

Smart plant Communicator System With IBM Cloud

Industrial/Practical Training Report

Submitted to the

Department of Electronics and Communication Engineering,

SESHADRIRAO GUDLAVALLERU ENGINEERING COLLEGE

In partial fulfillment of the requirements,

for the award of the Degree of

Bachelor of Technology

in

Electronics and Communication Engineering

By

Meghana Kammela(19481A0493)

Department of Electronics and Communication Engineering

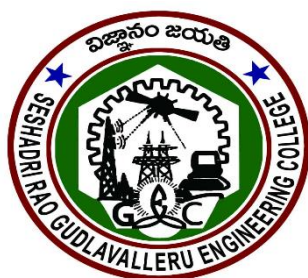
SESHADRIRAO GUDLAVALLERU ENGINEERING COLLEGE

SESHADRI RAO KNOWLEDGE VILLAGE

GUDLAVALLERU – 521356

ANDHRA PRADESH

2022-23



Project completion certificate

Certificate of Industrial Training Program

DECLARATION

I K. Meghana hereby declare that this industrial training report is the record of authentic work carried out by me during the period 21.03.2022 to 10.05.2022 in **Smart Bridge Pvt.Ltd** under the supervision of my training In charge Nikhil Gurram Smart Bridge Pvt.Ltd.

Signature:

Name of the student:

Kammela Meghana

Acknowledgement

I am very glad to express my deep indebtedness to all the employees of Smart Bridge Pvt.Ltd. Without their support and guidance, it would not have been possible for this training to have materialized and taken a concrete shape. I extend my deep gratitude to my training mentor – Sai prasanna kumar, Nikhil Gurram, Gnaneswar Bandari who supported us all the time. I owe my personal thanks to our HOD **Dr.Y.Rama Krishna** and our industrial training coordinator **Mr. N.Samba Murthy** for undergoing training at a reputed company like Smart Bridge Pvt.Ltd.

We would like to take this opportunity to express our profound sense of gratitude to our beloved Principal **Dr.G.V.S.N.R.V.Prasad**, for providing us all the required facilities. We express our thanks to Teaching and Non-Teaching staff of electronics and communication engineering department who helped us directly or indirectly in completion of our internship.

Meghana Kammela
(19481A0493)

CONTENTS

TITLE

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

CHAPTER 01:

- 1.0** Introductions to IoT
- 1.1** Introduction to open Hardware Platforms and Tinkercad Circuits
- 1.2** Arduino – Tinkercad
- 1.3** Introduction to Python and Python Basics
 - 1.3.1** Python Introduction
 - 1.3.2** Python Basics
 - 1.3.3** Conditional Statements and Functional basics
 - 1.3.4** Data Variables and Basic Variables
 - 1.3.5** Lists, Tuples and Dictionaries
 - 1.3.6** Functions and Modules
 - 1.3.7** Files IO and Python Inbuild Libraries
 - 1.3.8** Python OOPS Concepts
 - 1.3.9** Networking – Socket Programming

CHAPTER 02:

- 2.0** IoT Communication Technologies and Protocols
 - 2.0.0** Communication Technologies
 - 2.0.1** Communication Protocol
- 2.1** IBM Cloud Platforms
 - 2.1.0** Introduction to IBM Cloud and its Architecture
 - 2.1.1** IBM IoT Device creation and connecting Internal Simulator
 - 2.1.2** Connecting to IBM IoT Platform and sending sensor data using python code

CHAPTER 03:

- 3.0** IBM Cloud Services
 - 3.0.0** IBM Cloudant
 - 3.0.1** IBM Object Storage
- 3.1** Nodered
 - 3.1.0** Retrieving Data from IBM IoT Platform
 - 3.1.1** Sending Command to the Device

CHAPTER 04:

- 4.0** MIT App Inventor
 - 4.0.0** Display the Sensor Values in the mobile application.
 - 4.0.1** Controlling the appliances using Mobile App
- 4.1** Flask
 - 4.1.0** Flask Tutorials
 - 4.1.1** Flask Basics
 - 4.1.2** Flask graphics and Introduction to IBM Cloud Functions
- 4.2** Applications
 - 4.2.0** Smart Security Using Nodered Service
- 4.3** IBM Cloud Functions

CHAPTER 05:

- 5.0** Smart Plant Communicator System Project
 - 5.0.0** Purpose Of The Project
 - 5.0.1** Block Diagram
 - 5.0.2** Hardware/Software designing
- 5.1** Flow Chart
- 5.2** Advantages and Disadvantages
 - 5.2.0** Applications
- 5.3** Result
- 5.4** Conclusion

ABSTRACT

We have done our internship at **Smart Bridge** in Virtual mode. There are different areas in Electronics and communication Engineering and we have chosen Internet of Things (IoT), newly emerged technology with a group of students.

In our Internship, we have studied various Virtual Platforms like Tinker cad, IBM Cloud Platform and Python Programming. We have learnt how to use Virtual Environment to perform the Hardware Projects using the Tinker cad and ARDUINO. We have done the IoT based Smart Irrigation System.

As an application of these, we have done a project comprised of a Team of 4 members. Our Project name is IoT Analytics in Energy Management. With the help of Smart Internz Platform we started our project with the knowledge we have gained in this Internship. We successfully completed our Project using the Python Programming and IBM Cloud Services.

In this way the knowledge we have gained in the Smart Bridge company is useful to us.

Key words-IoT, Arduino, cloud, virtual.

LIST OF FIGURES

Figure No	Figure Name	Page No
1.	Internet of Things (IoT) Application in different Sectors	8
2.	IoT Architecture	9
3.	IoT Technology Stack	9
4.	Arduino UNO Board	10
5.	Experiments in Tinkercad	11
6.	Addition, Subtraction, Multiplication, Division using Python	13
7.	String, Hello World in Python Programming	13
8.	Various Operations using Python.	14
9.	Socket Programming Code- Creating Socket & Waiting for Connection	21
10.	Command Prompt Output-Client Data	21
11.	Output from Python Compiler - From Server.	22
12.	IBM IoT Platform Architecture	29
13.	IBM account Creation & IoT Service	30-32
14.	Python Programming to send data	34
15.	Cloudant in catalog page	35
16.	Cloudant database creating page	36
17.	Cloudant launch dashboard	36
18.	Creating database	37
19.	Editor Page	37
20.	Data is created in j.son format	38
21.	Object storage	38
22.	Creating object storage	39
23.	Creating buckets	39
24.	Data is stored	40
25.	Node-Red flow diagram	41
26.	IoT Simulator	42
27.	Sending Command to device	42
28.	Receiving Data from web	42
29.	Creating a MIT APP	43
30.	Displaying the sensor values	44
31.	Adding textbox 1 & 2	45
32.	Controlling the mobile app application	45
33.	Block Codes	46
34.	Smart Plant Communicator System With IBM Cloud	51
35.	Results	56-59

LIST OF TABLES

S.No	Table Name	Page No
1.	Arduino UNO Feature	10
2.	Modes description in Opening files in python	18
3.	Python File Methods Description	19
4.	Difference between GET and POST	26
5.	HTTP Status Codes	27
6.	2xx Successful	27
7.	4xx Client Error	27

CHAPTER 01

1.0 Introductions to IoT:

What is IoT?

- Connecting everyday things embedded with electronics, software and sensors to the internet enabling them to collect and exchange data.

Benefits of IoT:

1. Improved Performance
2. Reduced Cost
3. Improved Data Collection
4. Improved Customer Engagement
5. New Revenue Streams

Applications:

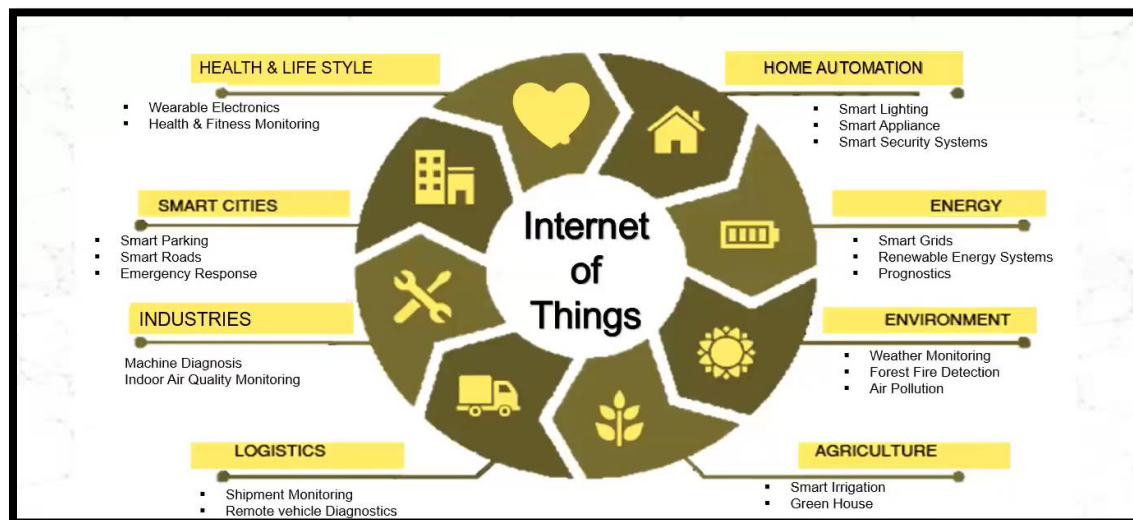


Fig 1: Internet of Things (IoT) Application in different Sectors

What is IoT Architecture?

IoT architecture is the system of numerous elements: sensors, protocols, actuators, cloud services, and layers. Given its complexity, there exist 4 stages of IoT architecture. Such a number is chosen to steadily include these various types of components into a sophisticated and unified network.

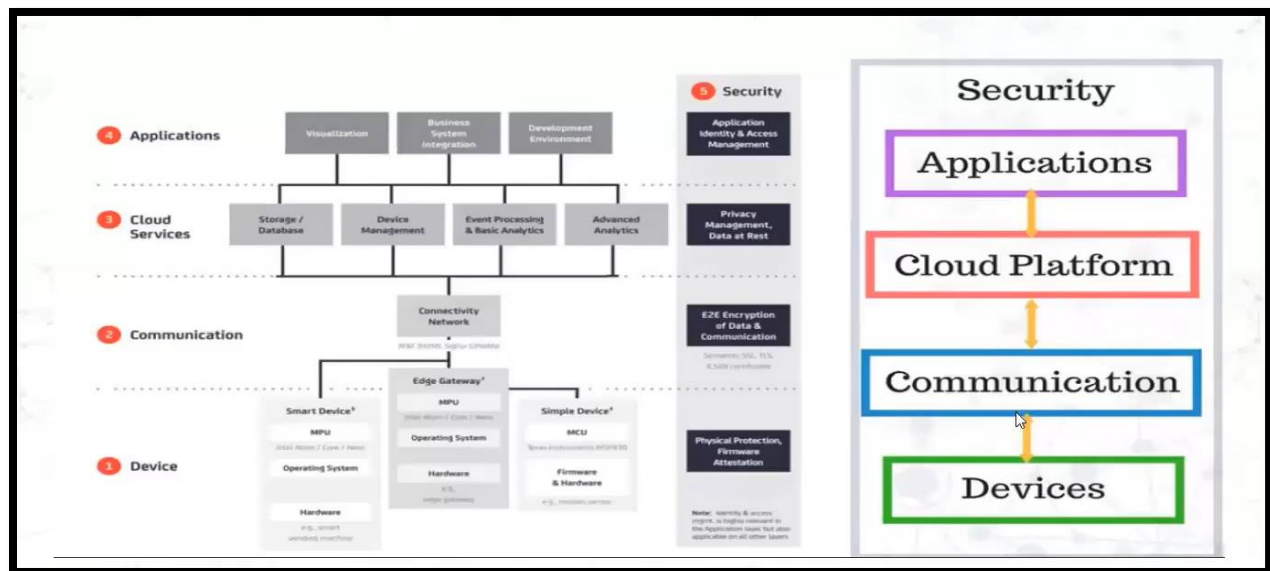


Figure 2: IoT Architecture

IoT Technology Stack:

- The IoT technology stack is nothing else than a range of technologies, standards and applications, which lead from the simple connection of objects to the Internet to the most easy and most complex applications that use these connected things, the data they gather and communicate and the different steps needed to power these applications

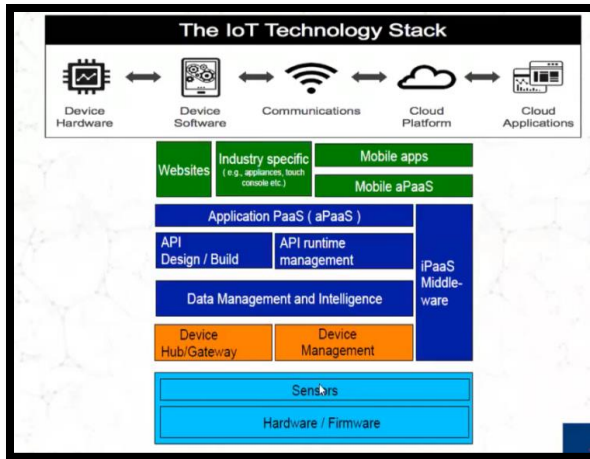


Figure 3: IoT Technology Stack

1.1 Introduction to open Hardware Platforms and Tinker cad Circuits:

What is Arduino?

- Arduino is an open-source platform used for building electronics projects.
- Easy tool for fast prototyping.
- Consists of both a physical programmable circuit board and a piece of software.

Why Arduino?

- Open-Source Platform
- Inexpensive
- Does not need a separate piece of hardware
- Arduino IDE uses a simplifies version of C++
- Cross-Platform
- Provides a standard form factor.

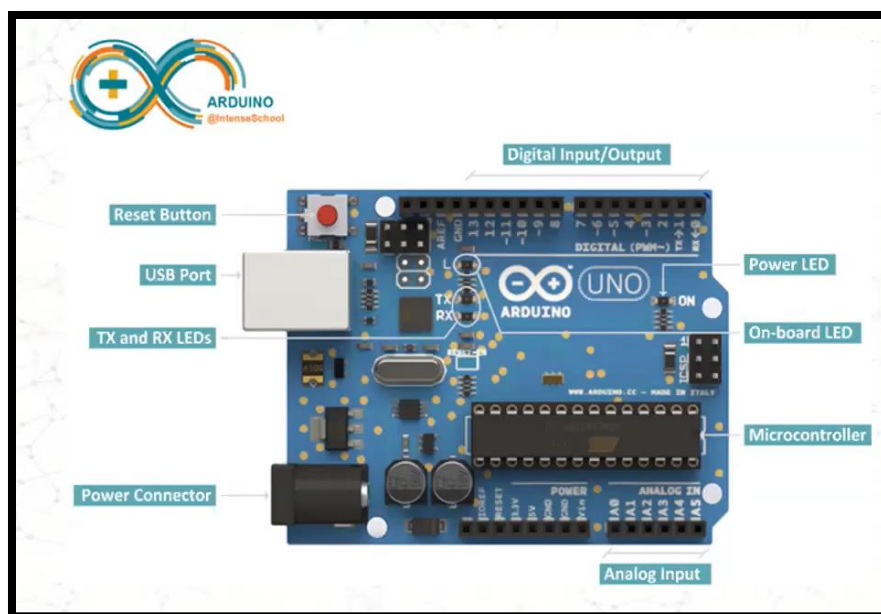


Figure 4: Arduino UNO Board

Arduino UNO Features:

Operating Voltage	5V and 3.3V
Input Voltage (recommended)	7-12 V
Input Voltage (limits)	6-20 V
Digital I/O Pins	14 (6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3 V Pin	50 mA
Flash Memory	32 KB (ATmega328)
SRAM	2 KB (ATmega328)
EEROM	1 KB (ATmega328)
Clock Speed	16 MHz
Microcontroller	ATmega328

Table 1: Arduino UNO Feature

1.2 Arduino – Tinkercad:**Use of Thinker Cad**

- Tinkercad is a free, easy-to-use app for 3D design, electronics, and coding. It's used by teachers, kids, hobbyists, and designers to imagine, design, and make anything!

List of the Experiments done with Thinkercad:

- Buzz
- Smart Door

- Blinker with TMP
- PIR Sensor
- LED Blink
- Piezo
- Ultrasonic Sensor
- Temp Sensor
- Servo Motor
- LED with Potentiometer
- Digital Input and Output

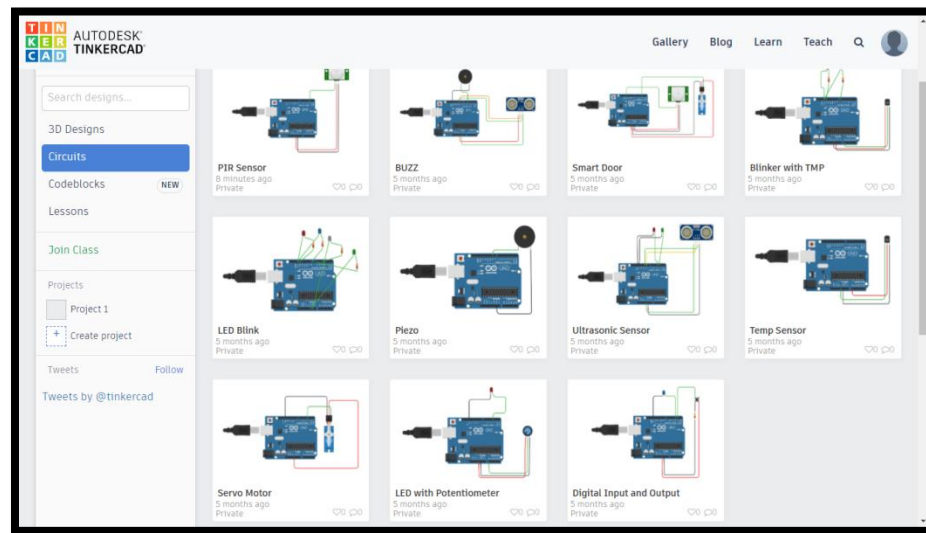


Figure 5: Experiments in Tinkercad

1.3 Introduction to Python and Python Basics:

1.3.1 Python Introduction:

What is Python?

