

Module 5

IoT Physical Devices and Endpoints:

RaspberryPi:

Introduction to RaspberryPi

- ▶ The Raspberrypi is a series of credit card sized single-board computers developed in the United Kingdom by the Raspberrypi Foundation.
- ▶ To promote the teaching of basic computer science in schools and developing countries.
- ▶ Accessories such as keyboard and mouse are not included



Several generations of Raspberrypi have been released.

- ▶ The first generation (Raspberrypi 1 Model B) was released in February 2012, followed by a simpler and inexpensive model A.
- ▶ In 2014, the foundation released a board with an improved design in Raspberry 1 Model B+.
- ▶ The Raspberrypi 2 which added more RAM was released in February 2015.
- ▶ Raspberrypi 3 model B released in February 2016 is bundled with on-board Wi-Fi and Bluetooth.
- ▶ **Raspberry Pi 4 Model B** was released in June 2019 with a 1.5 GHz 64-bit quad core [ARM Cortex-A72](#) processor, on-board 802.11ac [Wi-Fi](#), [Bluetooth 5](#), full [gigabit Ethernet](#)



-
- ▶ All models feature a Broadcom system on a chip (SoC), which includes an ARM compatible central processing unit (CPU) and an on chip graphics processing unit (GPU, a Video Core IV).
 - ▶ CPU speed ranges from 700 MHz to 1.5GHz for the Pi 4
 - ▶ On board memory range from 256MB to 1GB RAM.
 - ▶ Secure Digital (SD) cards are used to store the operating system and program memory in either the SDHC or Micro SDHC sizes.
 - ▶ Most boards have between one and four USB slots, HDMI and composite video output, and a 3.5 mm phone jack for audio.
 - ▶ Lower level output is provided by a number of GPIO pins which support common protocols like I2C.
 - ▶ The B-models have an Ethernet port and the Pi 3 has on board Wi-Fi 802.11n and Bluetooth.
-



-
- ▶ The Foundation provides Raspbian, a Debian-based Linux distribution for download, as well as third party ubuntu, windows 10 IoT core, RISC OS, and specialized media center distributions.
 - ▶ Python and Scratch are the main programming language, with support for many other languages.



Why RaspberryPi?

- ▶ Inexpensive, Cross-platform, Simple, clear programming environment, Open source and extensible software.



Table 8-1: Technical Specification of Raspberry Pi models

RaspberryPi	Model A+	Model B	Model B+	2, Model B	Model 3
Quick Summary	Cheapest, smallest single board computer	The original Raspberry Pi.	More USB and GPIO than the B. Ideal choice for schools	most advanced Raspberry Pi.	Newest with wireless connectivity
Chip	Broadcom BCM 2835			Broadcom BCM2836	Broadcom BCM 2837
processor	ARMv6 single core			ARMv7 quad core	4×ARM Cortex-A53
Processor speed	700 MHz			900 MHz	1.2GHz
Voltage and power draw	600mA @ 5V			650mA @ 5V	
GPU	Dual core Videocore IV Multimedia Co-Processor				Broadcom Videocone IV
Size	65×56mm	85×56mm			
Memory	256 MB SDRAM @ 400 MHz	512 MB SDRAM @ 400 MHz		1 GB SDRAM @ 400 MHz	1 GB LPDDR2 (900 MHz)

Table 8-1: Technical Specification of Raspberry Pi models

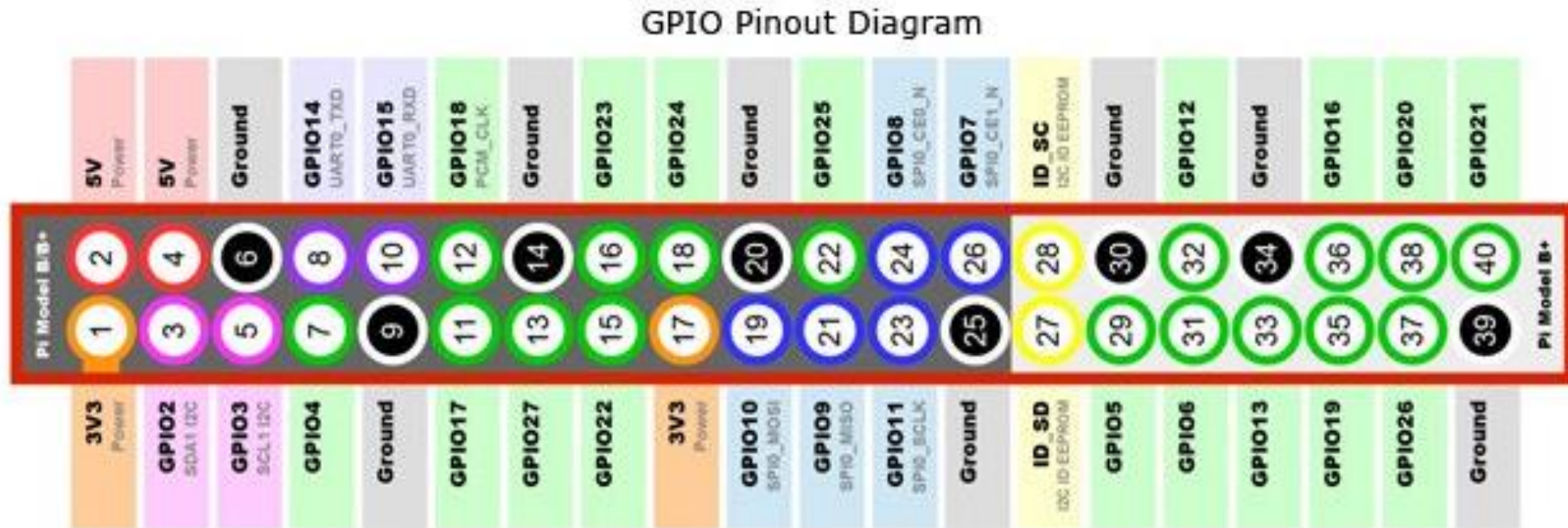
RaspberryPi	Model A+	Model B	Model B+	2, Model B	Model 3
Storage	Micro SD Card	SD Card	Micro SD Card	Micro SD Card	Micro SD Card
GPIO	40	26	40 4		
USB 2.0	1	2			
Ethernet	None	10/100mb Ethernet RJ45J ack			
Wireless	None				2.4GHz 802.11n wireless
Bluetooth	None				Bluetooth 4.1 Classic, Bluetooth Low Energy
Audio	Multi-Channel HD Audio over HDMI, Analog Stereo from 3.5mm Headphone Jack				

Operating Systems	Raspbian RaspBMC, Arch Linux, Rise OS, OpenEL EC Pidora
Video Output	HDMI Composite RCA
Supported Resolutions	640×350 to 1920×1200, including 1080p, PAL & NTSC standards
Power Source	Micro USB



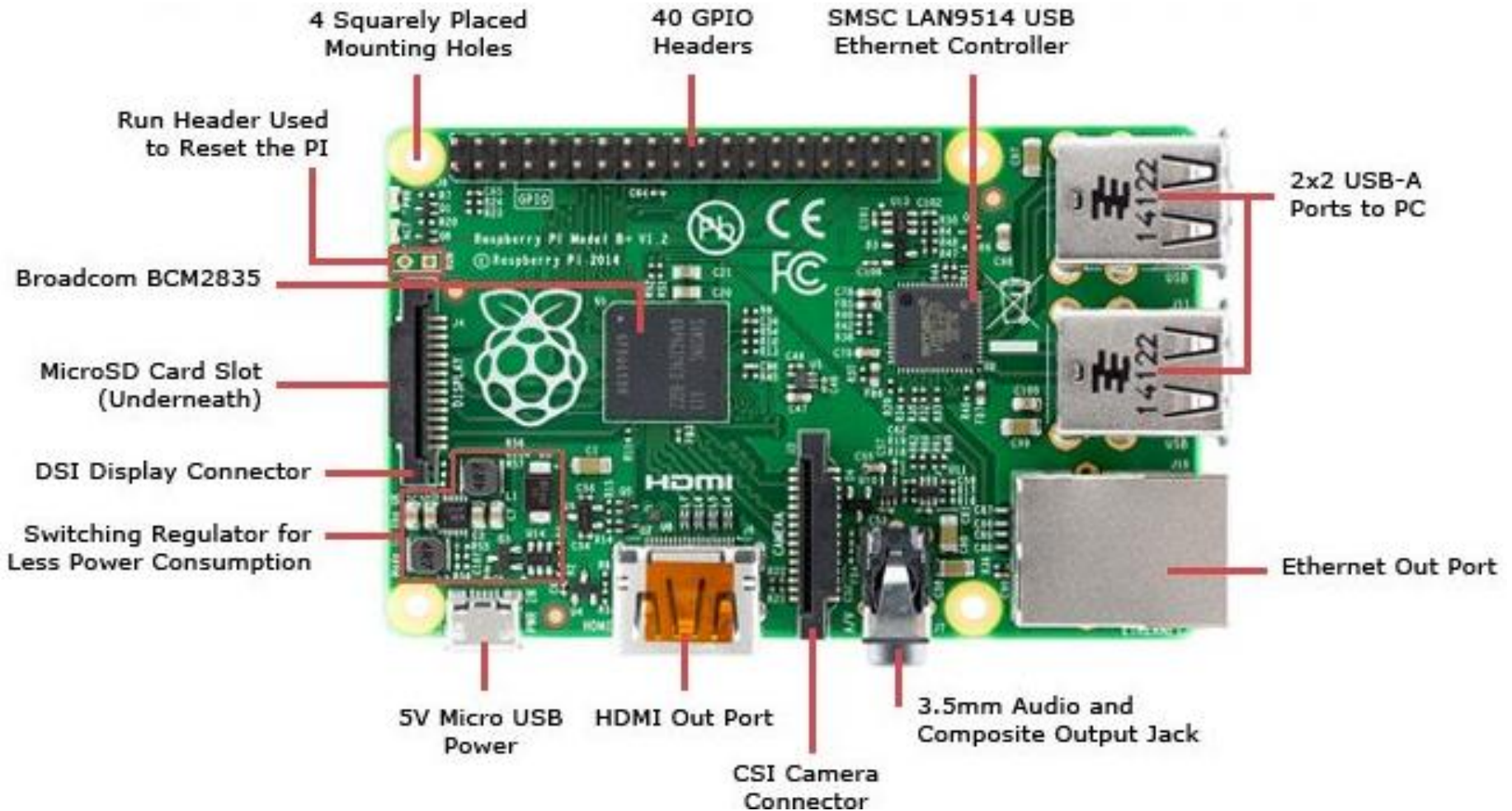
Exploring the Raspberrypi Learning Board

Raspberry Pi2 Model B and its GPIO



<https://www.jameco.com/Jameco/workshop/circuitnotes/raspberry-pi-circuit-note.html>

Raspberry Pi2 Model B and its GPIO



Exploring the Raspberrypi Learning Board

- ▶ Processor
- ▶ Power Source
- ▶ SD card
- ▶ GPIO(General Purpose Input and Output)
- ▶ DSI Display X
- ▶ Audio Jack
- ▶ Status LEDs
- ▶ Ethernet Port
- ▶ CSI Connector
- ▶ JTAG Headers
- ▶ HDMI



Processor

- ▶ The Broadcom **BCM2835** SoC used in the **first generation** Raspberrypi, which includes a 700 MHz ARM 1176JZF-S processor, Video Core IV graphics processing unit (GPU) and RAM.
- ▶ This has a level 1 (L1) cache of 16KB and a level 2 (L2) cache of 128 KB. The level 2 cache is used primarily by the GPU.
- ▶ The **Raspberrypi 2** uses a Broadcom **BCM2836** SoC with a 900 MHz 32-bit quad-core ARM cortex A7 processor (as do many current smart phones), with 256KB shared L2 cache.
- ▶ The **Raspberrypi3** uses a Broadcom **BCM2837** SoC with a 1.2GHz 64-bit quad core ARM Cortex A53 processor, with a 512KB shared L2 cache.



Power Source

- ▶ The recommended and easiest way to power the Raspberrypi is via the Micro USB port.
- ▶ The recommended input voltage is 5V, and the recommended input current is 2A.
- ▶ Can function on lower current power supplies e.g. 5V @ 1A.
- ▶ Any excessive use of the USB ports or even heavy CPU/GPU loading can cause the voltage to drop, and instability during use.
- ▶ The latest versions of the Raspberrypi B+/A+/2 have a “low voltage indicator icon” to notify the user if there is a problem with the power.



SD Card

- ▶ The Raspberry Pi does not have built in OS and storage. You can plug in an SD card with a linux image with SD card slot. Require atleast an 8GB SD card for New Out-of-the-Box Software.(NOOBS).

GPIO(General Purpose Input Output)

- ▶ There are four types of pin on Raspberry Pi- true GPIO pins, I2C interface pins, SPI interface pins and serial Rx and Tx pins.
- ▶ can be controlled by the client at run time.

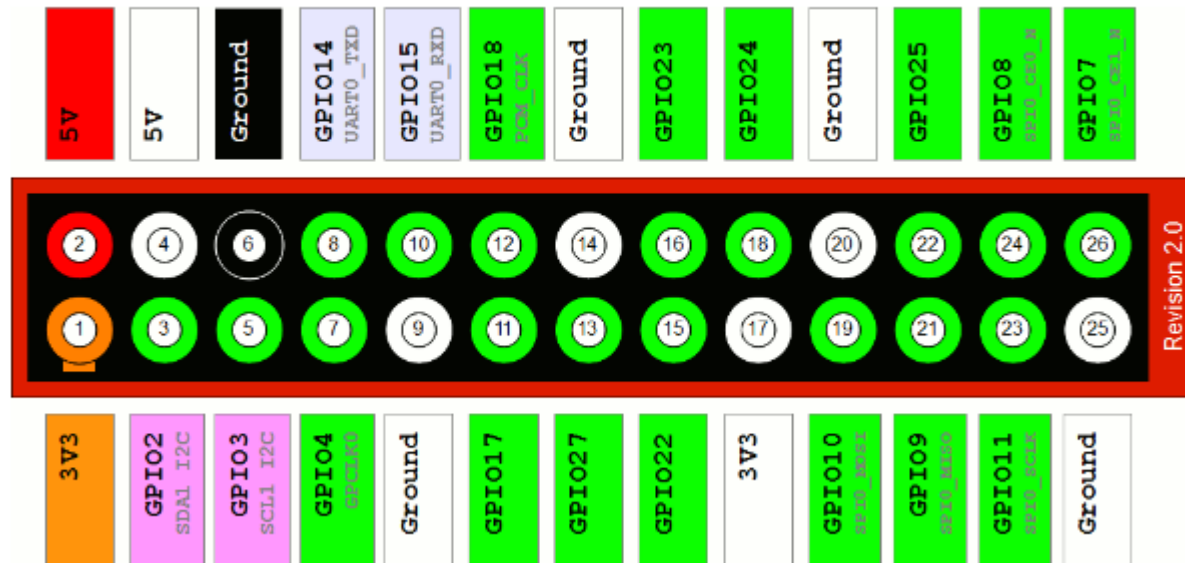
GPIO capabilities may include:

- ▶ GPIO pins can be designed to be input or output
- ▶ GPIO pins can be empowered/crippled
- ▶ Input values are meaningful (normally high=1, low=0).
- ▶ output values are writable/meaningful
- ▶ Input values can frequently be utilized as IRQs.



