TASK 1

By

K. Venkata Suresh Varma
Reg: 170953214

RASPBERRYPI AND ITS SPECS

- Raspberry Pi is a small computer which has an operating system in it
- Quad core Broadcom BCM2837 SoC (ARM processor).
- It has 40 GPIO pins , 4 USB ports , Micro SD card slot , Bluetooth 4.2 , Integrated WIFI .

RASPBERRY PI V/S ARDUINO

Raspberry Pi processor is faster 1.4 GHZ • Running an OS	Arduino is Slow 16MHZ
64 bit Processor	8 bit Processor
More Memory • 1 GB SRAM and Unlimited flash memory	Less Memory • 2KB SRAM and 32KB flash memory
Lower IO voltages 3.3 V	Higher Voltage 5V
Sensitive to Power	Control Operations and not sensitive to Power
Has an Operating System • Application • Library Functions • System calls • Microcontroller	Only for Simple Tasks ❖ Application to Microcontroller

GETTING STARTED WITH RASPBERRYPI

❖ Raspberry pi as an IoT device!
 Raspberry Pi can be used as an IoT device when we are interacting with the sensors and acurators.

❖ Setup of the Raspberry pi

- Plugging in the Monitor and Keyboard
- Getting an Operating system
- Installing an OS using NOOBS
- Raspi-Config
- Exploring options like Expand File system , Enable boot to desktop .
- Internationalization and Rastrack.
- Overclocking:
  Overclocking can increase the clock speed of the raspberrypi , which results in fast results but there can be high increase in temperature of the device.

OPERATING SYSTEM : LINUX

LINUX Basics:

The Shell:

- Interprets user input and execute commands.
- Text based user interface of an OS.
- Bash is the default shell for Raspian.
- Gives more precise control to the user
- Requires memorization for efficiency.

Console or Terminal:

- It is a Text entry and display device.
- Lx Terminal is used in the Raspian.

User name and Password:

Default:

Username: ‘pi’

Password: ‘raspberry’

Shell’s Path:

PATH is an environmental variable. It includes the most common program locations , such as /bin , /usr/bin/ , etc…

Command Syntax:

Linux is case sensitive.

Command [-argument] [-argument] [file]

```
Ex: ls       #List the files in current directory
    ls-l     #list files in long format
```

Getting help:

When we are stuck and need help we can use the 'man' command to display a command's manual.

Type 'man <command name>'

Ex: man ls #Get help on the ls command

man man #A manual about how to use manual

Commands for Navigating the File System:

a. pwd ("Print Workind Directory") #Shows the current location in the directory tree
b. cd ("Change Directory") #It returns to your hone directory.
   1. cd <dir_name> #changes current path to specified directory name
   2. cd ~ : '~' is an alias for home directory
   3. cd.. : Move up one directory
   4. cd- :Returns to previous directory
c. ls #List all the files in current directory

   1. ls <dir_name> #List all the files in specified directory
   2. ls -l : List files in long format
   3. ls -a : List all files including hidden files.
   4. ls -ld : A long list of directory
   5. ls /usr/bin/d* : Lis all the files starting with letter 'd'

Files and Directories:

Commands that can be used to find information about files and manipulate them.

a. touch: Creates a new empty file
b. file : To find what kind of a file it is.
c. cat : Display the content of the file
d. head : Display the first few lines
e. tail : Display the last few lines
f. cp : copies the file from one location to another.
g. mv : Moves a file to a new location or renames it.
h. rm : Delete a file
i. mkdir : Make directory
j. rmdir : Remove directory. (Only when the file is Empty)

File Permissions:

- Files have owners.
- They have access permissions . Read(r) , Write(w) , Execute(x).
- Different permission can be assigned according to type.
  User: The file Owner

Group: A Permission group

Other : All users

Root Account:

- The Root account have highest permission level
- Key files and directories are only accessible by root.
- Sometimes you need root privileges to Install a program and change the operating system.
- Use the ‘sudo’ command to gain root permission for a single command

Processes:

Process is execution of a program.

Linux allows multiple processes to run concurrently

Foreground vs Background Processes.

User can only see the Foreground processes , where as the background processes like reading an email , downloading a file , waiting for network connection and checking for viruses is taken care by the OS.