- General Purpose
- Interpreted
- High Level
- Dynamically Typed

General Purpose:

Python language is a general-purpose language because it can be applicable to all domains, such type of languages is generic and not specialized.

Interpreted:

An Interpreted language will read the Raw code without being explicitly compiled before execution.

High Level:

Python allows programmers to express their logic in a form which is very close to human language. This helps in many computing aspects like memory management and data management.

Dynamically Typed:

Python identifies the type of variables on the basis of what kind of data you have assigned to the variable.

Why Choose Python?

- Readily available and open source
- Huge Libraries
- Easy to understand and learn
- Big developer Communities
- Fewer code lines and larger Functionalities.

1.3.2 Python Basics:

Here is a list of the Python keywords. Enter any keyword to get more help.

False	class	from	or
None	continue	global	pass
True	def	if	raise
and	del	import	return
as	elif	in	try
assert	else	is	while
async	except	lambda	with
await	finally	nonlocal	yield
break	for	not	

Basic Python Programming: -



```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> a=70
>>> type(a)
<class 'int'>
>>> b=3.8
>>> type(b)
<class 'float'>
>>> a=5
>>> b=7
>>> a+b
12
>>> a-b
-2
>>> a*b
35
>>> a/b
0.7142857142857143
>>> a=5.0
>>> b=7.0
>>> a*b
35.0
>>> b-a
2.0
>>> x=3+5j
>>> type(x)
<class 'complex'>
>>> y=2-2.2j
>>> x+y
(5+2.8j)
>>> a=True
>>> type(a)
<class 'bool'>
>>> b=False
>>> type(b)
<class 'bool'>
>>> a < b
False
```

Figure 6: Addition, Subtraction, Multiplication, Division using Python.



```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> s1="hello"
>>> type(s1)
<class 'str'>
>>> print(s1)
hello
>>> s1
'hello'
>>> s2="welcome"
>>> s1+s2
'hellowelcome'
>>> print("Hello world")
Hello world
>>> print("Hello \n World")
Hello
World
>>> print("Hello \nWorld")
Hello
World
>>> print("Hello \t World")
Hello World
>>> print("Hello\tWorld")
Hello World
>>> s="Hello World"
>>> s.capitalize()
'Hello world'
>>> len(s)
11
>>> s[0]
'H'
>>> s[4]
'e'
>>> s[0:4]
'Hell'
>>> s[0:5]
'Hello'
>>>
```

Figure 7: String, Hello World in Python Programming.



```

Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> name="vinay kumar "
>>> name.lstrip()
'vinay kumar'
>>> name.rstrip()
'vinay kumar'
>>> name.strip()
'vinay kumar'
>>> s="This is python"
>>> s.find('is')
2
>>> s.find('python')
8
>>> s.replace('is','was')
'Thwas was python'
>>> s.upper()
'THIS IS PYTHON'
>>> s.lower()
'this is python'
>>> s.title()
'This Is Python'
>>> s[3]
'g'
>>> s="This is a animal"
>>>

```

Figure 8: Various Operations using Python.

1.3.3 Conditional Statements and Functional basics:

Conditional Statements:

A conditional statement in Python is handled by if statements and we saw various other ways we can use conditional statements like if and else over here.

- "if condition" – It is used when you need to print out the result when one of the conditions is true or false.
- "else condition"- it is used when you want to print out the statement when your one condition fails to meet the requirement
- "elif condition" – It is used when you have third possibility as the outcome. You can use multiple elif conditions to check for 4th,5th,6th possibilities in your code
- We can use minimal code to execute conditional statements by declaring all condition in single statement to run the code
- If Statement can be nested.

Syntax:

```

if expression
    Statement
else expression
    Statement

```

Basic Functions:

What is function in Python?

- A function is a group of related statements that performs a specific task.

- Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable.
- It avoids repetition and makes the code reusable.

Syntax:

```
Def function_name(parameters) :  
    """docstring"""  
    Statements(s)
```

Function definition that consists of the following components:

1. Keyword **def** that marks the start of the function header.
2. A function name to uniquely identify the function. Function naming follows the same rules of writing identifiers in python.
3. Parameters (arguments) through which we pass values to a function. They are optional.
4. A colon (:) to mark the end of the function header.
5. Optional documentation string (docstring) to describe what the function does.
6. One or more valid python statements that make up the function body. Statements must have the same indentation level (usually 4 spaces).
7. An optional **return** statement to return a value from the function.

Types of Functions:

1. Built-in functions - Functions that are built into Python.
2. User-defined functions- Functions defined by the users themselves.

1.3.4 Data Variables and Basic Variables:

Data Types in Python:

Every value in Python has a data type. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.

There are various data types in Python. Some of the important types are listed below.

1. Python Numbers:

- Integers, floating point numbers and complex numbers fall under Python numbers category. They are defined as int, float, and complex classes in Python.
- We can use the **type()** function to know which class a variable or a value belongs to. Similarly, the **isinstance()** function is used to check if an object belongs to a particular class.

2. Python Strings:

- String is a sequence of Unicode characters. We can use single quotes or double quotes to represent strings. Multi-line strings can be denoted using triple quotes, `'''` or `"""`.

3. Python Set:

- Set is an unordered collection of unique items. Set is defined by values separated by comma inside braces `{ }`. Items in a set are not ordered.

1.3.5 Lists, Tuples and Dictionaries:**1. Python List:**

- List is an ordered sequence of items. It is one of the most used data type in Python and is very flexible. All the items in a list do not need to be of the same type.
- Declaring a list is pretty straight forward. Items separated by commas are enclosed within brackets [].
- We can use the slicing operator [] to extract an item or a range of items from a list. The index starts from 0 in Python.

2. Python Tuple:

- Tuple is an ordered sequence of items same as a list. The only difference is that tuples are immutable. Tuples once created cannot be modified.
- Tuples are used to write-protect data and are usually faster than lists as they cannot change dynamically.
- It is defined within parentheses () where items are separated by commas.
- We can use the slicing operator [] to extract items but we cannot change its value.

3. Python Dictionaries:

- Dictionary is an unordered collection of key-value pairs.
- It is generally used when we have a huge amount of data. Dictionaries are optimized for retrieving data. We must know the key to retrieve the value.
- In Python, dictionaries are defined within braces { } with each item being a pair in the form **key:value**. Key and value can be of any type.
- We use key to retrieve the respective value. But not the other way around.

1.3.6 Functions and Modules:**Python Functions:**

- Functions are a handy way to isolate a particular part of your program's functionality and make it reusable. Modules are a way to collect a number of helpful functions in one file, which you can then use in multiple projects and share with other programmers.
- We've already been using functions extensively in our programs; functions are one of the main building blocks of Python programming. Whenever you've typed something like **len(x)** or **type(y)** or even **random.choice([1, 2, 3])**, you've been using functions. It's just that these functions come pre-defined by Python.

Python Modules:

- A Python module is a file that contains one or more function definitions. Modules are a handy way of keeping related functions together, so that you can easily reuse them between projects, or share them with other programmers.
- Making a module is easy. Just make a file that has a **.py** extension and contains only **import** statements and function definitions. Here's a module called **restaurantutils** that contains many of the functions.

1.3.7 Files IO and Python Inbuilt Libraries:

Files:

- Files are named locations on disk to store related information. They are used to permanently store data in a non-volatile memory.
- RAM is volatile which loses its data when the computer is turned off, we use files for future use of the data by permanently storing them.
- When we want to read from or write to a file, we need to open it first. When we are done, it needs to be closed so that the resources that are tied with the file are freed.

Hence, in Python, a file operation takes place in the following order:

1. Open a file
2. Write
3. Read (perform operation)
4. Close the file

1. Opening Files in Python:

- Python has a built-in **open()** function to open a file. This function returns a file object, also called a handle, as it is used to read or modify the file accordingly.
- We can specify the mode while opening a file. In mode, we specify whether we want to read **r**, write **w** or append **a** to the file. We can also specify if we want to open the file in text mode or binary mode.

Mode	Description
r	Opens a file for reading. (default)
w	Opens a file for writing. Creates a new file if it does not exist or truncates the file if it exists.
x	Opens a file for exclusive creation. If the file already exists, the operation fails.
a	Opens a file for appending at the end of the file without truncating it. Creates a new file if it does not exist.
t	Opens a file for appending at the end of the file without truncating it. Creates a new file if it does not exist.
b	Opens in binary mode.
+	Opens a file for updating (reading and writing)

Table 2: Modes description in Opening files in python

2. Writing to files in Python:

- In order to write into a file in Python, we need to open it in write **w**, append **a** or exclusive creation **x** mode.
- We need to be careful with the **w** mode, as it will overwrite into the file if it already exists.
- Writing a string or sequence of bytes for binary files is done using the **write()** method. This method returns the number of characters written to the file.
- This program will create a new file named **test.txt** in the current directory if it does not exist. If it does exist, it is overwritten.

3. Reading Files in Python:

- To read a file in Python, we must open the file in reading **r** mode.
- There are various methods available for this purpose. We can use the **read(size)** method to read in the **size** number of data. If the **size** parameter is not specified, it reads and returns up to the end of the file.
- We can see that the **read()** method returns a newline as **'\n'**. Once the end of the file reached, we get an empty string on further reading.
- We can change our current file cursor position using the **seek()** method. Similarly, the **tell()** method returns our current position in number of bytes.

4. Closing Files in Python:

- When we are done with performing operations on the file, we need to properly close the file.
- Closing a file will free up the resources that were tied with the file. It is done using the **close()** method available in Python.
- Python has a garbage collector to clean up unreferenced objects but we must not rely on it to close the file.

Python File Methods:

There are various methods available with the file object. Some of them have been used in the above examples.

Here is the complete list of methods in text mode with a brief description:

Method	Description
close()	Closes an opened file. It has no effect if the file is already closed.
detach()	Separates the underlying binary buffer from the TextIOBase and returns it.
fileno()	Returns an integer number (file descriptor) of the file.
flush()	Flushes the write buffer of the file stream.
isatty()	Returns True if the file stream is interactive.

read(n)	Reads at most n characters from the file. Reads till end of file if it is negative or None .
readable()	Returns True if the file stream can be read from.
readline(n =-1)	Reads and returns one line from the file. Reads in at most n bytes if specified.
readlines(n =-1)	Reads and returns a list of lines from the file. Reads in at most n bytes/characters if specified.
seek(offset , from , SEEK_SET)	Changes the file position to offset bytes, in reference to from (start, current, end).
seekable()	Returns True if the file steam supports random access.
tell()	Returns the current file location.
truncate (size = None)	Resizes the file stream to size bytes. If size if not specified, resizes to current location.
writable()	Returns True if the file stream can be written to.
write(s)	Writes the string s to the file and returns the number of characters written.
writelines(lines)	Writes a list of lines to the file.

Table 3: Python File Methods Description

1.3.8 Python OOPS Concepts:

Python Object Oriented Programming:

- Python is a multi-paradigm programming language. It supports different programming approaches.
- One of the popular approaches to solve a programming problem is by creating objects. This is known as Object-Oriented Programming (OOP)

An object has two characteristics:

- Attributes
- Behavior

In Python, the concept of OOP follows some basic principles:

1) Class:

- A Class is a blueprint for the object.
- We can think of class as a sketch of a parrot with labels. It contains all the details about the name, colors, size etc. Based on these descriptions, we can study about the parrot. Here, a parrot is an object.

2) Object:

- An object (instance) is an instantiation of a class. When class is defined, only the description for the object is defined. Therefore, no memory or storage is allocated.
- 3) Methods:**
- Methods are functions defined inside the body of a class. They are used to define the behaviors of an object.
- 4) Inheritance:**
- Inheritance is a way of creating a new class for using details of an existing class without modifying it. The newly formed class is a derived class (or child class). Similarly, the existing class is a base class (or parent class).
- 5) Encapsulation:**
- Using OOP in Python, we can restrict access to methods and variables. This prevents data from direct modification which is called encapsulation. In Python, we denote private attributes using underscore as the prefix i.e single _ or double __.
- 6) Polymorphism:**
- Polymorphism is an ability (in OOP) to use a common interface for multiple forms (data types).
 - Suppose, we need to color a shape, there are multiple shape options (rectangle, square, circles). However we could use the same method to color any shape. This concept is called Polymorphism.

Key Points on OOPS:

- 1) Object-Oriented Programming makes the program easy to understand as well as efficient.
- 2) Since the class is sharable, the code can be reused.
- 3) Data is safe and secure with data abstraction.
- 4) Polymorphism allows the same interface for different objects, so programmers can write code efficient code.

1.3.9 Networking - Socket Programming:

Networking:

A computer network is a group of computers that use a set of common communication protocols over digital interconnections for the purpose of sharing resources located on or provided by the network nodes.

OSI Model:

- 1) Physical
- 2) Data Link
- 3) Network
- 4) Transport
- 5) Session
- 6) Presentation
- 7) Application.

TCP/IP & Socket Programming:

A Socket Programming interface provides the routines required for interprocess communication between applications, either on the local system or spread in a distributed, TCP/IP based network environment. Socket is uniquely defined by

- Internet Address (IP Address)
- Communication Protocol
- Port

```
ser.py - C:/Users/SmartBridgePC/Desktop/ser.py (3.8.2)
File Edit Format Run Options Window Help

import socket

s=socket.socket() #creation of socket

print("Socket is created")

s.bind(("localhost",3333)) # assign IP and port to socket

s.listen(3) #Listening limit for client

print("Waiting for connection")

while True:
    c,addr=s.accept() #it will accept the connection from client
    print("Connected with",addr)
    data=c.recv(1024).decode() #to recieve the data from client
    c.send(bytes("Welcome",'utf-8'))
    print("Client Data",data)
    c.close()
```

Figure 9: Socket Programming Code- Creating Socket & Waiting for Connection

The screenshot shows a Python 3.8.2 Shell window with the following text:

```

Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('127.0.0.1', 59783))
s.listen(5)
while True:
    client, address = s.accept()
    client.send('Message from Server Welcome')
    data = client.recv(1024)
    print(data)

```

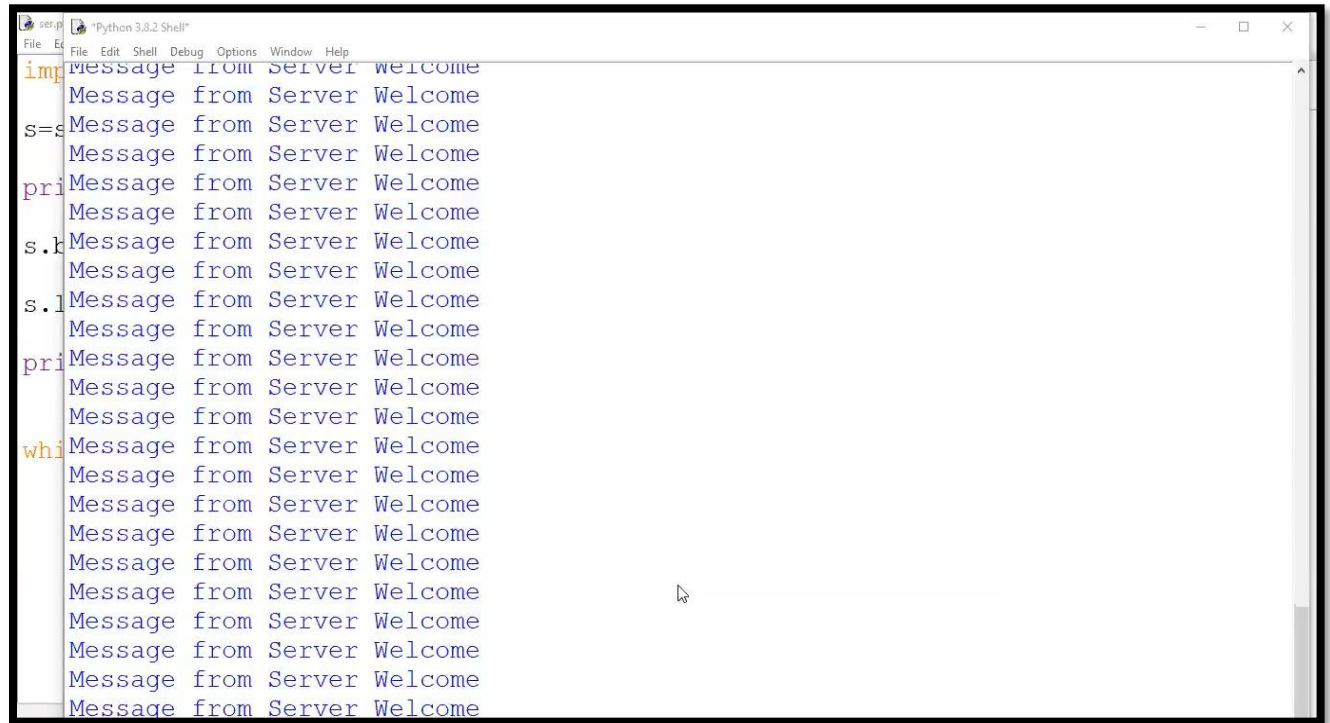
The output of the program is shown in the Command Prompt window:

```

C:\Python38\python ser.py
Client Data hello
Connected with ('127.0.0.1', 59783)
Client Data hello
Connected with ('127.0.0.1', 59784)
Client Data hello
Connected with ('127.0.0.1', 59785)
Client Data hello
Connected with ('127.0.0.1', 59786)
Client Data hello
Connected with ('127.0.0.1', 59787)
Client Data hello
Connected with ('127.0.0.1', 59788)
Client Data hello
Connected with ('127.0.0.1', 59789)
Client Data hello
Connected with ('127.0.0.1', 59790)
Client Data hello
Connected with ('127.0.0.1', 59791)
Client Data hello
Connected with ('127.0.0.1', 59792)
Client Data hello
Connected with ('127.0.0.1', 59793)
Client Data hello
Connected with ('127.0.0.1', 59794)
Client Data hello
Connected with ('127.0.0.1', 59795)
Client Data hello
Connected with ('127.0.0.1', 59796)
Client Data hello

```

Figure 10: Command Prompt Output-Client Data



```

Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('localhost', 8080))
while True:
    message = s.recv(1024)
    print(message.decode())
    if message == b'exit':
        break
    s.send('Welcome from client'.encode())

```

The screenshot shows a Python 3.8.2 Shell window with a menu bar (File, Edit, Shell, Debug, Options, Window, Help). The code in the editor is a simple socket client that connects to 'localhost' on port 8080. It enters a loop where it receives messages from the server and prints them. The output in the shell window shows 15 lines of 'Message from Server Welcome'.

