Workshop Basic Raspberry Class 1 – Raspberry Fundamentals

MSc. David Velásquez Rendón

UNIVERSIDAD EAFIT ®

Contents

Raspberry

¿What is?

- Embebbed single board computer (SBC).
- Allows creation of computational and basic electronic projects.
- Allows installation of different operative systems (based on Linux).
- Can be connected to an Arduino to expand his I/O Peripherals.
- Used mostly for computation and processing (multi-core ARM processor).
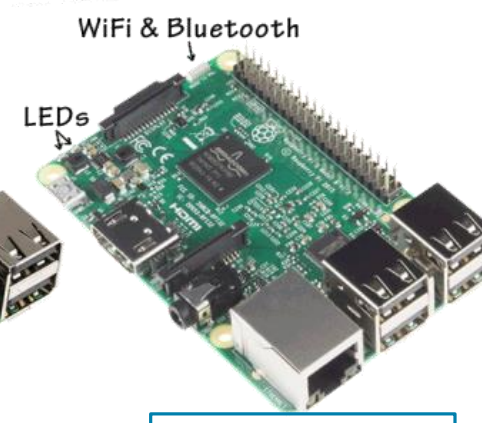- Commonly used for Python Learning.
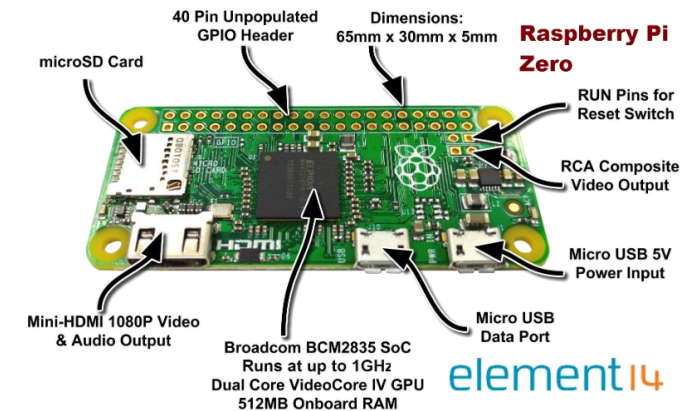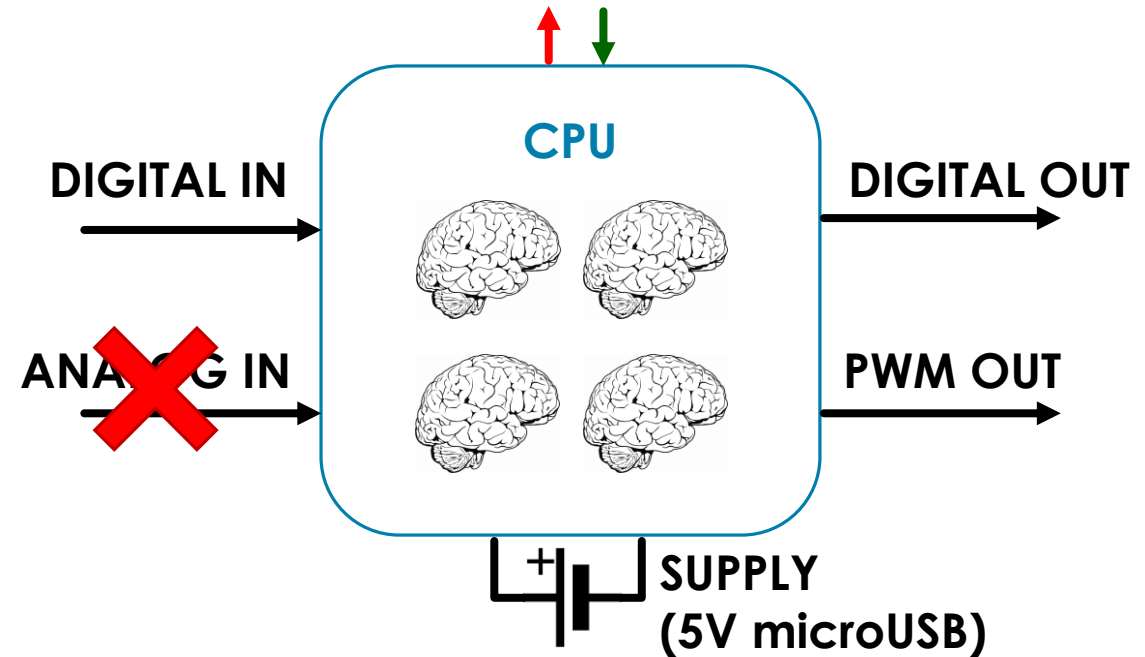
Raspberry Types

