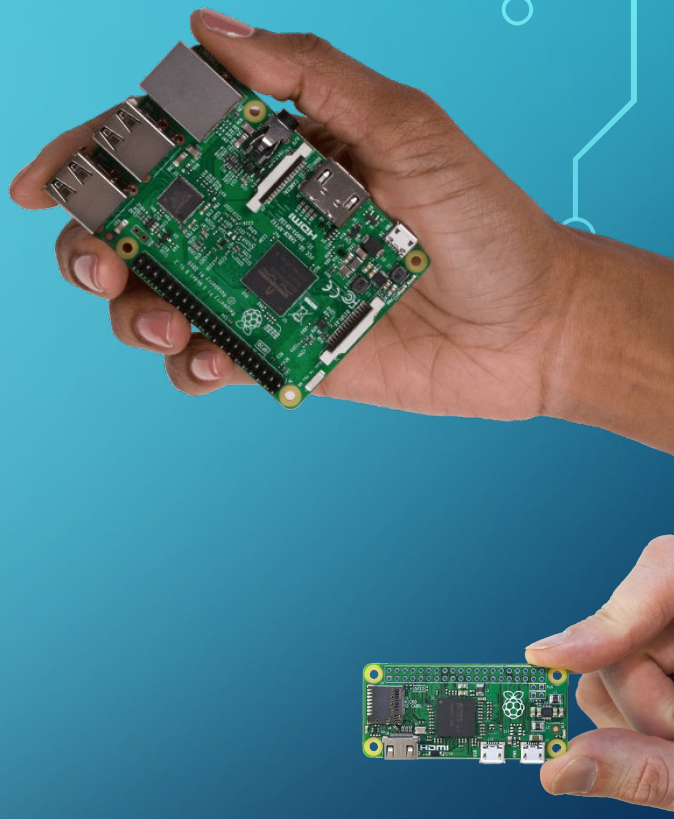
INTRO TO RPI (PART 1)

BY SUTD IEEE

AGENDA

- Setting up your RPi
  - Install Raspbian
  - Configure Wi-Fi
  - Set static IP
  - Setup SSH
  - Setup VNC
- Using the RPi through the Terminal
- Using the RPi like an Arduino
  - GP I/O Pins with Python

WHAT'S AN RPI?!



- Single Board Computer
- Runs Linux (Most of the time)
- Small
  - System on a Chip (SOC)
  - All/ most important components are on a single chip (CPU, memory, ~~storage~~, IO)
- Access to GP I/O Pins (Input and Output)
  - Like an Arduino

USES OF RPI'S

- Web Servers
- Cloud Servers
- Home Automation
- Home Security
- Arcade Games
- Supercomputing (Clusters)
- Cryptocurrency mining
- Robotics
- …

INSTALLING RASPBIAN ON THE RPI'S SD CARD

- Copy Etcher and the Raspbian image to your computer
- Connect the microSD Card
- Open Etcher
- Select the Raspbian image
- Click 'Flash!'

SETTING UP WI-FI

- Raspbian does not have a GUI that supports WPA2 Protocol
  - Cannot connect to SUTD_Student directly!
  - Need to configure it manually
- Configure network in `/etc/wpa_supplicant/wpa_supplicant.conf`

SETTING UP WI-FI

- Open Terminal
- `sudo nano /etc/wpa_supplicant/wpa_supplicant.conf`
- Add this to the file:

```
network={
        ssid="SUTD_Student"
        key_mgmt=WPA-EAP
        eap=PEAP
        identity="100XXXX"
        password="YOUR_PASSWORD"
        phase1="peaplabel=0"
        phase2="auth=MSCHAPV2"
}
```

- Reboot

SET STATIC IP ADDRESS

- What's an IP(v4) Address?
- Why? So we can connect to the RPi at the same address every time.
- Open Terminal:

```
ip -4 addr show | grep global
ip route | grep default | awk '{print $3}'
cat /etc/resolv.conf
```

SET STATIC IP ADDRESS

- `sudo nano /etc/dhcpcd.conf`
- Add this to the file:

```
interface wlan0
static ip_address=10.1.1.31/24
static routers=10.1.1.1
static domain_name_servers=10.1.1.1
```