Figure 11: Output from Python Compiler - From Server.

CHAPTER 02

1.3 IoT Communication Technologies and Protocols:

2.0.0 Communication Technology:

Several Communication Protocols and Technology used in the internet of Things. Some of the major IoT technology and protocol (IoT Communication Protocols) are Bluetooth, Wi-Fi, Radio Protocols, LTE-A, and Wi-Fi-Direct. These IoT communication protocols cater to and meet the specific functional requirement of an IoT system.

There are 6 IoT Communication Protocols/ Technology, let us look each one of them.

1. Bluetooth:

- A Wireless technology standard for exchanging data over short distances.
- Common use of Bluetooth technology is in hands-free devices such as headsets used with mobile phones.
- Uses short-wavelength UHF radio waves in the ISM band from 2.4 to 2.485 GHz.

- Bluetooth waves typically can only travel 33 feet or less.
- Bluetooth networks referred to as piconets use a master/slave model to control.
- A single master device can be connected to up to seven different slave devices.

Bluetooth Addresses:

- Every Single Bluetooth Device has a unique 48-bit address, abbreviated as BD_ADDR, presented in the form of a 12-digit hexadecimal value.
- The most-significant half – organization unique identifier (OUI)
- The lower 24-bits – unique part of the address.

Connection Process:

- **Inquiry** – If two Bluetooth devices know absolutely nothing about each other, one must run an inquiry to try to **discover** the other.
- **Paging (Connecting)** - Paging is the process of forming a connection between two, Bluetooth devices. Before this, each device needs to know the address of the other (found in the inquiry process).
- **Connection**
 - **Active Mode** – This is the regular connected mode.
 - **Sniff Mode** – This is a power saving mode.
 - **Hold Mode** – Hold mode is a temporary, power-saving mode.
 - **Park Mode** – Park is the deepest of sleep modes.

Bluetooth Low Energy:

BLE is meant for situations where battery life is preferred over high data transfer speeds.

Applications:

- Mesh Profiles
- Health Care Profiles
- Sports and Fitness Profiles
- Asset Tracking
- Indoor Navigation

2. Zigbee:

- Zigbee is an IEEE 802.15.4- based specification for a suite of high-level communication protocols used to create PAN.
- Consumes less power.
- Cost-effective wireless technology
- Zigbee is low-power, low data rate, and close proximity wireless ad hoc network.

Device Types:

- Zigbee Coordinator(ZC)
- Zigbee Router (ZR)
- Zigbee End Device (ZED)

3. Wi-Fi:

- WIFI (WLAN) utilizes the IEEE 802.11 standard.
- WIFI provides Internet access to devices that are within the range 66 feet from AP.
- Under ideal conditions, 2.4 GHz Wi-Fi will support up to 450 Mbps or 600 Mbps, depending on the class of the router, 5 GHz Wi-Fi will support up to 1300 Mbps.

Access Point Mode:

- An Access Point connects wired and wireless networks together and enables the sending and receiving of data between wireless clients and the wired network.
- The wireless SSID, also known as the 'Network Name' is the Service Set Identification controls access to a given wireless network.

4. RFID:

- RFID belongs to a group of technologies referred to as Automatic Identification and Data Capture (AIDC)
- Radio-frequency Identification (RFID) uses electromagnetic fields to automatically identify and track tags attached to objects. The tags contain electronically stored information.
- An RFID system consists of a reader, which emits an RF signal via an antenna. The chip receives the energy via an attached antenna (termed an RFID) and modulates the RF signal in order to respond through its antenna so that information can be transferred to the reader.
- Short range RFID is about 10cm, but long range can go up to 200m.
- Passive tags collect energy from a nearby RFID reader's interrogating radio waves.
- Active tags have a local power source (such as a battery) and may operate hundreds of meters from the RFID reader.

5. NFC:

Near-Field Communication (NFC) is a set of communication protocols that enable two electronic devices, one of which is usually a portable device such as a Smartphone, to establish communication by bringing them within 4 cm.

- Uses electromagnetic induction between two loop antennas located within each other's near field, effectively forming an air-core transformer.
- Frequency: ISM band of 13.56 MHz on ISO
- Data rates: 106 Kbits/s.

6. LoRa:

- LoRa (Long Range) is a spread spectrum modulation technique derived from chirp spread spectrum (CSS) technology
- Lora can achieve a distance of 15-20 km
- Lora can achieve high distance communication without using much power
- Operates on very low bandwidth (Max 5.5Kbps)

2.0.1 Communication Protocol:

A communication protocol is a system of rules that allow two or more entities of a communications system to transmit information via any kind of variation of a physical quantity. The protocol defines the rules, syntax, semantics and synchronization of communication and possible error recovery methods.

There are some Communication Protocols. Let's look at them:

1. HTTP:

- The Hypertext Transfer Protocol (HTTP) is an application-level protocol
- HTTP is a TCP/IP based communication protocol
- Used to deliver data (HTML files, image files, query results, etc.) on the World Wide Web.
- The default port is TCP 80.

Features:

- HTTP is Connectionless
- HTTP is media Independent
- HTTP stateless

Difference between GET and POST

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data.
History	Parameters remain in browser history	Parameters are not saved in browser history.
Restrictions on data type	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum	No restrictions

	URL length is 2048 characters)	
Restrictions on data types	Only ASCII characters allowed	No restrictions. Binary data is also allowed.
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs.
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL.

Figure 4: Difference between GET and POST

2. HTTP Client Request:

- An HTTP exchange takes place over a TCP/IP socket.
- The client opens a socket
- Connects to the HTTP server via the port the server is listening to, and issues a command.
- The command is routed to the server via the internet
- The server receives the command and does something, typically involving a file lookup.

3. HTTP Status Codes:

S.NO	Code and Description
1	1xx: Informational This means request received and continuing process.
2	2xx: Success This means the action was successfully received, understood, and accepted.
3	3xx: Redirection This means further action must be taken in order to complete the request.
4	4xx: Client Error This means the request contains bad syntax or cannot be fulfilled
5	5xx: Server Error This server failed to fulfill an apparently valid request

Table 5: HTTP Status Codes

2xx: Successful

Message	Description
200 OK	The request is OK

Table 6: 2xx Successful**4xx: Client Error**

Message	Description
400 Bad Request	The server did not understand the request
401 Unauthorized	The requested page needs a username and a password
403 Forbidden	Access is forbidden to the requested page
404 Not Found	The server cannot find the requested page

Table 7: 4xx Client Error**4. HTTPS:**

- Hyper Text Transfer Protocol Secure (HTTPS) is the secure version of HTTP. It uses SSL/TLS for secure encrypted communications.
- SSL (Secure Socket Layer) is a cryptographic protocol enhancement to HTTP, which defines how client and server should communicate with each other securely. TLS (Transport Layer Security) is the successor of SSL.
- The default port number for https is 443.
- Although HTTPS is secure by its design, the SSL/TLS handshake process consumes a significant time before establishing an HTTPS connection. It normally cost 1-2 seconds and drastically slows down the startup performance of a website.

5. MQTT:

- When a device (a client) wants to send data to the broker, we call this operation a “publish”.
- When a device (a client) wants to receive data from the broker, we call this operation a “subscribe”.

Broker:

- The broker is responsible for receiving all messages, filtering the messages, determining who is subscribed to each message, and sending the message to these subscribed clients.
- A broker has capabilities for both subscribing and publishing.
- Its primary function is to queue the received messages from the publisher and transmit the messages received to the subscriber client accordingly.

QoS: Quality of Service:

- The Quality of Service (QoS) level is an agreement between the sender of a message that defines the guarantee of delivery for a specific message.
- There are 3 QoS levels in MQTT:
 - **At most once (0):** This does not guarantee if a message has been delivered successfully.
 - **At least once (1):** This guarantee the message will be delivered at least once. But can be sent more than once.
 - **Exactly once (2):** This guarantees that the message is sent only once.
- The higher the QoS, the higher it consumes bandwidth to process the transmission.

2.1 IBM Cloud Platforms:

Cloud:

Cloud Computing is simply on-demand delivery of computing services over the internet.

Cloud Computing is categorized into different service models:

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Infrastructure as a Service (IaaS)

Software as a Service (SaaS):

- Simplest to use and easier to deploy
- No software to install or configure
- Everything is available by web browser.
- Microsoft's office and slack two popular SaaS products.

Platform as a Service (PaaS):

- Developers and programmers upload their content to pre-configured servers.
- No installation or maintaining the server software or operating system.

Infrastructure as a Service (IaaS):

- Dedicated virtual systems that you manage and maintain yourself.
- Greatest degree of flexibility and customization in cloud computing.
- Amazon web services (AWS) and Digital ocean are widely known as IaaS provider.

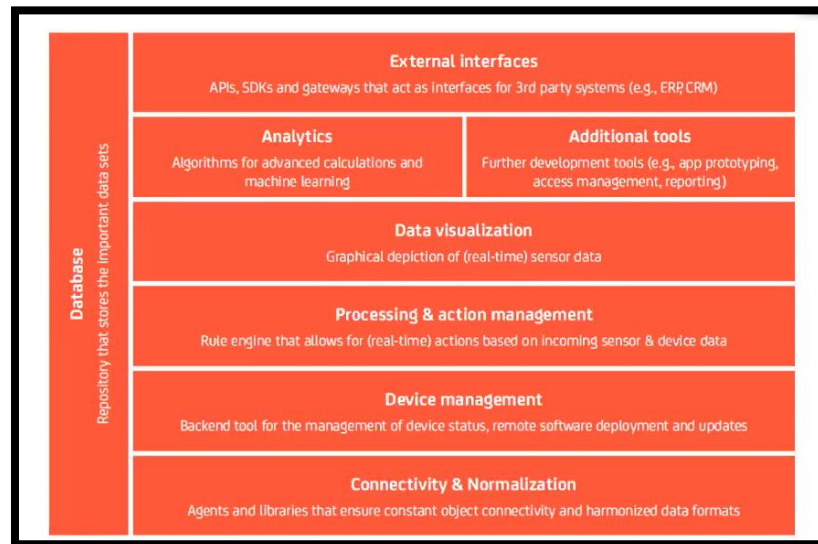
IBM IoT Platform Architecture:

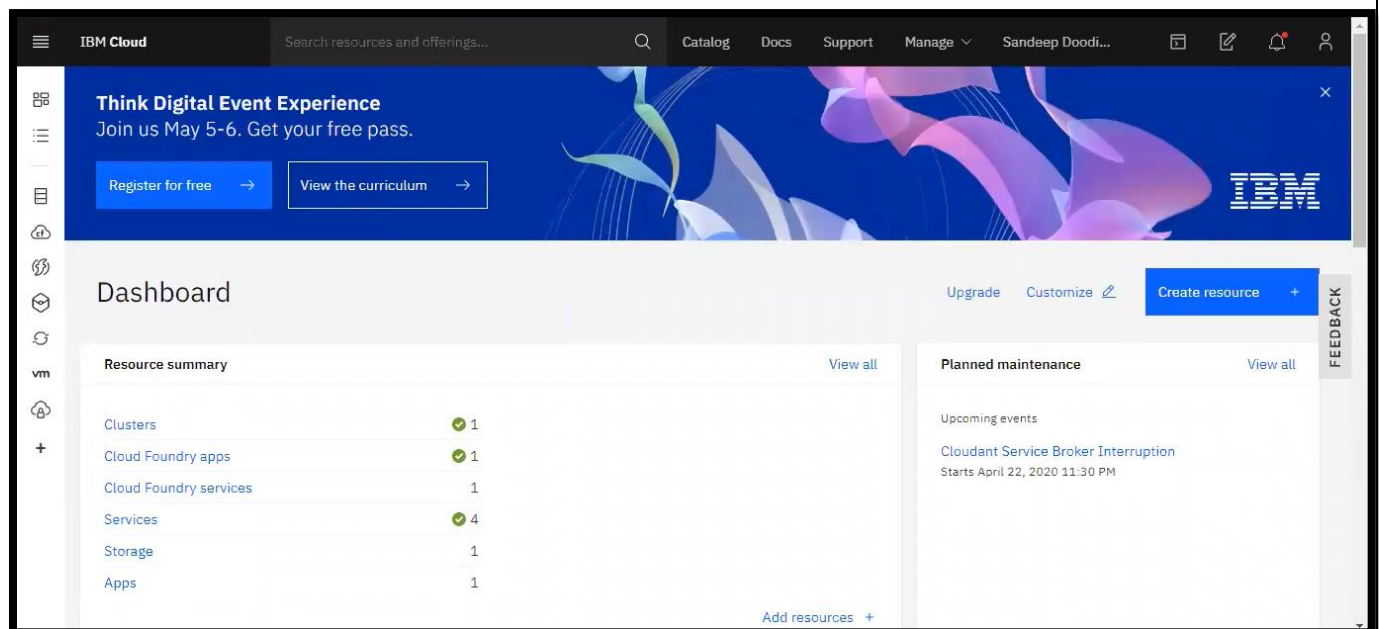
Figure 12: IBM IoT Platform Architecture

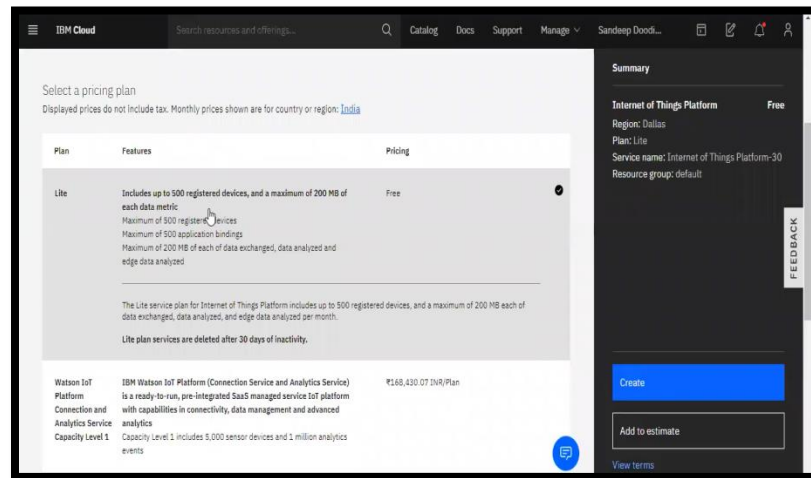
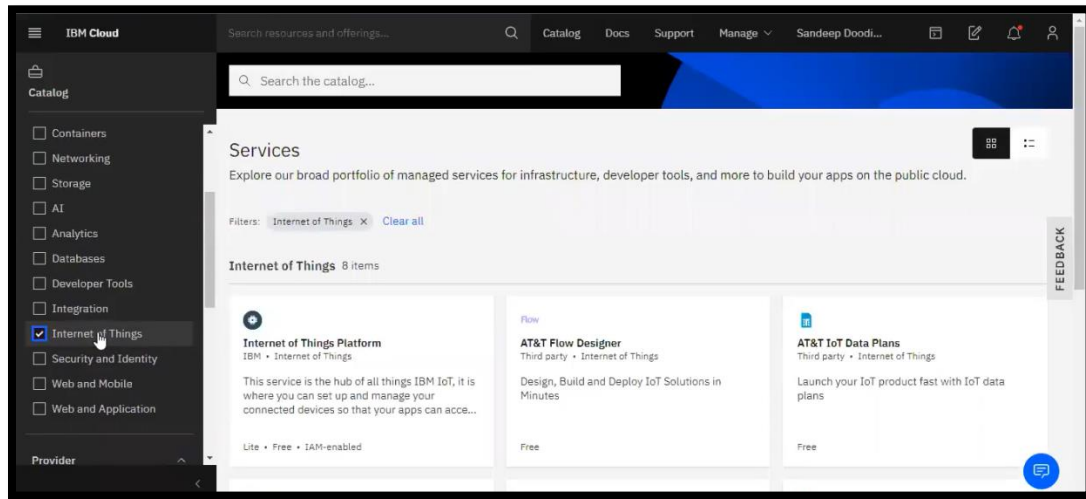
1. **Connectivity & Normalization**: Brings different protocols and different data formats into one “software” interface ensuring accurate data streaming and interaction with all devices.
2. **Device Management**: Ensures the connected “things” are working properly, seamlessly running patches and updates for software and applications running on the device or edge gateways.
3. **Database**: Scalable storage of device data brings the requirements for hybrid cloud-based data bases to a new level in terms of data volume, variety, velocity and veracity.
4. **Processing & Action Management**: Brings data to life with rule-based event-action-triggers enabling execution of “smart” actions based on specific sensor data.
5. **Analytics**: Performs a range of complex analysis from basic data clustering and deep machine learning to predictive analytics extracting the most value out of the IoT data-stream.
6. **Visualization**: Enables humans to see patterns and observe trends from visualization dashboards where data is vividly portrayed through line-, stacked-, or pie charts, 2D- or even 3D-models.
7. **Additional Tools**: Allow IoT developers prototype, test and market the IoT use case creating platform ecosystem apps for visualizing, managing and controlling connected devices.
8. **External Interfaces**: Integrate with 3rd-party systems and the rest of the wider IT-ecosystem via built-In application programming interfaces (API), software development kits (SDK), and gateways.