COMMUNICATION & OTHER PERIPHERALS

DIGITAL IN

CPU

DIGITAL OUT

ANALOG IN

PWM OUT

**SUPPLY (5V microUSB)**

Raspberry Pi B+
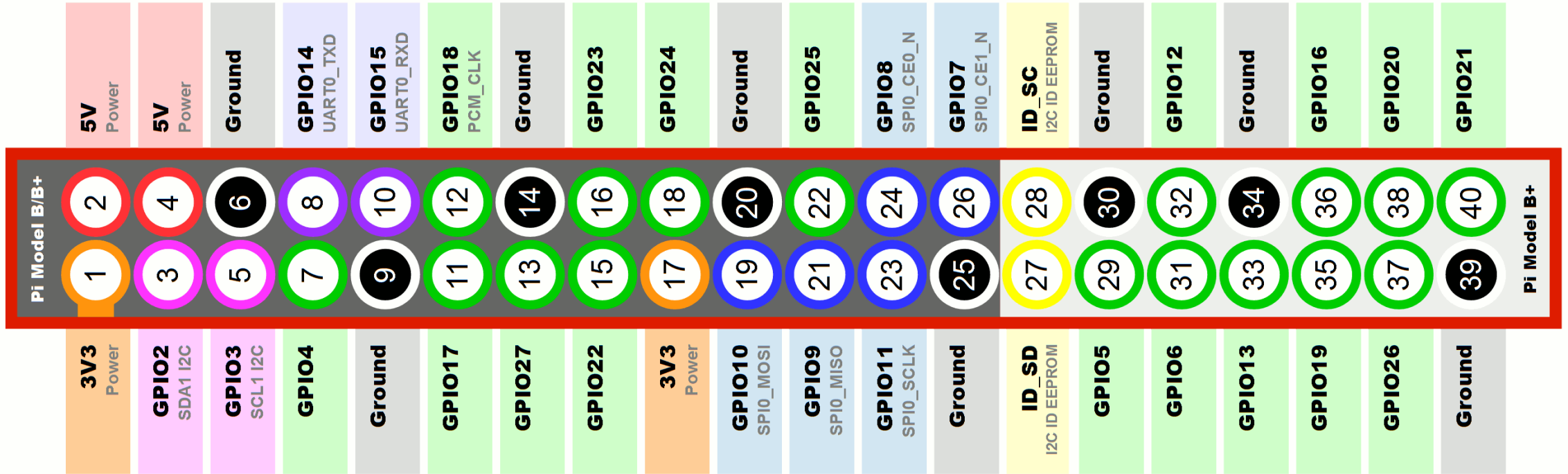

Raspberry Pi 2

Raspberry Pi 3
Model B

Raspberry Zero

Rasberry Pi 3 Model B Specs

- CPU **1.2GHz 64-bit quad-core ARMv8 64 bits**.

- **Supply:** Using micro USB with **5V**.

- On board **Bluetooth 4.1**, **Wi-Fi 802.11n** & **10/100 LAN**.

- **4 x USB 2.0** Ports.

- **26 x General Purpose Input Output** (GPIO) pins at **3.3V**.

- **1 x Full size HDMI** Video Output.

- **1 x 3.5 mm Audio Output** Jack.

- **1 x CSI Camera** Port.

- **Maximum current per I/O digital pin:** 16 mA (50 mA max for all connected GPIO).

- **GPU:** Broadcom VideoCore IV, OpenGL ES 2.0, MPEG-2 y VC-1, 1080p30 H.264/MPEG-4 AVC.

- **SDRAM:** 1 GB (shared with GPU).

- **microSD Card Slot** (min 8GB).

