DCS
Power To Design The Desired

CENTER FOR SKILL AND ENTREPRENEURSHIP DEVELOPMENT

(CSED)

INDUSTRIAL INTERNET OF THINGS

PROGRAM CODE: TERMINAL COMMANDS AND PYTHON PROGRAMS

PROGRAM NAME: IIOT3

+91 20 27 276939

+91 79722 67361

425, Pride Purple Square, Kalewadi Chowk,
Aundh Ravet Road, Wakad, Pune: 411057
Maharashtra, India

info@dcsplm.com
www.DYSMECH.com

DYSMECH COMPETENCY SERVICES PVT. LTD.

FILE SYSTEMS

ls

- The ls command lists the content of the current directory (or one that is specified). It can be used with the -l flag to display additional information (permissions, owner, group, size, date and timestamp of last edit) about each file and directory in a list format. The -a flag allows you to view files beginning with . (i.e. dotfiles).

cd

- Using cd changes the current directory to the one specified. You can use relative (i.e. cd directoryA) or absolute (i.e. cd /home/pi/directoryA) paths.

pwd

- The pwd command displays the name of the present working directory: on a Raspberry Pi, entering pwd will output something like /home/pi.

mkdir

- You can use mkdir to create a new directory, e.g. mkdir newDir would create the directory newDir in the present working directory.

rmdir

- To remove empty directories, use rmdir. So, for example, rmdir oldDir will remove the directory oldDir only if it is empty.

rm

- The command rm removes the specified file (or recursively from a directory when used with -r). Be careful with this command: files deleted in this way are mostly gone for good!

cp

- Using cp makes a copy of a file and places it at the specified location (this is similar to copying and pasting). For example, cp ~/fileA /home/otherUser/ would copy the file fileA from your home directory to that of the user otherUser (assuming you have permission to copy it there). This command can either take FILE FILE (cp fileA fileB), FILE DIR (cp fileA /directoryB/) or -r DIR DIR (which recursively copies the contents of directories) as arguments.

mv

- The mv command moves a file and places it at the specified location (so where cp performs a 'copy-paste', mv performs a 'cut-paste'). The usage is similar to cp. So mv ~/fileA /home/otherUser/ would move the file fileA from your home directory to that of the user otherUser. This command can either take FILE FILE (mv fileA fileB), FILE DIR (mv fileA /directoryB/) or DIR DIR (mv /directoryB /directoryC) as arguments. This command is also useful as a method to rename files and directories after they've been created.

touch

- The command touch sets the last modified time-stamp of the specified file(s) or creates it if it does not already exist.

cat

- You can use cat to list the contents of file(s), e.g. cat this File will display the contents of this File. Can be used to list the contents of multiple files, i.e. cat *.txt will list the contents of all .txt files in the current directory.

head

- The head command displays the beginning of a file. Can be used with -n to specify the number of lines to show (by default ten), or with -c to specify the number of bytes.

tail

- The opposite of head, tail displays the end of a file. The starting point in the file can be specified either through -b for 512 byte blocks, -c for bytes, or -n for number of lines.

chmod

- You would normally use chmod to change the permissions for a file. The chmod command can use symbols u (user that owns the file), g (the files group) , and o (other users) and the permissions r (read), w (write), and x (execute). Using chmod u+x *filename* will add execute permission for the owner of the file.

chown

- The chown command changes the user and/or group that owns a file. It normally needs to be run as root using sudo e.g. sudo chown pi:root *filename* will change the owner to pi and the group to root.

ssh

- ssh denotes the secure shell. Connect to another computer using an encrypted network connection. For more details see SSH (secure shell)

scp

- The scp command copies a file from one computer to another using ssh. For more details see SCP (secure copy)

sudo

- The sudo command enables you to run a command as a superuser, or another user. Use sudo -s for a superuser shell. For more details see Root user / sudo

dd

- The dd command copies a file converting the file as specified. It is often used to copy an entire disk to a single file or back again. So, for example, dd if=/dev/sdd of=backup.img will create a backup image from an SD card or USB disk drive at /dev/sdd. Make sure to use the correct drive when copying an image to the SD card as it can overwrite the entire disk.

df

- Use df to display the disk space available and used on the mounted filesystems. Use df -h to see the output in a human-readable format using M for MBs rather than showing number of bytes.

unzip

- The unzip command extracts the files from a compressed zip file.

tar

- Use tar to store or extract files from a tape archive file. It can also reduce the space required by compressing the file similar to a zip file.

