

# THE RASPBERRY PI PLATFORM AND PYTHON PROGRAMMING FOR THE RASPBERRY PI

By:

Mohamed Aziz Tousli

Saifeddine Barkia

## ABOUT (I)

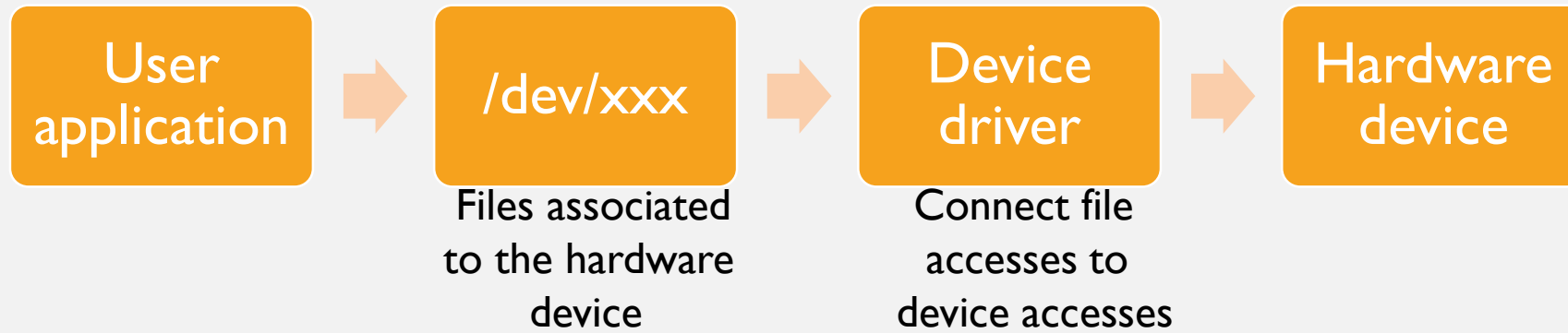
- ✓ ARM microprocessor: Brodcrom ARM Cortex A7
- ✓ 40 GPIO Pins (General Purpose Input Output)
- ✓ 4 USB Ports (Keyboard, Mouse)
- ✓ HDMI Port
- ✓ Ethernet Port
- ✓ Micro SD Slot (OS)

- PS: ARM design processors and sell its license, they don't built them – ARM Intellectual Property
- ARM Processor Family:
  - Classic Processors (ARM)
  - Embedded Processors (Cortex)
  - Application Processors (Cortex)

Raspberry Pi	Arduino
OS (Libraries, Functions)	No OS
Faster Processor (1,4GHz)	Slower Processor (16MHz)
64 bit	8 bit
More Memory	Less Memory
Lower I/O Voltage (3,3V) <b><u>SENSIBLE</u></b>	Higher I/O Voltage (5V)

## ABOUT (2)

- ❑ Text-based interface (console) vs Graphic interface (desktop)



- ❑ NOOBS: New Out Of Box Software → Raspbian (OS Linux-based)
- ❑ raspi-config: tool to setup different options
- ❑ Overclocking: Increase the clock frequency → Increase the internal voltage level (15% of processor)
  - ❑ Quicker execution of instructions (one instruction per clock)
  - ❑ Signals have shorter time to travel (reduce time over a clock cycle)
  - ❑ Temperature of device increases → Shortens device life