- **Electronic Communications:** Serial UART + I2C + SPI.
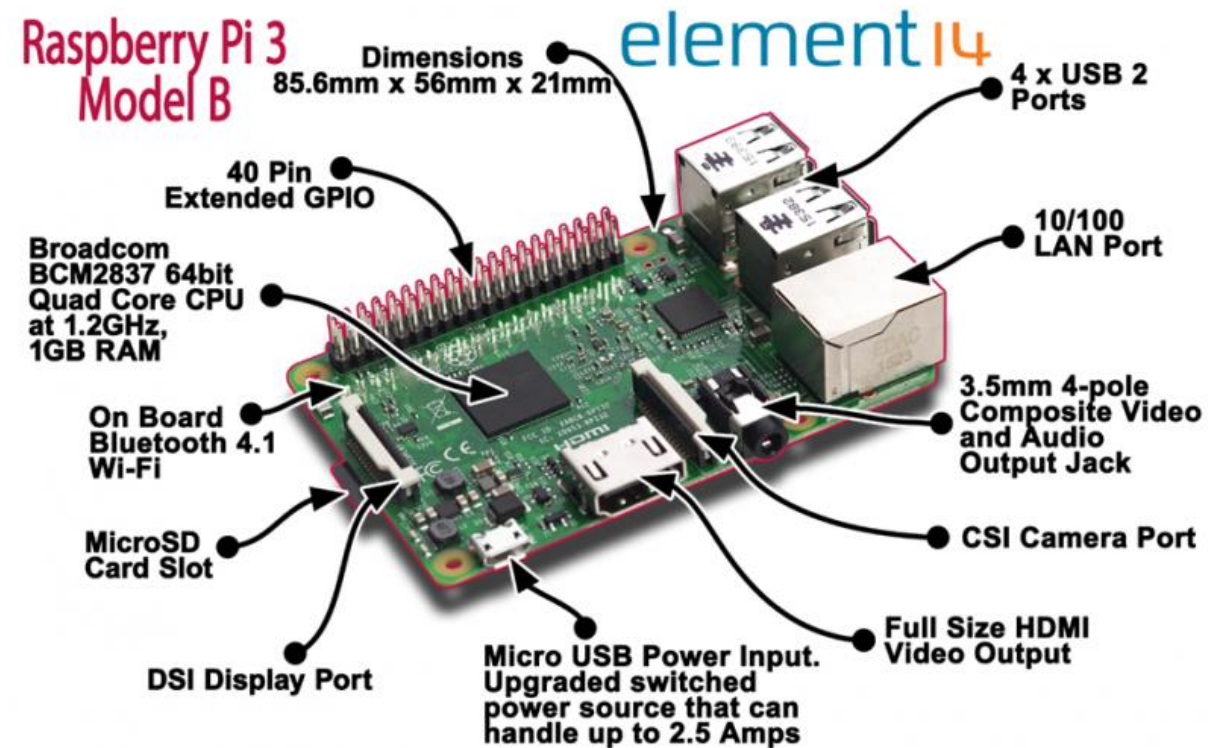
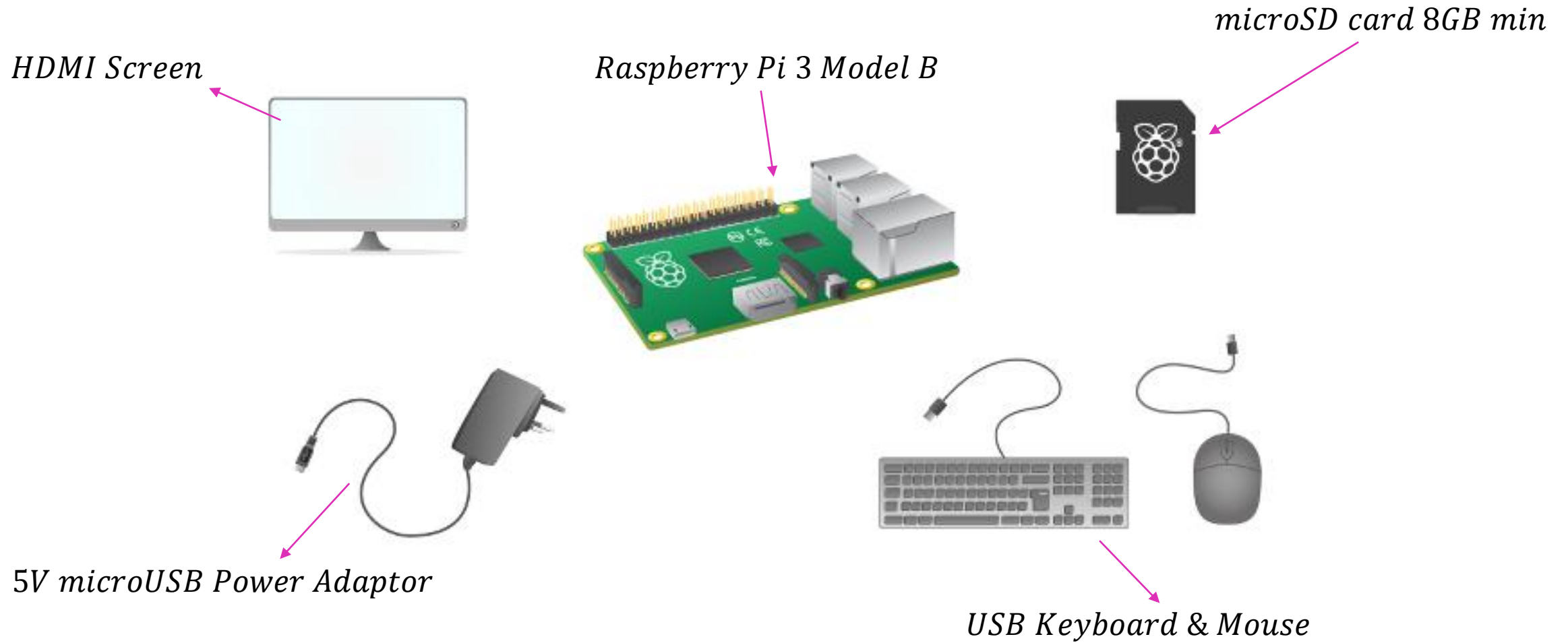- **26 x GPIO** at 3.3V.
- **2 x 3.3V Supply** Pins.
- **2 x 5V Supply** Pins.
- **GPIO7..GPIO11: Serial Peripheral Interface (SPI)** Pins.
- **GPIO14, GPIO15: Serial (UART)** Pins.
- **GPIO2, GPIO3: I2C** Pins.
- **ID_SD, ID_SC: I2C ID EEPROM**.