- To create a compressed file, use tar -cvzf *filename.tar.gz* *directory/* To extract the contents of a file, use tar -xvzf *filename.tar.gz*

pipes

- A pipe allows the output from one command to be used as the input for another command. The pipe symbol is a vertical line |. For example, to only show the first ten entries of the ls command it can be piped through the head command ls | head

tree

- Use the tree command to show a directory and all subdirectories and files indented as a tree structure.

&

- Run a command in the background with &, freeing up the shell for future commands

wget

- Download a file from the web directly to the computer with wget. So wget https://www.raspberrypi.org/documentation/linux/usage/commands.md will download this file to your computer as commands.md

curl

- Use curl to download or upload a file to/from a server. By default, it will output the file contents of the file to the screen.

man

- Show the manual page for a file with man. To find out more, run man man to view the manual page of the man command.

Networking

ping

- The ping utility is usually used to check if communication can be made with another host. It can be used with default settings by just specifying a hostname (e.g. ping raspberrypi.org) or an IP address (e.g. ping 8.8.8.8). It can specify the number of packets to send with the -c flag.

nmap

- nmap is a network exploration and scanning tool. It can return port and OS information about a host or a range of hosts. Running just nmap will display the options available as well as example usage.

hostname

- The hostname command displays the current hostname of the system. A privileged (super) user can set the hostname to a new one by supplying it as an argument (e.g. hostname new-host).

ifconfig

- Use ifconfig to display the network configuration details for the interfaces on the current system when run without any arguments (i.e. ifconfig). By supplying the command with the name of an interface (e.g. eth0 or lo) you can then alter the configuration: check the manual page for more details.

SUPER USER COMMANDS

- Sudo apt install
- Sudo apt-get install
- Sudo apt-get autoclean
- Sudo apt install tree
- Sudo tree
- Sudo apt purge tree
- Sudo apt install pip
- Sudo apt install nano
- Sudo apt install spell
- sudo apt-get install python-rpi.gpio python3-rpi.gpio

- sudo rpi-update
- Sudo passwd root
- Sudo reboot
- Sudo shutdown -h now
- https://www.linux-magazine.com/Online/Features/Convert-an-Android-Device-to-Linux

DCS
Power To Design The Desired

PYTHON PROGRAMS

Topics :

- Basic Programs
- Array Programs
- List Programs
- String Programs
- Dictionary Programs
- Tuple Programs
- Searching and Sorting Programs
- Pattern Printing
- Date-Time Programs

Basic Programs:

1. Python program to add two numbers
2. Python Program for factorial of a number
3. Python Program for simple interest
4. Python Program for compound interest
5. Python Program to check Armstrong Number
6. Python Program for Program to find area of a circle
7. Python program to print all Prime numbers in an Interval
8. Python program to check whether a number is Prime or not
9. Python Program for n-th Fibonacci number
10. Python Program for Fibonacci numbers
11. Python Program for How to check if a given number is Fibonacci number?
12. Python Program for n\'th multiple of a number in Fibonacci Series
13. Program to print ASCII Value of a character
14. Python Program for Sum of squares of first n natural numbers
15. Python Program for cube sum of first n natural numbers

Array Programs:

1. Python Program to find sum of array
2. Python Program to find largest element in an array
3. Python Program for array rotation
4. Python Program for Reversal algorithm for array rotation
5. Python Program to Split the array and add the first part to the end
6. Python Program for Find reminder of array multiplication divided by n
7. Reconstruct the array by replacing arr[i] with (arr[i-1]+1) % M
8. Python Program to check if given array is Monotonic

List Programs:

1. Python program to interchange first and last elements in a list
2. Python program to swap two elements in a list
3. Python program to remove Nth occurrence of the given word
4. Python | Ways to find length of list
5. Python | Ways to check if element exists in list
6. Different ways to clear a list in Python
7. Python | Reversing a List
8. Python | Cloning or Copying a list
9. Python | Count occurrences of an element in a list
10. Python program to find sum of elements in list
11. Python | Multiply all numbers in the list
12. Python program to find smallest number in a list
13. Python program to find largest number in a list
14. Python program to find second largest number in a list
15. Python program to find N largest elements from a list
16. Python program to print even numbers in a list
17. Python program to print odd numbers in a List
18. Python program to print all even numbers in a range
19. Python program to print all odd numbers in a range
20. Python program to count Even and Odd numbers in a List
21. Python program to print positive numbers in a list
22. Python program to print negative numbers in a list

23. Python program to print all positive numbers in a range
24. Python program to print all negative numbers in a range
25. Python program to count positive and negative numbers in a list
26. Remove multiple elements from a list in Python
27. Python | Remove empty tuples from a list
28. Python | Program to print duplicates from a list of integers
29. Python program to find Cumulative sum of a list
30. Break a list into chunks of size N in Python
31. Python | Sort the values of first list using second list

