the Pi

Setting a static IP address

Setting a local host name

Configuring the Wi-Fi dongle

Working with Pi GPIO

Working with Pi UART

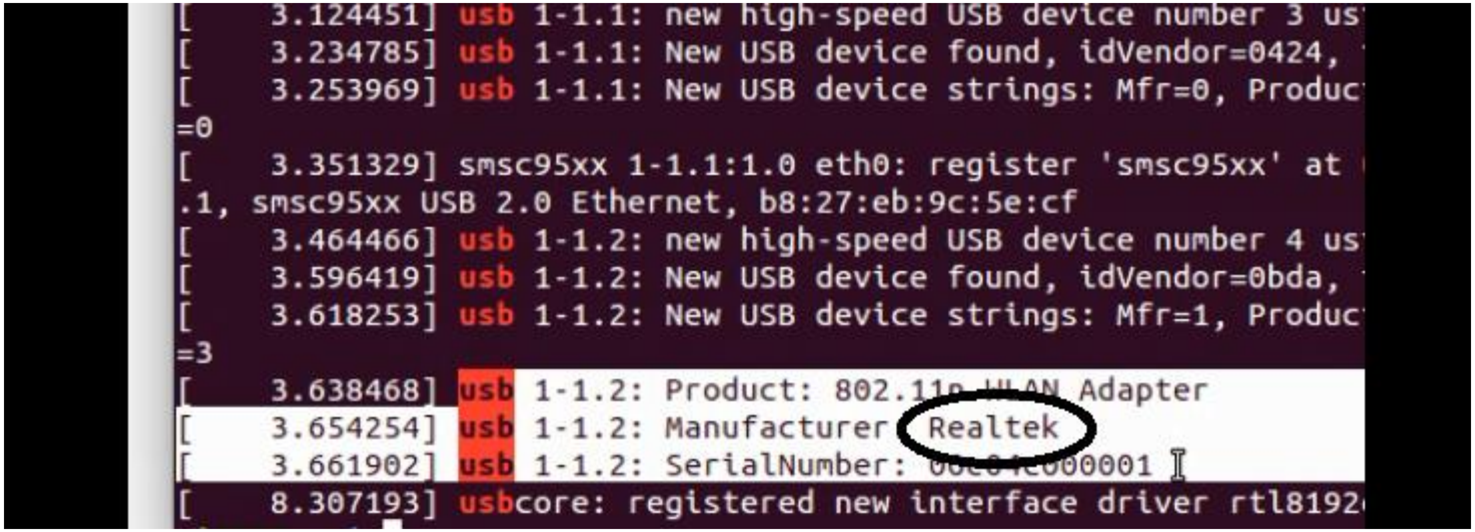
Working with Pi PWM

Configuring the Wi-Fi dongle

- First to know the manufacturer of the WiFi dongle type:

  $ dmesg | grep usb

- You will see something like this



```
[    3.124451] usb 1-1.1: new high-speed USB device number 3 us
[    3.234785] usb 1-1.1: New USB device found, idVendor=0424,
[    3.253969] usb 1-1.1: New USB device strings: Mfr=0, Produc
=0
[    3.351329] smsc95xx 1-1.1:1.0 eth0: register 'smsc95xx' at
.1, smsc95xx USB 2.0 Ethernet, b8:27:eb:9c:5e:cf
[    3.464466] usb 1-1.2: new high-speed USB device number 4 us
[    3.596419] usb 1-1.2: New USB device found, idVendor=0bda,
[    3.618253] usb 1-1.2: New USB device strings: Mfr=1, Produc
=3
[    3.638468] usb 1-1.2: Product: 802.11n WLAN Adapter
[    3.654254] usb 1-1.2: Manufacturer: Realtek
[    3.661902] usb 1-1.2: SerialNumber: 00e04c000001
[    8.307193] usbcore: registered new interface driver rtl8192
```

- As we can see it is Realtek.