Raspberry Pi 3 Required Equipment

microSD card 8GB min

HDMI Screen

Raspberry Pi 3 Model B

5V microUSB Power Adaptor

USB Keyboard & Mouse

➡ Most common OS is based in **linux** distribution **Debian** (**Raspbian**).

➡ To install it on the microSD download New Out Of the Box Software (**NOOBS**)
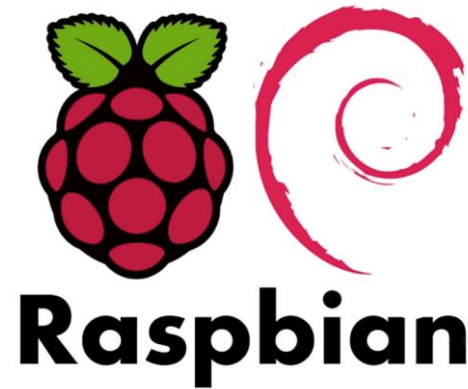
Raspbian

➡ There are other third party OS for Raspberry:

UbuntuMATE

Windows 10 IoT Core

OSMC

recallbox

➡ Ubuntu based custom image

➡ Windows 10 OS for IoT

➡ Open Source Media Centre

➡ Retro-console emulator

Raspbian Basic CLI Commands

- Raspbian is compatible with all linux terminal commands. Following are typical commands:

```
pwd: Prints the current directory
mkdir FOLDERNAME: Creates a new folder
cd FOLDER: Goes to specified directory
ls: Shows all the files in the current directory
lsusb: Shows all the USB connected devices
sudo COMMAND: Executes a command as Admin
sudo shutdown -h: Shutdowns the Raspberry
sudo shutdown -r now: Restarts the Raspberry
sudo raspi-config: Access the configuration
sudo apt-get update: Updates all packages from repo
sudo apt install arduino: Installs Arduino IDE
sudo apt install python-serial: Installs library to connect python with Serial port
ls /dev/tty*: Lists all the tty ports that can be used (e.g. Arduino connected to raspberry)
sudo apt install python-rpi.gpio: Installs Raspberry GPIO library for python
```

- **import**
  - Imports an external library and associates a name to use its functions.
  - Sintax: **import** `LIBRARY` **as** `OBJNAME`
    - `LIBRARY`: Required library to import
    - `OBJNAME`: Name given to use library functions as class object.