String Programs:

1. Python program to check if a string is palindrome or not
2. Reverse words in a given String in Python
3. Ways to remove i'th character from string in Python
4. Python | Check if a Substring is Present in a Given String
5. Find length of a string in python (4 ways)
6. Python program to print even length words in a string
7. Python | Program to accept the strings which contains all vowels
8. Python | Count the Number of matching characters in a pair of string
9. Python program to count number of vowels using sets in given string
10. Remove all duplicates from a given string in Python
11. Python | Program to check if a string contains any special character
12. Generating random strings until a given string is generated
13. Find words which are greater than given length k
14. Python program for removing i-th character from a string
15. Python program to split and join a string
16. Python | Check if a given string is binary string or not
17. Python | Find all close matches of input string from a list
18. Python program to find uncommon words from two Strings
19. Python | Swap commas and dots in a String
20. Python | Permutation of a given string using inbuilt function
21. Python | Check for URL in a String

22. Execute a String of Code in Python
23. String slicing in Python to rotate a string
24. String slicing in Python to check if a string can become empty by recursive deletion
25. Python Counter| Find all duplicate characters in string

Dictionary Programs:

1. Python | Sort Python Dictionaries by Key or Value
2. Handling missing keys in Python dictionaries
3. Python dictionary with keys having multiple inputs
4. Python program to find the sum of all items in a dictionary
5. Python | Ways to remove a key from dictionary
6. Ways to sort list of dictionaries by values in Python – Using itemgetter
7. Ways to sort list of dictionaries by values in Python – Using lambda function
8. Python | Merging two Dictionaries
9. Program to create grade calculator in Python
10. Python | Check order of character in string using OrderedDict( )
11. Python | Find common elements in three sorted arrays by dictionary intersection
12. Dictionary and counter in Python to find winner of election
13. Find all duplicate characters in string
14. Print anagrams together in Python using List and Dictionary
15. K'th Non-repeating Character in Python using List Comprehension and OrderedDict
16. Check if binary representations of two numbers are anagram
17. Python Counter to find the size of largest subset of anagram words
18. Python | Remove all duplicates words from a given sentence
19. Python Dictionary to find mirror characters in a string
20. Counting the frequencies in a list using dictionary in Python
21. Python | Convert a list of Tuples into Dictionary
22. Python counter and dictionary intersection example (Make a string using deletion and rearrangement)
23. Python dictionary, set and counter to check if frequencies can become same
24. Scraping And Finding Ordered Words In A Dictionary using Python

25. Possible Words using given characters in Python

Tuple Programs:

1. Create a list of tuples from given list having number and its cube in each tuple
2. Sort a list of tuples by second Item

Searching and Sorting Programs:

1. Python Program for Binary Search (Recursive and Iterative)
2. Python Program for Linear Search
3. Python Program for Insertion Sort
4. Python Program for Recursive Insertion Sort
5. Python Program for QuickSort
6. Python Program for Iterative Quick Sort
7. Python Program for Selection Sort
8. Python Program for Bubble Sort
9. Python Program for Merge Sort
10. Python Program for Iterative Merge Sort
11. Python Program for Heap Sort
12. Python Program for Counting Sort
13. Python Program for ShellSort
14. Python Program for Topological Sorting
15. Python Program for Radix Sort
16. Python Program for Binary Insertion Sort
17. Python Program for Bitonic Sort
18. Python Program for Comb Sort
19. Python Program for Pigeonhole Sort
20. Python Program for Cocktail Sort
21. Python Program for Gnome Sort
22. Python Program for Odd-Even Sort / Brick Sort
23. Python Program for BogoSort or Permutation Sort
24. Python Program for Cycle Sort

25. Python Program for Stooge Sort

Pattern Printing Programs:

1. Program to print the pattern 'G'
2. Python | Print an Inverted Star Pattern
3. Python 3 | Program to print double sided stair-case pattern
4. Print with your own font using Python !!