- ▶ The **GPIO.BOARD** option specifies that you are referring to the pins by the number of the pin the plug - i.e the numbers printed on the board (e.g. P1) and in the middle of the diagrams below.
- ▶ The **GPIO.BCM** option means that you are referring to the pins by the "Broadcom SOC channel" number, these are the numbers after "GPIO" in the green rectangles around the outside

Pi 1 Model B Revision 2.0:



DSI Display X

- ▶ Raspberrypi Connector S2 is a display serial interface (DSI) for connecting a liquid crystal display (LCD) panel using a 15-pin ribbon cable.
- ▶ The mobile industry processor interface (MIPI) inside the Broadcom BCM2835 IC feeds graphics data directly to the display panel through this connector.

Audio Jack

- ▶ A standard 3.5 mm TRS connector is accessible on the RPi for stereo sound.
- ▶ Any earphone or 3.5mm sound link can be associated straightforwardly.
- ▶ USB mics or USB sound cards can be utilized.



Status LED

- ▶ ACT- SD card Access
- ▶ PWR- 3.3 V power is present
- ▶ FDX- Full duplex LAN connected
- ▶ LNK –Link/Network Activity
- ▶ 100- 100Mbit LAN connected



Status LEDs

- ▶ There are 5 status LEDs on the RPi that demonstrate the status:
 - ▶ — OK - SD Card Access (by means of GPIO 16) - named as “OK” on Model B Rev 1.0 sheets and “ACT” on Model B Rev 2.0 and Model A sheets .
 - ▶ — POWER - 3.3 V Power - named as “PWR” on all the boards
 - ▶ — FDX - Full Duplex (LAN) (Model B) - marked as “FDX” on all the boards
 - ▶ — LNK - Link/Activity (LAN) (Model B) - marked as “LNK” on all the boards
 - ▶ — 10M/100 - 10/100Mbit (LAN) (Model B) - named as “10M” on Model B Rev 1.0 boards and “100” on Model B Rev 2.0 and Model A boards USB Ports.
- ▶ There is 1 port on Model A, 2 on Model B and 4 on Model B+ operates at a current upto 100mA.
- ▶ An external USB powered hub is required to draw current more than 100mA.



Ethernet port:

- ▶ Ethernet port is accessible on Model B and B+. Comes with RJ45 Ethernet ports. You can connect to ethernet port or WiFi to provide internet connectivity. The Ethernet ports are controlled by Microchip LAN9512 LAN controller chip.

CSI connector (CSI)

- ▶ Camera Serial Interface is a serial interface outlined by MIPI (Mobile Industry Processor Interface) organization.
- ▶ CSI interface can be used to connect Camera module to Raspberry Pi.



JTAG headers: (Joint Test Action Group)

- ▶ is an industry standard for verifying designs and testing printed circuit boards after manufacture.

HDMI:

- ▶ HDMI port provides both video and audio output. You can connect the Raspberry Pi to a monitor using an HDMI cable



Description of System on Chip (SoC)

- ▶ An integrated circuit (also known as a "chip") that integrates all or most components of a computer or other electronic system.
- ▶ These components almost always include a central processing unit (CPU), memory, input/output ports and secondary storage – all on a single substrate or microchip, the size of a coin
- ▶ SoCs are exceptionally regular in the portable gadgets because of low power utilization.



An SoC comprises of:

- ▶ Processor cores can be a microcontroller, microprocessor digital signal processor (DSP) or application-specific instruction set processor (ASIP). Multiprocessor SoCs have more than one processor core .
- ▶ SoC include choice of ROM, RAM, EEPROM and flash memory.
- ▶ SoCs require timing sources to generate clock signals,
- ▶ System-on-chip peripherals including counter-timers, real-time timers and power-on reset generators.
- ▶ External interfaces-for example, USB, FireWire, Ethernet, USART, SPI.
- ▶ Simple interfaces including ADCs and DACs.
- ▶ SoCs also include voltage regulators and power management circuits.



Accessories:

- ▶ Camera
- ▶ Gertboard
- ▶ USB Hub



Camera:

- ▶ The Raspberry Pi camera board contains a 5 Mpixel (v1) & 8Mpixel(v2) sensor, and interfaces through a strip link to the CSI connector on the Raspberry Pi.
- ▶ In Raspbian, the user must enable the use of the camera board by running Raspi-config and selecting the camera option.

Gertboard

- ▶ A Raspberry Pi Foundation sanctioned device, designed for educational purposes, that expands the Raspberry Pi's GPIO pins to allow interface with and control of LEDs, switches, analogue signals, sensors and other devices.



USB Hub

- ▶ Suggested extra for the Pi.
- ▶ A fueled USB Hub with 7 additional ports is accessible at all online stores.
- ▶ It is mandatory to utilize a USB Hub to associate outer hard plates or different adornments that draw control from the USB ports, as the Pi can't offer energy to them.



Raspberry Pi interfaces

- ▶ Raspberry Pi has serial, SPI and I2C interfaces

Serial:

- ▶ The serial interface on Raspberry Pi has receive(rx) and transmit(Tx) pins for communication with serial peripherals.

SPI: Serial Peripheral Interface(SPI)

- ▶ Synchronous serial data protocol used for communicating with one or more peripheral devices. In an SPI connection, there is one master device and one or more peripheral devices.
 - ▶ There are five pins on Raspberry Pi for SPI interface:
 - ▶ MISO(Master In Slave Out): Master line for sending data to the peripherals.
 - ▶ MOSI(Master out Slave In): Slave line for sending data to the master.
 - ▶ SCK(Serial Clock): Clock generated by master to synchronize data transmissions.
 - ▶ CE0(Chip Enable 0):To enable or disable devices.
 - ▶ CE1(Chip Enable 1):To enable or disable devices
-



I2C (Inter-Integrated Circuit)

- ▶ I2C interface pins on Raspberry Pi allow you to connect hardware modules. I2C interface allows synchronous data transfer with just two pins-SDA(data line) and SCL(clock line).
- ▶ The two lines are called SCL and SDA. SCL is the clock line for synchronizing transmission. SDA is the data line through which bits of data are sent or received.



Raspberrypi Operating Systems

Various operating systems can be installed on Raspberrypi through SD cards.

- ▶ Raspbian, a Debian-based Linux operating system.

Other third party operating systems

- ▶ Ubuntu MATE,
- ▶ Snappy Ubuntu Core,
- ▶ Windows 10 IoT Core,
- ▶ RISC OS
- ▶ specialized distributions for the Kodi media center and class room management

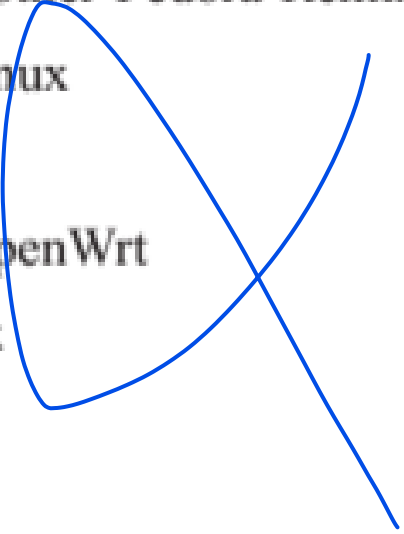


Operating Systems (not Linux based):

- ▶ RISC OS Pi
- ▶ FreeBSD
- ▶ NetBSD
- ▶ Plan 9 from Bell Labs and Inferno
- ▶ Windows 10 IoT core- smallest edition of Windows 10 offered by Microsoft that runs natively on the RaspberryPi 2.
- ▶ xv6-is a modern reimplementation of sixth edition Unix OS for teaching purposes, it is ported to RaspberryPi from MIT Xv6, which can boot NOOBs.
- ▶ Haiku-is an open source BeOS clone that has can be compile for the RaspberryPi and several other ARM boards



8.3.2 Operating Systems (Linux based):

- ✓ Xbian-using Kodi open source digital media center
 - ✓ openSUSE
 - ✓ Raspberry Pi Fedora remix
 - ✓ Pidora, another Fedora Remix optimized for the RaspberryPi
 - ✓ Gentoo Linux
 - ✓ Diet Pi
 - ✓ CentOS\OpenWrt
 - ✓ Kali Linux
 - ✓ Ark OS
 - ✓ Kano OS
 - ✓ Nard SDK
- 

8.3.3 Media center operating systems

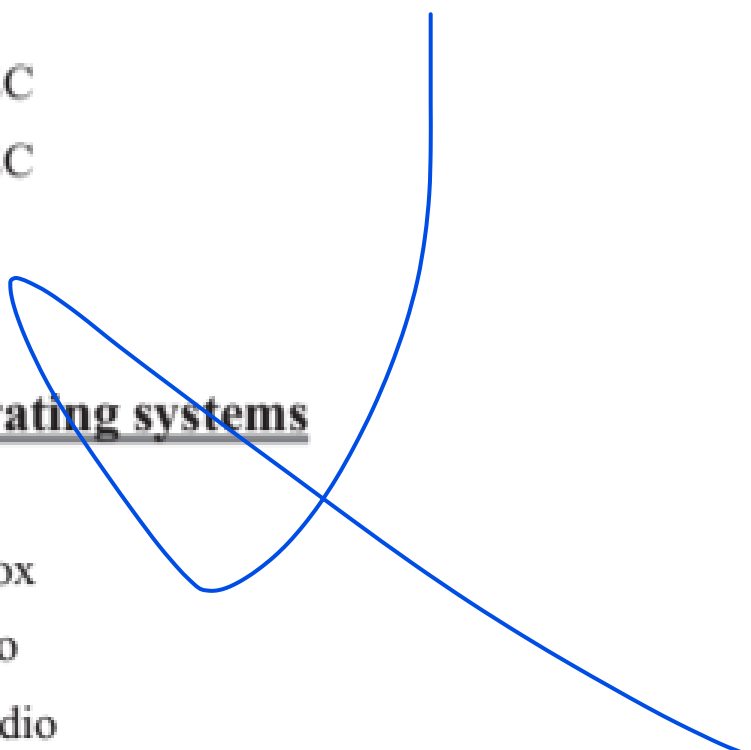
- ✓ OSMC
- ✓ OpenELEC
- ✓ LibreELEC
- ✓ Xbian
- ✓ Rasplex

8.3.4 Audio operating systems

- ✓ Volumio
- ✓ Pimusicbox
- ✓ Runeaudio
- ✓ moOdeaudio

8.3.5 Recalbox

- ✓ Happi Game Center
- ✓ Lakka
- ✓ ChameleonPi
- ✓ Piplay



Operating System Setup on Raspberrypi

- ▶ Preinstalled NOOBS operating system is already available in many authorized as well as independent seller.
- ▶ There are many other operating system for Raspberrypi in the market like NOOBS, Raspbian and third party operating systems are also available like UBUNTU MATE, OSMC, RISC OS etc.
- ▶ To setup an operating system we need a SD card with minimum capacity of 8GB.

Download Raspbian:

- ▶ Download latest Raspbian image from raspberry Pi official website:
- ▶ <https://www.raspberrypi.org/downloads/>



Formatting SD card:

- ▶ Format the SD card before copying NOOBS onto it. To do this
- ▶ 1. Download SD formatter 4.0 from SD Association website for either Windows or Mac.
- ▶ 2. Follow the instructions to install the software.
- ▶ 3. Insert the SD card into the computer or laptops SD card reader and make a note of the drive letter allocated to it
- ▶ 4. In SD formatter, select the drive letter the SD card is and format it.



OS installation:

Follow the steps to install operating system in the SD card.

- ▶ 1. Go to raspberry pi foundation website and click on DOWNLOAD section.
 - ▶ 2. Click on NOOBS, then click on the “Download ZIP” button under NOOBS(offline and network install) and select a folder to save this ZIP file
 - ▶ 3. Extract all the files from ZIP.
 - ▶ 4. Once SD card has been formatted, drag all the files in the extracted NOOBS folder and drop them onto the SD card drive.
 - ▶ 5. The necessary files will then be transferred to the SD card.
 - ▶ 6. When this process has finished, safely remove the SD card and insert it into the Raspberry Pi.
-



First Boot:

- ▶ 1. Plug in the keyboard, mouse, and monitor cables.
- ▶ 2. Now plug the USB cable into the Raspberrypi.
- ▶ 3. Now Raspberrypi will boot, and a window will appear with a list of different operating systems that we can install. We recommend using Raspbian- tick the box next to Raspbian and clicking on install.
- ▶ 4. Raspbian will then run through its installation process.
- ▶ 5. When the install process has completed, the Raspberrypi configuration menu (raspi-config) will load. Here we should set the time and date for our region, enable a Raspberrypi camera board, or even create users.



LOGIN Information

- ▶ The default login for Raspbian is username “pi” with the password “raspberry”.
- ▶ To load the graphical user interface, type “startx” and press “Enter”.