RPi.GPIO Library functions

- **import** `RPi.GPIO` **as** `GPIO`
  - Imports RPi.GPIO and names the library GPIO for using its functions

- `GPIO.`**setmode**`(MODE)`
  - Sets desired pins referencing allocation (using pin # in Raspberry board or the pin label as seen in the slide 5).
    - `MODE`: Selected MODE. This can be **GPIO.BOARD** for the corresponding pin # in the GPIO 40 pins or **GPIO.BCM** for pin label (see slide 5)

- `GPIO.`**setup**`(PORT/PIN, MODE)`
  - Configures the desired GPIO PORT/PIN in the required `MODE (input or output)`
    - `PORT/PIN`: The pin # that will be configured.
    - `MODE`: **GPIO.IN** or **GPIO.OUT**.

- `GPIO.`**output**`(PORT/PIN, VALUE)`
  - Writes a logical state to an output pin: a HIGH logic state (3.3V) or a LOW logic state (0V)
    - `PORT/PIN`: The pin # that will be written
    - `VALUE`: **1** or **0** / **True** or **False**.

- `GPIO.`**input**`(PORT/PIN)`
  - Reads and returns the logic state value of a digital input pin
    - `PORT/PIN`: The input pin # that will be read
    - Returns **1** or **0** depending on the logic state value of the input pin that was read

time Library functions

- **import** `time`
  - Imports time library.

- `time.`**sleep**`(SECS)`
  - Pauses the program execution for a desired time (in seconds)
    - SECS: The number of secs that is desired to pause the program

Python Variables

- Python doesn't require explicit declaration of variables.

- Automatically declares variable type after assignation (=).

- Example:

```python
counter = 80          # An integer assignment
temp    = 27.6        # A floating point
name    = "David"     # A string

print counter
print miles
print name
```

- Output:

```
80
27.6
David
```

Python Typical Operators

	SYMBOL	DESCRIPTION
ARITHMETIC	=	Assignment
	+	Addition
	-	Subtraction
	*	Multiplication
	/	Division
	%	Module
COMPARATIVE	==	**Equal to:** $x == y$ is equivalent to: x is equal to y?
	<>	**Not equal to:** $x <> y$ is equivalent: x is not equal to y?
	<	Less than
	>	Greater than
	<=	Less than or equal to
	>=	Greater than or equal to
BOOLEANS	&	AND
	|	OR
	~	Negation (NOT)
ACCUMULATORS	+=	**Addition assignment:** $y += x$ is equivalent to: $y = y + x$
	-=	**Subtraction assignment:** $y -= x$ is equivalent to: $y = y - x$
	*=	**Multiplication assignment:** $y *= x$ is equivalent to: $y = y * x$
	/=	**Division assignment:** $y /= x$ is equivalent to: $y = y/x$

Python Program Structure for HW

Library declaration(e.g: `import` `RPi.GPIO` `as` `GPIO`)
I/O Pin Labeling (e.g: `LEDPIN = 36`)
Constant declaration (e.g: `CONTMAX = 10`)
Variable declaration (e.g: `temperature = 0.0`)