Configuring the Wi-Fi dongle

- Then type the following commands:

```
$ sudo apt-cache search firmware wireless
$ sudo apt-get install firmware-realtek
```

- Then edit the interfaces file

```
$ sudo nano /etc/network/interfaces
```

- Replace the following lines:

```
allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
```

  With the following lines:

```
auto wlan0
iface wlan0 inet dhcp
wpa-conf /etc/wpa.conf
```

Configuring the Wi-Fi dongle

- Then type the following command:

  $ sudo nano /etc/wpa.conf

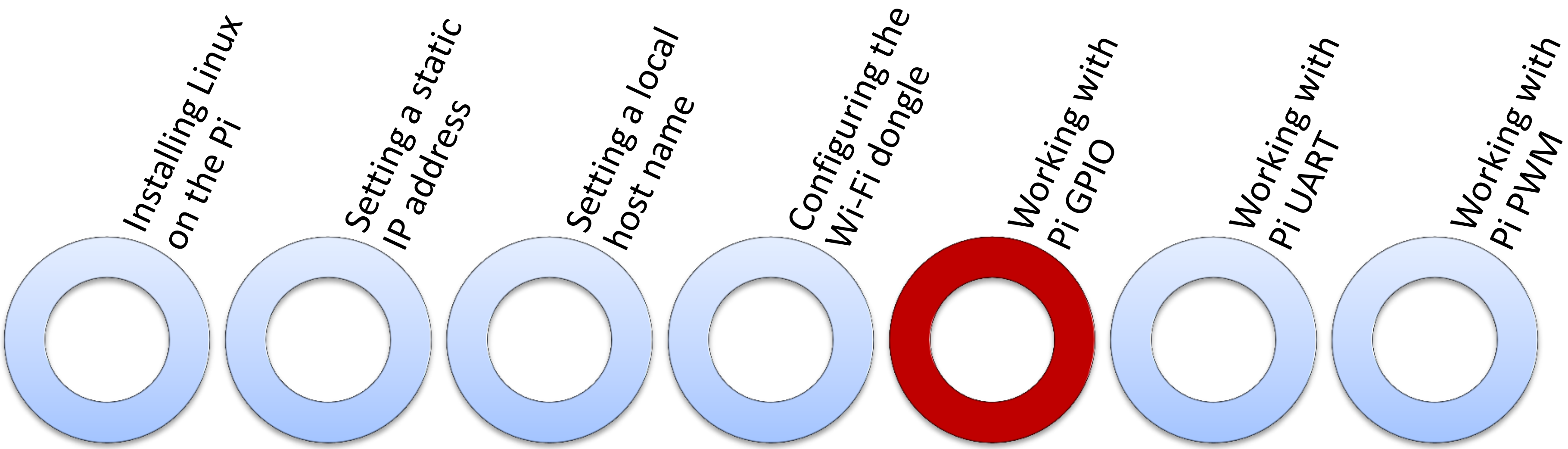
- Type the following lines, replace “networkname” with your WiFi network name, and “network passkey” with your WiFi network pass key:

```
network={
        ssid= “networkname”
        key_mgmt=WPA-PSK
        psk= “netowork passkey”
}
```

*Note that there is no space after the = sign

- Now shutdown and remove the LAN cable then start the Pi with the WiFi dongle only and it will automatically access the network, you can access it remotely using the same name “raspberrypi.local” or the name you gave to it.