- 'ps' command is used to display the current running commands along with process ID.
- 'kill' is used to terminate a process.

We may have to choose a text editor inorder to write the commands.

Ex: Emacs , vi , vim , Nano.

Nano is a common text editor.

Shut down:

We should not just unplug the Linux machine.

Proper shutdown procedure is needed to place data structures in a good state.

Use the command "Shutdown" to turn off the machine which flushes all the buffers and closes all the files.

Raspberry Pi as an IoT device:

- Can be used as a laptop/Desktop.
- To use it as a part of an IoT device programming is needed.

Many languages can be used but **Python** is the best choice.

PYTHON LANGUAGE

High level language , easy to use.

- Do not need to explicitly declare data types
- No pointers are present
- Object oriented programming , includes classes and objects.
- Slow when compared to c & c++ as it is interpreted but not compiled.
- Hard to meet deadlines.

Programming Environment:

Two possible environments

- Integrated Development Environment (IDE).

Executing Python code:

1. Interactive: Execute lines interactively in the python console.
2. Batch: Execute an entire python program.
3. To execute from IDLE , type the code requires and press Run and then press Run Module. The code will be executed.

PYTHON OVERVIEW

- Python has number of built-in data types such as integers, floats, complex numbers, strings, lists, tuples, dictionaries, and file objects.

- These can be manipulated using language operators, built-in functions, library functions, or a data type's own methods.

- Python provides conditional and iterative control flow through an if-elif-else construct along with while and for loops. It allows function definition with flexible argument-passing options.

- Exceptions (errors) can be raised using the raise statement and caught and

handled using the try-except-else construct.

Number Types

Integers
1, –3, 42, 355, 888888888888888, –7777777777
Floats—3.0, 31e12, –6e-4
Complex numbers—3 + 2j, –4- 2j, 4.2 + 6.3j
Booleans—True, False

- Arithmetic operators: + (addition), – (subtraction), * (multiplication), / (division), ** (exponentiation), and %(modulus).
- Note that integers are of unlimited size
- Integers

```
>>> x = 5 + 2 - 3 * 2
>>> x
1
>>> 5 / 2
2.5
>>> 5 // 2
2
>>> 5 % 2
1
>>> 2 ** 8
256
>>> 1000000001 ** 3
1000000003000000003000000001
```

Floats

```
>>> x = 4.3 ** 2.4
>>> x
```

```
        33.137847377716483
        >>> 3.5e30 * 2.77e45
        9.6950000000000002e+75
        >>> 1000000001.0 ** 3
        1.000000003e+27
```

Complex numbers

```
    >>> (3+2j) ** (2+3j)
    (0.68176651908903363-2.1207457766159625j)
    >>> x = (3+2j) * (4+9j) //
    [(3*4)+(3*9j)+(2j*4)+(2j*9j)]□12+27j+8j+18j2
    >>> x
    12+35j+18*(-1)□-6+35j
    (-6+35j)
    >>> x.real
    -6.0
    >>> x.imag
    35.0
```

- Operators that could be used on complex numbers are +, -, *, /.
- % cannot be used

Boolean

```
>>> x = False
True and False returns False
True or False returns True
>>> not x
True
>>> y = True * 2 // True is 1, False is 0
>>> y
2
```

‘#’ is used to write the comments in the python language.

LISTS

```
[]          #Empty list
[1]
[1, 2, 3, 4, 5, 6, 7, 8, 12]
[1, "two", 3, 4.0, ["a", "b"], (5,6)]
```

• A list can contain a mixture of other types as its elements, including strings, tuples, lists, dictionaries, functions, file objects, and any type of number.

```
>>> x = ["first", "second", "third", "fourth"]
>>> x[0]
'first'
>>> x[2]
'third' can be indexed from its front or back.
>>> x[-1]     #From the end of the list.
'fourth'
>>> x[-2]
'third'
>>> x[1:-1]
```

```
['second', 'third']   # prints values from position 1
to position (-1)-1=-2

>>> x[0:3]
['first', 'second', 'third']
>>> x[-2:-1]
['third']
>>> x[:3]
['first', 'second', 'third']
>>> x[-2:]
['third', 'fourth']
```