Subroutines or functions declaration:
Example for subroutine:
```python
def hello():                     #Example of a subroutine that prints "Hello" in the console output
    print("Hello World")         #Prints "Hello World"


Example for function:
def sum(x, y):                   #Example of a function that sums two numbers "x" y "y" and returns the result
    return x + y
```
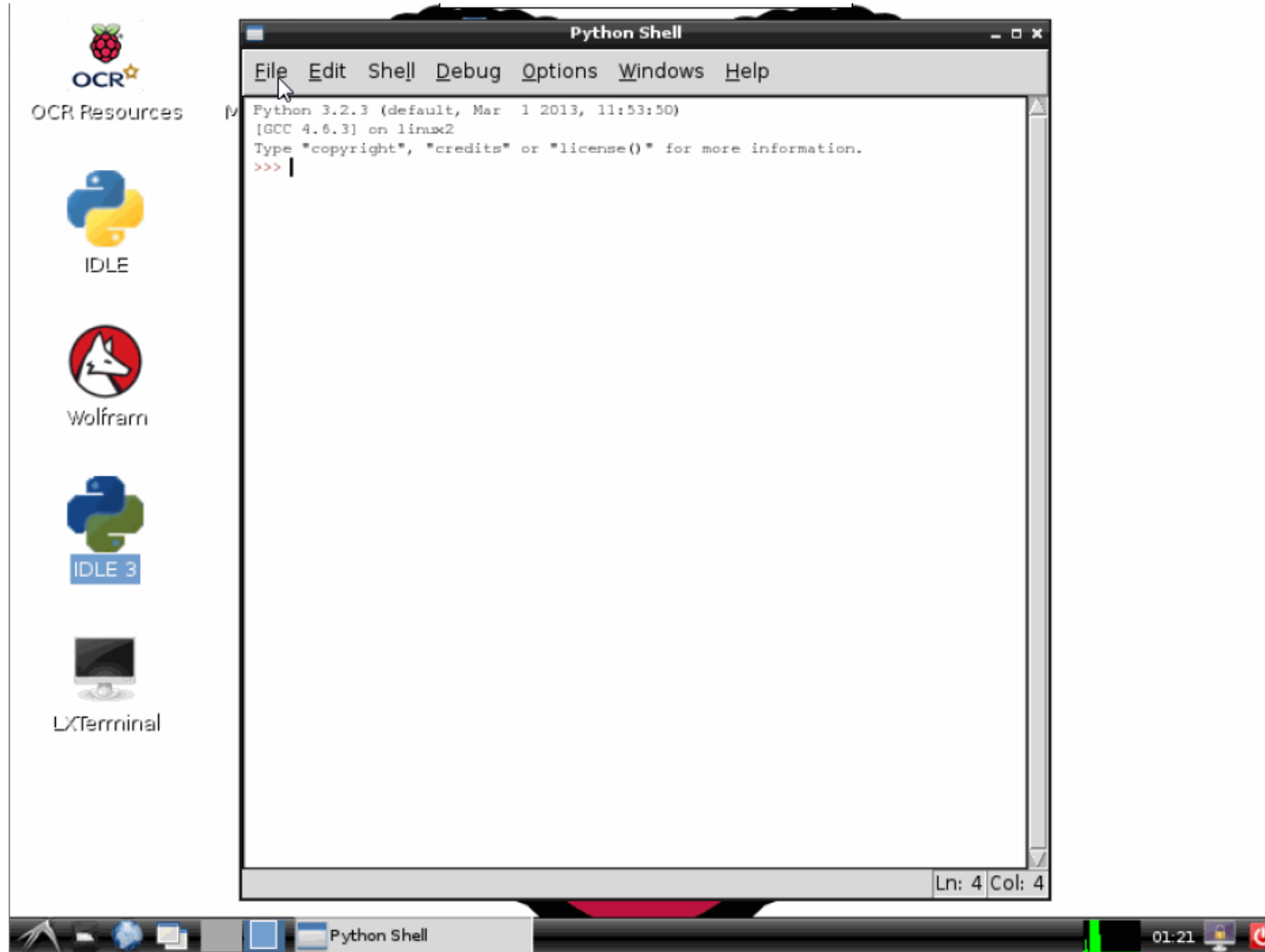
Pin configuration and cleaning:
```python
#SETUP
#CONFIGURATION: Indicate which pins are inputs and which are outputs
#->setmode and setup functions must be used for this part
#CLEANING: For safety, it is important to clean used outputs with the purpose that they are turned off at the beginning
of the program. Use the function GPIO.output(PIN,False).
#COMMUNICATIONS: For example, for communications with Arduino, import Serial library at Library declaration and use the
function ser = serial.Serial("/dev/ttyACM0", 9600) to begin this communications.
```