Date-Time Programs:

1. Python program to convert time from 12 hour to 24 hour format

More Python Programs:

1. Python Program to Reverse a linked list
2. Python Program for Find largest prime factor of a number
3. Python Program for Efficient program to print all prime factors of a given number
4. Python Program for Product of unique prime factors of a number
5. Python Program for Find sum of odd factors of a number
6. Python Program for Coin Change
7. Python Program for Tower of Hanoi
8. Python Program for Sieve of Eratosthenes
9. Python Program to Check if binary representation is palindrome
10. Python Program for Basic Euclidean algorithms
11. Python Program for Extended Euclidean algorithms
12. Python Program for Number of elements with odd factors in given range
13. Python Program for Common Divisors of Two Numbers
14. Python Program for Maximum height when coins are arranged in a triangle
15. Python Program for GCD of more than two (or array) numbers
16. Python Program for Check if count of divisors is even or odd
17. Python Program for Find minimum sum of factors of number
18. Python Program for Difference between sums of odd and even digits
19. Python Program for Program to Print Matrix in Z form
20. Python Program for Largest K digit number divisible by X
21. Python Program for Smallest K digit number divisible by X
22. Python Program for Print Number series without using any loop

23. Python Program for Number of stopping station problem
24. Python Program for Program to calculate area of a Tetrahedron
25. Python Program for focal length of a spherical mirror
26. Python Program for Find the perimeter of a cylinder
27. Check if a triangle of positive area is possible with the given angles
28. Python Program for Number of jump required of given length to reach a point of form (d, 0) from origin in 2D plane
29. Python Program for Finding the vertex, focus and directrix of a parabola
30. Python program to find the most occurring character and its count
31. Python Program for Find sum of even factors of a number
32. Python Program for Check if all digits of a number divide it
33. Python program to convert float decimal to Octal number
34. Python program to convert floating to binary
35. Check whether a number has consecutive 0's in the given base or not
36. Python Program for Number of solutions to Modular Equations
37. Python Program for Triangular Matchstick Number
38. Python Program for Legendre\'s Conjecture
39. Python program to check if a string contains all unique characters
40. Python program to copy odd lines of one file to other

Adding two numbers

```
num1 = 15
num2 = 12

# Adding two nos
sum = num1 + num2

# printing values
print("Sum of {0} and {1} is {2}" .format(num1, num2, sum))
```

Add Two Numbers With User Input

# Store input numbers

num1 = input('Enter first number: ')

num2 = input('Enter second number: ')

# Add two numbers

sum = float(num1) + float(num2)

# Display the sum

print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))

In the program below, we take a temperature in degree Celsius and convert it into degree Fahrenheit. They are related by the formula:

celsius * 1.8 = fahrenheit - 32

Source Code

# Python Program to convert temperature in celsius to fahrenheit

# change this value for a different result

celsius = 37.5

# calculate fahrenheit

fahrenheit = (celsius * 1.8) + 32

print('%0.1f degree Celsius is equal to %0.1f degree Fahrenheit' %(celsius,fahrenheit))

formula : celsius = (fahrenheit - 32) / 1.8

Python Conditions and If statements

Python supports the usual logical conditions from mathematics:

- Equals: `a == b`
- Not Equals: `a != b`
- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`

These conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the `if` keyword.

If statement:

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

Indentation

Python relies on indentation (whitespace at the beginning of a line) to define scope in the code. Other programming languages often use curly-brackets for this purpose.

Example

If statement, without indentation (will raise an error):

```
a = 33
b = 200
if b > a:
print("b is greater than a") # you will get an error
```

Elif

The elif keyword is pythons way of saying "if the previous conditions were not true, then try this condition".

EXAMPLE

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

Else

The else keyword catches anything which isn't caught by the preceding conditions.

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

In this example a is greater than b, so the first condition is not true, also the elif condition is not true, so we go to the else condition and print to screen that "a is greater than b".

You can also have an else without the elif:

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

Short Hand If

If you have only one statement to execute, you can put it on the same line as the if statement.

```
if a > b: print("a is greater than b")
```

Short Hand If ... Else

If you have only one statement to execute, one for if, and one for else, you can put it all on the same line:

```
a = 2
b = 330
print("A") if a > b else print("B")
```

You can also have multiple else statements on the same line:

```
a = 330
b = 330
print("A") if a > b else print("=") if a == b else print("B")
```

And

The and keyword is a logical operator, and is used to combine conditional statements:

```
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```

Or

The or keyword is a logical operator, and is used to combine conditional statements:

```
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions is True")
```

Nested If

You can have if statements inside if statements, this is called *nested* if statements.