- Reboot

WHAT IS SSH (SECURE SHELL)

- It is a protocol that allows a computer to remotely log in to another through the Terminal, allowing you to use that computer.
  - In this case the RPi

SETTING UP SSH

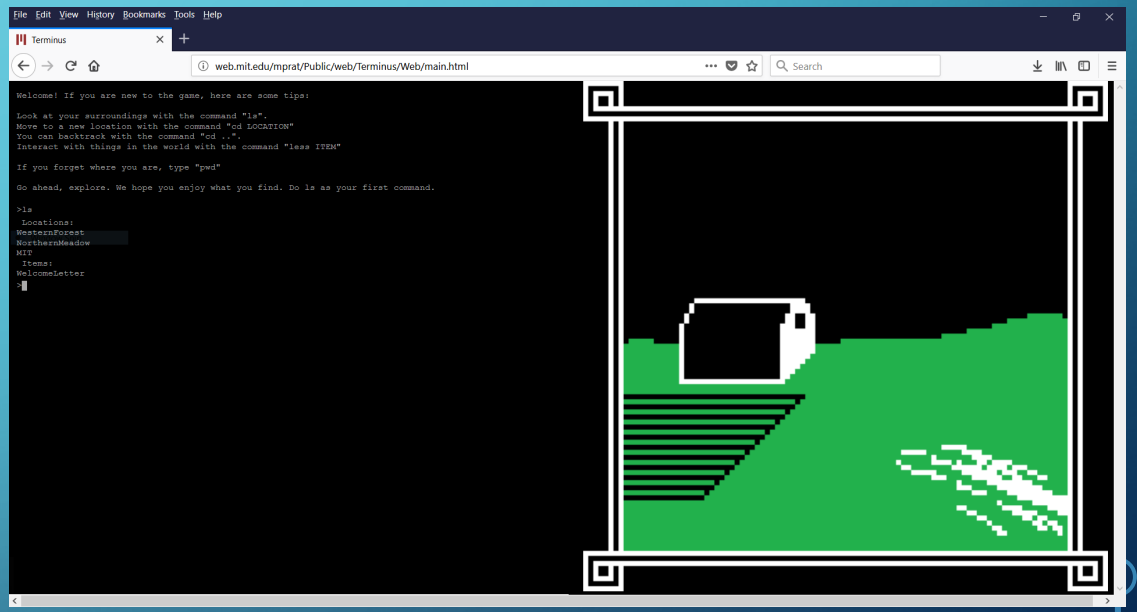
- GUI
  - Open ‘Applications Menu’ (Top Left) >> ‘Preferences’ >> ‘Raspberry Pi Configuration’
  - Select the ‘Interfaces’ tab; enable SSH
- Command Line
  - Run: `sudo raspi-config`
  - Navigate to interfaces; enable SSH

SETTING UP VNC (VIRTUAL NETWORK COMPUTING)

- RealVNC – Free!
- GUI
  - Open ‘Applications Menu’ (Top Left) >> ‘Preferences’ >> ‘Raspberry Pi Configuration’
  - Select the ‘Interfaces’ tab; enable VNC
- Command Line
  - Run: `sudo raspi-config`
  - Navigate to interfaces; enable VNC