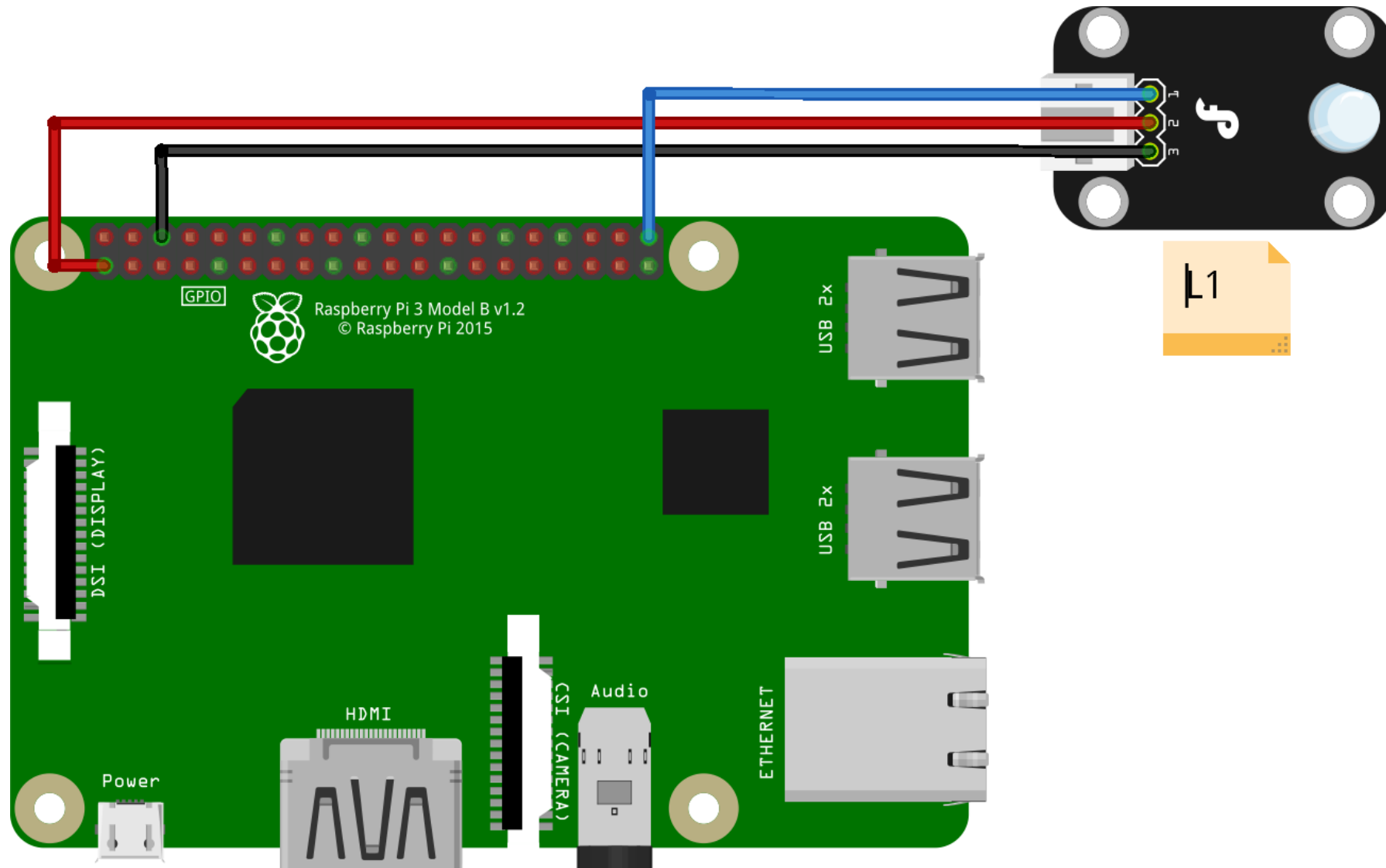**Infinite loop (Main program - Execution)**:
```python
#EXECUTION
while True:
    #Main program
```

https://www.codeproject.com/KB/boards-embedded-devices/850842/Image7.gif

- **Example: In PIN 40 (GPIO21) there is a LED (L1) connected. Blink the LED ½ second ON and ½ second OFF.**