2.1.1 IBM IoT Device creation and connecting Internal Simulator:

Steps for IBM IoT device creation and connecting Internal Simulator:

1. First we have to Create IBM Cloud Account at <https://www.ibm.com/en/cloud> .
2. Next we have login into our account.
3. You will display the IBM dashboard. In that dashboard you can see our active IBM Services.
4. Click on catalog we will get the list of the services provided by the IBM Cloud.
5. Create Internet of Things Service.
6. After service is Created, we have to add devices to the IoT Platform.
7. We can add the devices like Raspberry Pie, Arduino etc.
8. We have to connect device to the IBM virtual IoT simulator <http://watson-iot-sensor-simulator.mybluemix.net/> .





The screenshot shows the 'Browse Devices' page in the IBM Watson IoT Platform. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A user profile in the top right shows 'sandeep.doodigani@gmail.com' and 'ID: sj98sz'. A sidebar on the left contains various icons. The main content area has a 'Browse Devices' title, a 'Diagnose' button, and a table with columns: Device ID, Status, Device Type, Class ID, and Date Added. A search bar and a 'Device Simulator' toggle are also present.

IBM Watson IoT Platform

sandeep.doodigani@gmail.com
ID: sj98sz

Browse Action Device Types Interfaces

Add Device

Browse Devices

All Devices Diagnose

This table shows a summary of all devices that have been added. It can be filtered, organized and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Search by Device ID

Device Simulator

Device ID	Status	Device Type	Class ID	Date Added
-----------	--------	-------------	----------	------------

Cookie Preferences

The screenshot shows the 'Add Device' wizard in the IBM Watson IoT Platform. The wizard has four steps: Identity, Device Information, Security, and Summary. The 'Identity' step is currently active, showing a form to select a device type and enter a unique ID. The 'Device Type' field is set to 'raspberrypi' and the 'Device ID' field is set to '654321'. There are 'Cancel' and 'Next' buttons at the bottom.

IBM Watson IoT Platform

ID: sj98sz

Browse Action Device Types Interfaces

Add Device

Identity Device Information Security Summary

Select a device type for the device that you are adding and give the device a unique ID.

Device Type: raspberrypi

Device ID: 654321

Cancel Next

Cookie Preferences

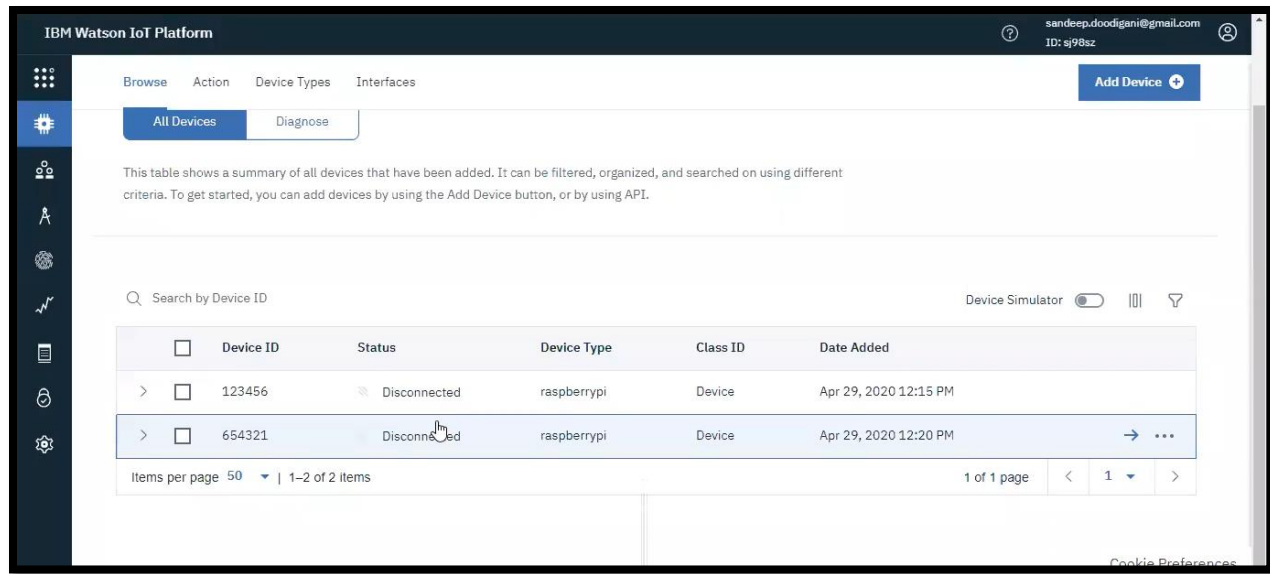
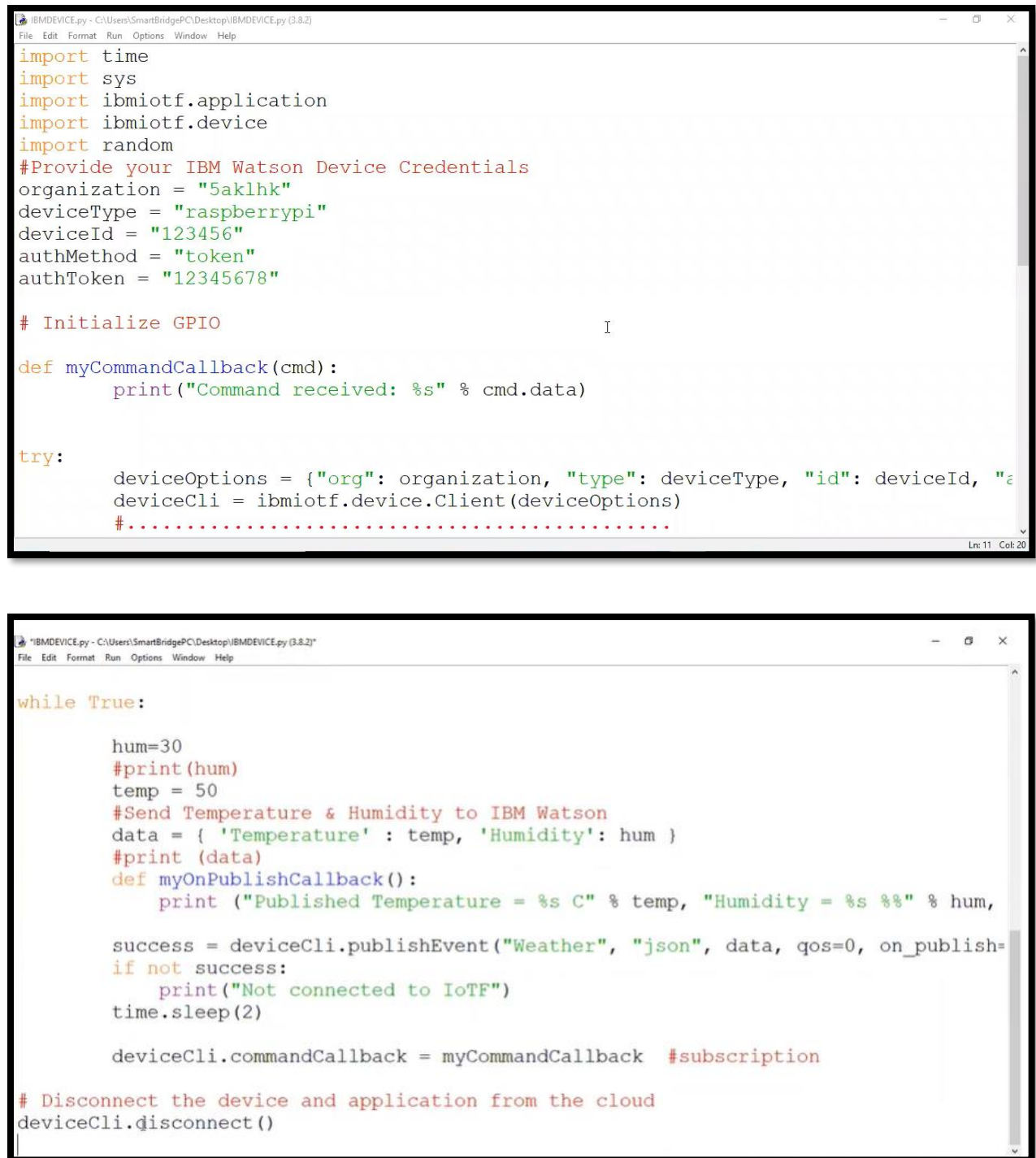


Figure 13: IBM account Creation & IoT Service

2.1.2 Connecting to IBM IoT Platform and sending sensor data using python code:

- Previously we created IoT Platform and connected to Virtual IBM IoT simulator.
- Now we using python code to perform IoT
- By using Device data in the IoT Platform we can send data by using the Python code.
- To use Python code, we have to install some library files.
- We can check whether the required library files are available in our laptop/Computer by using command prompt.
- For installation use **pip install ibmiotf** in command prompt



```

IBMDEVICE.py - C:\Users\SmartBridgePC\Desktop\IBMDEVICE.py (3.8.2)
File Edit Format Run Options Window Help

import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
#Provide your IBM Watson Device Credentials
organization = "5aklhk"
deviceType = "raspberrypi"
deviceId = "123456"
authMethod = "token"
authToken = "12345678"

# Initialize GPIO

def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data)

try:
    deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "authMethod": authMethod}
    deviceCli = ibmiotf.device.Client(deviceOptions)
    #.....

while True:
    hum=30
    #print(hum)
    temp = 50
    #Send Temperature & Humidity to IBM Watson
    data = { 'Temperature' : temp, 'Humidity': hum }
    #print (data)
    def myOnPublishCallback():
        print ("Published Temperature = %s C" % temp, "Humidity = %s" % hum,
        success = deviceCli.publishEvent("Weather", "json", data, qos=0, on_publish=
    if not success:
        print("Not connected to IoT")
        time.sleep(2)

    deviceCli.commandCallback = myCommandCallback #subscription

# Disconnect the device and application from the cloud
deviceCli.disconnect()

```

Figure 14: Python Programming to send data

CHAPTER 03

3.0 IBM Cloud Services

3.0.0 IBM Cloudant:

What is IBM Cloudant?

IBM Cloudant is a fully managed JSON document database that offers independent serverless scaling of throughput capacity and storage.

How to open IBM Cloudant and use it?

First fall open IBM cloud and there will open a dashboard page and click on catalog and there will see all the services of IBM. and click on catalog then left side we see services option click on it and search for Cloudant, and there will see a Cloudant data base and click on it.

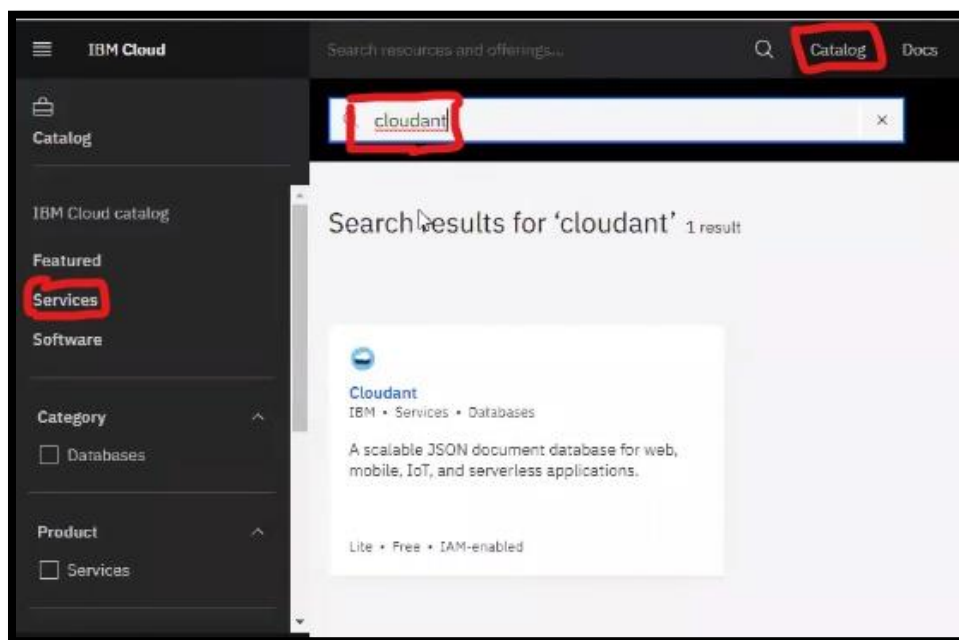


Fig18: Cloudant in catalog page

In Cloudant we store the data in JSON format. The Cloudant page will be open and fill necessary credentials and click on create.

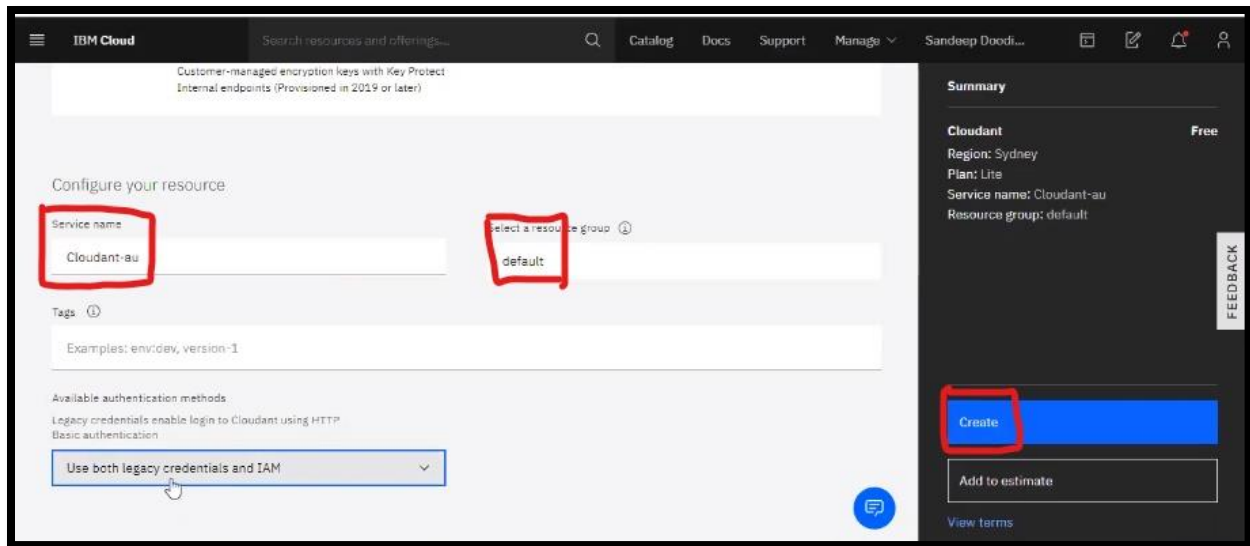


Fig 19: Cloudant database creating page

Now open Cloudant service and click on launch dashboard page.