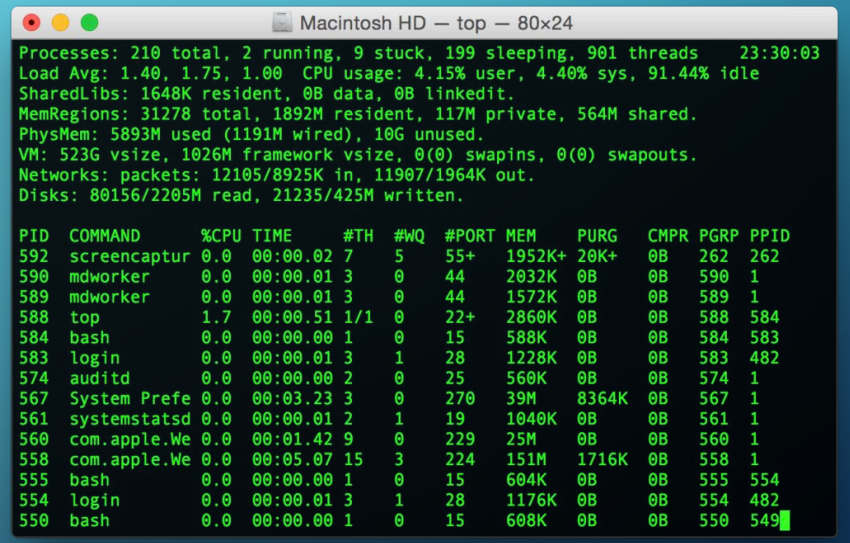
USING THE RPI THROUGH THE TERMINAL

- www.mprat.org/Terminus/
- Learn Linux commands to navigate and control the file system.



LINUX COMMANDS & TOOLS

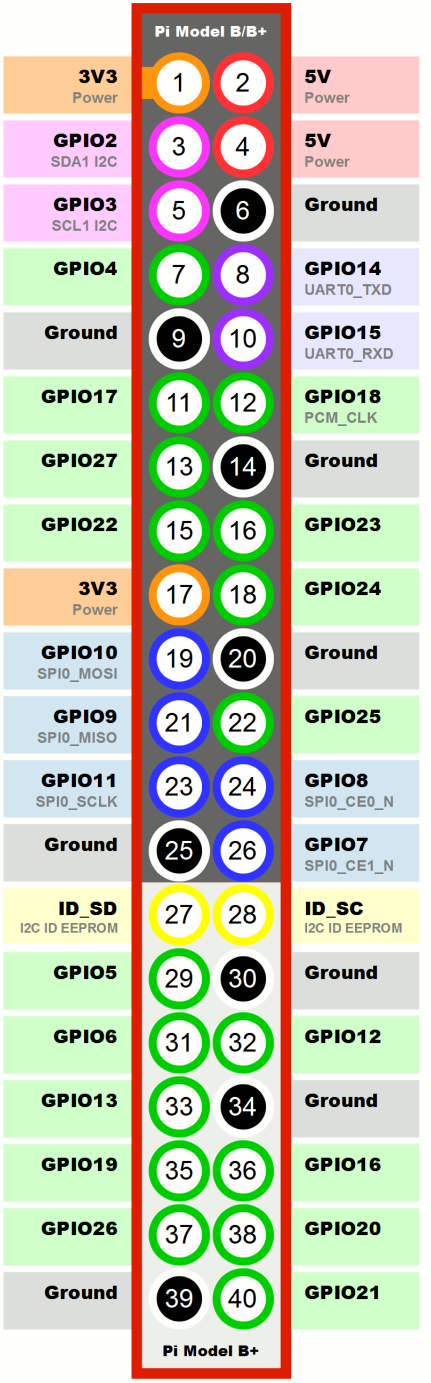
- cd
- ls
- touch
- rm
- mkdir
- rmdir
- nano/ vi
- mv
- cp
- find
- cat
- ifconfig



http://www.informit.com/blogs/blog.aspx?uk=The-10-Most-Important-Linux-Commands

NOW FOR THE ELECTRONICS STUFF

- Program the RPi's GP I/O Pins
  - General Purpose Input/ Output
  - https://pinout.xyz/#
- Use it like an Arduino
- Can be done using Python, C, C++, NodeJS, Bash, etc.