L1

```python
#Library declaration
import RPi.GPIO as GPIO
import time

#I/O pin labeling
L1 = 40 #Label LED connected in pin 40 as "L1"

#Constant declaration
TBLINK = 0.5 #Blink constant TBLINK initialized on 0.5s

#SETUP
#I/O Pin Configuration
GPIO.setmode(GPIO.BOARD) #Configures all pins reference using pin #
GPIO.setup(L1, GPIO.OUT) #Set pin L1 as Output
#Output cleaning
GPIO.output(L1,0) #Turn OFF L1 (also posible GPIO.output(L1,False))

#EXECUTION
while True:
    GPIO.output(L1,1) #Turn ON L1
    time.sleep(TBLINK); #Delay of TBLINK secs(0.5s)
    GPIO.output(L1,0) #Turn OFF L1
    time.sleep(TBLINK); #Delay of TBLINK secs(0.5s)
```
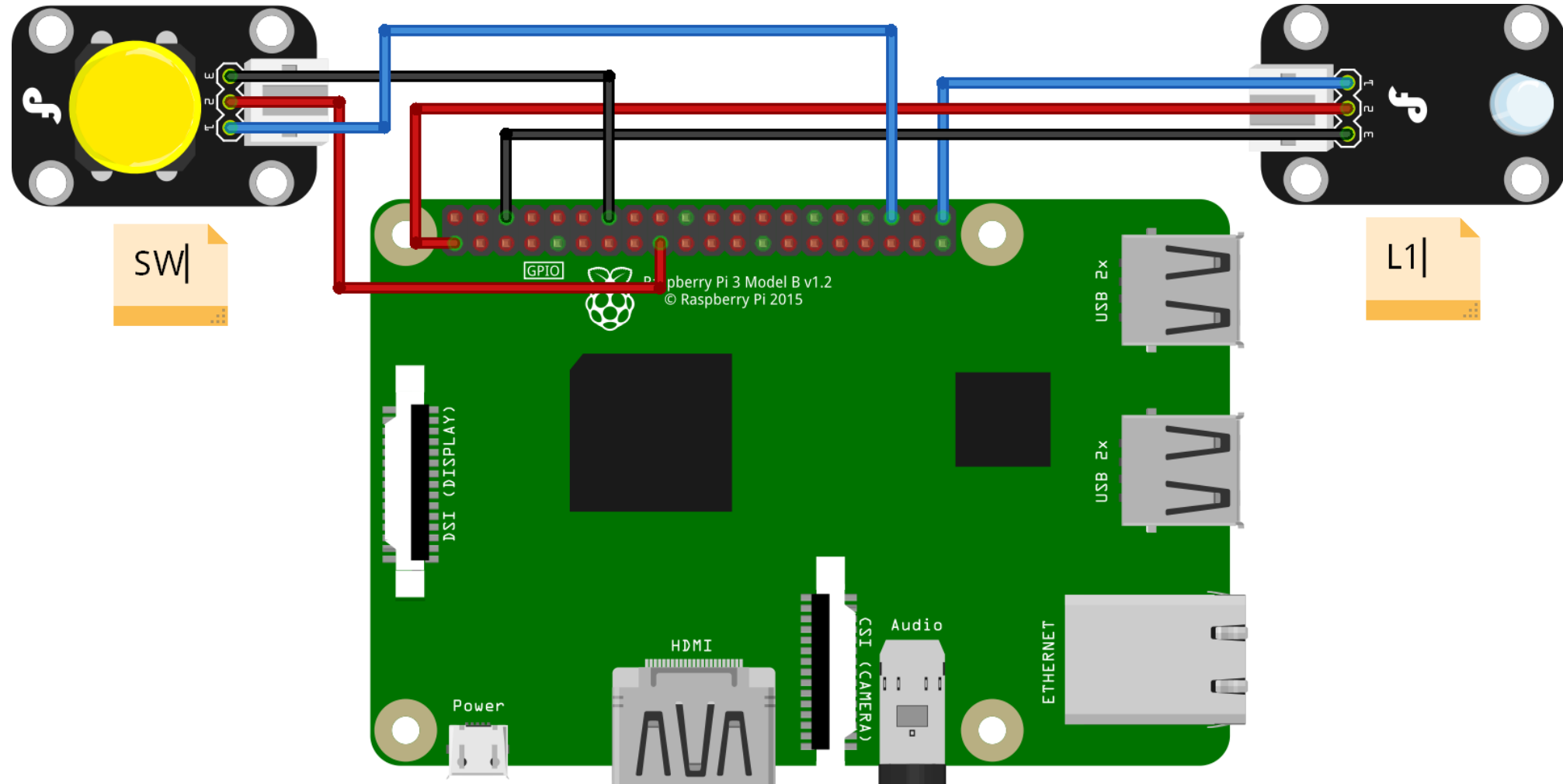
For java, use the following example: http://pi4j.com/example/control.html

Example 1.2 – If statement with digital input

- Example: In the PIN 36 (GPIO16) there is a switch (SW) and in the PIN 40 there is a LED (L1). Turn ON the LED if the switch is activated, in other case, turn off the LED

Example 1.2 – If statement with digital input

```python
#Library declaration
import RPi.GPIO as GPIO

#I/O pin labeling
L1 = 40 #Label LED connected in pin 40 as "L1"
SW = 36 #Label Switch connected in pin 36 as "SW"

#SETUP
#I/O Pin Configuration
GPIO.setmode(GPIO.BOARD) #Configures all pins reference using pin #
GPIO.setup(L1, GPIO.OUT) #Set pin L1 as Output
GPIO.setup(SW, GPIO.IN) #Set pin SW as Output

#Output cleaning
GPIO.output(L1,0) #Turn OFF L1 (also posible GPIO.output(L1,False))

#EXECUTION
while True:
    if GPIO.input(SW) == 1:
        GPIO.output(L1,1) #Turn ON L1
    else:
        GPIO.output(L1,0) #Turn OFF L1
```

Thanks!