• Can add, remove, and replace elements in a list or to obtain an element or a new list that is a slice from it:

```
>>> x = [1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> x[1] = "two"
>>> x[8:9] = []
>>> x
[1, 'two', 3, 4, 5, 6, 7, 8]
>>> x[5:7] = [6.0, 6.5, 7.0]
>>> x
[1, 'two', 3, 4, 5, 6.0, 6.5, 7.0, 8]
>>> x[5:]
[6.0, 6.5, 7.0, 8]
```

Built in function:

len(x)   : Returns the number of elements in list x
max(x) : returns the maximum number in the list x

min(x) : returns the minimum number in the list x

operators:
obj in x : Returns True if object obj is in list x. Otherwise, returns False

list1+list2 :returns a new list and does not modify list1 or list2

list1*n :Returns list1 repeated by n times. Does not modify list1

del list1[index] : removes(deletes) the element at the index position in the list

• Operators + and * creates new list.

```
>>> x = [1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> len(x)
9
>>> [-1, 0] + x
[-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> x.reverse()
>>> x
[9, 8, 7, 6, 5, 4, 3, 2, 1]
```

LIST Methods

list1.append(object) : appends a single object

list1.count(value) : Counts the number of occurrences of value in the list1
list1.reverse() : reverses the list
list.insert(index,object) :Inserts an object at index position
list1.index(object) : returns index of an object
list1.remove(value) : Removes the first occurrence of the value from list

list1.pop() : Removes the last item from the list

list1.sort() : Sorts the list having similar kind ofobjects
list1.clear() : Makes the list empty

STRINGS

- Strings can be delimited by single (' '), double (" "), triple single (''' '''), or triple double (""" """) quotations and can contain tab (\t) and newline (\n) characters.
- Strings are also immutable.
  - The operators in, +, and *
  - Built-in functions : len, max, and min

  Examples:
```
>>>"hello" in "hello world"
True
>>> "Hello"+ "world"
‘Hello world”
```

```
>>>"Hello"*3
“HelloHelloHello”
>>>len(“something”)
9
>>> max("Returns the character with
greatest ascii value in this string")
‘w’
>>> min(“"Returns the character with least
ascii value in this string”)
```

• The print function outputs strings. Other Python data types can be easily converted to strings and formatted:

```
>>> e = 2.718
>>> x = [1, "two", 3, 4.0, ["a", "b"], (5, 6)]
>>> print("The constant e is:", e, "and the
list x is:", x)
The constant e is: 2.718 and the list x is: [1,
'two', 3, 4.0, ['a', 'b'], (5, 6)]
>>>var1 = 'Hello World!‘
>>>var2 = "Python Programming“
>>> print "var1[0]: ",var1[0]
>>>print "var2[1:5]: ", var2[1:5]
```

• Objects are automatically converted to string representations for printing.

• The % operator provides a formatting capability

```
>>> print("the value of %s is: %.2f" % ("e",
e))
the value of e is: 2.72
```

- Reverse of string s: s[::-1]
- Reverse of a number n: int(str(n)[::-1])

FUNCTIONS

Functions are defined using the key word def.

Ex:

```
def test():
        print('A test function')
>>>test()
A test function.
```

All the instructions in a function are indented.

Functions can take arguments and return values.

Ex:

```
def area (r):
        return 3.14*r*r
>>>area(6)
113.04
```

CONTROL FLOW

Statements that change the order in which lines of code are executed.

The if-elif-else statement

• The elif and else clauses are optional
• There can be any number of elif clauses.
• Python uses indentation to delimit blocks.

IF clause

```
>>>x = 5
if x < 5:
    y = -1
    z = 5
elif x > 5:
     y = 1
     z = 11
else:
     y = 0
     z = 10
print(x, y, z)
Output: 5,0,10.
```

FOR loop

```
x=[0,1,2,3,4,5] #list
for i in x:
        print(i)

Output: 012345

for i in [0,1,2,3,4,5]:
        print(i)

for i in "abc","def","xyz":
        print(i)
        Output:
```

```
abc
def
xyz
```

```
for i in 1,5.5,5+8j,True,"str":
        print(i)
```