GP I/O WITH PYTHON

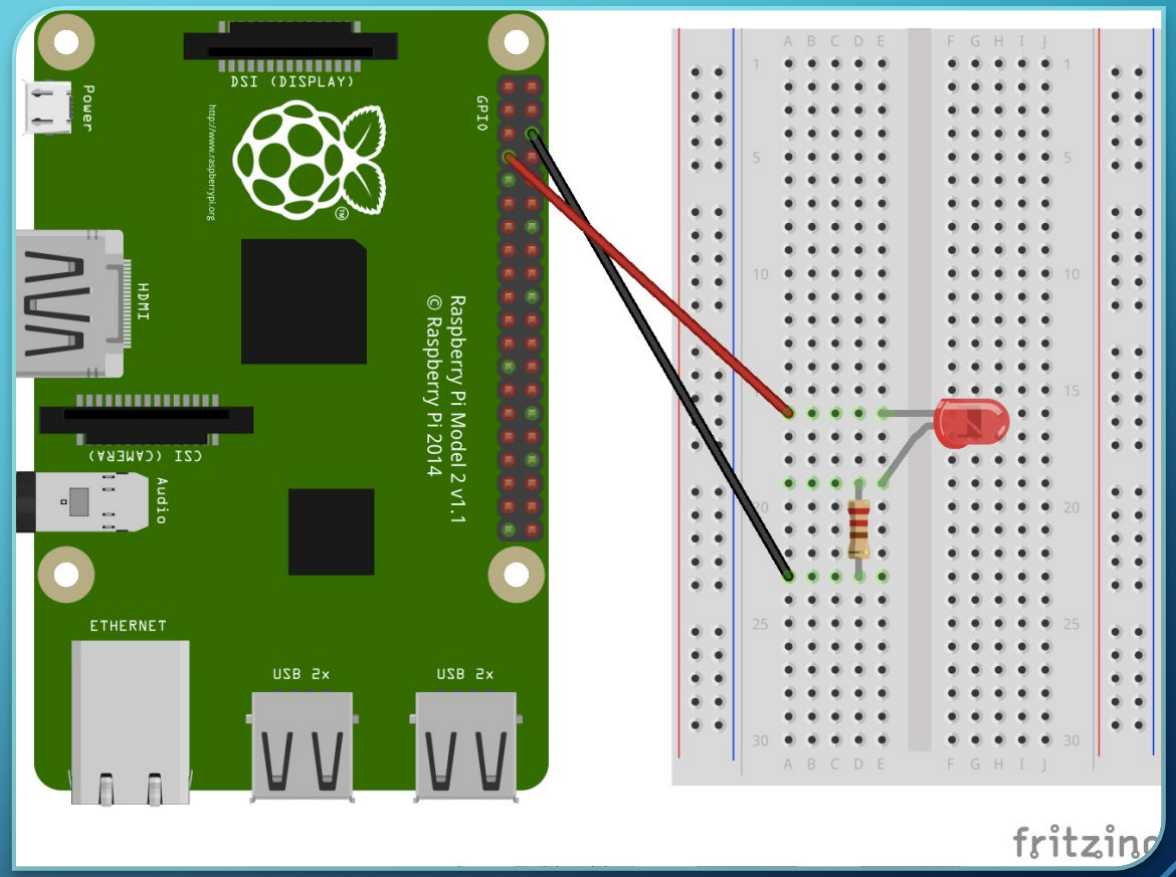
- Install RPI.GPIO (In Terminal)
  - `sudo pip3 install rpi.gpio`
- In Python File
  - `import RPi.GPIO as GPIO`
- Documentation
  - https://sourceforge.net/p/raspberry-gpio-python/wiki/Examples/

GP I/O WITH PYTHON

- **GPIO.setmode(MODE)** => MODE is GPIO.BOARD or GPIO.BCM
- GPIO.setup(channel, GPIO.HIGH) => channel can be a list of channels
- GPIO.setup(channel, GPIO.HIGH, initial=GPIO.HIGH)
- GPIO.input(channel)
- GPIO.output(channel) => channel can be a list of channels
- GPIO.PWM(channel,frequency)
- **GPIO.cleanup()**

ACTIVITY #1: BLINKING LED

- Connect +ve lead of LED (Longer leg) to BCM26
  - Refer to https://pinout.xyz/# !
- Connect a resistor from the –ve lead of the LED to an empty space
- Connect the resistor to a GND pin
  - Refer to https://pinout.xyz/# !