Working with Pi GPIO



Installing Linux on the Pi

Setting a static IP address

Setting a local host name

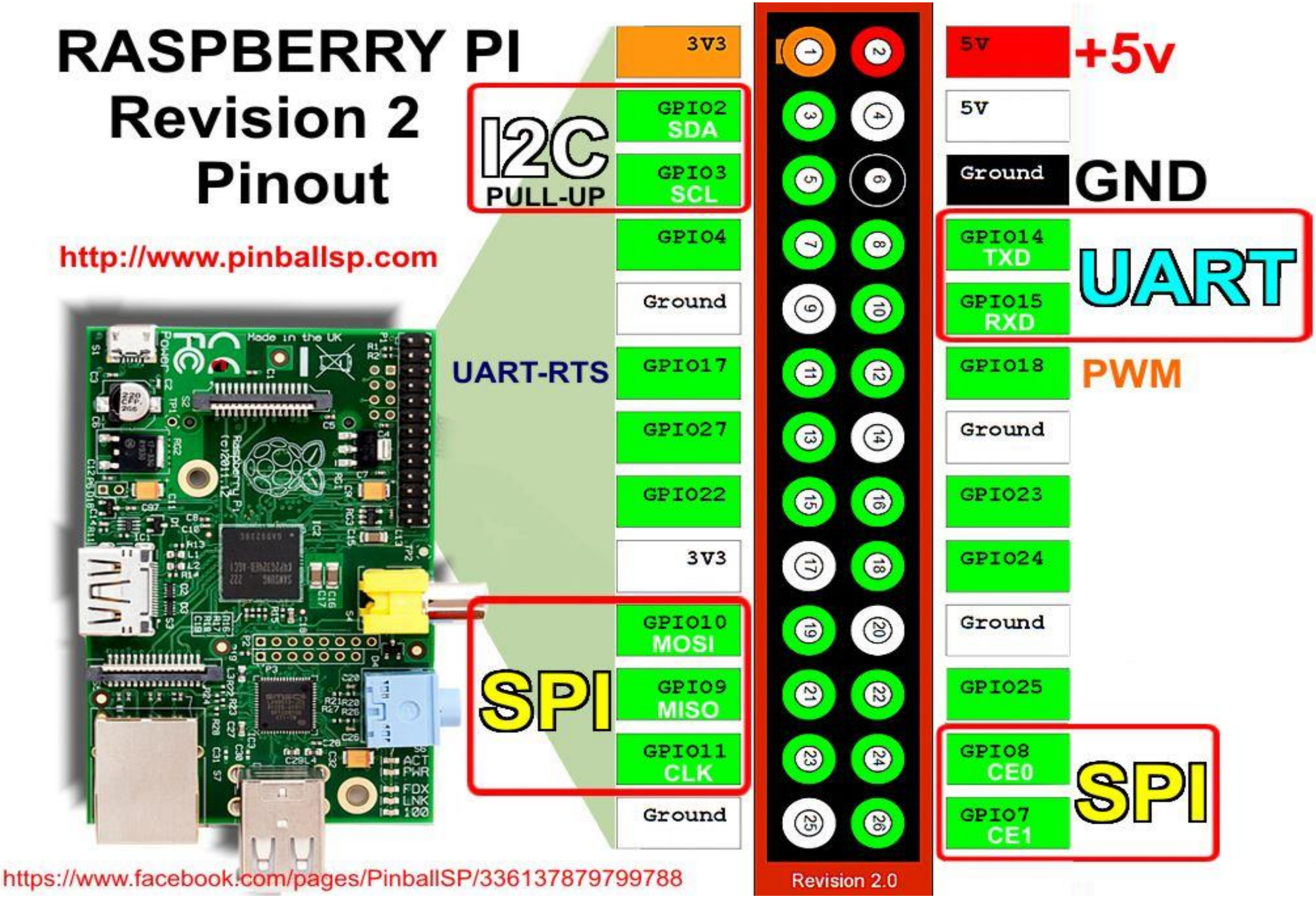
Configuring the Wi-Fi dongle

Working with Pi GPIO

Working with Pi UART

Working with Pi PWM

Working with Pi GPIO



3V3	1	2	5V
GPIO2 SDA	3	4	5V
GPIO3 SCL	5	6	Ground
GPIO4	7	8	GPIO14 TXD
Ground	9	10	GPIO15 RXD
GPIO17	11	12	GPIO18
GPIO27	13	14	Ground
GPIO22	15	16	GPIO23
3V3	17	18	GPIO24
GPIO10 MOSI	19	20	Ground
GPIO9 MISO	21	22	GPIO25
GPIO11 CLK	23	24	GPIO8 CE0
Ground	25	26	GPIO7 CE1