Output
:

```
1
5.5
(5+8j)
True
str
```

```
                                    #for each letter in string
for letter in 'Python':
        print('Current Letter :', letter)

fruits = ['banana', 'apple', 'mango']
for fruit in fruits:
        print('Current fruit :', fruit)
```

Output:

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n
Current fruit : banana
Current fruit : apple
Current fruit : mango

for x in range(0, 3):
        print("We're on time",x)

Output:
We're on time 0
We're on time 1
We're on time 2

range(stop) # returns range object
range(start,stop[,step]) # returns range object

for x in range(1,10,2):
        print(x)

Output : 1 3 5 7 9
```

While Loop

```
count = 0
        while count <= 10:
                print('The count is:', count)
                count = count + 1
                print ("Good bye!“)
```

output: Good is printed 10 times.

GPIO PINS IN RASPBERRY PI

GPIO: General Purpose Input Output pins.

Raspberry pi contains 40 GPIO pins.

There are some dedicated power and ground pins

- 3.3 V (pins 1,17) , 5V (2,4) , Gnd (6,9,14,20,30,39)

Input and output are 0V and 3.3V.
The pins are given the names as follows
GPIO 2 (3) #3rd pin
GPIO 3 (5) #5th pin
GPIO 4 (7) #7th pin

- Pins labelled with “GPIOxx” can be used as general purpose Input Output pins.
- Some pins are multi-functional and they are labelled with extra labels.

UART pins

UART is an asynchronous serial communication protocol.

It takes bytes of data and transmits the individual bits in a sequential fashion.

UART is not clocked.

The pins used are GPIO 14 (Tx) and GPIO 15 (Rx).

Tx : Transmit

Rx : Receive

PROTOCOL PINS

SPI Communication pins:

Serial Peripheral Interface is a protocol in which the data is transferred between microcontrollers and small peripherals like registers , SD card etc..

There is a separate clock and data line along with select line by which we can choose the device that we want to communicate to.

The communication is Synchronous as it uses a clock .

The clock is an oscillating signal that tells exactly when to sample the data on the data line.

The side that produces the Clock is the Master and the It communicates with the slaves.

The pins used in this communication are:

MOSI(19) , MISO(21) , SCLK(23) , CE0(24) , CE1(26).

The clock is generated on the SCLK pin at the master and Master Out Slave In(MOSI) pin is used to send the data from master to slave .

Master In Slave Out(MISO) pin is used to communicate from slave to the master using the same clock generated by the master.

The 2 pins CE0 and CE1 are used to enable the chips.

Advantages :

- Supports Multiple slaves.
- The receive hardware can be simple shift register.
- It's faster than the Asynchronous serial

Disadvantages:

- Requires more signal lines that other communication methods.
- Communication must be well defined in advanced. SPI is used to communicate with the sensors that have very specific command structure.
- Separate slave select lines must be present to communicate with different slaves

I2C Pins:

Inter – Integrated Circuit Protocol is used to communicate among multiple masters and slaves present in a single circuit.

It's a Serial communication protocol b/w two chips or drives that relatively share same clock.

It's a Synchronous protocol , so they have to be close enough to share the same clock.

Unlike SPI it only requires 2 wires for the communication.

Each I2C bus contains two signals SCL and SDA.

SCL: The clock signal generated by the current bus master and the slave device may force the clock to be low , so as to delay the master from sending more data.

SDA: The data signal where the data is sent in Sync with clock.

The pins that are enabled on the GPIO are

GPIO2 (3)

GPIO3 (5)

The Communication in I2C is more complex than UART or SPI sol.

GPIO ACCESS IN PYTHON

In Python we use the GPIO libraries for controlling and the observing the pins in the Raspberry Pi.

The command “import Rpi.GPIO as GPIO” is used to import the GPIO library with the name GPIO.

As it is a GPIO operation we need the root permission to run the program which can be achieved by using the “sudo” keyword to access the root.

PIN Numbering Modes:

Two ways to refer to the gpio pins.

1. The number of the pins in their order on the board.
   GPIO.setmode(GPIO.BOARD) command is used to select the pin numbering mode.
2. The Broadcom Soc number which may vary according to the device version.
   GPIO.setmode(GPIO.BCM)

Pin Direction and assignment:

GPIO.setup(13 , GPIO.OUT) # command used to setup 13th pin as the output pin.

GPIO.output(13,TRUE) # command to set the 13th pin High.

Python Example for Blinking LED:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(13,GPIO.OUT)
while True:
        GPIO.output(13,TRUE) # set to high
        Time.sleep(1) #for delay of 1 sec.
        GPIO.output(13,FALSE) # set to low
        Time.sleep(1)
```

Reading Input pins:

```
GPIO.setup(13,GPIO.IN)
#setting the pin direction as input
Value=GPIO.input(13)
```

Read the value on input pin.