ACTIVITY #1: BLINKING LED

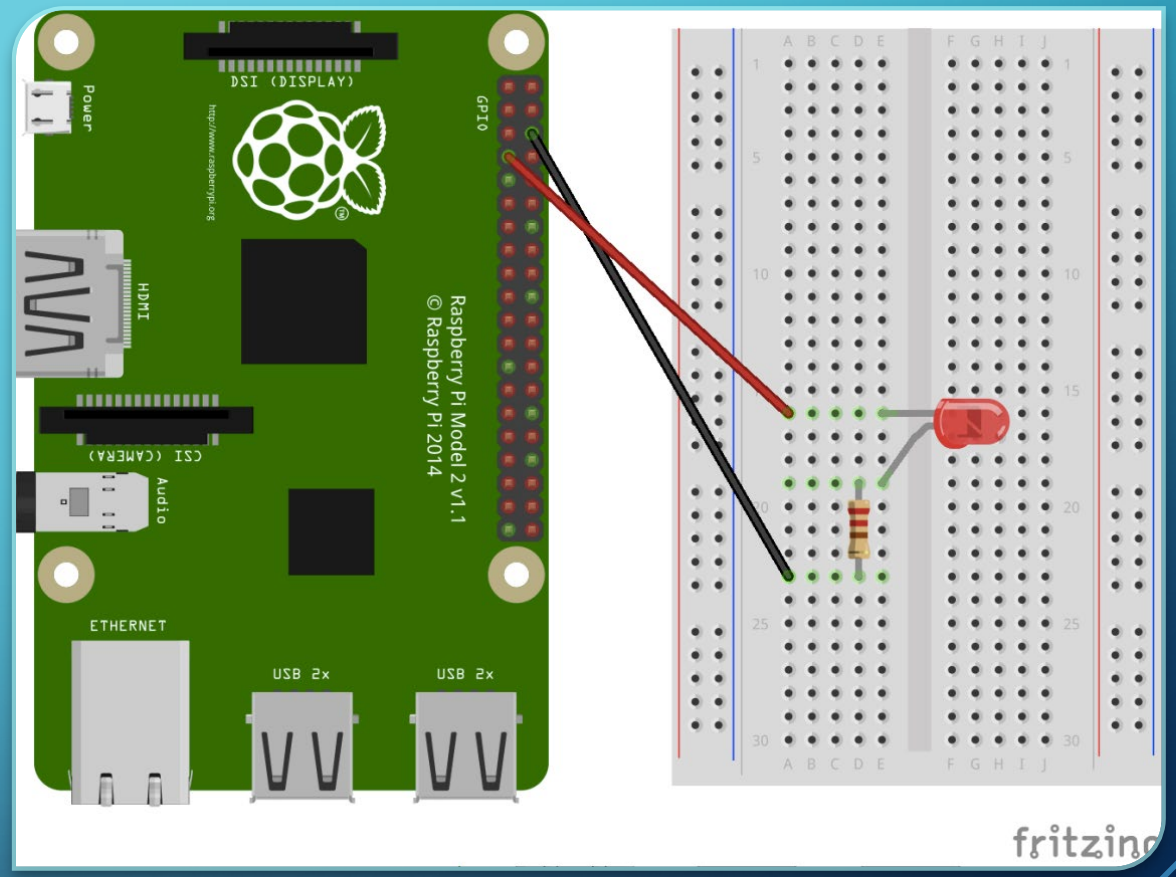
- `import RPi.GPIO as GPIO`
- `from time import sleep`
- `import sys`
- `GPIO.setmode(GPIO.BCM)`
- `GPIO.setup(26,GPIO.OUT)`
- `GPIO.output(26,GPIO.HIGH)`
- `sleep(1) // Sleep for 1s`

ACTIVITY #1: BLINKING LED

```
Try:
        while True:
                # Do Something
finally:
        GPIO.cleanup()
        sys.exit()
```

ACTIVITY #2: FADING LED

- Connect +ve lead of LED (Longer leg) to BCM26
  - Refer to https://pinout.xyz/# !
- Connect a resistor from the –ve lead of the LED to an empty space
- Connect the resistor to a GND pin
  - Refer to https://pinout.xyz/# !