Figure 8-3: Raspbian desktop

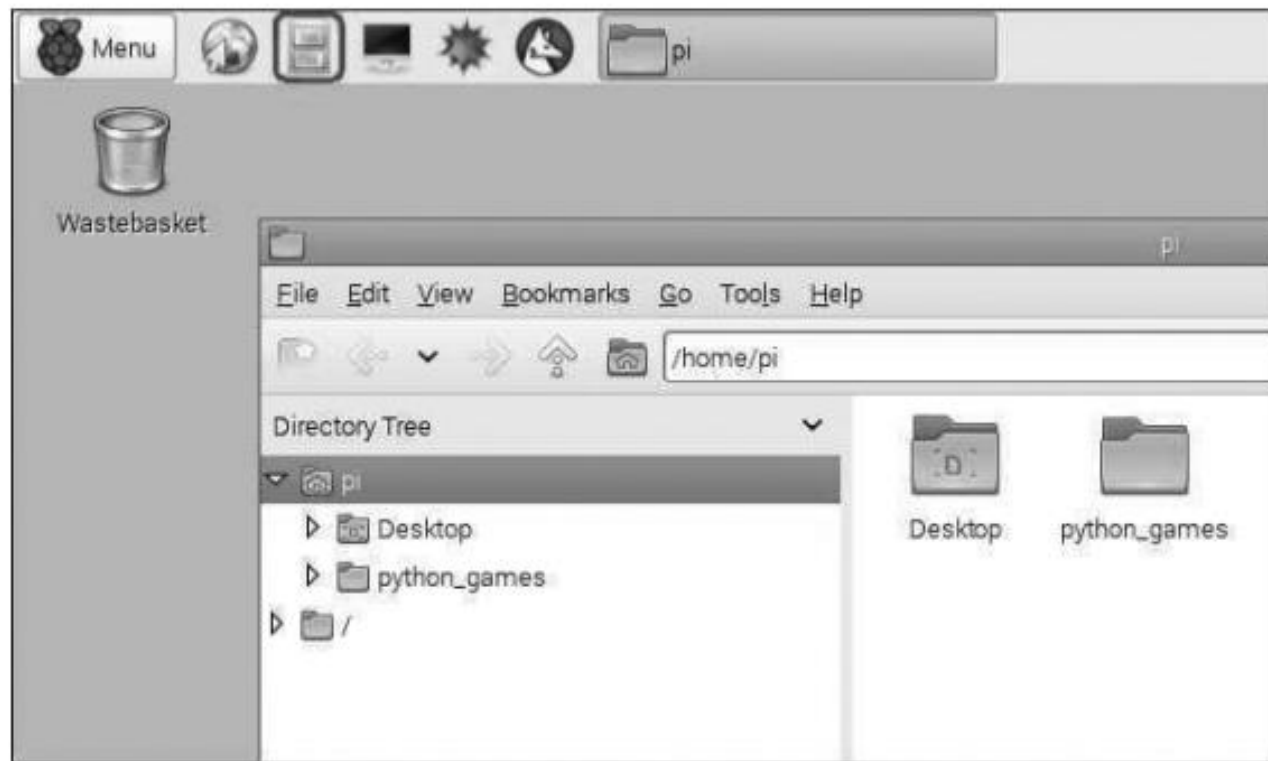


Figure 8-4: File explorer on Raspberry Pi

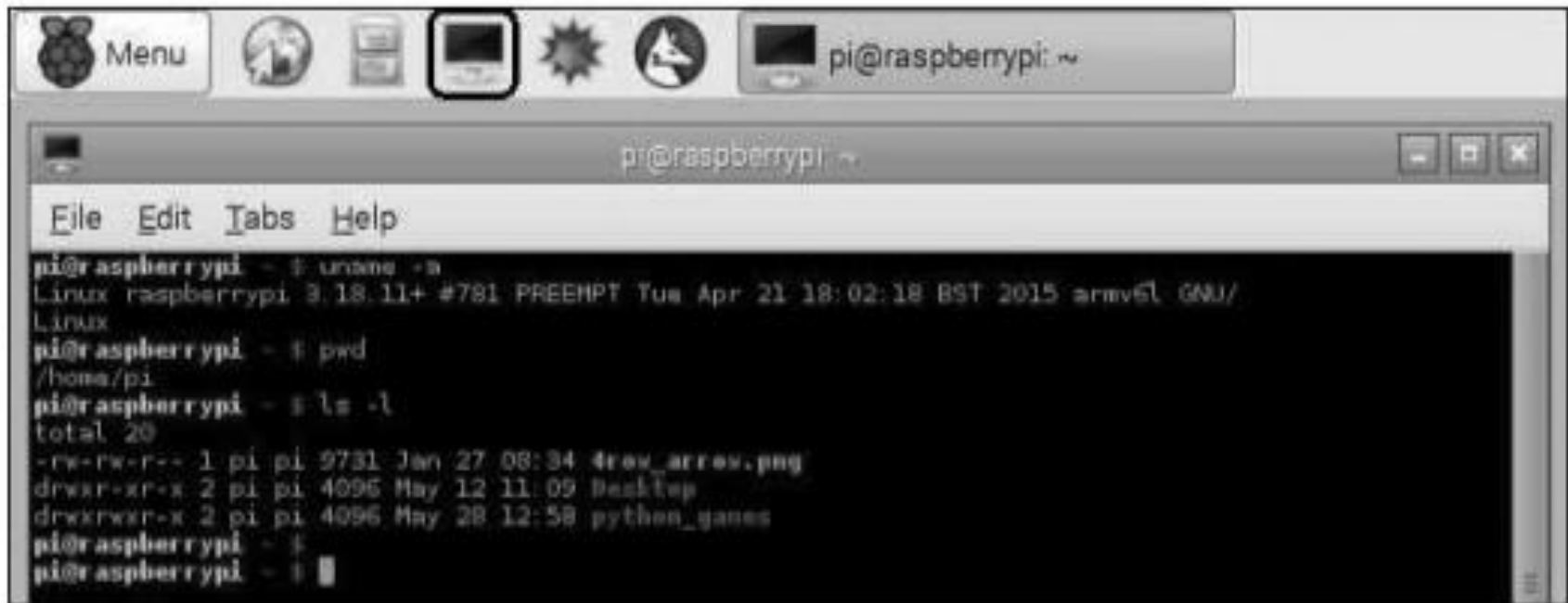


Figure 8-5: Console on RaspberryPi.



Figure 8-6: Browser on RaspberryPi

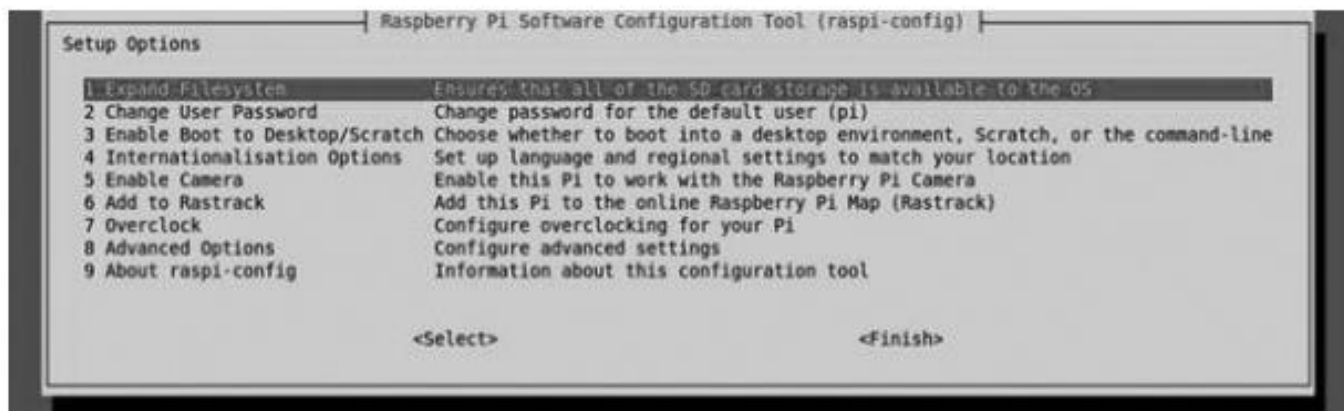


Figure 8-7: RaspberryPi configuration tool

RaspberryPi commands

General commands for RaspberryPi:

- ▶ **raspi-config:** Configuration settings menu
- ▶ **clear:** clears data from terminal.
- ▶ **date:** current date.
- ▶ **reboot:** Reboot immediately
- ▶ **shutdown -h now:** Shutdown system immediately
- ▶ **nano example.txt:** Opens the example.txt in the text editor nano.
- ▶ **poweroff:** To shutdown immediately.
- ▶ **shutdown -h 01:22:** To shutdown at 1:22 AM.
- ▶ **apt-get update:** Synchronizes the list of packages on our system to the list in the repositories. Use this before installing new packages to make sure we are installing the latest version
- ▶ **apt-get upgrade:** Upgrades all of the software packages we have installed.
- ▶ **startx:** Opens the GUI



RaspberryPi commands

Directory and File commands:

- ▶ **mkdir new_directory:** Creates a new directory named new_directory.
- ▶ **mv new_folder:** Moves the file or directory named “new_folder” to a specified location
- ▶ **rm new_file.txt:** Deleted the file new_file.txt.
- ▶ **rmdir new_directory:** Deletes the directory “new_directory” only if it is empty.
- ▶ **touch new_file.txt:** creates a new, empty file named new_file.txt in the current directory.
- ▶ **cat new_file.txt:** Displays the contents of the file new_file.txt
- ▶ **cd /xyz/abc:** Changes the current directory to the /xyz/abc directory.
- ▶ **ls -l:** lists files in the directory.



RaspberryPi commands

Networking and Internet Commands:

- ▶ **iwconfig:** check which wireless adapter is currently active.
- ▶ **ifconfig:** wireless connection status.
- ▶ **ping:** tests connectivity between two devices connected on a network.
- ▶ **wget http://www.website.com/new_file.txt:** Downloads the file new_file.txt from the web and saves it to the current directory.
- ▶ **nmap:** Scans the network and lists connected devices, protocol, port number and other information.
- ▶ **iwlist wlan0 scan:** list of currently available wireless networks.



RaspberryPi commands

System information commands:

- ▶ **cat /proc/meminfo:** shows details about memory
- ▶ **cat /proc/version:** shows which version of raspberrypi we are using.
- ▶ **df -h:** shows information about available disk space.
- ▶ **df/:** shows how much free disk space is available.
- ▶ **free:** shows how much free memory is available.
- ▶ **hostname -I:** shows ip address of the raspberrypi.
- ▶ **lsusb;** lists the usb hardware connected to raspberrypi.
- ▶ **vcgencmd measure_temp:** shows the temperature of the cpu.
- ▶ **vcgencmd get_mem arm && vcgencmd get_mem gpu:** shows the memory split between the cpu and gpu.



Programming RaspberryPi with Python:

Table 8-3: Simple python programs on RaspberryPi

Program	Code
Print hello world	<code>print("hello world")</code>
program to add two numbers code	<code>a=1.2 b=5.3 sum=float(a)+float(b) print("the sum of {0} and {1} is {2}".format(a,b,sum))</code>



program to roll a dice	<pre>import random min=1 max=6 roll_again="yes" while roll_again=="yes" or roll_again=="y" print("rolling the dices") print("the values are") print(random.randint(min,max)) print(random.randint(min,max))</pre>
-------------------------------	---



program to find the ip address of raspberrypi	<pre>import urllib import re print("we will try to open this url, in order to get ip address") url=http://checkip.dyndns.org print(url)</pre>
program to generate password	<pre>import string from random import* characters=string.ascii_letters+string.punctuation+string.digits password=''.join(choice(characters) for x in range(randint(8,16))) print(password)</pre>
program to print fibonacci series	<pre>a,b=0,1 while b<200: print(b) a,b=b,a+b</pre>



Program 1:Printing to a terminal

Interfacing programs on Raspberrypi

Program #1	Printing to a terminal
Components required	Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse and Power supply
Description	Now let us start with a basic example of printing a message "Hello World" using Python Programming. Box shows how to print a greeting message to the console.
Code	File:Hello world.py #Access the python working environment #!/usr/bin/python #Print a message Hello world on to the terminal print("Hello World!")
Output	A message "Hello world" will prints on the console.



Programming RaspberryPi with Python

- ▶ 1. Blinking an LED
- ▶ 2. Push button for physical input
 - ▶ Switch used to control LED
- ▶ 3. Interact with the user
 - ▶ Switch used to control LED depending on user choice



Program 2: Blinking an LED

Program #2	Blinking an LED
Components required	Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse and Power supply, 1 Red LED and Blue LED, 2 1K resistors.
Description	Now let us look at an example of controlling the state of LEDs from ON to OFF state or vice versa from Raspberry Pi. Figure shows the schematic diagram of connecting an LEDs to Raspberry Pi. Box shows how to turn the LED on/off from by running the code given. In this example the LEDs are connected to GPIO pin 17 and 27 respectively which is initialized as output pin. The state of the LED is toggled by the executing the two programs given in the Box. Box
	having the code changes the state of the LED twice whereas Box having the code runs an infinite loop to flicker LEDs from ON/OFF state every second. Run the code given below and observe at the desired output.



LED Blink

- ▶ **import RPi.GPIO as GPIO**

- ▶ Tells the Python interpreter (the thing that runs the Python code) that it will be using a 'library' that will tell it how to work with the Raspberry Pi's GPIO pins.

- ▶ **import time**

- ▶ imports the Time library so that we can pause the script later on.

- ▶ **GPIO.setmode(GPIO.BCM) or GPIO.BOARD**

- ▶ Each pin on the Raspberry Pi has several different names, so you need to tell the program which naming convention is to be used.

- ▶ **GPIO.setup(18, GPIO.OUT)**

- ▶ **GPIO.output(18, GPIO.HIGH)**

- ▶ This turns the GPIO pin 'on'. What this actually means is that the pin is made to provide power of 3.3volts. This is enough to turn the LED in our circuit on.