It only reads the digital value as there is no analog Read equivalent and there is no Analog to Digital convertor.

Pulse Width Modulation:

Pulse width modulator can produce a square wave that can vary the width of high or low signal without changing the frequency of the wave.

It produces signals with same frequency but different duty cycle.

This phenomena can be used to control an analog device from a digital circuit by varying the frequency and duty cycle of the signal.

PWM Initialization:

The Command “GPIO.PWM(a,b)” is used to initialize the PWM with “a” as the Pin number and “b” as the frequency of the signal.

PWM functions are present in the GPIO library.

```
pwm_obj=GPIO.PWM(10,100) #creates an object of pwm on 10th pin with frequency 100.
```

```
pwm_obj.start(100) #generates the PWM signal with 100% duty cycle.
```

```
pwm_obj.ChangeDutyCycle(50) #command used to change the duty cycle to 50%.
```

PWM frequency is not accurate off by over 50% duty cycle and 10KHz.

Frequency cannot be changed as it cannot be controlled easily as there is no function to do so.

We can do it manually by the code:

```
while True:
        GPIO.output(10,TRUE)
        time.sleep(0.5)
        GPIO.ouput(10,FALSE)
        time.sleep(0.5)
```

This generates a square wave with frequency 1Hz.

GRAPHIC USER INTERFACE(GUI)

We can use the GUI to access the GPIO as it is supported by the operating system in the Raspberry Pi , and not in the Arduino.

In GUI there are various visual entities that user can interact with , like Buttons, menus, sliders, scrollbars, drawing surfaces for drawing.

The execution of these processes or jobs is controlled by the user and not by the programmer.
Ex: The file is opened only when the button is clicked , the OS has to wait till the user clicks the button.
Event Loop:
Typically a program will wait for the user to activate one of its widget
Ex: Push a button , select a menu , draw on a drawing surface etc..
It will do all the things in sequence and wait for the user for the next instruction.

Tkinter Library:
It is a GUI Library.

Tkinter Library provides tools for writing programs that use graphics.
The GUI widgets like buttons , menus , labels , scrollbars, etc.. are accessed through the Tkinter library.
A canvas widget is a widget on which we can create arbitrary drawings using lines, squares, rectangles , ovals, images , text , etc..

Sample Tkinter Program:

```
from Tkinter import*  #Imports all the classes from
the Tkinter Library.
 root = TK() #Construction function for a window.
root.geometry(‘800 X 600’) #specifies the window
size.
c=Canvas(root,width=800,height=600) # Canvas
function creates the window with specified
geometry.
c.pack()  # pack fuctn. Makes it appear on the
screen.
r= c.create_rectangle(0,0,50,50,fill= ‘red’ , outline
= ‘red’)
```

```
# creates an obj r with required specs in the canvas
and opens the window with a red square in the
corner.
```

INTERACTION:
The widgets can be used to make the changes in the registers in the Raspberry Pi i.e when a button is pressed , a value in the GPIO pins can be varied.

Example:
Scale Widget.
Code:

```
from Tkinter import*
master=Tk()
w=Scale(master, from_=0, to=100,
orient=HORIZONTAL)
w.pack() # To make it appear on screen.
```

This draws a Scale widget . Slider can be moved by the user.
When we include a command in the Scale function as argument , we can use the command to change a specific value in the gpio.
Ex:

```
w=Scale(master, from_=0,
to=100,orient=HORIZONTAL, command =
update)
#here the command is a call back function.
def update(duty):
        pwm.ChangeDutyCycle(float(duty))
```

When the scale is changed , the update function is called back making changes in the Duty cycle.