```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

The pass Statement

if statements cannot be empty, but if you for some reason have an if statement with no content, put in the pass statement to avoid getting an error.

```
a = 33
b = 200

if b > a:
  pass
```

Python While Loops

Python Loops

Python has two primitive loop commands:

- while loops
- for loops

The while Loop

With the while loop we can execute a set of statements as long as a condition is true.

```
i = 1
while i < 6:
  print(i)
  i += 1
```

The break Statement

With the break statement we can stop the loop even if the while condition is true:

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

The continue Statement

With the continue statement we can stop the current iteration, and continue with the next:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

The else Statement

With the else statement we can run a block of code once when the condition no longer is true:

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

Python For Loops

Python For Loops

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

```python
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

Looping Through a String

Even strings are iterable objects, they contain a sequence of characters:

```python
for x in "banana":
  print(x)
```

The break Statement

With the `break` statement we can stop the loop before it has looped through all the items:

```python
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

example 2

```python
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    break
  print(x)
```

The continue Statement

With the `continue` statement we can stop the current iteration of the loop, and continue with the next:

```python
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

The range() Function

To loop through a set of code a specified number of times, we can use the range() function,

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

```
for x in range(6):
  print(x)
```

The range() function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: range(2, 6), which means values from 2 to 6 (but not including 6):

```
for x in range(2, 6):
  print(x)
```

The range() function defaults to increment the sequence by 1, however it is possible to specify the increment value by adding a third parameter: range(2, 30, 3):

```
for x in range(2, 30, 3):
  print(x)
```

Else in For Loop

The else keyword in a for loop specifies a block of code to be executed when the loop is finished:

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

Nested Loops

A nested loop is a loop inside a loop.

The "inner loop" will be executed one time for each iteration of the "outer loop":

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

The pass Statement

for loops cannot be empty, but if you for some reason have a for loop with no content, put in the pass statement to avoid getting an error.

```
for x in [0, 1, 2]:
  pass
```

Rpi with LED

RPI with ultrasonic sensor

SENSING WITH PYTHON

The HC-SR04 Ultrasonic sensor we'll be using in this tutorial for the Raspberry Pi has four pins: ground (GND), Echo Pulse Output (ECHO), Trigger Pulse Input (TRIG), and 5V Supply (Vcc). We power the module using Vcc, ground it using GND, and use our Raspberry Pi to send an input signal to TRIG, which triggers the sensor to send an ultrasonic pulse. The pulse waves bounce off any nearby objects and some are reflected back to the sensor. The sensor detects these return waves and measures the time between the trigger and returned pulse, and then sends a 5V signal on the ECHO pin.

ECHO will be "low" (0V) until the sensor is triggered when it receives the echo pulse. Once a return pulse has been located ECHO is set "high" (5V) for the duration of that pulse. Pulse duration is the full time between the sensor outputting an ultrasonic pulse, and the return pulse being detected by the sensor receiver. Our Python script must therefore measure the pulse duration and then calculate distance from this.

IMPORTANT. The sensor output signal (ECHO) on the HC-SR04 is rated at 5V. However, the input pin on the Raspberry Pi GPIO is rated at 3.3V. Sending a 5V signal into that unprotected 3.3V input port could damage your GPIO pins, which is something we want to avoid! We'll need to use a small voltage divider circuit, consisting of two resistors, to lower the sensor output voltage to something our Raspberry Pi can handle.

Now that we've hooked our Ultrasonic Sensor up to our Pi, we need to program a Python script to detect distance!

The Ultrasonic sensor output (ECHO) will always output low (0V) unless it's been triggered in which case it will output 5V (3.3V with our voltage divider!). We therefore need to set one GPIO pin as an output, to trigger the sensor, and one as an input to detect the ECHO voltage change.

First, import the Python GPIO library, import our time library (so we make our Pi wait between steps) and set our GPIO pin numbering.

```
import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BCM)
```

Next, we need to name our input and output pins, so that we can refer to it later in our Python code. We'll name our output pin (which triggers the sensor) GPIO 23 [Pin 16] as TRIG, and our input pin (which reads the return signal from the sensor) GPIO 24 [Pin 18] as ECHO.

```
TRIG = 23

ECHO = 24
```

We'll then print a message to let the user know that distance measurement is in progress. . . .

```
print "Distance Measurement In Progress"
```

Next, set your two GPIO ports as either inputs or outputs as defined previously.

```
GPIO.setup(TRIG,GPIO.OUT)

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

Then, ensure that the Trigger pin is set low, and give the sensor a second to settle.

```
GPIO.output(TRIG, False)

print "Waiting For Sensor To Settle"

time.sleep(2)
```

The HC-SR04 sensor requires a short 10uS pulse to trigger the module, which will cause the sensor to start the ranging program (8 ultrasound bursts at 40 kHz) in order to obtain an echo response. So, to create our trigger pulse, we set out trigger pin high for 10uS then set it low again.

```
GPIO.output(TRIG, True)

time.sleep(0.00001)

GPIO.output(TRIG, False)
```

Now that we've sent our pulse signal we need to listen to our input pin, which is connected to ECHO. The sensor sets ECHO to high for the amount of time it takes for the pulse to go and come back, so our code therefore needs to measure the amount of time that the ECHO pin stays high. We use the "while" string to ensure that each signal timestamp is recorded in the correct order.