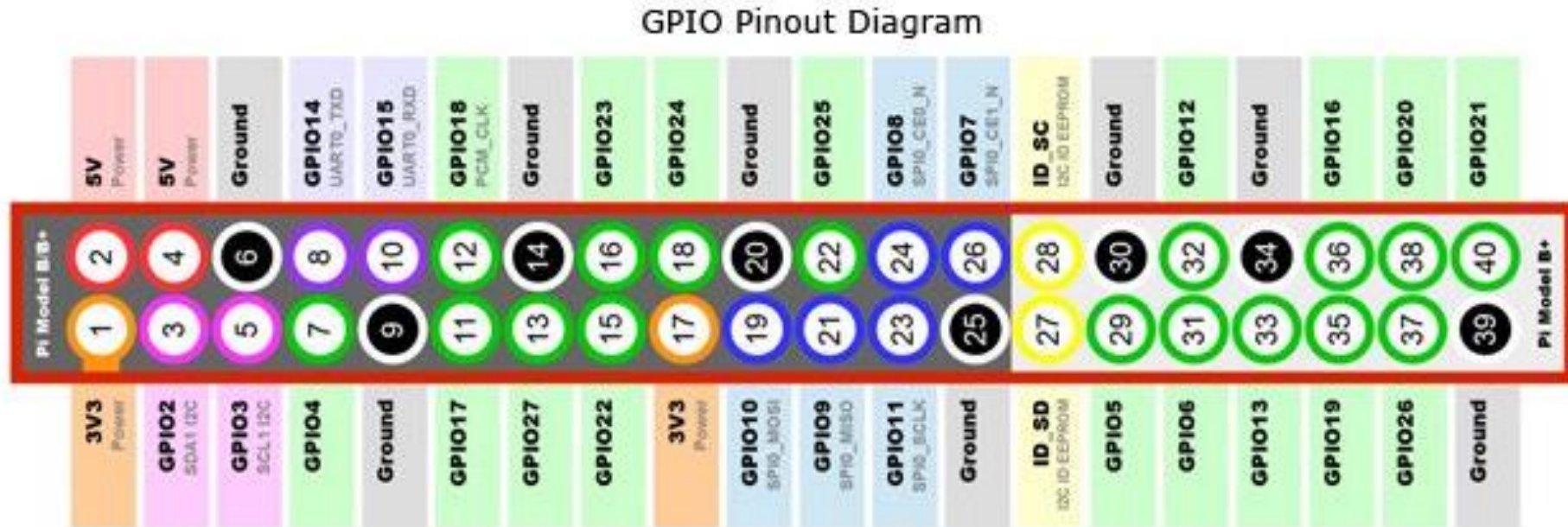
- ▶ **GPIO.output(18, GPIO.LOW)**

- ▶ This turns the GPIO pin 'off', meaning that the pin is no longer supplying any power.



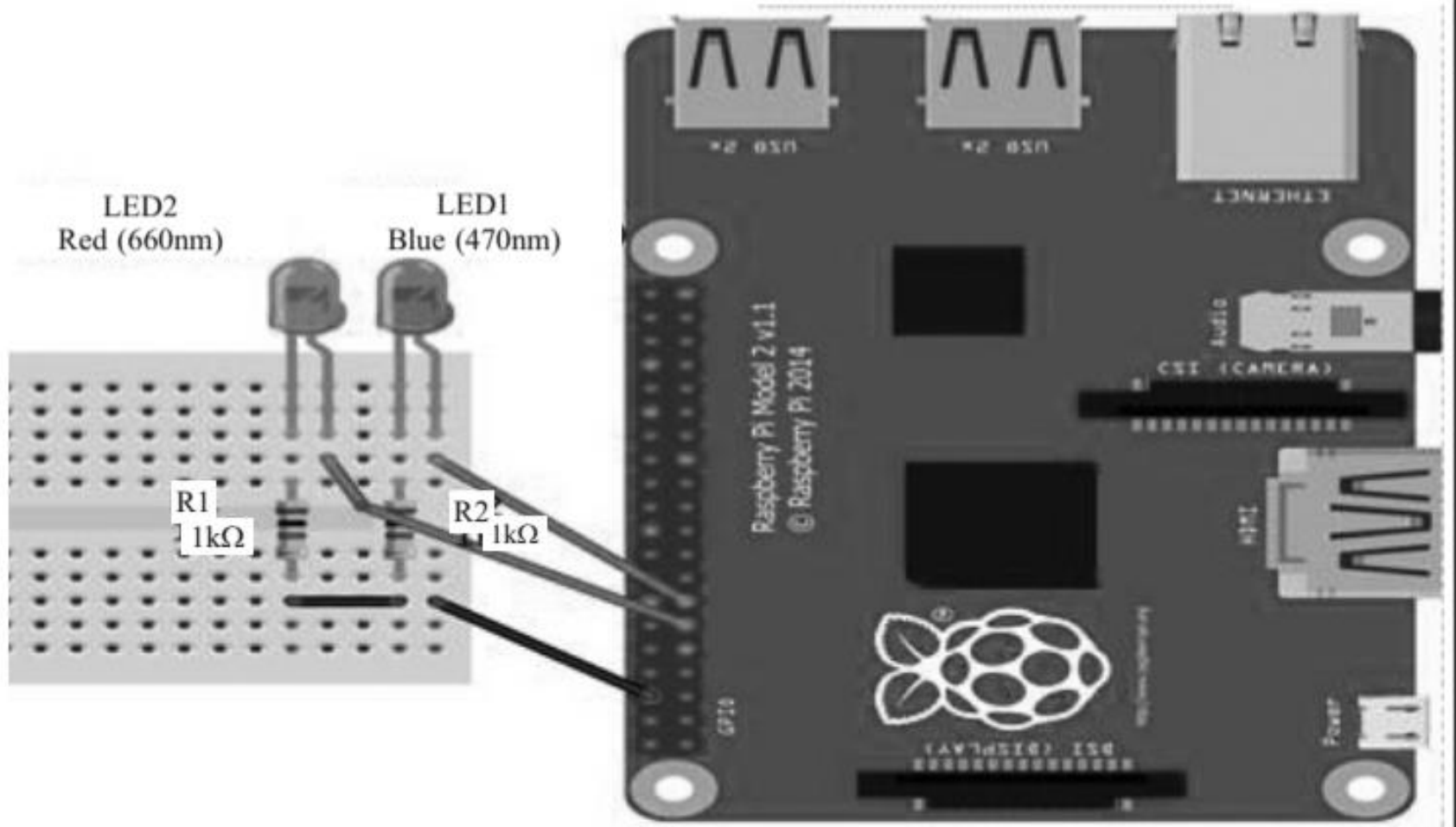
Exploring the Raspberrypi Learning Board

Raspberry Pi2 Model B and its GPIO



<https://www.jameco.com/Jameco/workshop/circuitnotes/raspberry-pi-circuit-note.html>

Circuit Diagram



Key points

1. Create file “blink.py”
2. Create file “blink_ever.py”
3. Enter the above code
4. Run the python file “sudo python blink.py” << Watch the LEDs blink 2
5. Run the python file “sudo python blink_ever.py” << Watch the LEDs blink forever



Code

File: blink.py

```
#Access the python working environment
#!/usr/bin/python
#import the time module so as to switch LEDs on/off with the time elapsed
#import the RPi.GPIO library
import RPi.GPIO as GPIO
#use one of the two numbering system either BOARD numbers/BCM
# Refer to the channel numbers on the Broadcom SOC.
GPIO.setmode(GPIO.BCM)
#Configure Pin 17 as an OUTPUT
GPIO.setup(17,GPIO.OUT)
#Configure Pin 27 as an OUTPUT
GPIO.setup(27,GPIO.OUT)
#Turn up LEDs on pin 17
GPIO.output(17,GPIO.HIGH)
#Turn up LEDs on pin 27
GPIO.output(27,GPIO.HIGH)
#wait for 1 second
time.sleep(1)
#Turn up LEDs off on pin 17
GPIO.output(17,GPIO.LOW)
#Turn up LEDs off on pin 27
GPIO.output(27,GPIO.LOW)
#wait for 1 second
time.sleep(1)
```

File:blink_ever.py

```
#Access the python working environment
#!/usr/bin/python
#import the time module so as to switch LEDs on/off with the time elapsed
import time
#import the RPL.GPIO library
import RPi.GPIO as GPIO
#use one of the two numbering system either BOARD numbers/BCM
# Refer to the channel numbers on the Broadcom SOC.
GPIO.setmode(GPIO.BCM)
#Configure pin 17 and 27 to be an OUTPUT pins
GPIO.setup(17,GPIO.OUT)
GPIO.setup(27,GPIO.OUT)
#Use while construct which runs infinite number of times there by blinking
    LEDs forever
while 1:
    #Turn up LEDs on
    GPIO.output(17,GPIO.HIGH)
    GPIO.output(27,GPIO.HIGH)
    time.sleep(1)
    #Turn up LEDs off
    GPIO.output(17,GPIO.LOW)
    GPIO.output(27,GPIO.LOW)
    time.sleep(1)
```

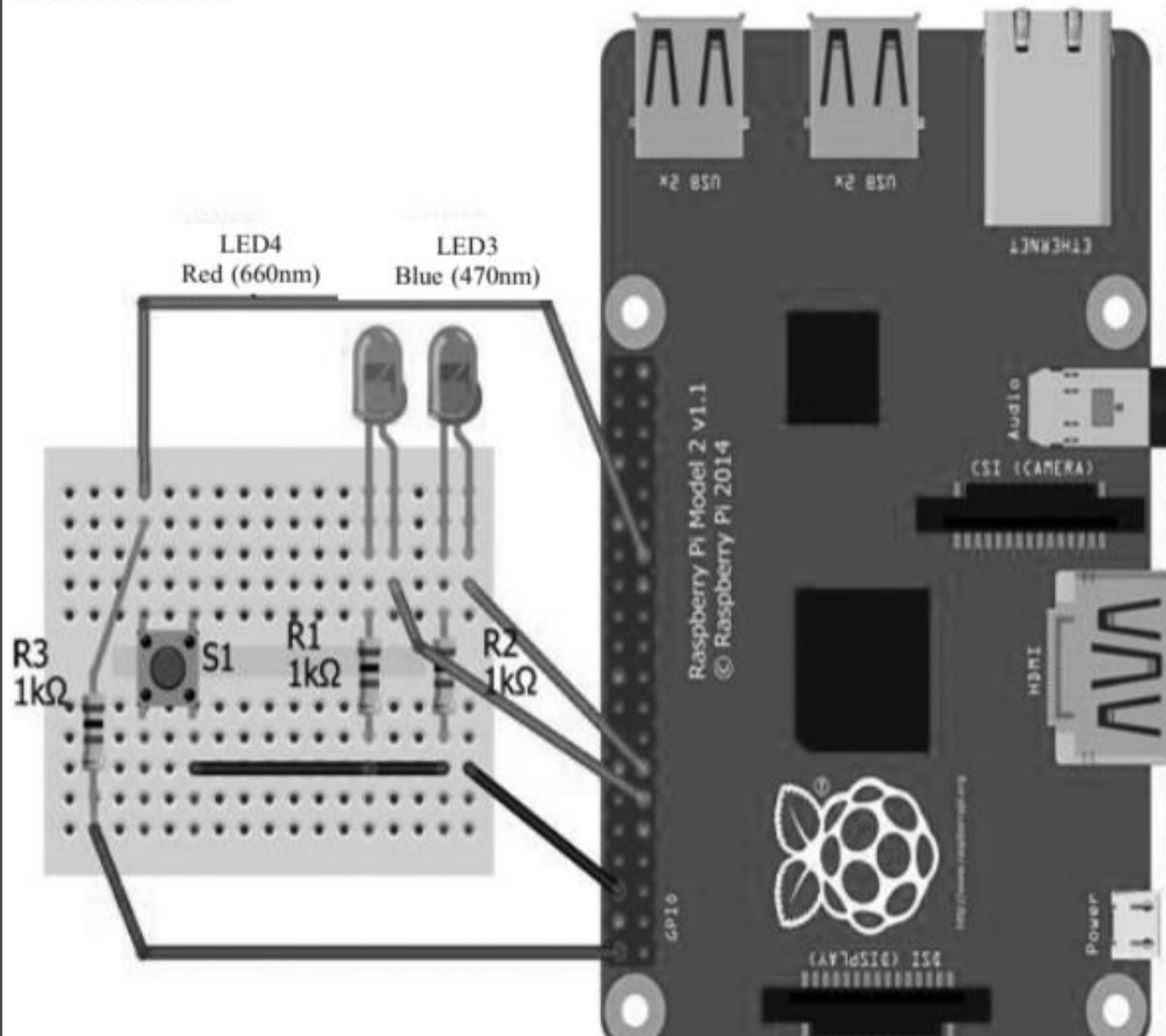
Output
LEDs turns on/off twice when blink.py file is executed and LEDs keeps changing their state forever when blink_forever.py file is executed.



Program #3	Push button for physical input
Components required	Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse and Power supply, 1 Red LED and Blue LED, 2 1K resistors, push button and jumper wires.
Description	Now let us look at an example involving LEDs and a switch that is used to control LED. Figure shows the schematic diagram of connecting an LEDs and a switch to Raspberry Pi. Box shows a python program for controlling an LED with a switch.In the infinite while loop the value of pin 10 is checked and the state of the LED is toggled if the switch is pressed. This example shows how to get input from GPIO pins and process the state of the LEDS. Run the code given below and observe at the desired output.