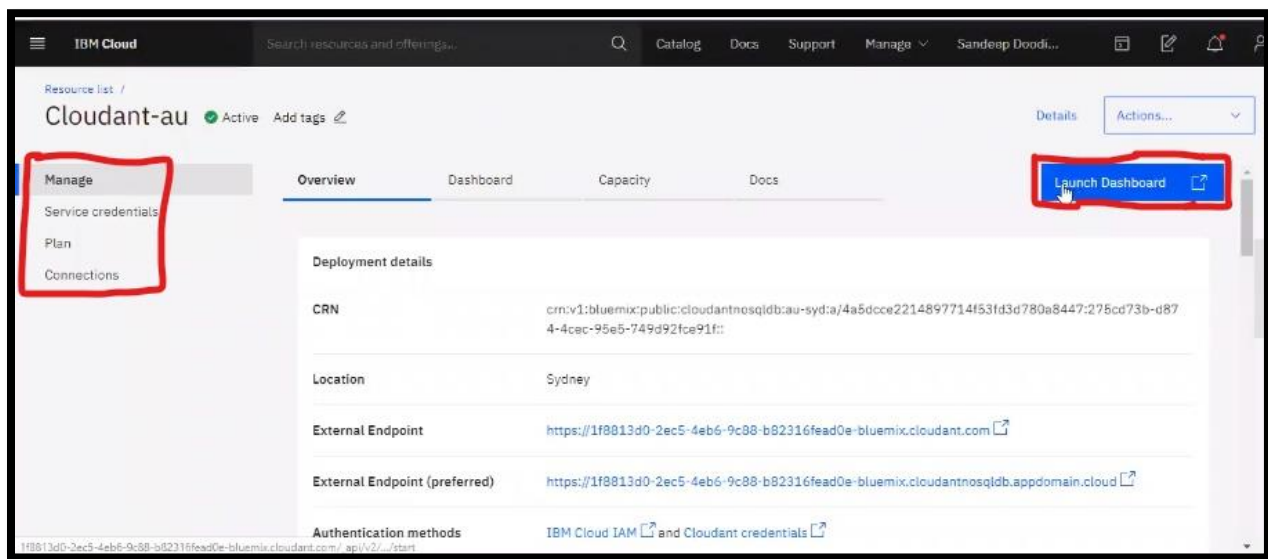


Fig 20: Cloudant launch dashboard

And database page will be open for us. and click on create database and there a pop page will be open on right side and create a database name and there we have an option is partitioning in that we have to click on non-partitioned and then click on create button.

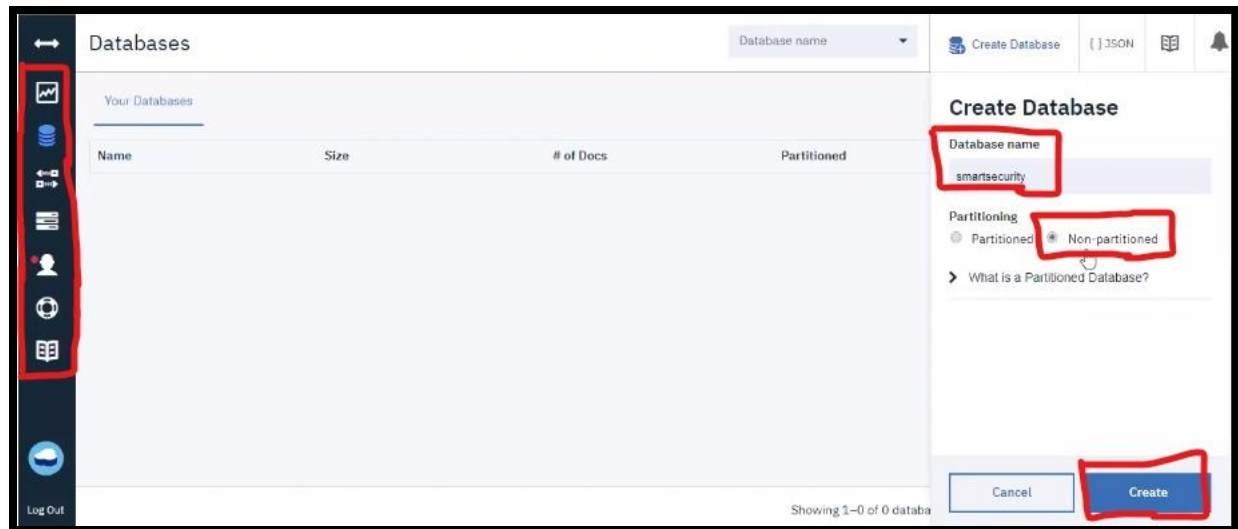


Fig 21: creating database

And we create our data here and document will be created in it. To run the python code in it. We have to open database page and check data is created or not.

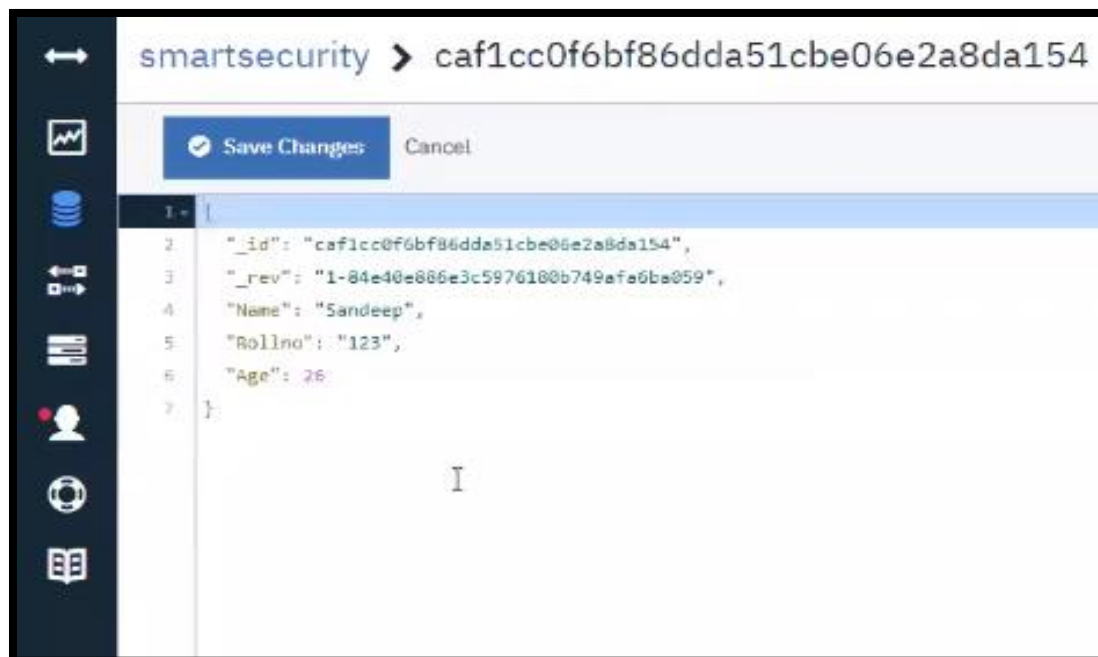


Fig 22: Editor page



The screenshot shows the IBM Cloud Object Storage interface. On the left is a sidebar with navigation options: All Documents, Query, Permissions, Changes, and Design Documents. The main area displays a document in JSON format. The document has a single entry with a key and a value.

id	key	value
ca1cc0f6b186dda51cbe06e2a8da154	ca1cc0f6b186dda51cbe06e2a8da154	["rev": "1-84e40e886e3c5976180b7..."]

At the bottom, it indicates 'Showing document 1 - 1. Documents per page: 20'.

Fig 23: Data is created in j.son format

3.0.1 IBM Object Storage:

What is IBM object storage?

IBM Object Storage makes it possible to store practically limitless amounts of data, simply and cost effectively. It is commonly used for data archiving and backup; for web and mobile applications; and as scalable, persistent storage for analytics.

How to open IBM Object Storage and use it?

First fall open IBM cloud and there will open a dashboard page and click on catalog and there will see all the services of IBM. and click on catalog then left side we see a services option click on it. Search for Object Storage, there we observe Object Storage then click on it.

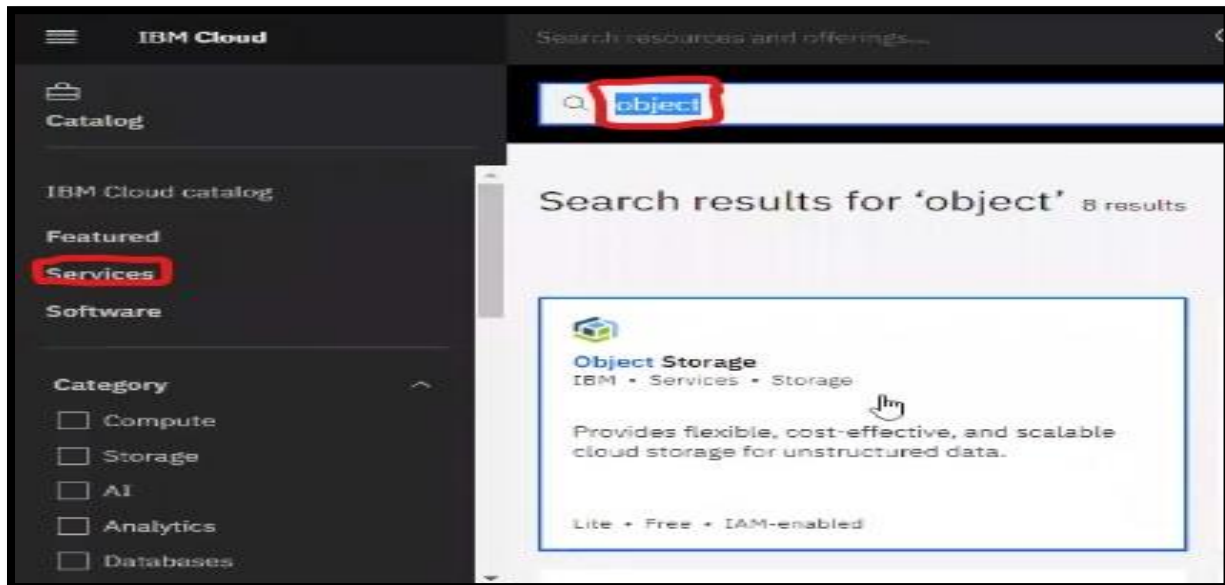


Fig 24: Object storage

Object storage page will be open then click on create option then service will be created.

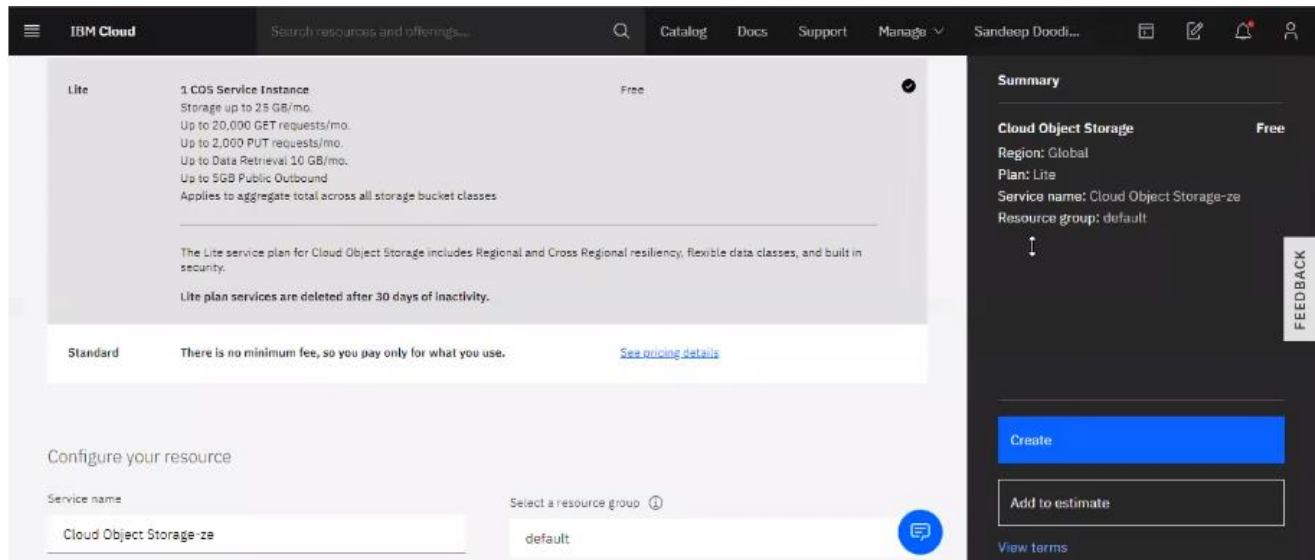


Fig 25: Creating object storage

Object storage will be open and click on buckets and then click on create bucket button there are different kind of buckets are there. We have to click on custom bucket and bucket should always be unique name and click on create bucket.

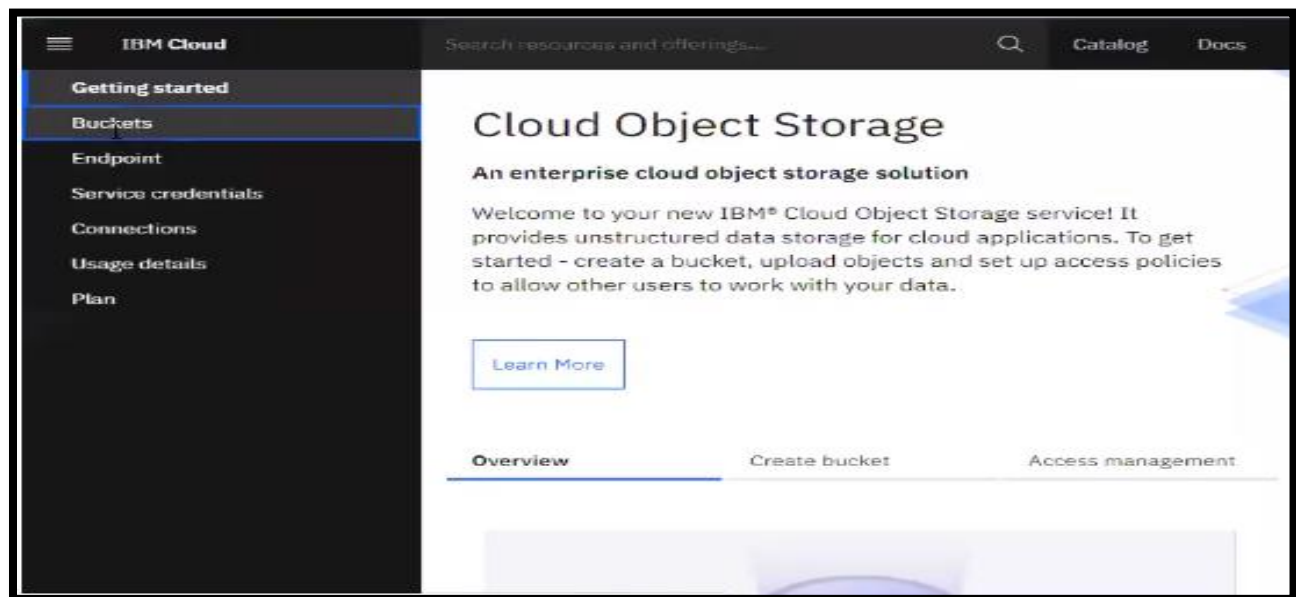


Fig 26: Creating buckets

There will have a drag and drop option then click on it. A pop page will be opened and select any file you want to store in it.

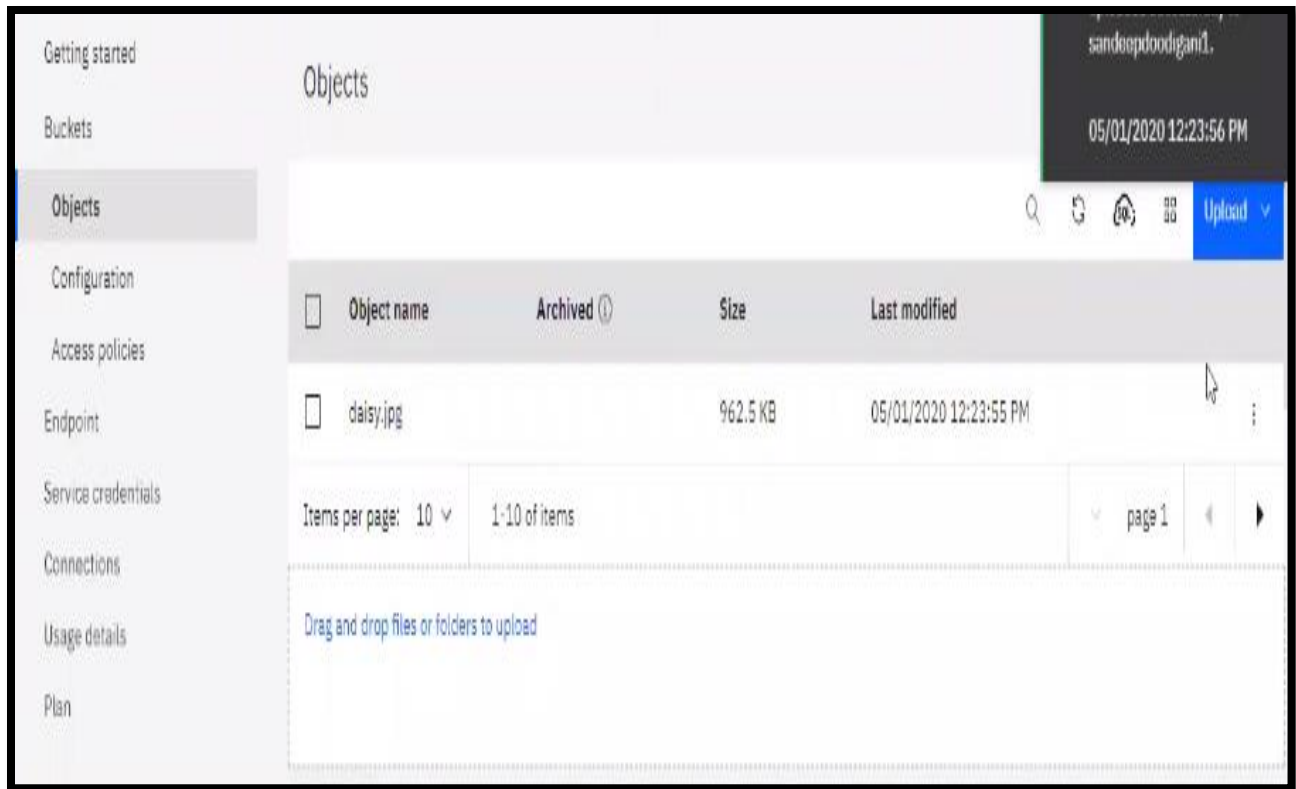


Fig 27: Data is stored

3.1 NODERED

WHAT IS Nodered?