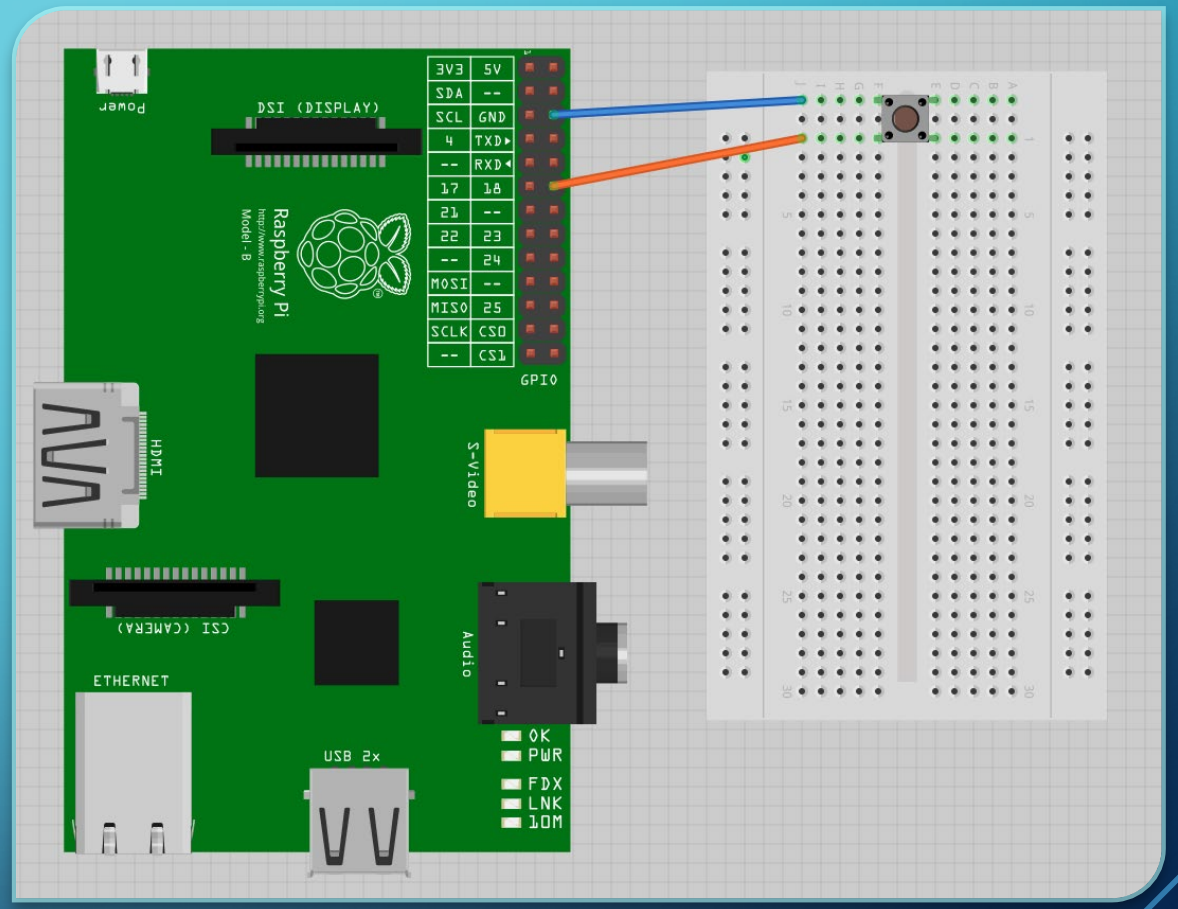


ACTIVITY #2: FADING LED

- `pwm = GPIO.PWM(26,1000)`
- `pwm.start(0)`
- `pwm.ChangeDutyCycle(x) // 0 ≤ x ≤ 100`
- `for i in range(100):`
- `pwm.stop()`

ACTIVITY #3: PUSH BUTTON

- Connect one end of the button to BCM26
- Connect the other end on the same side to GND



ACTIVITY #3: PUSH BUTTON

- `GPIO.setup(26,GPIO.IN,pull_up_down=GPIO.PUD_UP)`
- `GPIO.add_event_detect(channel, GPIO.FALLING)`
-
```
if GPIO.event_detected(channel):
    #Do Something
```