Circuit Diagram



Key points	<ol style="list-style-type: none">1. Create file “button.py”2. Enter the above code3. To run the python code “sudo python button.py”
Code	<pre>File: button.py #Access the python working environment #!/usr/bin/python #Import os module to enable interrupts from a push button import os #import the time module so as to know the time the user as given the input from a push button</pre>



```
import time
#import the RPi.GPIO library
import RPi.GPIO as GPIO
#use one of the two numbering system either BOARD numbers/BCM
# Refer to the channel numbers on the Broadcom SOC.
GPIO.setmode(GPIO.BCM)
#configure pin 10 to be INPUT which reads the status of a switch button
GPIO.setup(10, GPIO.IN)
#print a message on to the terminal
print(" Button + GPIO ")
#Read the status of a button from GPIO pin 10
print GPIO.input(10)
#Run a infinite loop on the status of the button read
while True:
    if ( GPIO.input(10) == True ):
        print("Button Pressed")
        #Print the time when the input was given from the push button
        os.system('date')
        #Read the status of a button from GPIO pin 10
        print GPIO.input(10)
        #wait for 5 seconds
        time.sleep(5)
    else:
        #clear the system variables
        os.system('clear')
        #prompt the user to give an input
        print ("Waiting for you to press a button")
        time.sleep(1)
```

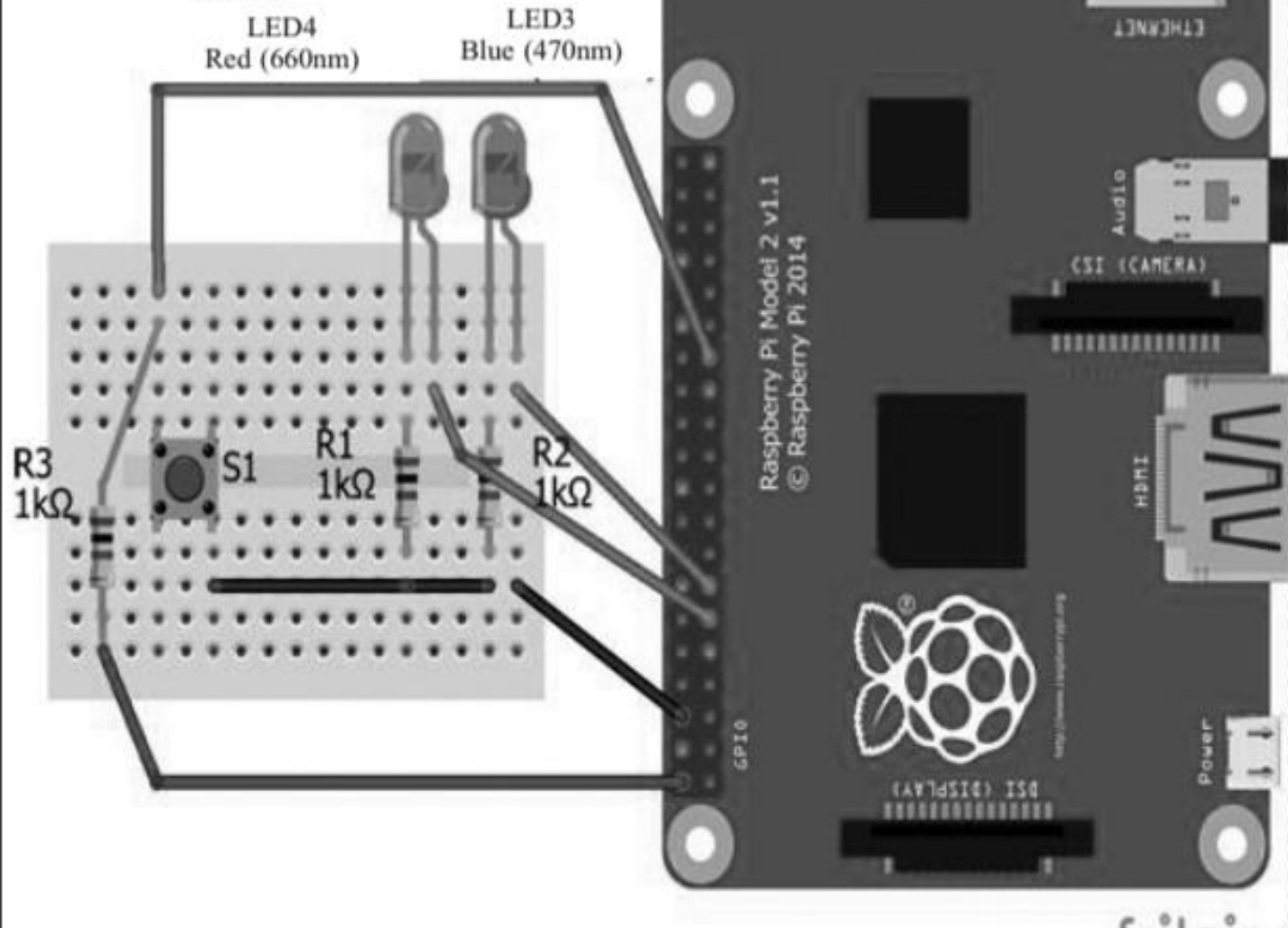
Output	Press the Push button switch to Turn ON/OFF the LED
---------------	---



Program #4	Interact with the user
Components required	Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse and Power supply, 1 Red LED and Blue LED, 2 1K resistors, push button and jumper wires.
Description	<p>Now let us look at an example involving LEDs and a switch that is used to control LED depending on what a user choose. Figure shows the schematic</p> <p>diagram of connecting an LEDs and a switch to Raspberry Pi. Box shows a python program for controlling an LED by reading two input values, one for which LED would user like to blink(option 1-for Red, 2- for Blue) and one more parameter to set the maximum number of times the LED should be flickered.</p> <p>This example shows how to get input from a user and process the state of the LEDS. Run the code given below and observe at the desired output.</p>



Raspberry Pi 2



File: user_input.py

#Access the python working environment

#!/usr/bin/python

#Import os module to enable interrupts from a push button

import os

#import the time module so as to switch LEDs on/off with the time elapsed

import time

#import the RPi.GPIO library

import RPi.GPIO as GPIO

#use one of the two numbering system either BOARD numbers/BCM

Refer to the channel numbers on the Broadcom SOC

GPIO.setmode(GPIO.BCM)

#configure pin 17 to be OUTPUT pin

GPIO.setup(17,GPIO.OUT)

#configure pin 27 to be OUTPUT pin

GPIO.setup(27,GPIO.OUT)



```
#Initialize variables for user input
led_ch = 0
counter = 0

#clear the python interpreter console
os.system('clear')

print "Which LED would you like to blink"
#Accept 1 for Red
print "1: Red?"
#Accept 2 for Blue
print "2: Blue?"

led_ch = input("Choose your option: ")
```



```
if led_ch == 1:
    #clear the python interpreter console
    os.system('clear')
    print "You picked the Red LED"
    counter = input("How many times would you like it to blink?: ")
    while counter > 0:
        #on LED on Pin 27
        GPIO.output(27,GPIO.HIGH)
        time.sleep(1)
        #off LED on pin 27
        GPIO.output(27,GPIO.LOW)
    time.sleep(1)

    #Record the number of counts on LED
    counter = counter - 1
```



```
if led_ch== 2:
    #clear the python interpreter console
    os.system('clear')
    print "You picked the Red LED"
    counter = input("How many times would you like it to blink?: ")
    while counter > 0:
        #on LED Pin 27
        GPIO.output(17,GPIO.HIGH)
        time.sleep(1)
        #off LED on pin 27
        GPIO.output(17,GPIO.LOW)
        time.sleep(1)
        #Record the number of counts on LED
        count = count - 1
```

Output	LED gets flickered on the inputs given by the user.
---------------	---



Program #5: Piezo Buzzer

Components Required

- ▶ Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse and Power supply, jumper wires, Breadboard, buzzer.

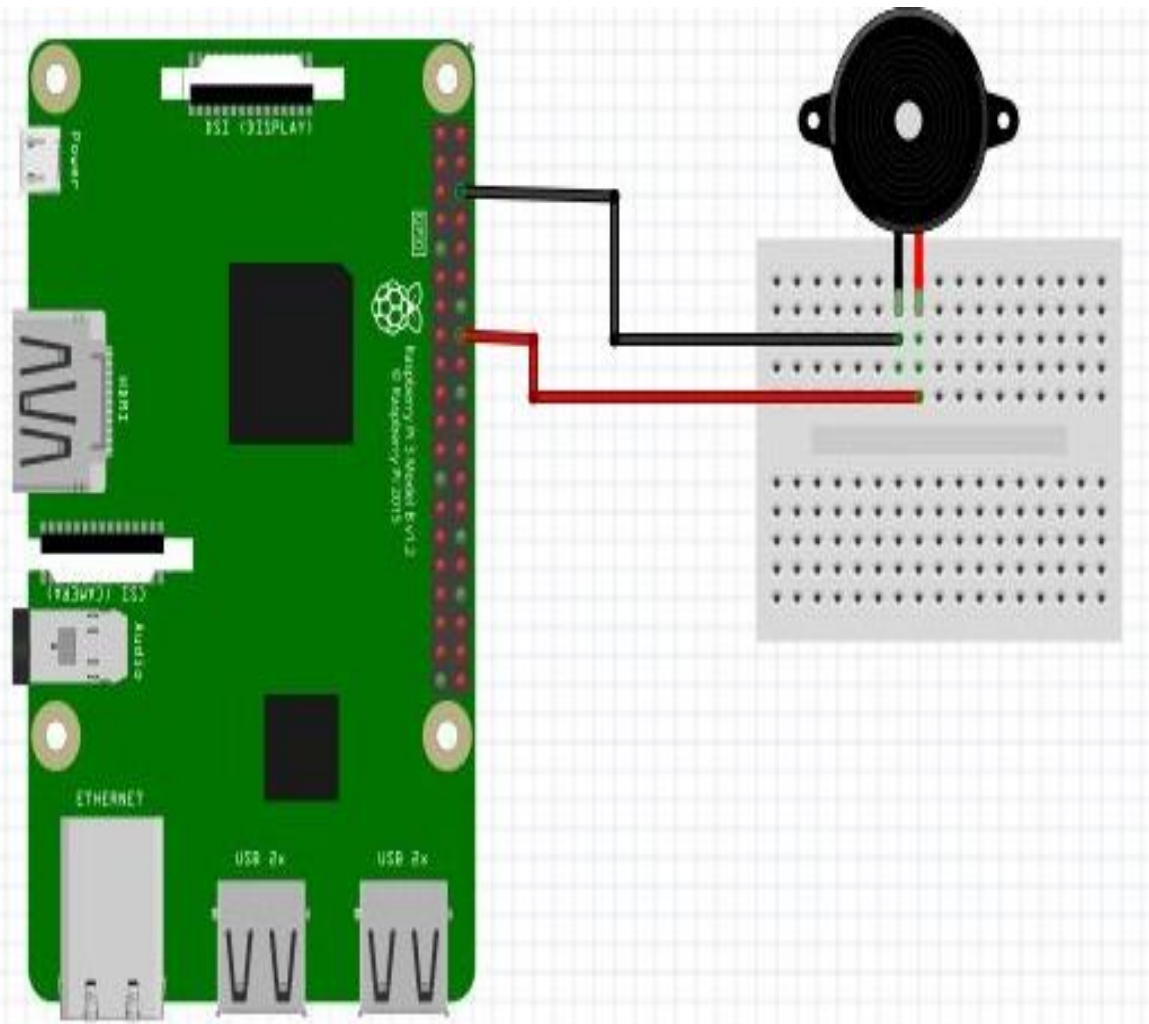
Description

- ▶ A Piezo buzzer used to beep a number of times the user choose.
 - ▶ connect an piezo buzzer to pin 23.
 - ▶ A python program for controlling an piezo buzzer by reading an input value which runs over a loop to beep number of times the user has choose.
-



3V3 Power	1	2	5V Power
GPIO2 SDA1, I2C	3	4	5V Power
GPIO3 SCL1, I2C	5	6	Ground
GPIO4	7	8	GPIO14 UART0_TX
Ground	9	10	GPIO15 UART0_RX
GPIO17	11	12	GPIO16 PCM_CLK
GPIO27	13	14	Ground
GPIO22	15	16	GPIO23
3V3 Power	17	18	GPIO24
GPIO10 SMB_ALERT	19	20	Ground
GPIO9 SMB_MISO	21	22	GPIO25
GPIO11 SMB_MOSI	23	24	GPIO6 SPI_CS0_N
Ground	25	26	GPIO7 SPI_CS1_N
ID_SD IO 30 EXP+IOE	27	28	ID_SC IO 30 EXP-IOE
GPIO5	29	30	Ground
GPIO6	31	32	GPIO12
GPIO13	33	34	Ground
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
Ground	39	40	GPIO21

GPIO Pinout Diagram



-
- ▶ Initialize pin 23 as an output pin with `GPIO.setup()` function.
 - ▶ In the main loop, you make a beep sound with `GPIO.output()` function and "pause" the program for 0.1 second with `sleep()` function.



Code

```
import os
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(23,GPIO.OUT)
loop_counter = 0
```



```
def morsecode ():
```

```
    #Dot Dot Dot
```

```
    GPIO.output(23,GPIO.HIGH)
```

```
    time.sleep(.1)
```

```
    GPIO.output(23,GPIO.LOW)
```

```
    time.sleep(.1)
```

```
    GPIO.output(23,GPIO.HIGH)
```

```
    time.sleep(.1)
```

```
    GPIO.output(23,GPIO.LOW)
```

```
    time.sleep(.1)
```

```
    GPIO.output(23,GPIO.HIGH)
```

```
    time.sleep(.1)
```



#Dash Dash Dah

```
GPIO.output(23,GPIO.LOW)
time.sleep(.2)
GPIO.output(23,GPIO.HIGH)
time.sleep(.2)
GPIO.output(23,GPIO.LOW)
time.sleep(.2)
GPIO.output(23,GPIO.HIGH)
time.sleep(.2)
GPIO.output(23,GPIO.LOW)
time.sleep(.2)
GPIO.output(23,GPIO.HIGH)
time.sleep(.2)
GPIO.output(23,GPIO.LOW)
time.sleep(.2)
```



```
os.system('clear')
print "Morse Code" loop_count = input("How many times
would you like SOS to loop?:")
while loop_count > 0:
    loop_counter = loop_counter - 1
    morsecode ()
```



Program #6: Temperature Sensor-DS18B20

Components Required

- ▶ Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse and Power supply, 1 DS18B20 temperature sensor.

Description

- ▶ An DS18B20 temperature sensor which reads out a temperature and records on to a terminal.
- ▶ A python program which records the temperature read by DS18B20 temperature sensor.
- ▶ An infinite loop runs over the sensor which records a temperature every second.



About DS18B20

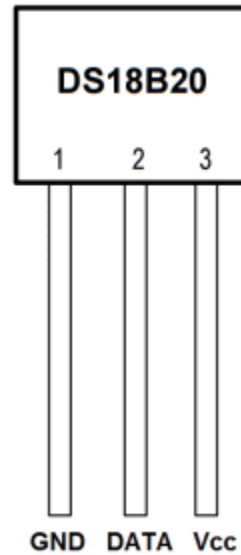
- ▶ A commonly used temperature sensor providing 9 bit to 12 bit digital Celsius temperature measurements.
- ▶ The DS18B20 communicates with the “**One-Wire**” **communication protocol**, a proprietary serial communication protocol that uses only one wire to transmit the temperature readings to the microcontroller.
- ▶ The DS18B20 can be operated in what is known as *parasite power* mode.
- ▶ Normally the DS18B20 needs three wires for operation: the Vcc, ground, and data wires. In parasite mode, only the ground and data lines are used, and power is supplied through the data line.

