To use Raspberry Pi 4,

Equipment required

Board

Sd card formatter

Raspberry pi os

Ubunutu

Etcher

Peripherals needed

5V adapter

Sd card

Sd card slot

External keyboard

Linux Commands

* pwd-cuurent working directory
* ls-to enter inside a directory to access file
* mkdir #name#- to create another directory
  + - we can create directory inside another
* cd #name#- to change the directory to another name
  + - to get inside a directory
    - cd .. - to go back to previous directory
    - cd ~-Home directory
    - cd /-to go to main starting directory/ root folder
* nano #filename#- to create a text file
  + - to exit from nano use Ctrl+X
    - to delete a line from a text file use Ctrl+K
    - to select a line use Ctrl+6
    - to paste a line ctrl+U
    - to search a particular text use ctrl+W
    - to save text in a file Ctrl+O
* cat #filename#- to display the contents of a file
* rm #filename#- remove the file from a directory
* rmdir #directorname#- to remove directory
* python #filename#- to run a python file
* wget #URL#- to download an image from a particular website
* git clone #URL# - to download github file into raspberry file
* sudo- helps to become super user
* sudo apt-get #filename# - to install a system or root file
* sudo apt-get remove #filename# - to remove a root file
* root directory files
* boot file contains all the program files required to start up a device
* bin contains all the commands required for linux
* etc(editable text files)-file which is editable
* sys-contains all system hardware files
* usr-contains user files

Virtual Network Computing (VNC) is a cross-platform technology that allows you to remotely control your Raspberry Pi's desktop interface from another computer or mobile device. This means you can control your Raspberry Pi from anywhere without needing a monitor.

git is a software which is used to connect raspberry pi to use github repositories

permission is required to change system files which can only be done by super user or root

sudo apt-get arduino- as this is system file we use sudo to install

Interfacing Raspberry PI GPIO's

40 pins(no analog pins so if we use analog sensors we use ADC chip for that)

enable communication with SPI(serial peripheral Interface)

SPI examples RFID,TFT LCD Color Screen

Device Comm Protocols Include UART,SPI,I2C,Serial Communication etc

Python library- RPi.GPIO

There are two types of mode to use in GPIO library

GPIO.BOARD- refers to pin's physcial location on the header

GPIO.BCM(broadcomm)- General Purpose Input/Output Pin Numbering

to set any mode we use GPIO.setmode(GPIO.BCM)-to set this mode

To detect which pin numbering system has been set (for example, by another Python module): mode = GPIO.getmode()

To classify a particular GPIO pin as input or Output, We use

I/p-GPIO.setup(pinnumber,GPIO.IN)

o/p-GPIO.setup(pinnumber,GPIO.OUT)

To give Output/Write at a particular pin, GPIO.output(pinnumber,GPIO.high)

GPIO.output(pinnumber,GPIO.low)

In Raspberry Pi, to give analog output we use PWM method

x=GPIO.PWM(pIn Number, Frequency)

x.start(50) ; PWM wave with 50% duty Cycle

x.ChangeDutyCycle(75) ;Changes Duty Cycle to 75%

By changing Duty Cycle we can provide variable Voltage

to take input from sensors Into Raspberry Pi,

x=GPIO.input(pin)

dir() returns the list and attributes of any object

To provide delay we use time module, time.delay(#time duration) - this helps provide a delay

DHT11 Sensor(used to measure Temp and Humidity)

The python library used for this is Adafruit\_DHT sensor library

Blinking LED Python Code

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BCM)

GPIO.setup(21, GPIO.OUT)

while (True):

GPIO.output(21, GPIO.HIGH)

time.sleep(1)

GPIO.output(21, GPIO.LOW)

time.sleep(1)

Traffic Lights Control System Python Code

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

GPIO.setup(36, GPIO.OUT)

GPIO.setup(38, GPIO.OUT)

GPIO.setup(40, GPIO.OUT)

while(True):

GPIO.output(36, GPIO.HIGH)

GPIO.output(38, GPIO.LOW)

GPIO.output(40, GPIO.LOW)

time.sleep(1)

GPIO.output(36, GPIO.LOW)

GPIO.output(38, GPIO.HIGH)

GPIO.output(40, GPIO.LOW)

time.sleep(1)

GPIO.output(36, GPIO.LOW)

GPIO.output(38, GPIO.LOW)

GPIO.output(40, GPIO.HIGH)

time.sleep(1)

Working with Buzzer

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

GPIO.setup(40, GPIO.OUT)

while (True):

GPIO.output(40, GPIO.HIGH)

time.sleep(1)

GPIO.output(40, GPIO.LOW)

time.sleep(1)

Working with DHT11 sensor code

import Adafruit\_DHT

While (True):

h,t=Adafruit\_DHT.read\_retry(11,21)

print("Temp is",t)

print("Humidity is",h)

Thinkspeak is useful for visualize,analyze sensor data as thinkspeak also uses MATLAB

Thinkspeak supports two data protocols HTTP,MQTT to exchange data between hardware and thingspeak

Works with many different programming languages such as Python,Arduino,C,C++,Javascript

Requests Library:

The requests library in Python is a popular HTTP library used for making HTTP requests to interact with web servers. It simplifies the process of sending HTTP requests and handling responses, making it easier for developers to work with APIs and web services.

We use this requests library for sending / retrieving data from IoT Devices.

This library provides a user-friendly interface to perform various HTTP methods such as GET, POST, PUT, DELETE, etc., allowing you to fetch data from URLs, send data to web servers, and interact with APIs.