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways. It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.

3.1.0 Retrieving Data from IBM IoT Platform:

First fall open IBM cloud and there will open a dashboard page and click on catalog and there will see all the services of IBM. and click on catalog then left side we see software option click on it. and search for Node-red and there will see a Node-Red App then click on it. Node-Red page will be open and back end it operates under Cloudant database. Node-Red is been developed on Node. JS and click on create and give some necessary credentials in it. Then click on Next and duplicate the same page and come back to dashboard page then click on services in that click on internet of things platform then click on launch button. So, it will be launch IBM IoT platform. In node-red page click on IBM cloud and click on cloud foundry apps in that click on Node-Red and then click on Visit App URL. then welcome page will be open of Node-Red and click on next finally click on finish. And click on Go to Node-Red flow Editor then open the Node-Red tool page. in Node-Red left side we have filter nodes and middle page we have work bench and right side we have console and every time we have to click on Deploy to save the flow. Connect necessary nodes and click on Deploy to save the flow of the Node-Red, in right side of the corner we have two buttons one is Node information and other one is Debug messages and then click on

the button on work bench then we see the debug information on the right side of the debug message area. We have to Retrieving Data from IBM IoT Platform and we have installed some libraries in it. Then click on menu button and go to manage palette and click on install and search for IBM IoT and click on install and then click on again install. and left side we have seen IBM IoT nodes and drag the node to work bench and double tab on the IBM IoT and insert API KEY and API token in it and click on add and come back same page then add some credentials in it. and then connected to the IBM IoT platform and whatever data coming in IoT platform that data comes to node-red flow also.

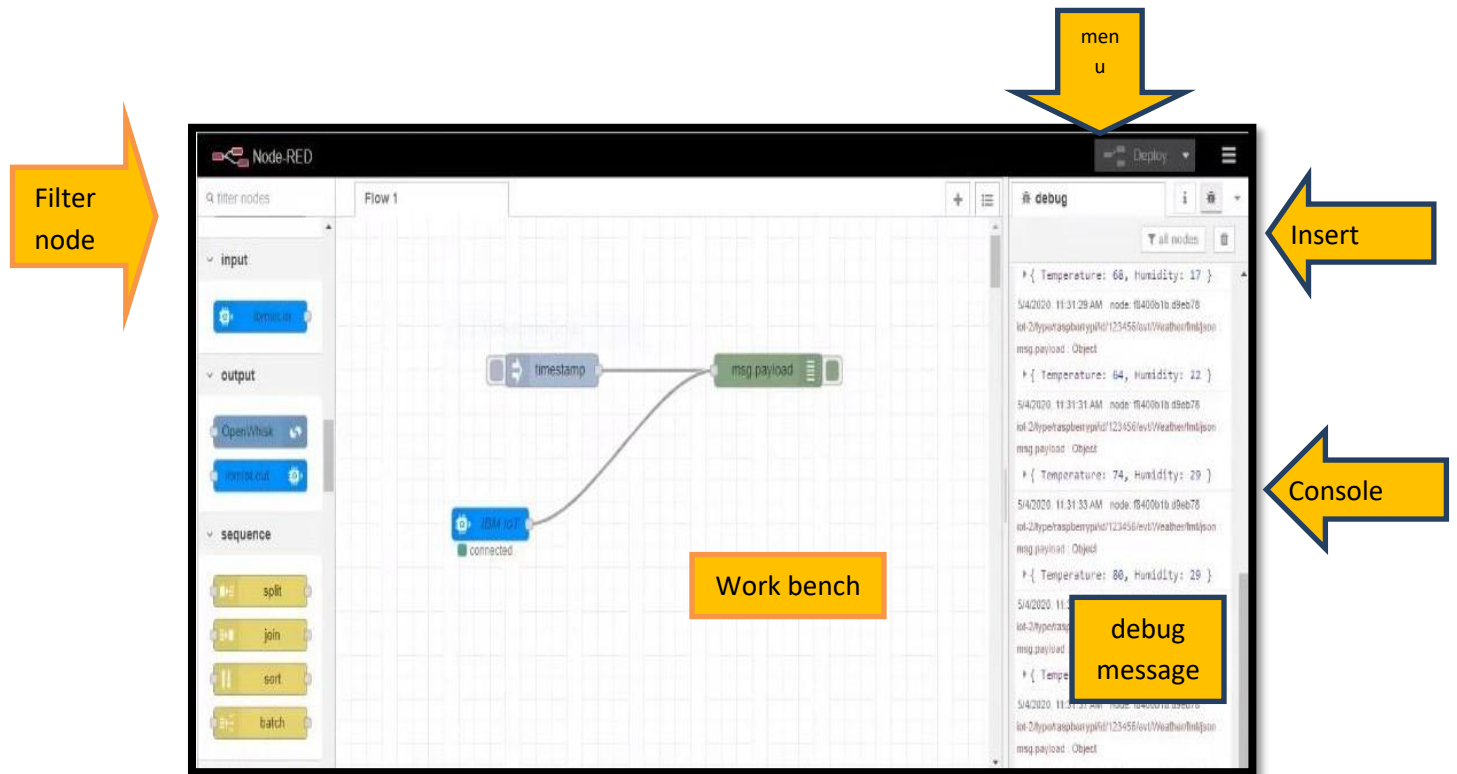


Fig 28: Node-Red flow diagram

3.1.1 Sending Command to the Device:

Sending commands to connected device by using IoT simulator. By adding additional functions to it, temperature and humidity functions to it to the message. Payload sees same values in the IoT simulator. Whatever information comes in the node-red. If we want to see the interface of UI then copy the node-red URL and open the new tab and paste URL in Web and add additional term UI in the web lastly and then press enter it is shown as figure.



Fig 29: IoT Simulator

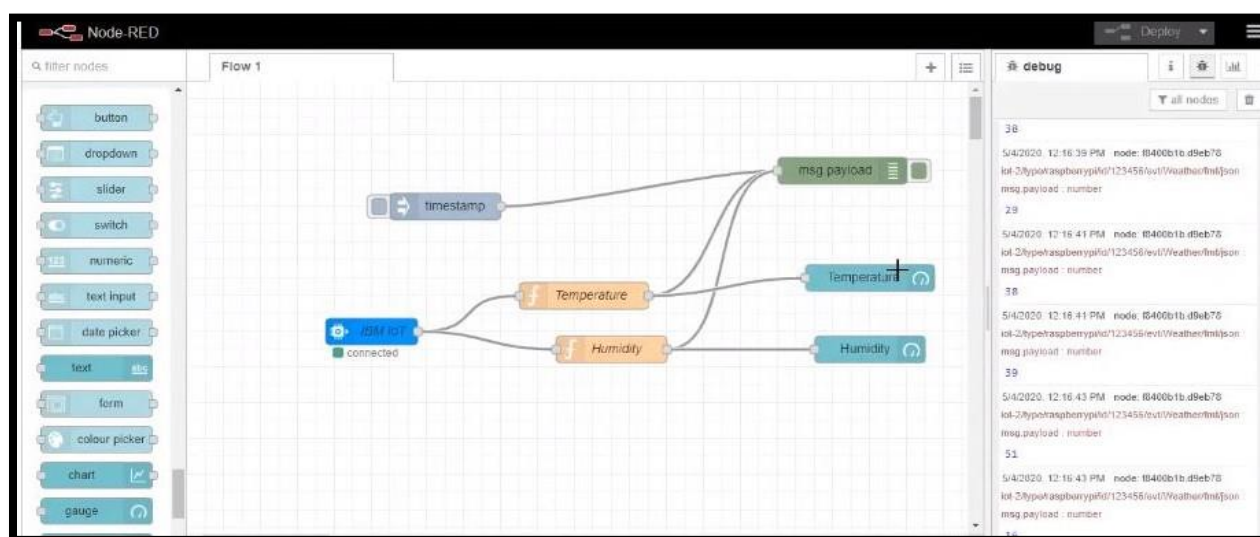


Fig 30: Sending command to device

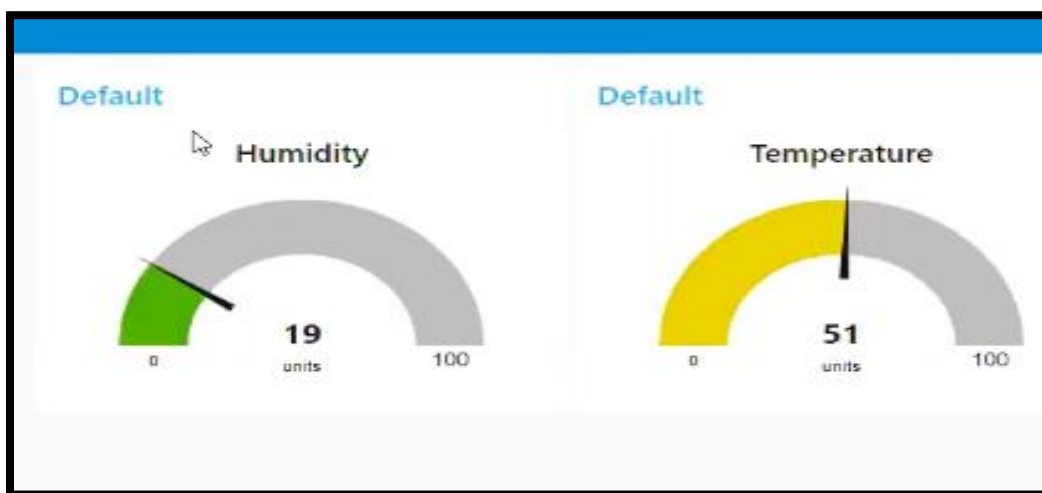


Fig 31: Receiving data from web

CHAPTER 04

4.0 MIT App Inventor:

What is MIT App Inventor?

- MIT App Inventor is an intuitive, visual programming environment that allows everyone to build fully functional apps for smart phones and tablets. It is free and open-source software.
- It is originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT).
- It allows newcomers to computer programming to create application software apps for two operating systems Android, and IOS.

How to download the created app?

- 1) Click on Build and save.apk file or from the QR code.
- 2) Install app from play store MIT AI2 Companion. Click on connect and click AI companion.

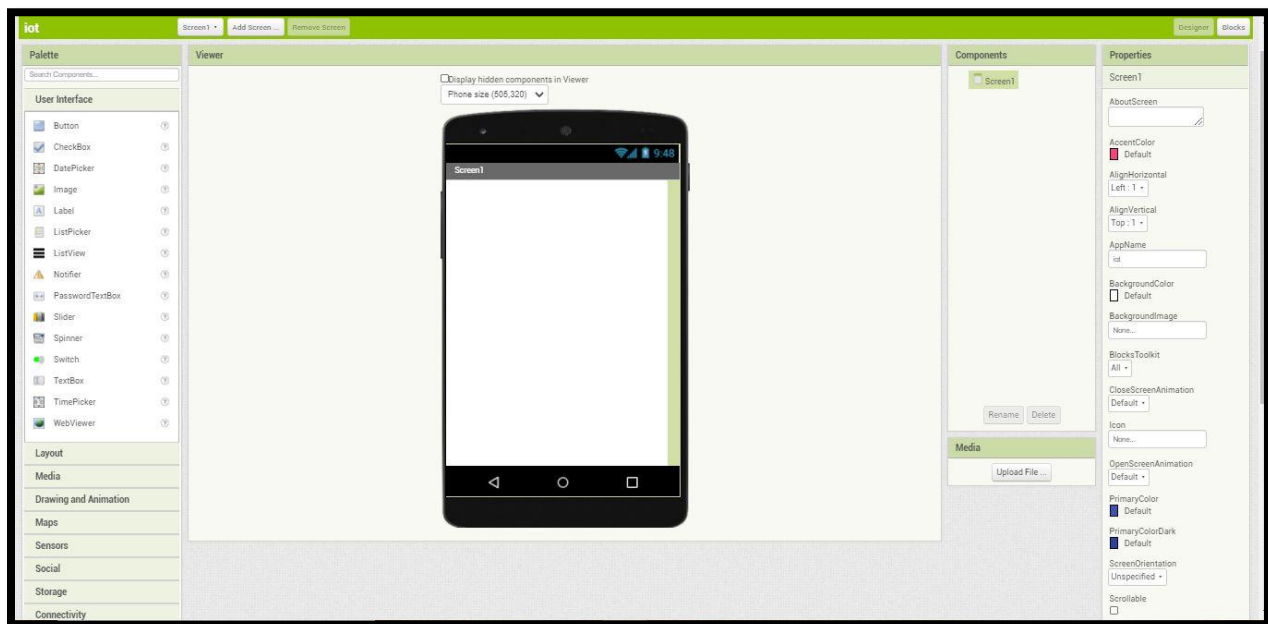


Fig 37: Creating a MIT APP

4.0.0 Display the Sensor Values in the mobile application:

Login to the MIT App inventor and click on the designer to display the sensor values of temperature values and humidity values text box is required, so drag the text box from palette and viewer.

To indicate whether it is temperature or humidity we need label drag and drop the label from palette to viewer for temperature, humidity and rename it as temperature and humidity from the properties of label.

The sensor values have to send from Node-Red to MIT app inventor for that http protocol is used. In Node-Red drag and drop http in and response. By clicking http GET method is used URL as "/data".

For function node is used to send JSON data to the http response and click deploy. copy the Node-Red URL upto.net and add "/data" for the temperature and the humidity.

Using this URL the output can be displayed in MIT App inventor, for that drag and drop Web viewer, clock from palette to the viewer.

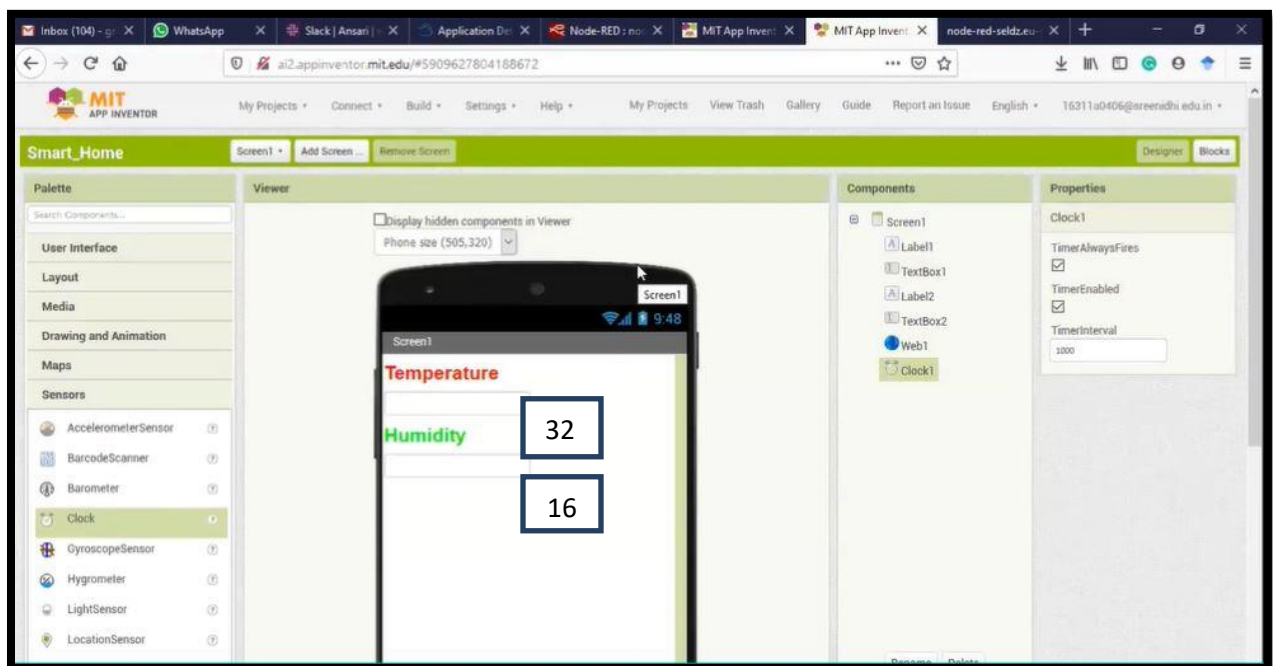


Fig 38: Displaying the sensor values

Click on blocks drag and drop clock1-do block to get the data from the web. At every second clock call web1. When data is obtained it is in JSON format Web1. getText-do block is used and set TextBox1.Text to look up pair key which decodes JSON data to actual data in temperature.