ACTIVITY #3: PUSH BUTTON - DEBOUNCE

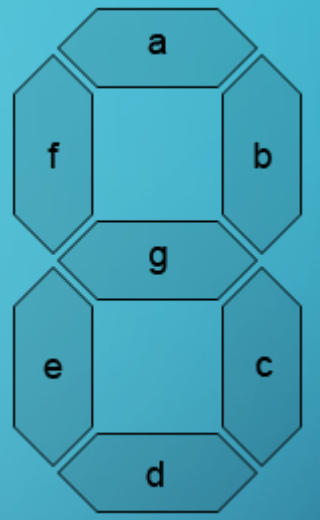
- Oscillation in mechanical switch in button => Multiple button presses
- Logic:
  - Wait for x ms after button pressed and until button is released
  - Only then register it as 1 button press

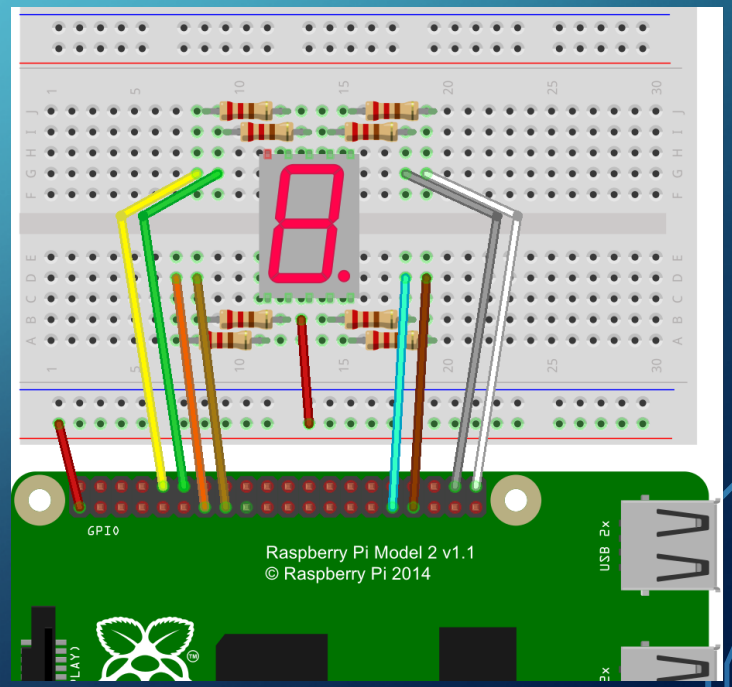
ACTIVITY #3.1: TRY IT YOURSELF

- Use a push button to **toggle** an LED on and off!
  - Press button => LED turns on & remains on
  - Press button again => LED turns off & remains off
- TRY!
- Hint: You are going to need to keep track of the state of the LED (is it current on/off)
- Syntax
  - ```
    if (SOME STATEMENT):
      # Do Something
    ```
    ```

ACTIVITY #4: 7-SEGMENT DRIVER

- Choose 7 I/O Pins on the Raspberry Pi
- Connect 7 resistors to the 7-Segment display
- Connect wires
- Code is the same as controlling an LED, just 7 of them!





ACTIVITY #4: 7-SEGMENT DRIVER

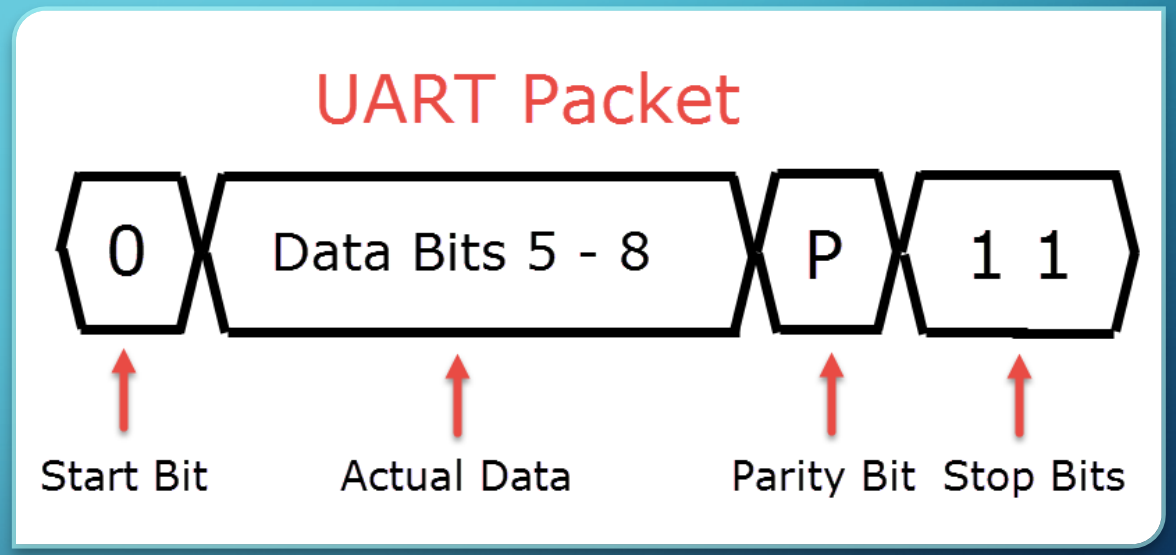
- Creating a function in Python:
  - `def my_function(param1, param2):`