To install this library use, pip install requests

Python code for sending temperature and humidity values to thingspeak cloud

import Adafruit\_DHT as dht

import time

import requests

while(True):

h,t=dht.read\_retry(11,21)

print("Temperature",t)

print("Humidity",h)

time.sleep(1)

url="https://api.thingspeak.com/update?api\_key=FF2QQAK4DCGGEGMB&field1=%s&field2=%s"%(t,h)

print(url)

requests.get(url)

IFTTT(If this then that)

IFTTT is a powerful platform that enables you to create applets—small programs that connect various online services and devices to perform automated actions based on specific triggers.

These applets can automate a wide range of tasks, from simple data logging to more complex integrations between different services.

Code: Sending email alerts based on Motion Detection

import RPi.GPIO as GPIO

import requests

GPIO.setmode(GPIO.BCM)

GPIO.setup(21,GPIO.IN)

url="https://maker.ifttt.com/trigger/motion\_detected/json/with/key/mkK5u4TfiAeqVhv7-LeXBA9fdrBW7-C6JcI4DansRye"

while(True):

x=GPIO.input(21)

if(x==1):

print("sending email")

r=requests.get(url)

print("response: ",r.text)

else:

print("No motion detected")

to get an URL we use ifttt and create another automated mail

SMTP(Simple Mail Transfer Protocol) is a TCP/IP protocol which is used to send emails between servers using internet

SMTP along with TLS(transport layer Protocol) also called SMTPS helps in

Emails encryption

Email Authentication

Email Data Tampering Protection

Code: Sending Email Alerts using SMTP Protocol

import smtplib #smtplib is used for sending emails

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BCM)

GPIO.setup(21,GPIO.IN)

s = smtplib.SMTP('smtp.gmail.com', 587)

s.starttls()

s.login("your gmail","app password")

message = "Motion detected"

while(True):

x=GPIO.input(21)

print(x)

time.sleep(2)

if(x==0):

s.sendmail("your gmail", "to email address", message)

s.quit() #to quit from smtp

AWS IOT core

used for

managed cloud service

connect devices securely

transfer data

Data protocols used are MQTT HTTP loraWAN

IOT products

Amazon Dynamo DB is a storage database that uses a documented oriented JSON(javascript object notation) data model

Amazon SNS(simple notification service) to send emails or sms

Amazon S3(simple storage service) provides unlimited storage for various obejct types

AWS lambda- serverless cloud computing service

Amazon Quicksight- Cloud based BI tool used to build visualizations perform analysis

An IOT thing refers to representation and record of your physical device in the cloud

Code: Interfacing RGB Led

1. # Import the RPi.GPIO library for working with the Raspberry Pi GPIO pins
2. import RPi.GPIO as GPIO
4. # Import the sleep function from the time module to introduce delays
5. from time import sleep
7. # Define the GPIO pin numbers for red, green, and blue LEDs
8. rPin = 16
9. gPin = 20
10. bPin = 21
12. # Set the GPIO mode to BCM numbering
13. GPIO.setmode(GPIO.BCM)
15. # Set up the GPIO pins for output
16. GPIO.setup(16, GPIO.OUT) # Red LED
17. GPIO.setup(20, GPIO.OUT) # Green LED
18. GPIO.setup(21, GPIO.OUT) # Blue LED
20. # Define a function to set the color of the RGB LED by turning on/off respective pins
21. def setColor(x, y, z):
22. GPIO.output(rPin, x) # Set the state of the red LED pin
23. GPIO.output(gPin, y) # Set the state of the green LED pin
24. GPIO.output(bPin, z) # Set the state of the blue LED pin
26. # Enter an infinite loop
27. while True:
28. # Set the LED color to Red
29. setColor(0, 1, 1) # Red
30. sleep(2) # Pause for 2 seconds
32. # Set the LED color to Green
33. setColor(1, 0, 1) # Green
34. sleep(2) # Pause for 2 seconds
36. # Set the LED color to blue
37. setColor(0, 0, 1) # Blue
38. sleep(2) # Pause for 2 seconds

#### *Python Program: Smart Weather Umbrella Complete Project*

1. # Import necessary libraries
2. import RPi.GPIO as GPIO
3. from time import sleep
4. import requests
6. # Define GPIO pin numbers for LEDs
7. rPin = 16
8. gPin = 20
9. bPin = 21
11. # Set the GPIO mode to use BCM numbering
12. GPIO.setmode(GPIO.BCM)
14. # Set up the GPIO pins as outputs
15. GPIO.setup(rPin, GPIO.OUT)
16. GPIO.setup(gPin, GPIO.OUT)
17. GPIO.setup(bPin, GPIO.OUT)
19. # Function to set the color of the RGB LED
20. def setColor(x, y, z):
21. GPIO.output(rPin, x)
22. GPIO.output(gPin, y)
23. GPIO.output(bPin, z)
25. # Main loop
26. while True:
27. # Ask the user to input the city name
28. city = input("Enter city name: ")
30. # Create the URL for the OpenWeatherMap API request
31. url = "https://api.openweathermap.org/data/2.5/weather?q=%s&appid=4561f60c5669a8cbb4952b2e7640b9db" % city
33. # Send a GET request to the OpenWeatherMap API
34. r = requests.get(url)
36. # Print the response from the API
37. print(r)
39. # Parse the JSON response
40. s = r.json()
42. # Extract temperature and humidity data from the JSON response
43. t = s["main"]["temp"]
44. h = s["main"]["humidity"]
46. # Print temperature and humidity information
47. print("Temperature: ", t)
48. print("Humidity: ", h)
50. # Set the color of the RGB LED based on temperature
51. if t > 285 and t <= 295:
52. setColor(1, 0, 1) # Green color
53. elif t > 295 and t <= 302:
54. setColor(1, 1, 0) # Blue color
55. else:
56. setColor(0, 1, 1) # Red color