To obtain the humidity value set TextBox2.Text to lookup in pairs key to humidity.

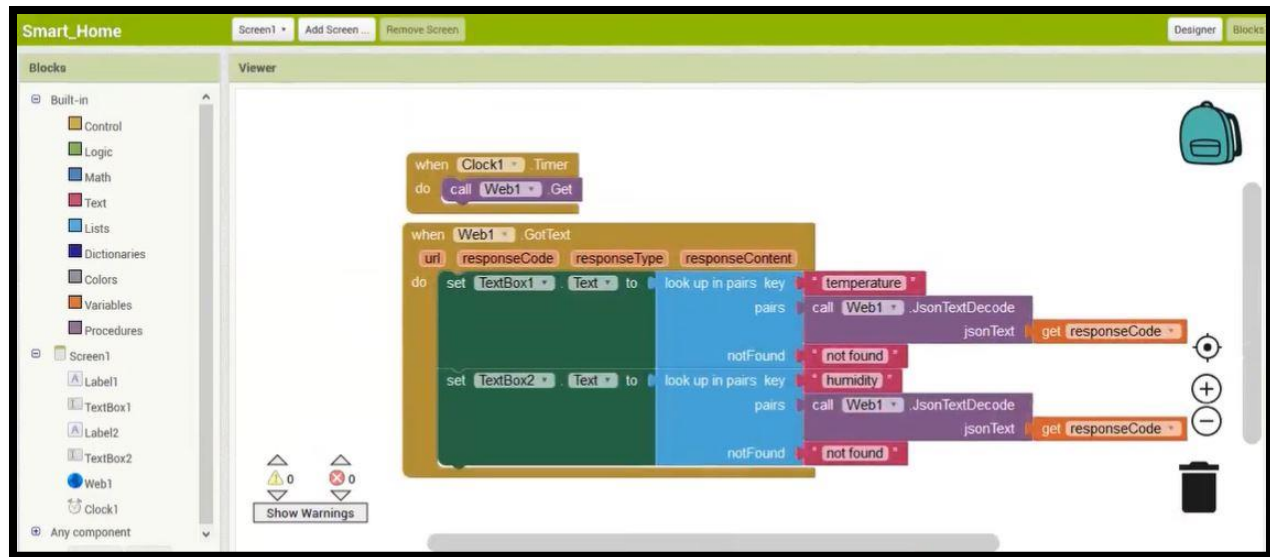


Fig 39: Adding textbox1 & 2

4.0.1 Controlling the applications using Mobile App:

By using buttons one can control home applications like light on and light off. Drag and drop two buttons from the palette to viewer and edit text as light on and light off set the alignment left or right or center. To send the data from the MIT to Node-Red using URL we can send the data by using HTTP nodes in the Node-Red. HTTP and HTTP response nodes are drag and drop in the Node-Red, function node is placed in between it. Data is sends in the form of commands from MIT to Node-Red. when button light on is pressed in MIT App this URL is used to send data "Node-Red URL /command? command=light on" and same in the case of light off button.

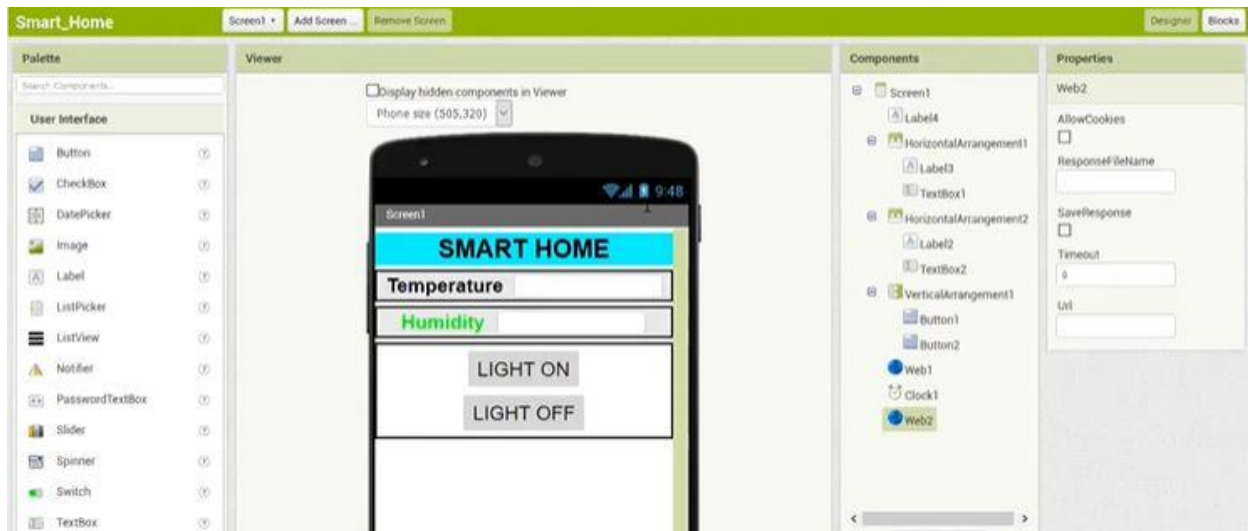


Fig 40: Controlling the mobile app application

In MIT App inventor drag and drop the web viewer from the blocks to the viewer to send the commands to Node-Red. Click on blocks drag and drop when button1 click set Web2 URL to light on and set button1 background color to red and button2 background color as green and call the URL.

When it calls the URL, it sends data to the Node-Red that light on. when button2 click set Web2 URL to light off and set button2 background color to red and set button1 background color to green.it sends the command to the Node-Red that light is off.

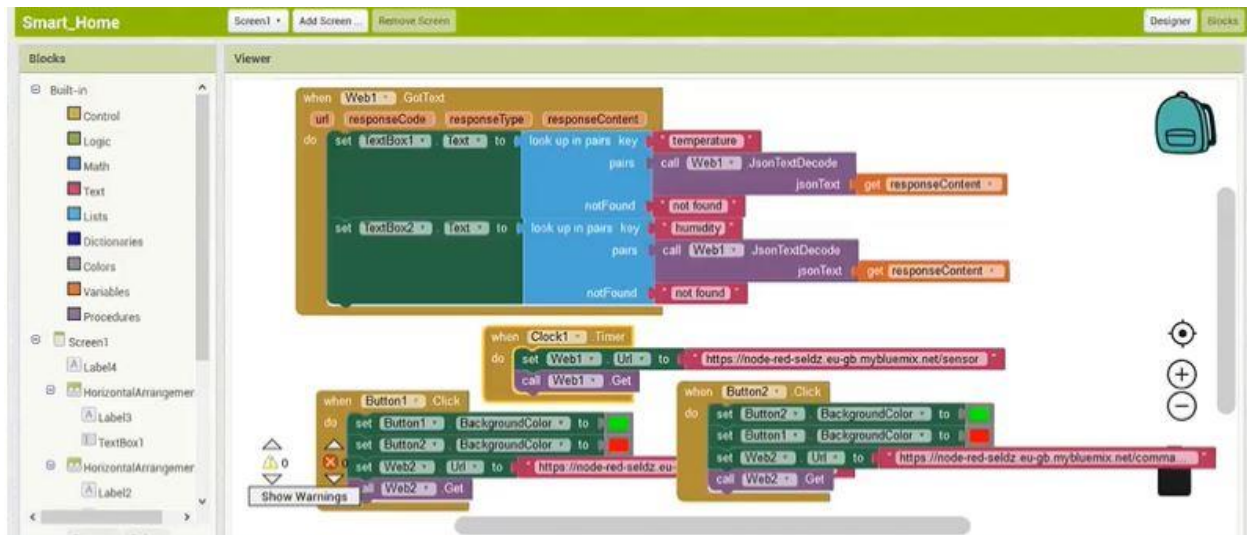


Fig 41: Block codes

4.1 Flask:

What is a Flask?

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It is based on Werkzeug WSGI toolkit and Jinja2 template engine.

What are the applications required to build?

- 1) Python
- 2) HTML
- 3) JavaScript

What is WSGI and Jinja2?

WSGI:

- Web Server Gateway Interface (WSGI) has been adopted as a standard for Python web application development.
- It is a specification for a universal interface between the web server and the web applications.

Jinja2:

- Jinja2 is a popular templating engine for Python. A web templating system combines a template with a certain data source to render dynamic web pages.
- To pass the data from python to html we can use Jinja2 template.

How to install Flask?

Open Command Prompt and click pip install Flask

4.1.0 Flask Tutorials & Flask Basics:

To use the Flask module in python, from flask import Flask is used.

Route Function:

The route() function of the Flask class is a decorator, which tells the application which URL should call the associated function.

Ex: @app.route('/')

Every route should have a unique function definition

__name__ Constructor:

Flask constructor takes the name of current module __name__ as argument.

Run Method:

Finally the run() method of Flask class runs the application on the local development server.

Output of the code:

Save the code using .py extension and run the code. A message in Python shell informs you that Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

Now the link copy and paste it on web browser to get the output, don't press CTRL+C.

Debug mode:

- A Flask application is started by calling the run() method. However, while the application is under development, it should be restarted manually for each change in the code. To avoid this inconvenience, enable debug support.
- The server will reload itself if the code changes. It will also provide a useful debugger to track the errors if any, in the application.
- The Debug mode is enabled by setting the debug property of the application object to True before running or passing the debug parameter to the run() method.
- app.run(debug=True)
- It is possible to build a URL dynamically, by adding variable parts to the rule parameter. This variable part is marked as <variable-name>.
- It is passed as a keyword argument to the function with which the rule is associated.
- ex: @app.route('/hello/<name>')

To call the html page using python Flask:

- To call the html file, first create the templates folder and python file in the same folder. All the html files should save in templates folder.

- Open new file and write the html code and save the file in the templates folder using the .html as the extension.
- In python file import Flask, Render_template from flask. Route function return render_template('html_file'). Render template is used to call the html file and send the data to the html.
- Copy the file path and paste in the command prompt and run the python code in command prompt.

HTTP Methods:

- 1) GET: Sends data in unencrypted form to the server.
- 2) HEAD: Same as GET, but without response body.
- 3) POST: Used to send HTML form data to server. Data received by POST method is not cached by server.
- 4) PUT: Replaces all current representations of the target resource with the uploaded content.

4.1.1 Flask graphics and Introduction to IBM Cloud Functions

To show the data from the sensors in the form of graph Flask graphics is used.

Html

4.2 Applications

4.2.0 Smart Security Using Node-RedService

- Login to the IBM Cloud and open the Cloudant database service, Node-Red and object storage. Give the service credentials of the object storage.
- From the end point give the resiliency as regional, location as jp-tok, copy and paste the public URL.
- Give the service credentials to the Cloudant service to the python code. Give the database name =sample.
- Open the Fast2sms website and login. click on Dev API and then open read API documentation. Click on Bulk SMS API and click Get Method.
- Copy the URL and paste it new tab and give the API key from Fast2sms and give the mobile number.

4.3 IBM Cloud Functions

- IBM Cloud Functions is a distributed compute service that executes application logic in response to requests from web or mobile apps. You can set up specific actions to occur based on HTTP-based API requests from web apps or mobile apps, and from event-based requests from services like Cloudant. Functions can run your code snippets on demand or automatically in response to events.
- All APIs are protected with HTTP Basic authentication. You can use the wskadmin tool to generate a new namespace and authentication.

CHAPTER 05

PROJECT

1. Purpose of the Project
2. Block Diagram
3. Hardware/Software Designing
4. Flowchart
5. Advantages & Disadvantages
6. Applications
7. Result
8. Conclusions

5.1 Purpose of the Project

The most important factors for the quality and productivity of plant growth are temperature, humidity, light and the soil moisture levels. Continuous monitoring of these environmental variables gives information to grow better how each factor affects growth and how to manage maximal growth of plants. Climate control and monitoring of the plant is one of important aspect. The aspects we are presenting resembles the concept of precision agriculture. The main motivation of the project is for the user to monitor the plants and also could manipulate the resources provided to the plants depending on the climate of the plant's location.

This could help user not only to give the resources to the plants everyday without much manual effort also helps the constant and healthy growth of a plant. This project is designed as a smart plant communicator system with IBM cloud based on IOT. In this project we used different modules such as IOT, Node red, Temperature, soil moisture levels, Humidity

5.2 Proposed Solution

This Project examines and compares some IOT regression methods. So that we tried to make a system which will monitor the health conditions of the plants. Our systems will check different environmental parameters and with the connection of an IOT platform we are going to have the information of the conditions of our plants and then we can take effective steps for the welfare of the plants. In this system the heart of our system will be worked as a cloud platform. If any change in the temperature, humidity and soil moisture levels occurs, a message will be sent to the user smart phone. Temperature, Humidity and Soil Moisture is visualized in the mobile app and also stored in the database. Motor can be turned ON/OFF using the mobile app.

5.3 Block Diagram

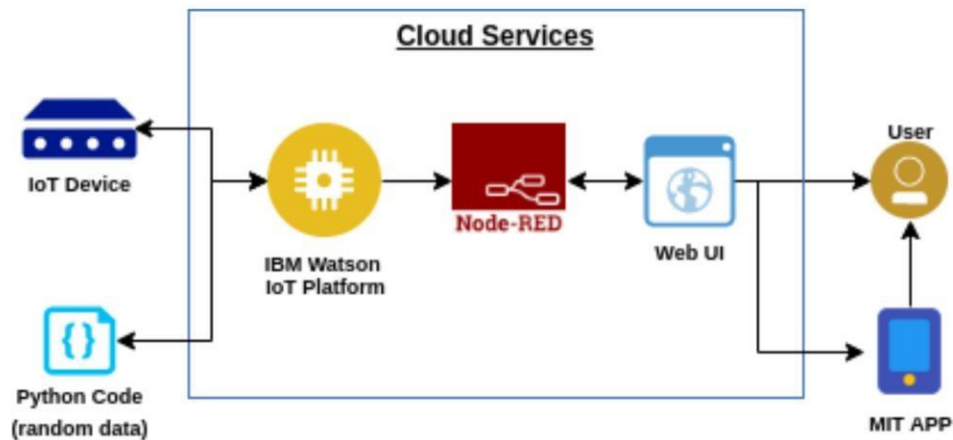


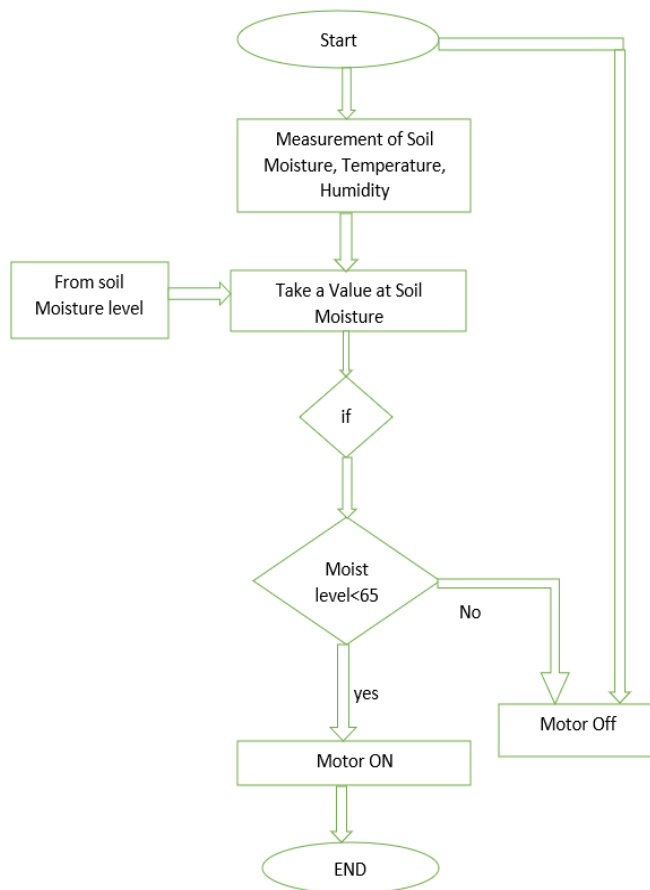
Fig: Smart plant communicator system with IBM cloud

5.4 Hardware/Software Designing

By using

Python
IOT Open Hardware Platforms
IOT Application Development
IOT Cloud Platform
IOT Communication Technologies
IOT Communication Protocols