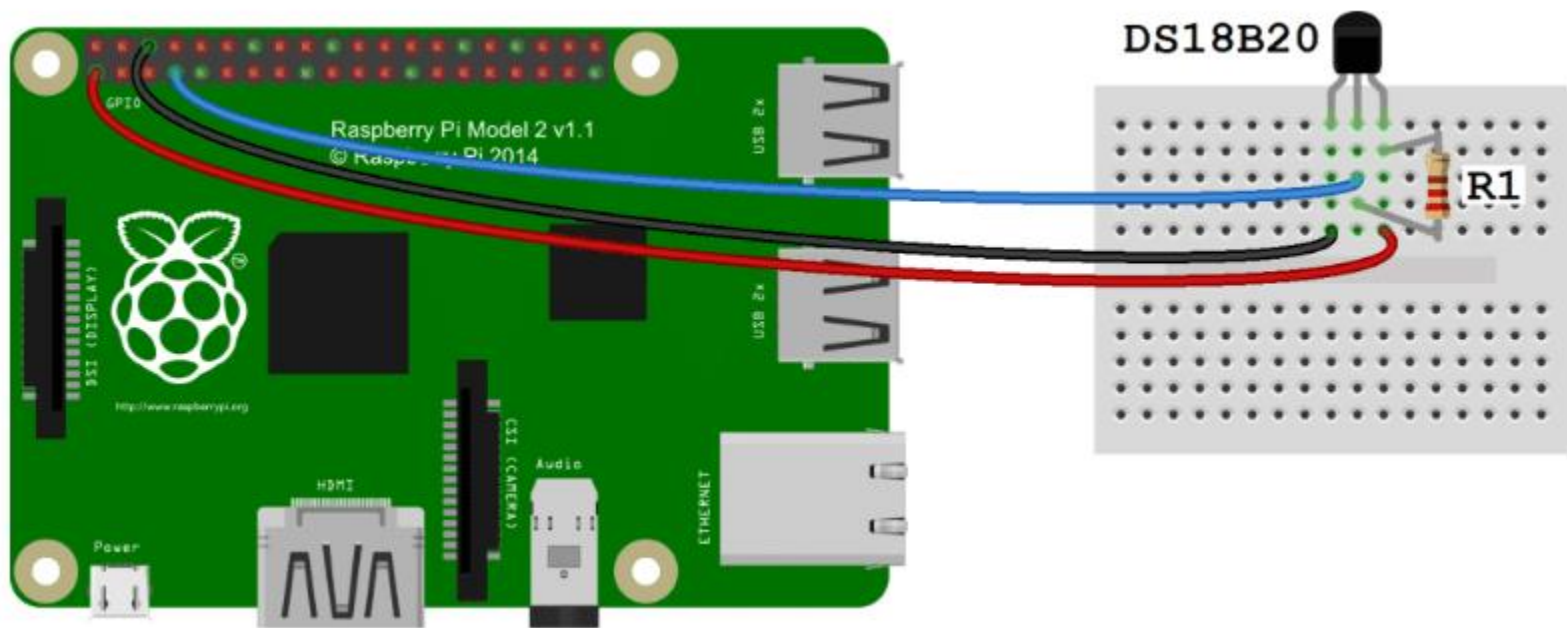


-
- ▶ A 64 bit ROM stores the device's unique serial code. This 64 bit address allows a microcontroller to receive temperature data from a virtually unlimited number of sensors at the same pin.
 - ▶ The address tells the microcontroller which sensor a particular temperature value is coming from.
 - ▶ The Raspberry Pi has drivers for one wired devices to be connected to GPIO pin-4 by default.
 - ▶ <https://electrosome.com/ds18b20-sensor-raspberry-pi-python/>
 - ▶ <https://www.circuitbasics.com/raspberry-pi-ds18b20-temperature-sensor-tutorial/>
-



DS18B20





ENABLE THE ONE-WIRE INTERFACE

modprobe

- ▶ used to add a [loadable kernel module](#) to the [Linux kernel](#)

1. `sudo modprobe wl-gpio`
 2. `sudo modprobe wl-therm`
 3. Change directories to the `/sys/bus/wl/devices` directory by entering: `cd /sys/bus/wl/devices`
 4. Now enter `ls` to list the devices:
28-000006637696 wl_bus_master1 is displayed in my case.
 5. Now enter `cd 28-XXXXXXXXXXXXXX` (change the X's to your own address)
For example, I would enter: `cd 28-000006637696`
 6. Enter `cat wl_slave` which will show the raw temperature reading output by the sensor:
-

► Output

```
81 01 4b 46 7f ff 0f 10 71 : crc=71 YES  
81 01 4b 46 7f ff 0f 10 71 t=24062
```

- The displayed data consist of two lines – first line representing the value read and CRC check and second line temperature in Celsius x 1000.



Code

```
import os      # import os module
import glob    # import glob module
import time    # import time module
#initialize the device
# load one wire communication device kernel modules
os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')
base_dir = '/sys/bus/w1/devices/'    # point to the address
# find device with address starting from 28*
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave' # store the details
def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()              # read the device details
    f.close()
    return lines
```



```
def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':      # ignore first line
        time.sleep(0.2)
        lines = read_temp_raw()
        equals_pos = lines[1].find('t=') # find temperature in the details
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string)/ 1000.0 # convert to Celsius
        temp_f = temp_c * 9.0/ 5.0 + 32.0    # convert to Fahrenheit
        return temp_c, temp_f

while True:
    print(read_temp())
    time.sleep(1)
```



Output

Current Room Temperature is recorded.

```
$ python temp_DS18B20.py
temp C=25.187000 temp F=77.336600
temp C=25.125000      temp F=77.225000
temp C=25.062000      temp F=77.111600
temp C=26.312000      temp F=79.361600
temp C=27.875000      temp F=82.175000
temp C=28.875000      temp F=83.975000
```



Program #7: Light sensor –LDR(Light Dependent Resistor)

Components Required

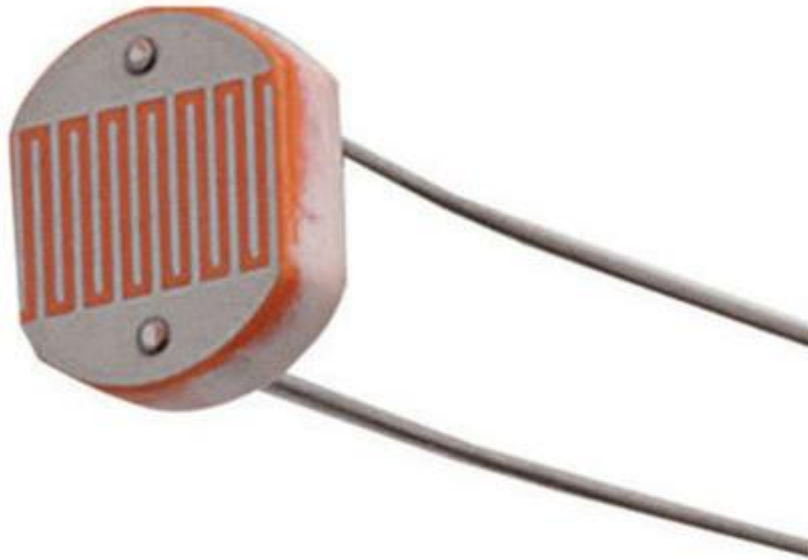
- ▶ Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse, Power supply, jumper wires, Breadboard, LDR Light Dependent resistor, 1uF Capacitor

Description

- ▶ An LDR sensor reads the intensity of light and records it in a text file.
- ▶ A python program which records the intensity of light to a text file.
- ▶ An infinite loop runs over the sensor which records intensity of light with date and time stamps every second.



LDR



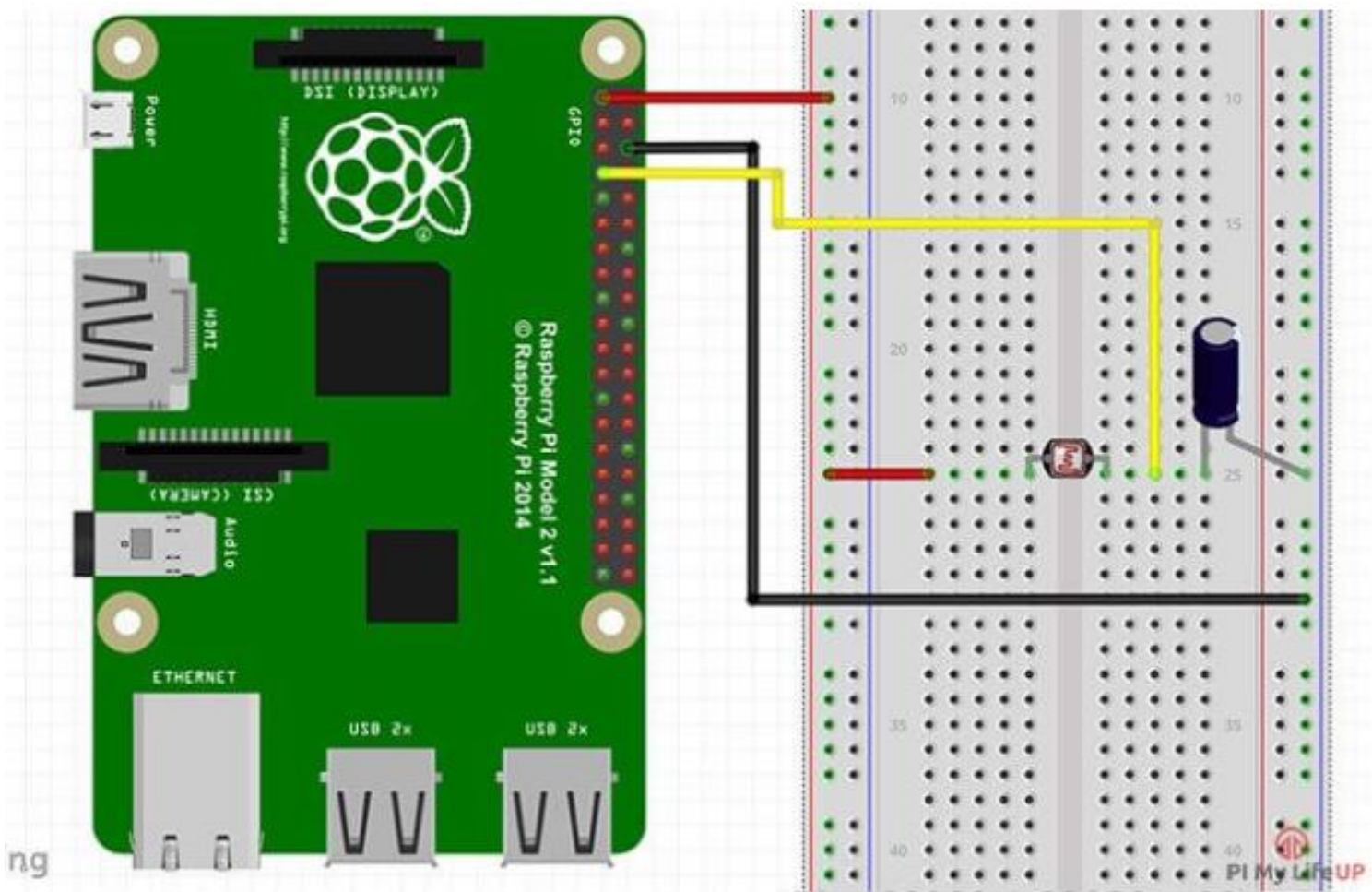
LDR(Photo Resistor)

- ▶ A photo conductive sensor.
 - ▶ It is a variable resistor that changes its resistance in a proportion to the light exposed to it. Its resistance decreases with the intensity of light.
 - ▶ As the capacitor gradually charges, the voltage that passes through the circuit and to the GPIO pin rises. Once the capacitor is charged to a certain point, its voltage rises above 1.4 volts and the Raspberry Pi will sense that GPIO pin 4 is HIGH.
 - ▶ If the resistance of the sensor increases, the capacitor will charge more slowly and the circuit will take more time to reach 1.4 volts.
 - ▶ The script essentially counts how long it takes for pin 4 to turn High and then uses this measurement to calculate the resistance of the Photoresistor.
-



-
- ▶ The Raspberry Pi will repeatedly display the resistance of the photoresistor.
 - ▶ If you place your finger over the photoresistor, the resistance will increase.
 - ▶ If you shine a bright light on the photoresistor, the resistance will decrease.
 - ▶ <https://pimylifeup.com/raspberry-pi-light-sensor/>





Circuit Diagram

- ▶ Connect one side of LDR sensor to 3.3V (Red Wire) and other side of LDR is connected to both GPIO pin 4 (yellow Wire) and 1uF Capacitor
- ▶ Connect other side of capacitor to negative. (Black Wire)



Code

```
import os
import datetime
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
def RCtimes(RCpins):
    readings = 0
    GPIO.setup(RCpins, GPIO.OUT)
    GPIO.output(RCpins, GPIO.LOW)
    time.sleep(.1)
```



```
GPIO.setup(RCpins, GPIO.IN)
```

```
#iterates 1 milliseconds over one cycle
```

```
while (GPIO.input(RCpins) == GPIO.LOW):
```

```
    readings += 1
```

```
return readings
```

```
while True:
```

```
    GetDateTime = datetime.datetime.now().strftime("%Y-%m-%d  
    %H:%M:%S")\
```

```
    LDRReading = RCtimes(4)
```

```
    print RCtimes(4)
```

```
    #Open a file
```

```
    fo = open("/home/pi/I0xI0/foo.txt", "wb")    # write only binary
```

```
    fo.write (GetDateTime)
```

```
    LDRReading = str(LDRReading)
```

```
    fo.write ("\n")
```

```
    fo.write (LDRReading)
```

```
    #Close opened file
```

```
    fo.close()
```

```
    time.sleep(1)
```



- RCtimes function returns a count which is proportional to light level.
- In this functions, LDR pin is set to output and low and then to input.
- At this point, capacitor starts charging through the resistor until input pin reads high(this happens when capacitor voltage becomes greater than 1.4V)
- Counter stopped when input reads high.
- The final count is proportional to light level as greater the amount of light, smaller is the LDR resistance and greater is the time taken to charge the capacitor

Output:

- Intensity of the light in the room is recorded on to a terminal as well as to text file.



Program #8: Passive infrared Sensor

Components Required

- ▶ Raspberry Pi + SD card, Monitor + HDMI Cable, Keyboard & Mouse and Power supply, jumper wires, Breadboard, PIR(Passive Infrared sensor) sensor, Buzzer.