    `## Your function goes here`
- Getting user input:
  - `my_input = input("Write something: ")`

SERIAL COMMUNICATION

- Send data bit by bit, instead of all at once
- Many protocols:
  - UART / USART
  - SPI
  - I2C
  - …

SERIAL COMMUNICATION (UART)

- Universal Asynchronous Receiver-Transmitter
- Star-Bit
- Data Bits
- Parity Bit (Optional)
- Stop Bit/s

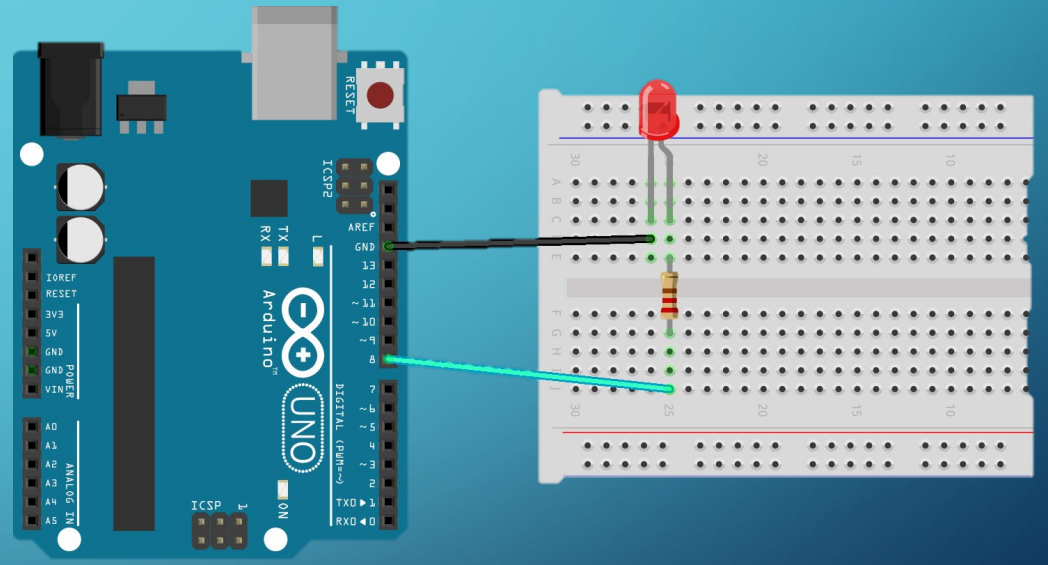


PYTHON SERIAL IN RPI (WITH ARDUINO)

- Why?
  - Offload processing and simple tasks to Arduino
  - Add more input/ output pins
  - Connect to other serial peripherals
- How?
  - Connect a USB cable from the Pi to the Arduino

ON ARDUINO

```
String inString = "";
void serialEvent() {
      char c = ' ';
      c = Serial.read();
      if (c == '\n') {
            // Do something
            inString = "";
      } else {
            inString += c;
      }
}
```



SPI AND I2C

- RPi has:
  - 3 SPI Bus's (Only one accessible via the headers)
  - 2 I2C Bus's accessible through the headers
  - I think

THE END