Working with Pi GPIO

- To interact with Pi GPIO (Make pi input or output, read pin value or write to pin logic high or logic low value), first create empty file:

```
$ nano GPIO_python.py
```

- Write the following lines in the file, here we are making LED ON for 5 seconds then OFF on GPIO17:

```
#!/usr/bin/python                # To make this script run by python

import RPi.GPIO as GPIO          #import python GPIO library
import time                      #import time library to use delay function
GPIO.setmode(GPIO.BCM)           Set numbering mode (BCM means when we
                                 say 17 in the next line we mean GPIO17 as
                                 shown in the  previous picture)
GPIO.cleanup()                   # Clean up previous GPIO configurations
GPIO.setup(17,GPIO.OUT)          # Set GPIO17 as output
GPIO.output(17,GPIO.HIGH)        # Output logic high to GPIO17 (LED ON)
time.sleep(5)                    # Delay 5 seconds
GPIO.output(17,GPIO.LOW)         # Output logic low to GPIO17 (LED OFF)
```

Working with Pi GPIO

- Now close the file, then add execute permission to it:

```
$ sudo chmod u+x GPIO_python.py
```

- Now execute the script by typing:

```
$ sudo ./GPIO_python.py
```

Working with Pi GPIO

- To make LED Flasher each 1 second on GPIO17, create new script as previous example:

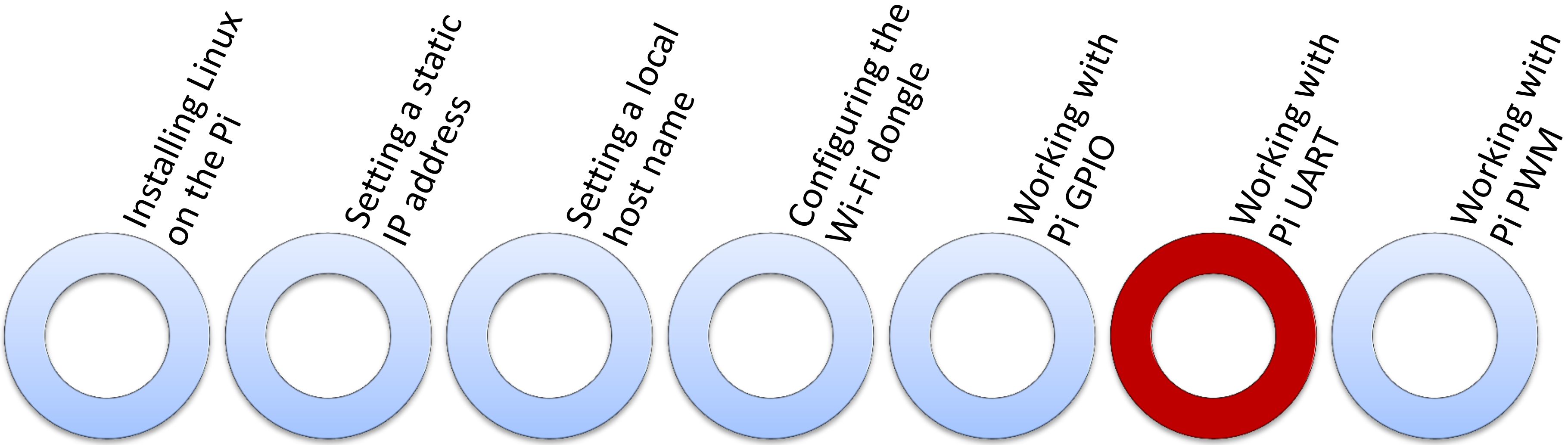
```
#!/usr/bin/python                # To make this script run by python

import RPi.GPIO as GPIO          #import python GPIO library
import time                      #import time library to use delay function
GPIO.setmode(GPIO.BCM)           # Set numbering mode (BCM means when
                                 we say 17 in the next line we mean GPIO17
                                 as shown in the  previous picture)
GPIO.cleanup()                   # Clean up previous GPIO configurations
GPIO.setup(17,GPIO.OUT)          # Set GPIO17 as output
while True:                      # Do it forever
        GPIO.output(17,GPIO.HIGH)        # Output logic high to GPIO17
                                         (LED ON)
        time.sleep(1)                    # Delay 1 second
        GPIO.output(17,GPIO.LOW)         # Output logic low to GPIO17 (LED
                                         OFF)
        time.sleep(1)                    # Delay 1 second
```