Description

- ▶ An PIR sensor detects the motion of an object.
- ▶ A python program detects the motion of an object and triggers a buzzer and print message as "Motion detected".
- ▶ An infinite loop runs over the PIR sensor which waits for any of the movements across its boundary.

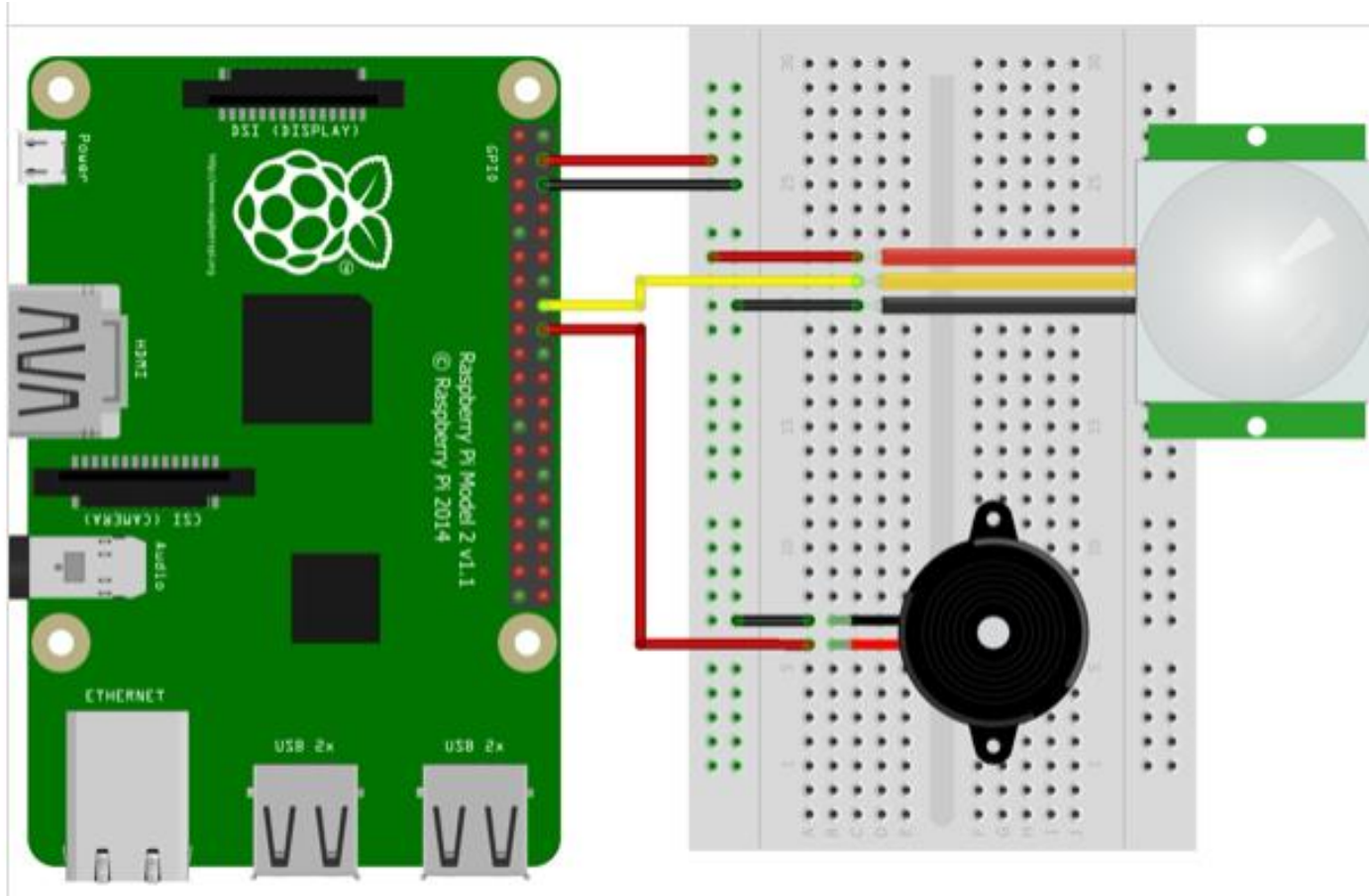


PIR

- ▶ PIR (passive infrared) motion sensor detects any movement of objects, human or animals.
- ▶ Sensor has three pins. Power (VCC), Ground (GND) and output (OUT) pin which gives logic high if motion is detected.
- ▶ <https://www.hackster.io/hardikrathod/pir-motion-sensor-with-raspberry-pi-415c04>



Circuit Diagram



```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.IN)          #PIR
GPIO.setup(24, GPIO.OUT)        #Buzzer
try:
    time.sleep(2)                # to stabilize sensor
    while True:
        if GPIO.input(23):      #If there is a movement, PIR sensor gives
                                input to GPIO 23
            GPIO.output(24, True)
            time.sleep(0.5)      #Buzzer turns on for 0.5 sec
            GPIO.output(24, False)
            print("Motion Detected...")
            time.sleep(5)        #to avoid multiple detection
            time.sleep(0.1)      #loop delay, should be less than detection delay
except:
    GPIO.cleanup()
```

Connecting Raspberry Pi via SSH (Secure Shell)

- ▶ You can access the command line of a Raspberry Pi remotely from another computer or device using SSH.
- ▶ SSH connection can't open programs that use a graphical interface
- ▶ The Raspberry Pi will act as a remote device: you can connect to it using a client on another machine.
- ▶ <https://magpi.raspberrypi.org/articles/ssh-remote-control-raspberry-pi>
- ▶ <https://www.raspberrypi.org/documentation/remote-access/ssh/>



Connecting Raspberry Pi via SSH

- ▶ Step 1. Set up your local network and wireless connectivity
- ▶ Step 2. Enable SSH
- ▶ Step 3: Activate SSH Client in Windows
- ▶ Step 4: Connect via SSH
- ▶ Step 5: Use PuTTY on a Windows PC



I. Set up your local network and wireless connectivity

- ▶ Make sure your Raspberry Pi is properly set up and connected.
- ▶ Ifconfig command or hostname -I to display IP address.
 - ▶ Note down the IP address of your Pi



▶ 2. Enable SSH

1. Launch `Raspberry Pi Configuration` from the `Preferences` menu
2. Navigate to the `Interfaces` tab
3. Select `Enabled` next to `SSH`
4. Click `OK`

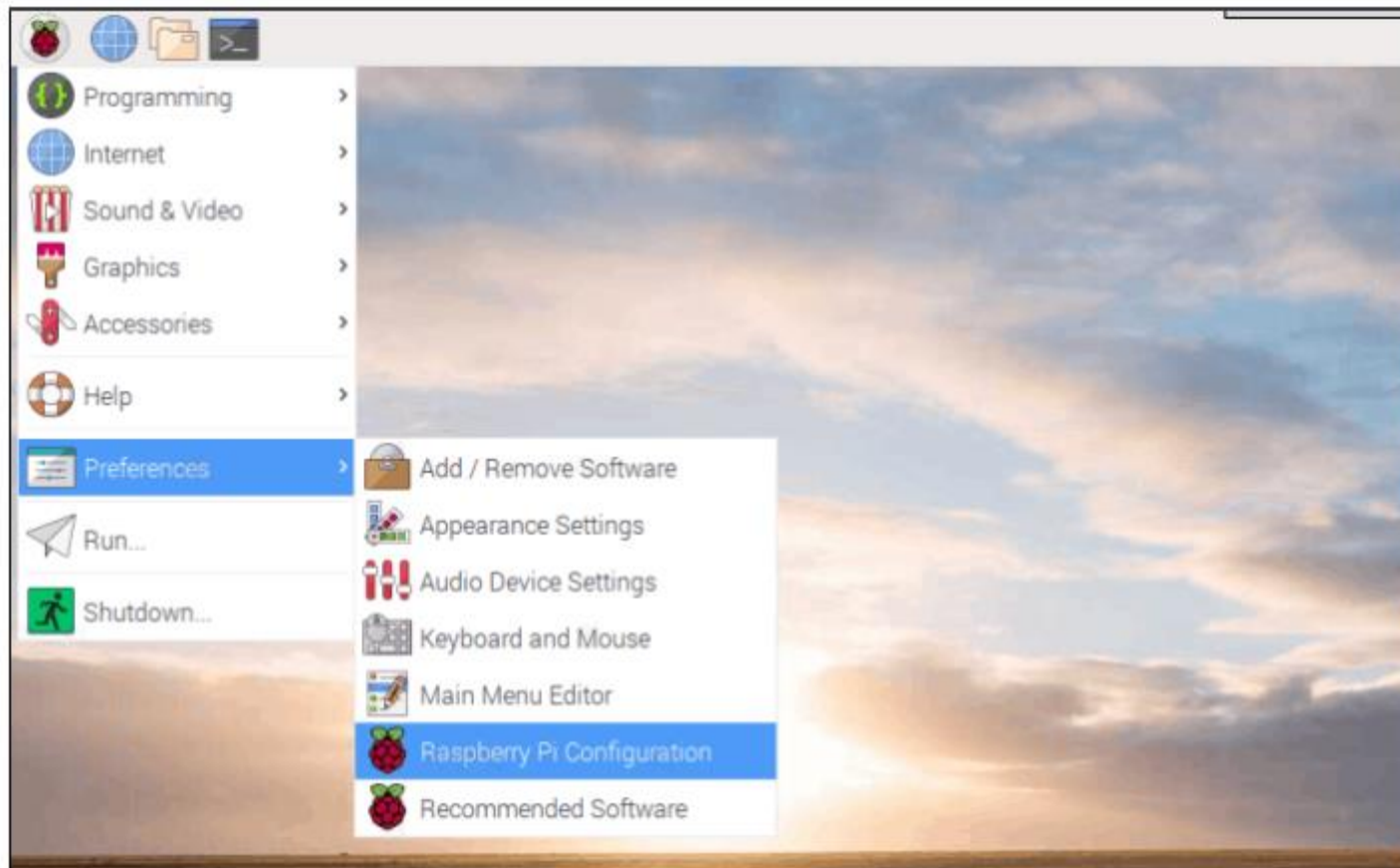
Alternatively, `raspi-config` can be used in the terminal:

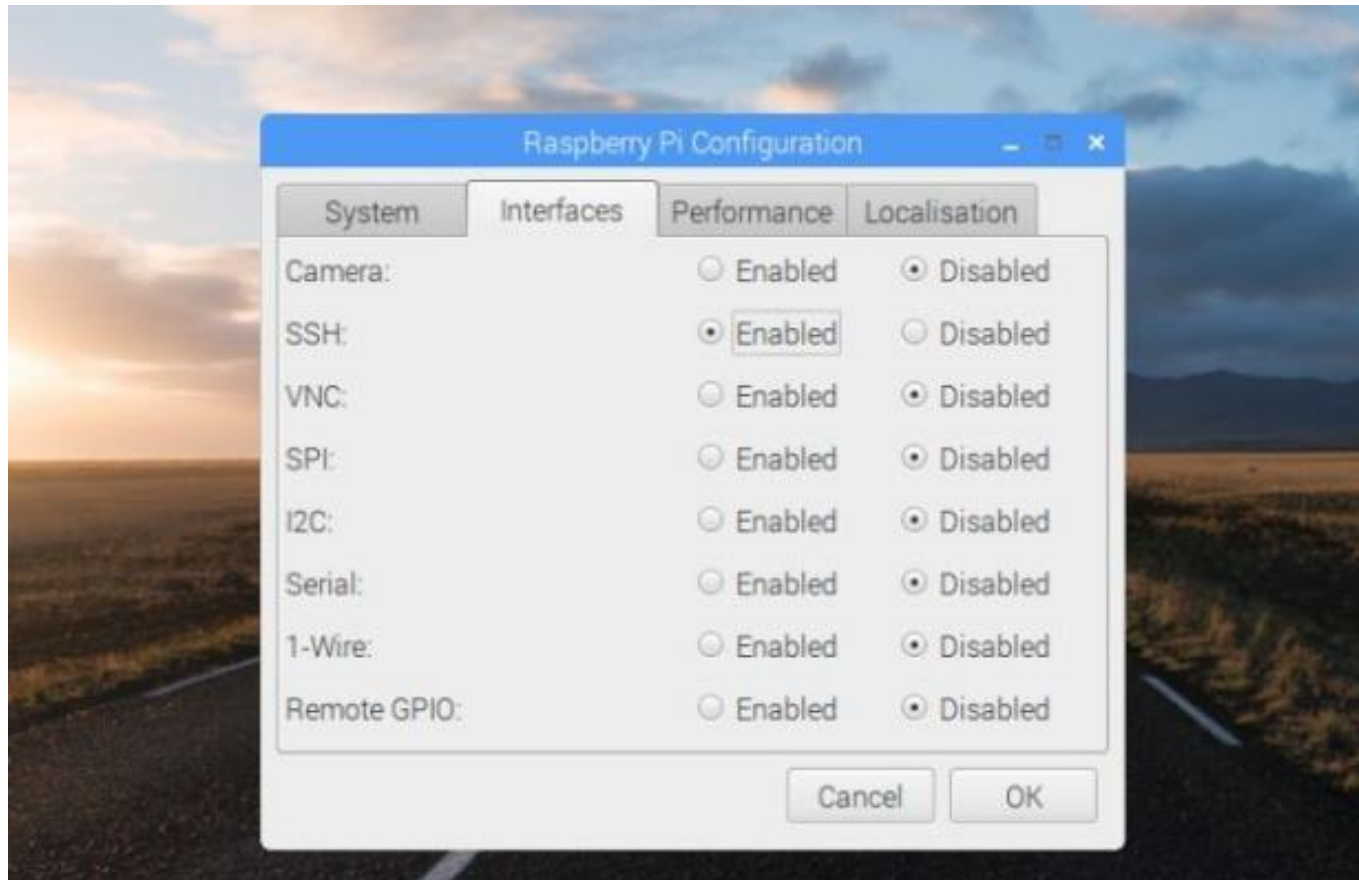
1. Enter `sudo raspi-config` in a terminal window
2. Select `Interfacing Options`
3. Navigate to and select `SSH`
4. Choose `Yes`
5. Select `OK`
6. Choose `Finish`

▶ Alternatively, use **systemctl** to start the services

- ▶ `sudo systemctl enable ssh`
- ▶ `sudo systemctl start ssh`







Step 3: Activate SSH Client in Windows

- ▶ Linux and macOS both support SSH out-of-the-box; skip ahead to Step 3 if you are using one of those operating systems
- ▶ Windows 10 supports SSH, but you need to activate it. Click on Search and look for 'Manage Optional Features'. Click it in Search to open the Settings window.
- ▶ Click 'Add a feature' and wait for the list of Optional Features to load. Scroll down the list to Open SSH Client (Beta). Click Install.