The time.time() function will record the latest timestamp for a given condition. For example, if a pin goes from low to high, and we're recording the low condition using the time.time() function, the recorded timestamp will be the latest time at which that pin was low.

Our first step must therefore be to record the last low timestamp for ECHO (pulse_start) e.g. just before the return signal is received and the pin goes high.

```
while GPIO.input(ECHO)==0:

  pulse_start = time.time()
```

Once a signal is received, the value changes from low (0) to high (1), and the signal will remain high for the duration of the echo pulse. We therefore also need the last high timestamp for ECHO (pulse_end).

```
while GPIO.input(ECHO)==1:

  pulse_end = time.time()
```

We can now calculate the difference between the two recorded timestamps, and hence the duration of pulse (pulse_duration).

```
pulse_duration = pulse_end - pulse_start
```

With the time it takes for the signal to travel to an object and back again, we can calculate the distance using the following formula.

$$Speed = \frac{Distance}{Time}$$

The speed of sound is variable, depending on what medium it's travelling through, in addition to the temperature of that medium. However, some clever physicists have calculated the speed of sound at sea level so we'll take our baseline as the 343m/s. If you’re trying to measure distance through water, this is where you’re falling down – make sure you’re using the right speed of sound!

We also need to divide our time by two because what we’ve calculated above is actually the time it takes for the ultrasonic pulse to travel the distance to the object and back again. We simply want the distance to the object! We can simplify the calculation to be completed in our Python script as follows:

$$34300 = \frac{Distance}{Time/2}$$

$$17150 = \frac{Distance}{Time}$$

$$17150 \times Time = Distance$$

We can plug this calculation into our Python script:

```
distance = pulse_duration x 17150
```

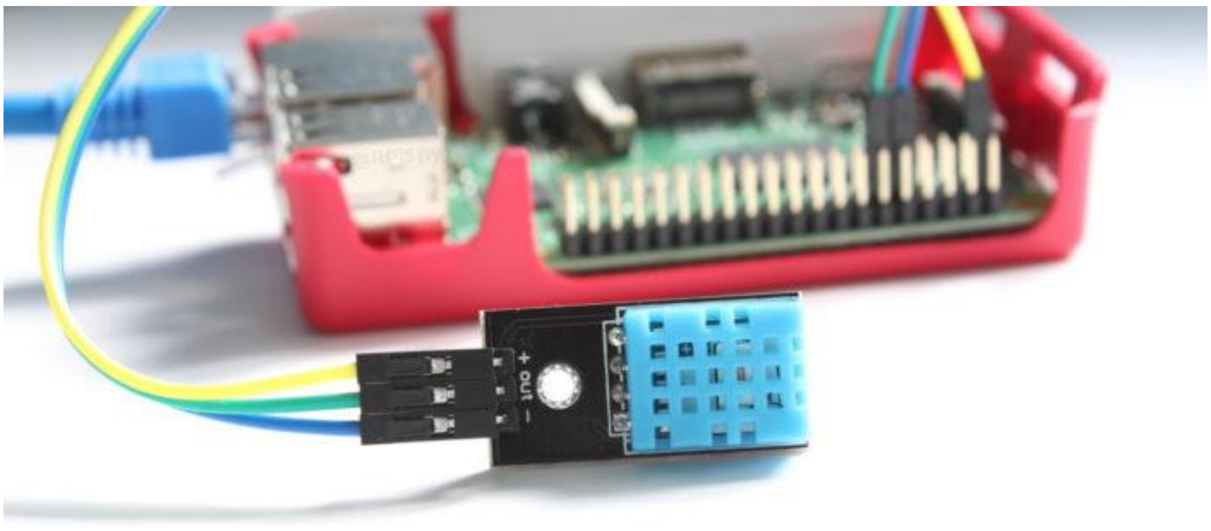
Then, we print the distance. The below command will print the word “Distance:” followed by the distance variable, followed by the unit “cm”

```
print "Distance:",distance,"cm"
```