Working with Pi GPIO

- To take input from GPIO4, if logic high the make the LED connected to GPIO17 to be ON, else make LED OFF, create new script:

```
#!/usr/bin/python                # To make this script run by python

#switch input script, if pressed LEDON, else LEDOFF
import RPi.GPIO as GPIO          #import python GPIO library
GPIO.setmode(GPIO.BCM)           # Set numbering mode
GPIO.cleanup()                   # Clean up previous GPIO configurations
GPIO.setup(4,GPIO.IN)            # Set GPIO4 as input
GPIO.setup(17,GPIO.OUT)          # Set GPIO17 as output

while True:                      # Do it forever
    if GPIO.input(4) == True:    # Take GPIO4 status, if Logic High, then
        GPIO.output(17,GPIO.HIGH)      # Output logic high to GPIO17
                                       (LED ON)
    else:                        # Else, GPIO4 status is Logic Low, then
        GPIO.output(17,GPIO.LOW)       # Output logic low to GPIO17 (LED
                                       OFF)
```

Working with Pi UART



Installing Linux on the Pi

Setting a static IP address

Setting a local host name

Configuring the Wi-Fi dongle

Working with Pi GPIO

Working with Pi UART

Working with Pi PWM

Working with Pi UART

- To interact with Pi UART, first we need to prevent the kernel from sending any messages to the UART port by editing these files:
  - First backup the original file

    ```
    $ sudo cp /boot/cmdline.txt /boot/cmdline_bp.txt
    ```

  - Change this file configurations

    ```
    $ sudo nano /boot/cmdline.txt
    ```

  - Remove ttyAMA0,115200, final line should be something like this:

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p6 rootfstype=ext4
elevator=deadline rootwait
```

  - Change this file configurations

    ```
    $ sudo nano /etc/inittab
    ```

  - Comment out the line that contains ttyAMA0 115200, by putting # in its start
- Reboot and now the UART port is ready to be used.

  ```
  $ sudo reboot
  ```

Working with Pi UART

- To send and receive bytes on UART port, create new script:

```
#!/usr/bin/python                       # To make this script run by python
import serial                           # import python serial library
import time                             # import python time library

portname = serial.Serial("/dev/ttyAMA0")     # choose UART device name
portname.baudrate = 9600                      # Set UART baud rate to be 9600
if portname.isOpen() == False :              # check if port is not opened
        portname.open()                      # Open the UART port
portname.flushInput()                        # Empty input and output buffers
portname.flushOutput()
portname.write("WELCOME")                    # Send “WELCOME” on the
                                             UART Tx pin
time.sleep(3)                                # Delay 3 seconds
data = portname.read(10)                     # Read 10 bytes from UART Rx pin
                                             and store it in variable called data
print data                                   # Print received bytes to screen
portname.close()                             # Close UART port
```

Working with Pi UART

- To observe what you receive on the serial port you can use a program like minicom, to install it:

```
$ sudo apt-get install minicom
```

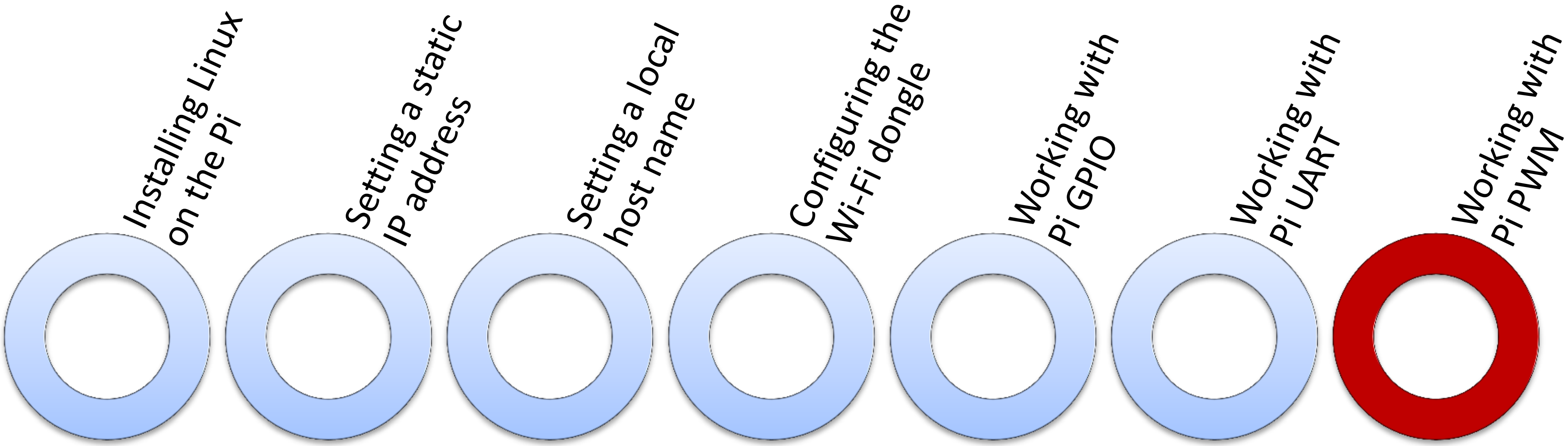
- To edit program configurations:

```
$ sudo minicom -s
```

- To edit port name press Shift+A, and change it to be /dev/ttyAMA0.
- To edit baud rate press Shift+E, then Shift+C for 9600.
- Now choose save as dfl to save this configurations as default, then Exit.
- To use the program at any time, type the following and observe what you receive on the UART port:

```
$ sudo minicom
```

Working with Pi PWM



Installing Linux on the Pi

Setting a static IP address

Setting a local host name

Configuring the Wi-Fi dongle

Working with Pi GPIO

Working with Pi UART

Working with Pi PWM

Working with Pi PWM

- To output a soft PWM signal on any GPIO for example GPIO27, create new script:

```
#!/usr/bin/python                  # To make this script run by python
import RPi.GPIO as GPIO             # import python GPIO library
import time                         # import python time library

GPIO.setmode(GPIO.BCM)              # Set numbering mode
GPIO.cleanup()                      # Clean up previous GPIO configurations

GPIO.setup(27, GPIO.OUT)            # Set GPIO27 as output
pwm_pin = GPIO.PWM(27, 60)          #C onfigure PWM on GPIO27 with
                                    frequency 60Hz
pwm_pin.start(50)                   # Start PWM signal with duty cycle 50%
time.sleep(5)                       # Delay 5 seconds
pwm_pin.stop()                      # Stop PWM signal
GPIO.output(27, GPIO.LOW)           # Output Logic low on GPIO27
GPIO.cleanup()                      # Clean up GPIO configurations
```

Working with Pi PWM

- Useful methods:

```
pwm_pin.ChangeFrequency(f)      # change PWM signal frequency to be f Hz

pwm_pin.ChangeDutyCycle(dc)     # change PWM signal duty cycle to be dc %
```