Step 4: Connect via SSH

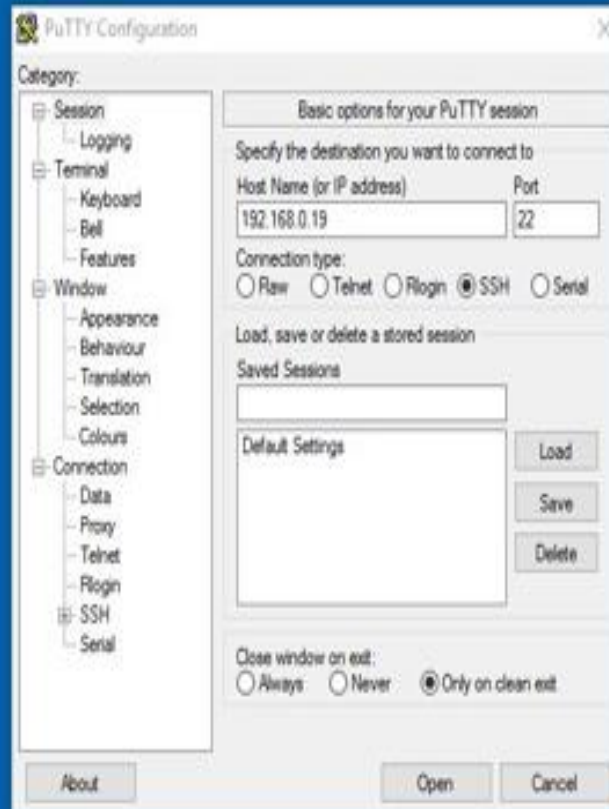
- ▶ Open Command Prompt on a Windows PC, or a Terminal window in Linux or macOS.
- ▶ Enter this command:
- ▶ `ssh pi@192.168.0.41`

Step 5: Use PuTTY on a Windows PC

- ▶ On older PCs you'll need to install PuTTY. Download the putty.exe file and click Run.
 - ▶ Enter the IP address of your Raspberry Pi in the 'Host Name (Or IP Address)' field. Don't change the 'Port' field. Click Open. You will get a PuTTY 'Security Alert' field. Click Yes. The terminal window displays 'login as:' Enter **pi** and press **RETURN**. Now enter the password for your Raspberry Pi.
-



Step 5: Use PuTTY on a Windows PC



SSH with PuTTY on a PC

Remote Access to RaspberryPi

Using VNC

- ▶ VNC Server is included with your Raspberry Pi.
- ▶ You will also need a VNC Viewer application for the Windows.
- ▶ Run the following commands to make sure you have the latest version:
 - ▶ `sudo apt-get update`
 - ▶ `sudo apt-get install realvnc-vnc-server`
- ▶ VNC Server is included with Raspbian but you still have to enable it.
- ▶ Select Menu > Preferences > Raspberry Pi Configuration > Interfaces and make sure VNC is set to Enabled
- ▶ Alternatively, run the command, `sudo raspi-config`, navigate to **Interfacing Options > VNC** and select **Yes**



Establishing a direct connection

- ▶ Direct connections are quick and simple providing you're joined to the same private local network as your Raspberry Pi.
- ▶ On your Raspberry Pi, discover its private IP address by double-clicking the VNC Server icon on the taskbar and examining the status dialog.





Connectivity



192.168.5.37

Connecting users can enter this address in [VNC Viewer](#)

[Sign in](#) to enable cloud connectivity or [learn more about the benefits](#)

▼ Other ways to connect

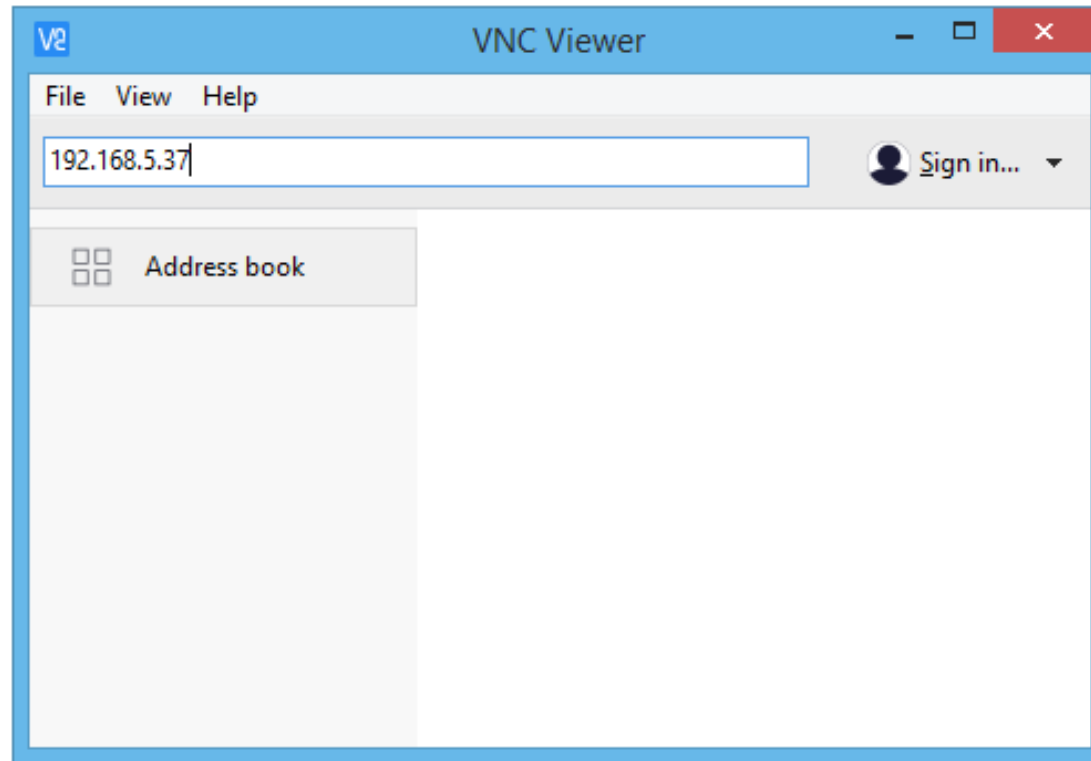
192.168.5.37

[2e02:390::501:192:cf2d:a576:9d1a:9d77]

Security



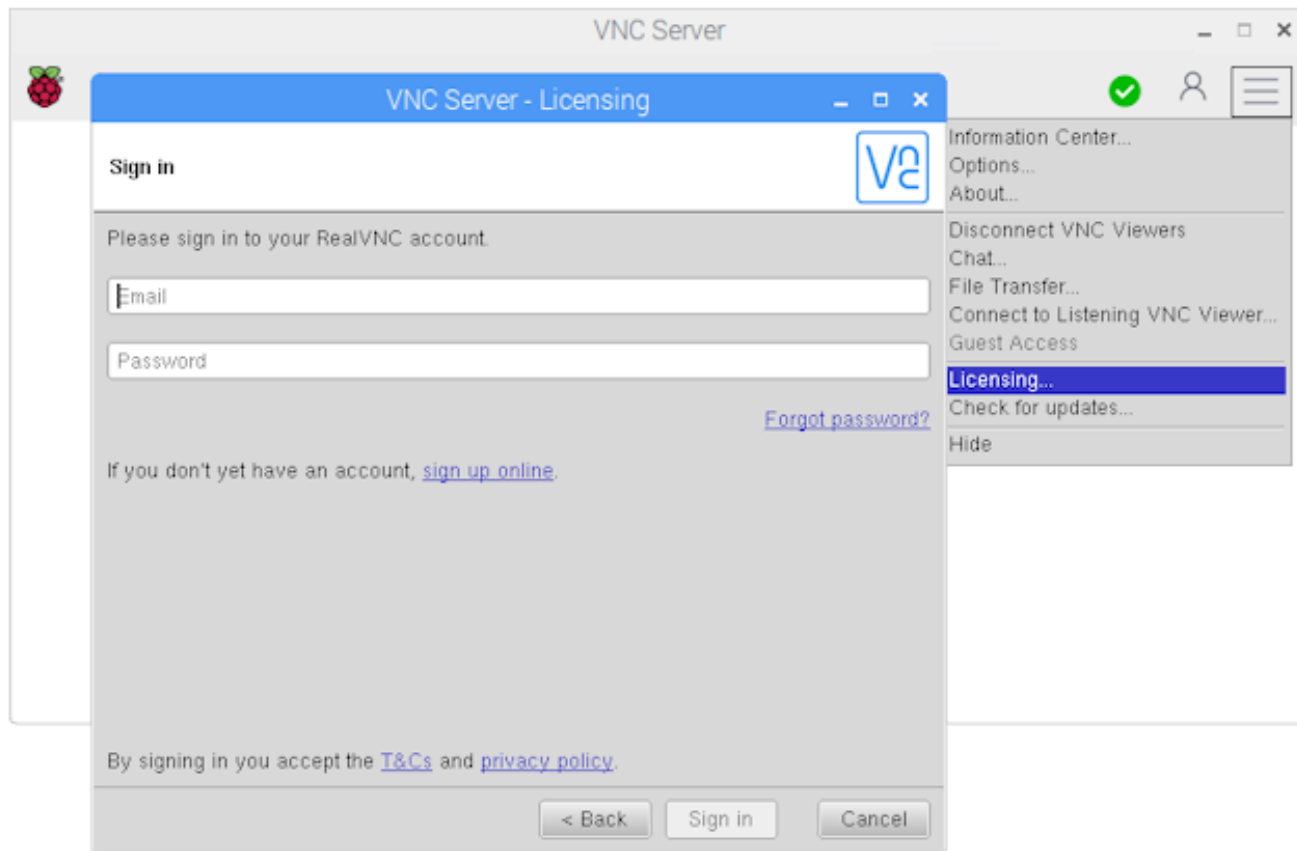
-
- ▶ On the device you will use to take control, run VNC Viewer and enter the IP address in the search bar:



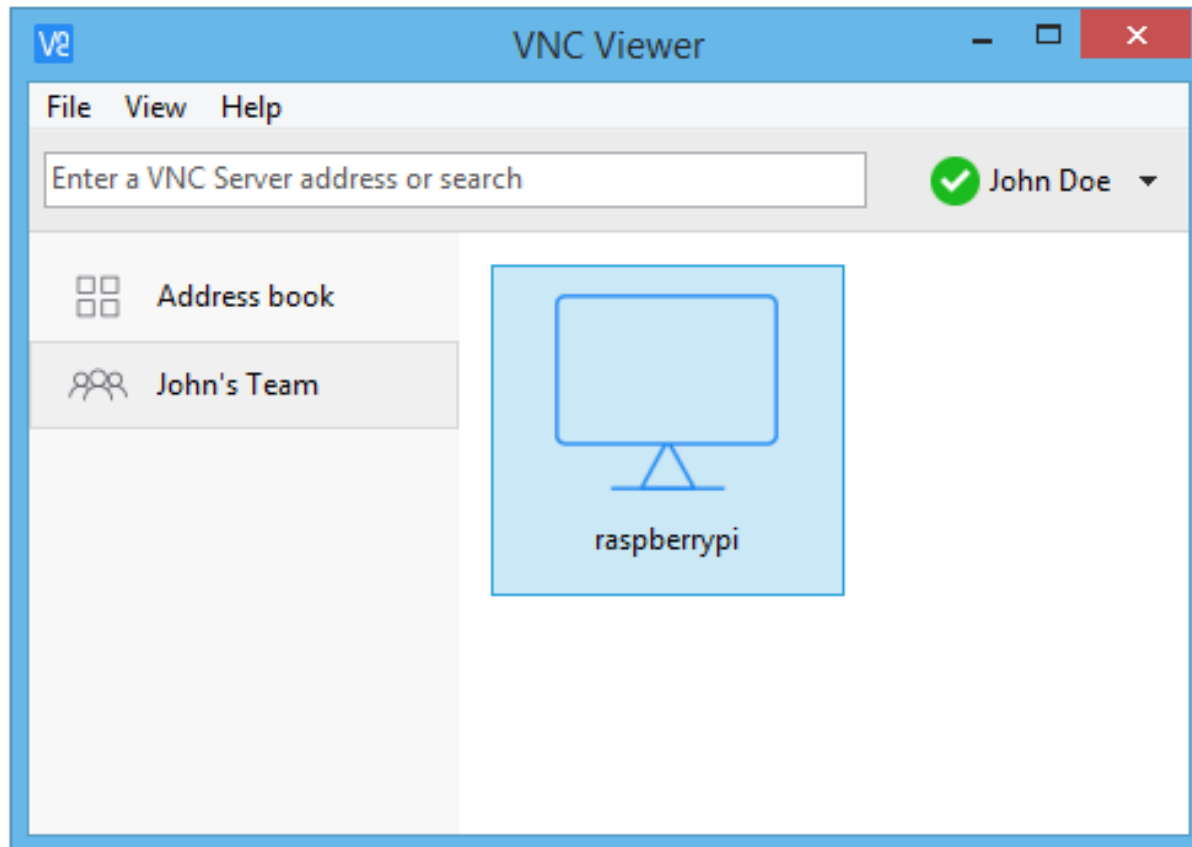
Establishing a cloud connection

- ▶ Sign up for a RealVNC account by entering your email address.
- ▶ On your Raspberry Pi, select **Licensing** from the VNC Server status menu, choose **Sign in to your RealVNC account**, and enter your new account email and password.





- ▶ On the device you will use to take control, run VNC Viewer and sign in using the same account credentials.
- ▶ In VNC Viewer, a connection to your Raspberry Pi automatically appears under the name of your team. Simply tap or double-click to connect:



Authenticating to VNC Server

- ▶ Enter the user name and password you normally use to *log on* to your user account on the Raspberry Pi.
- ▶ By default, these credentials are pi and raspberry.