# LINUX – RASPBIAN

- ❑ Shell: Text-based user interface that executes commands
- ❑ Bash (Bourne again shell): Default shell for Raspbian → LXTerminal (vs Terminal)
- ❑ **man commandName** #Manual of a command
- ❑ **pwd** #Current directory
- ❑ **cd ( ;arg ;.. ;path)** #Change directory
- ❑ **ls ( ;-l)** #Give contents of current directory → (d:directory,-file) (user/group/other) (rwx:read/write/execute)
- ❑ **Mkdir** #Make directory ; **rmdir ( ;-r {if not empty})** #Remove directory
- ❑ **nano file** #Create a nano editor file (**sudo apt-get install nano**)
- ❑ **cat ; head ; last ; tail fileName** #Print file content
- ❑ **cp originalName copyName** #Copy file
- ❑ **mv fileName directory** #Move file ; **mv fileName newFileName** #Rename file
- ❑ **sudo instruction** #Switch user account to root account → Gain the highest permission level
- ❑ Processor: Execution of a program; Background processor vs Foreground processor
- ❑ **ps** #Open task monitor ; PID: Process ID ; **kill PID** #End a processor ; **shutdown** #Close a processor
- ❑ GUI: Graphic User Interface ; File manager: Regular file interface; **startx** #Start the GUI

# GPIO (I)

## ☐ Pins:





















- ☐ **Power:** 3,3V (1,17) & 5V (2,4)
- ☐ **Ground:** 6, 9, 14, 20, 20, 30, 39
- ☐ **GPIO:** General Purpose Input Output (Only 3,3V)
- ☐ **GPMF:** General Purpose Multi Function (SDA, SCL, UART, SPI)
- ☐ `import Rpi.GPIO as GPIO` #Call GPIO Library
- ☐ `GPIO.setmode(GPIO.BOARD)` #Broadway numbering (In the circle)
- ☐ `GPIO.setmode(GPIO.GPIO.BCM)` #Broadcom numbering (In the box) (It changes from an Rpi to another)
- ☐ `GPIO.setup(pinNumber,mode)` #mode=GPIO.OUT/GPIO.IN #pinMode
- ☐ `GPIO.output(pinNumber,value)` #value=True/False #digitalWrite
- ☐ `while True` #void loop{
- ☐ `import time; time.sleep(numberInSeconds)` #delay
- ☐ `value=GPIO.input(pinMode)` #digitalRead
- ☐ PS: No analogRead, no analogWrite
- ☐ `sudo python3 test.py` #Execute a python script

## GPIO (2)

- ❑ PWM (Pulse With Modulation)
- ❑ Duty cycle = Fraction of the duration of high (of voltage)
- ❑ `pwm_obj=GPIO.PWM(pinNumber,frequencyInHz)`
- ❑ `pwm_obj.start(dutyCycle)` #Generate PWM to the pin #dutyCycle 0->100
- ❑ `pwm_obj.ChangeDutyCycle(dutyCycle)`
- ❑ PS: PWM frequency is not accurate because of OS (off by over 50% at 10kHz)
- ❑ Frequency control: (For more accuracy)

```
while True:
    GPIO.output(18, True)
    time.sleep(0.5)
    GPIO.output(18, False)
    time.sleep(0.5)
```

# Raspberry Pi2 GPIO Header

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I <sup>2</sup> C)		DC Power 5v	04
05	GPIO03 (SCL1 , I <sup>2</sup> C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I <sup>2</sup> C ID EEPROM)		(I <sup>2</sup> C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Early Models

Late Models

# GUI-BASED PROGRAMS

- ❑ Widgets: Visual entities you can interact with (button, menu..)
- ❑ Event loop: Wait for an event → Execute (by the user) if there is an event → Wait again
- ❑ `from Tkinter import *` #Python library for widgets
- ❑ `root=Tk()` #Create a window on the screen
- ❑ `root.geometry('widthxheight')` #Give size of geometry
- ❑ `c=Canvas(root,width=W,height=H)` #Create a canvas
- ❑ `c.pack()` #Make canvas appear on the screen
- ❑ `r=c.create_rectangle(x,y,w,h,fill='color',outline='color')` //Create a rectangle
- ❑ `w=Scale(root,from_=min,to=max,orient=HORIZONTAL/VERTICAL,command=callBackFunction)`
- ❑ `def callBackFunction(duty)` #Function called when user changes scale #Duty is the value of the scale