Finally, we clean our GPIO pins to ensure that all inputs/outputs are reset

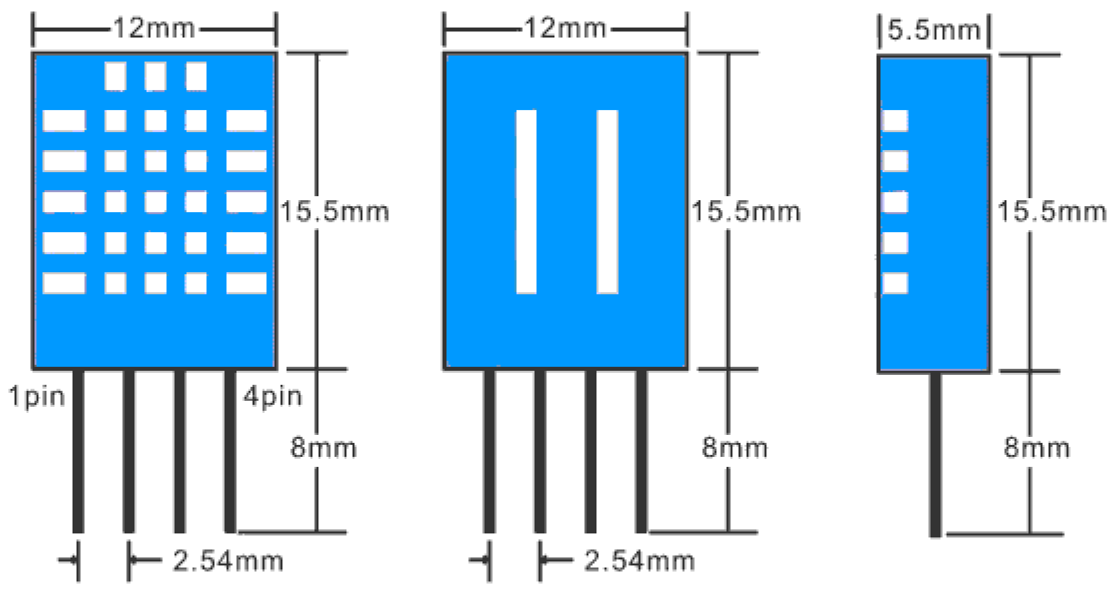
```
GPIO.cleanup()
```

DHT11 WITH RASPBERRY PI



DHT11 Specifications

The device itself has four pins but one of these is not used. You can buy the 4-pin device on its own or as part of a 3-pin module.

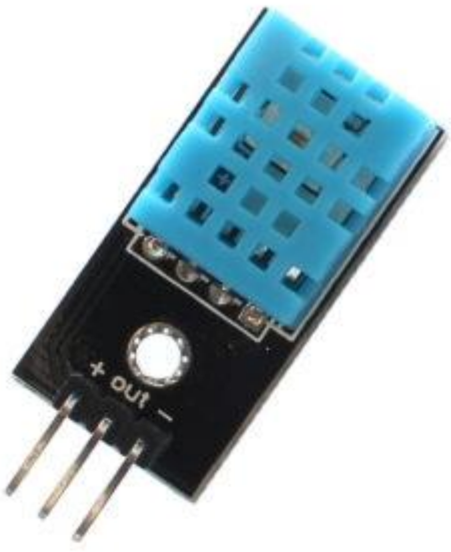


The modules have three pins and are easy to connect directly to the Pi's GPIO header.

- Humidity : 20-80% (5% accuracy)
- Temperature : 0-50℃ (±2℃ accuracy)

The manufacturers do not recommended that you read data from this device more than once per 2 seconds. If you do you may get incorrect readings.

Hardware Setup



The 4-pin device will require a resistor (4.7K-10K) to be placed between Pin 1 (3.3V) and Pin 2 (Data).

The 3-pin modules will usually have this resistor included which makes the wiring a bit easier. For this reason I got hold of the module which I could then attach to the Pi with a piece of 3-way Dupont cable.

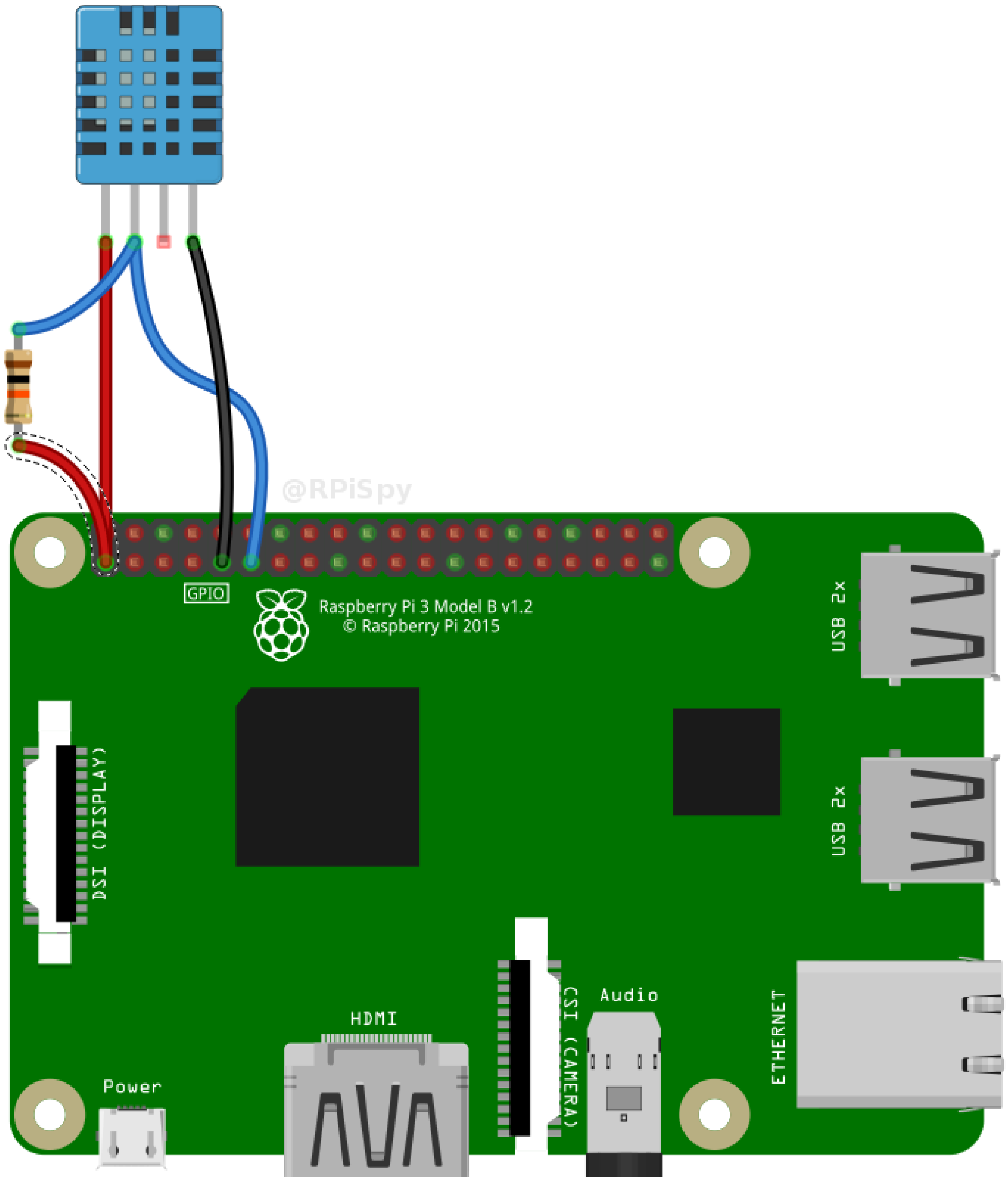
Different suppliers may wire the module pins differently so check the PCB markings to identify Vcc (+), data and Ground (-).

The 3 pins should be connected to the Pi as shown in the table below :

DHT Pin	Signal	Pi Pin
1	3.3V	1
2	Data/Out	11 (GPIO17)

DHT Pin	Signal	Pi Pin
3	not used	–
4	Ground	6 or 9

Your data pin can be attached to any GPIO pin you prefer. In my example I am using physical pin 11 which is GPIO 17. Here is a 4-pin sensor connected to the Pi's GPIO header. It has a 10K resistor between pin 1 (3.3V) and 2 (Data/Out).



Python Library

The DHT11 requires a specific protocol to be applied to the data pin. In order to save time trying to implement this yourself it's far easier to use the Adafruit DHT library.

The library deals with the data that needs to be exchanged with the sensor but it is sensitive to timing issues. The Pi's operating system may get in the way while performing other tasks so to compensate for this the library requests a number of readings from the device until it gets one that is valid.

```
import Adafruit_DHT

# Set sensor type : Options are DHT11,DHT22 or AM2302
sensor=Adafruit_DHT.DHT11

# Set GPIO sensor is connected to
gpio=17

# Use read_retry method. This will retry up to 15 times to
# get a sensor reading (waiting 2 seconds between each retry).
humidity, temperature = Adafruit_DHT.read_retry(sensor, gpio)

# Reading the DHT11 is very sensitive to timings and occasionally
# the Pi might fail to get a valid reading. So check if readings are valid.
if humidity is not None and temperature is not None:
  print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature,
humidity))
else:
  print('Failed to get reading. Try again!')
```