5 .FLOWCHART



Python Code:

```
#IBM Watson IOT Platform

#pip install wiotp-sdk

import wiotp.sdk.device

import time

import random

myConfig = {
    "identity": {
        "orgId": "ttmy0h",
        "typeId": "plant",
        "deviceId": "123654"
    },
    "auth": {
        "token": "12345678"
    }
}

def myCommandCallback(cmd):
    print("Message received from IBM IoT Platform: %s" % cmd.data['command'])
    m=cmd.data['command']
    if m=="pump ON":
        print("The pump is switched ON")
    else:
        print("The pump is switched OFF")
```



```
client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)

client.connect()

while True:

    temp=random.randint(-20,125)

    hum=random.randint(0,100)

    soilmoistLevel=random.randint(0,100)

    user=random.randint(0,10)

    if soilmoistLevel<65:

        print("Motor on")

    else:

        print("Motor off")

    myData={'temperature':temp, 'humidity':hum, 'soilmoistureLevel':soilmoistLevel, 'guest':user}

    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0,
onPublish=None)

    print("Published data Successfully: %s", myData)

    client.commandCallback = myCommandCallback

    time.sleep(2)

client.disconnect()
```

7 ADVANTAGES & DISADVANTAGES

Advantages:

- 1) Ability to save water
- 2) Reduced water consumption
- 3) Safe
- 4) No manpower required
- 5) Require smaller water sources
- 6) Reduce soil erosion and nutrient leaching

Disadvantages:

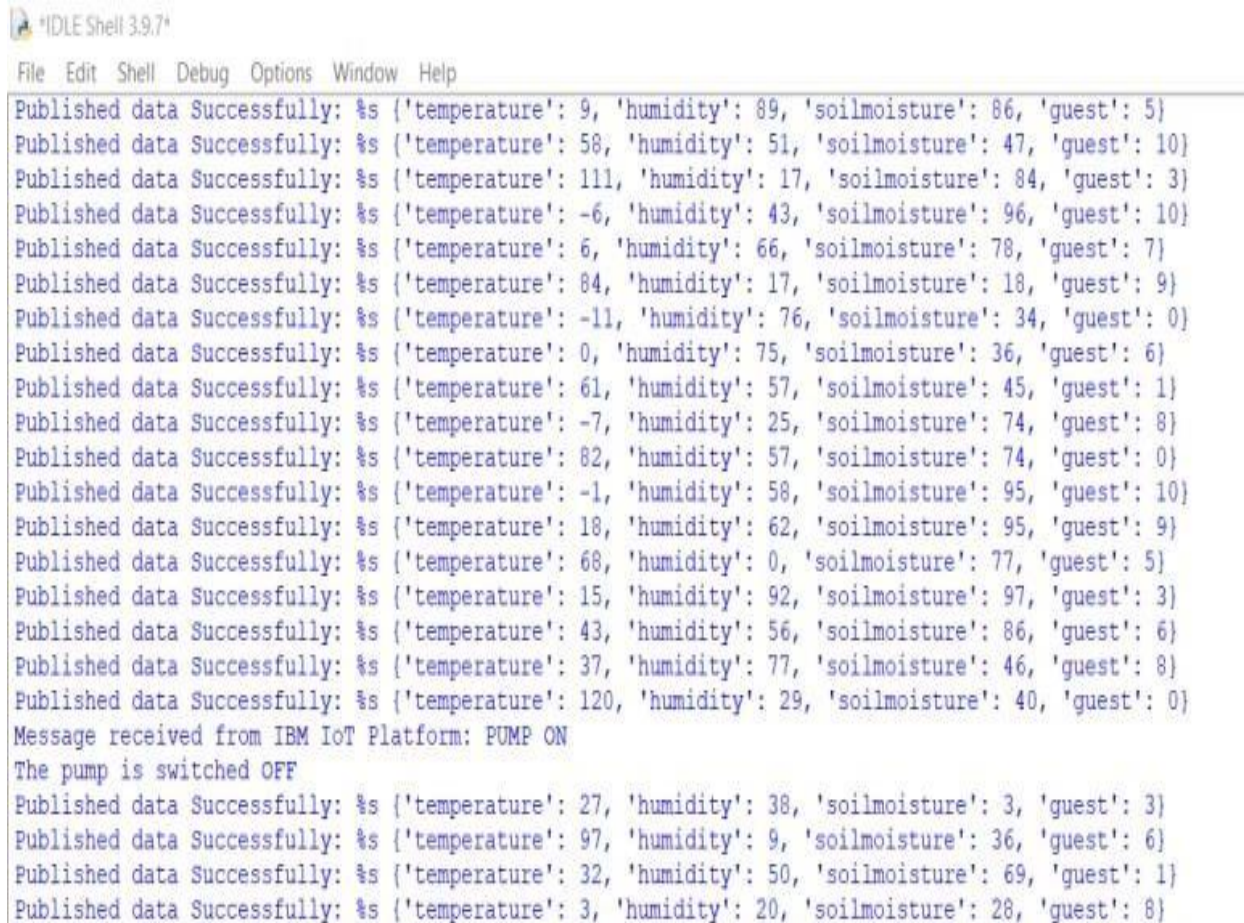
- 1) The smart plant need availability on internet continuously. Rural part of the developing countries did not fulfill this requirements. Moreover internet is slower.
- 2) Fault sensor or data processing engines can cause faulty decisions which may lead to over use of water, fertilizers and other wastage of resources.

8 APPLICATIONS

We can implement in various fields like

- 1) It is suitable to check the soil moisture levels, which may help to start a better growth of a plant
- 2) It may also be used in rural areas
- 3) May be implemented by small agriculturists

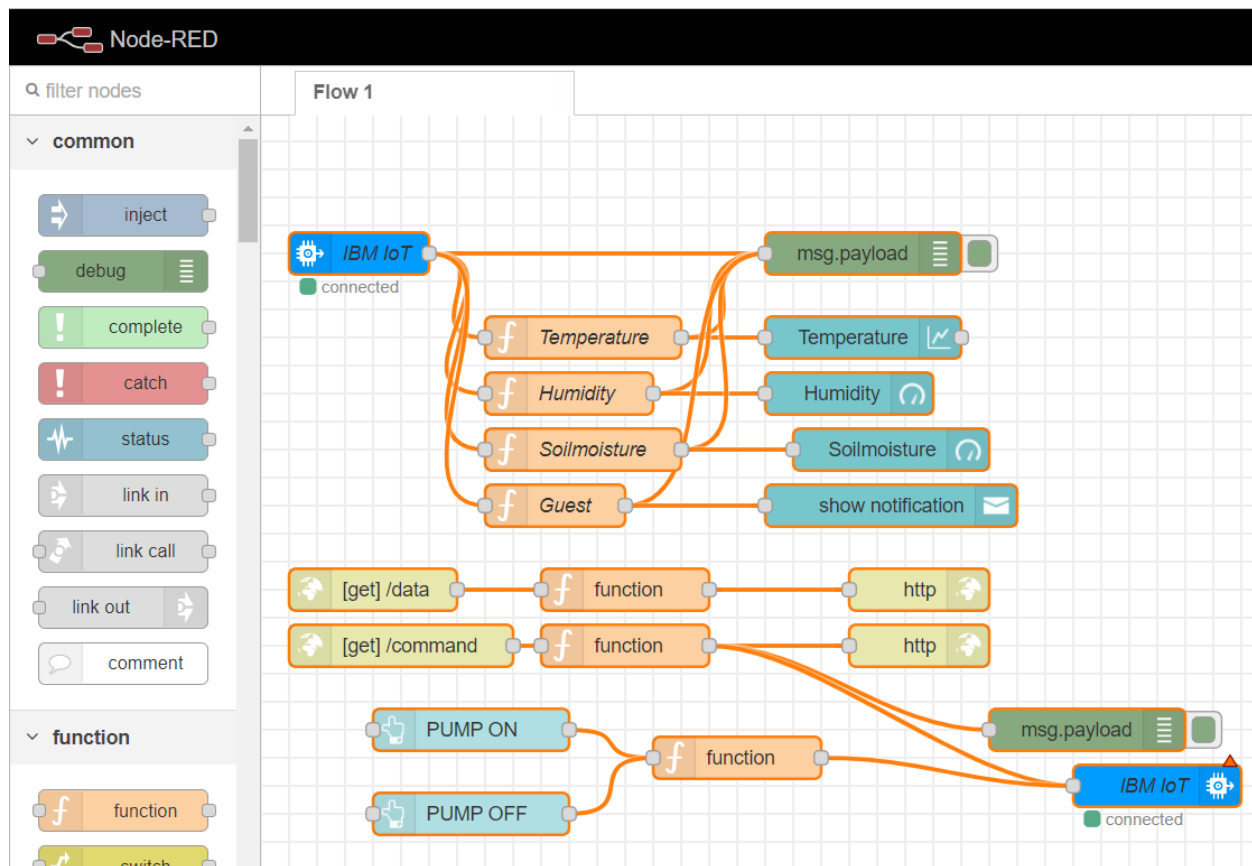
RESULTS

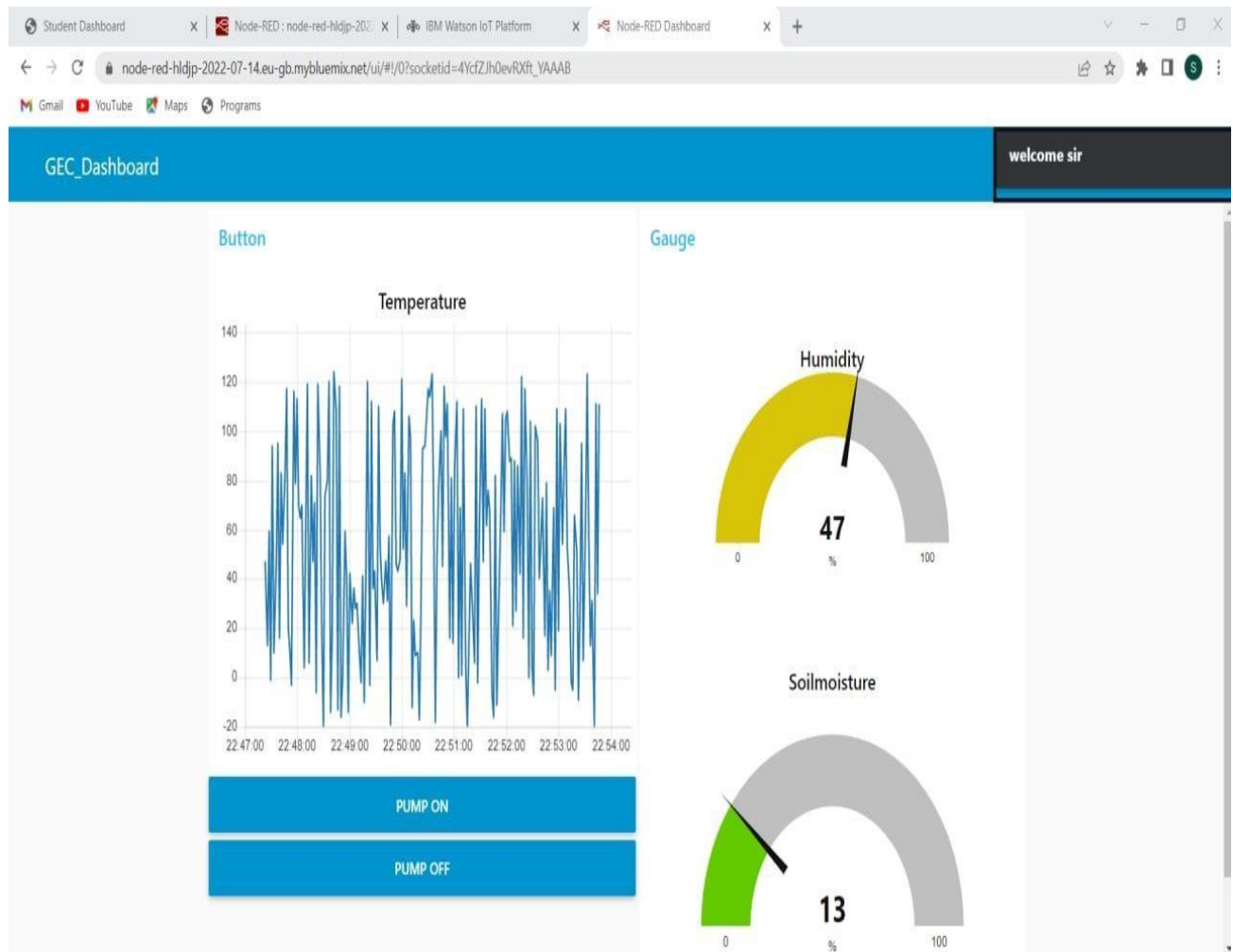


```

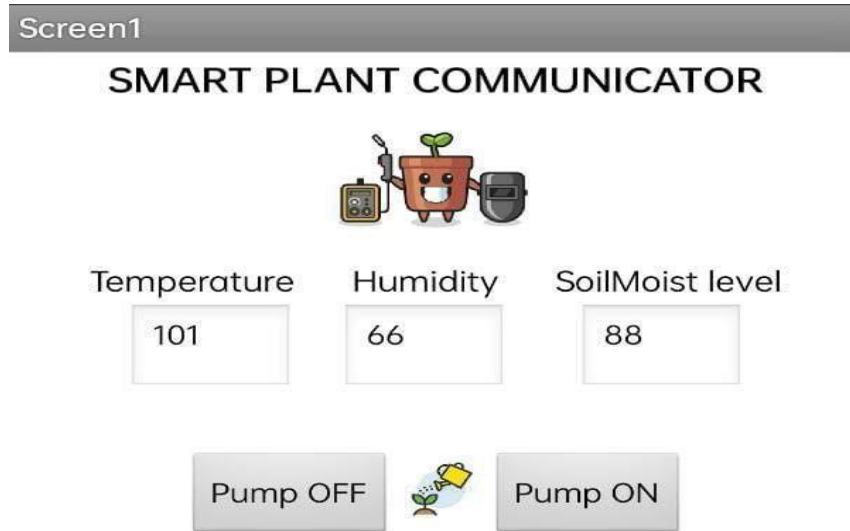
IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
Published data Successfully: %s {'temperature': 9, 'humidity': 89, 'soilmoisture': 86, 'guest': 5}
Published data Successfully: %s {'temperature': 58, 'humidity': 51, 'soilmoisture': 47, 'guest': 10}
Published data Successfully: %s {'temperature': 111, 'humidity': 17, 'soilmoisture': 84, 'guest': 3}
Published data Successfully: %s {'temperature': -6, 'humidity': 43, 'soilmoisture': 96, 'guest': 10}
Published data Successfully: %s {'temperature': 6, 'humidity': 66, 'soilmoisture': 78, 'guest': 7}
Published data Successfully: %s {'temperature': 84, 'humidity': 17, 'soilmoisture': 18, 'guest': 9}
Published data Successfully: %s {'temperature': -11, 'humidity': 76, 'soilmoisture': 34, 'guest': 0}
Published data Successfully: %s {'temperature': 0, 'humidity': 75, 'soilmoisture': 36, 'guest': 6}
Published data Successfully: %s {'temperature': 61, 'humidity': 57, 'soilmoisture': 45, 'guest': 1}
Published data Successfully: %s {'temperature': -7, 'humidity': 25, 'soilmoisture': 74, 'guest': 8}
Published data Successfully: %s {'temperature': 82, 'humidity': 57, 'soilmoisture': 74, 'guest': 0}
Published data Successfully: %s {'temperature': -1, 'humidity': 58, 'soilmoisture': 95, 'guest': 10}
Published data Successfully: %s {'temperature': 18, 'humidity': 62, 'soilmoisture': 95, 'guest': 9}
Published data Successfully: %s {'temperature': 68, 'humidity': 0, 'soilmoisture': 77, 'guest': 5}
Published data Successfully: %s {'temperature': 15, 'humidity': 92, 'soilmoisture': 97, 'guest': 3}
Published data Successfully: %s {'temperature': 43, 'humidity': 56, 'soilmoisture': 86, 'guest': 6}
Published data Successfully: %s {'temperature': 37, 'humidity': 77, 'soilmoisture': 46, 'guest': 8}
Published data Successfully: %s {'temperature': 120, 'humidity': 29, 'soilmoisture': 40, 'guest': 0}
Message received from IBM IoT Platform: PUMP ON
The pump is switched OFF
Published data Successfully: %s {'temperature': 27, 'humidity': 38, 'soilmoisture': 3, 'guest': 3}
Published data Successfully: %s {'temperature': 97, 'humidity': 9, 'soilmoisture': 36, 'guest': 6}
Published data Successfully: %s {'temperature': 32, 'humidity': 50, 'soilmoisture': 69, 'guest': 1}
Published data Successfully: %s {'temperature': 3, 'humidity': 20, 'soilmoisture': 28, 'guest': 8}
  
```

Node-Red:





MIT APP:



This system also allows to control the amount of water delivered to plants when it is needed based on the climatic conditions and moisture level by continuous monitoring

CHAPTER 06

6.0 Conclusion

In conclusion, I am well satisfied with my training. I have learned many new concepts, acquired a number of new technical skills and improved another group of existing skills. What I liked most about my training is that it is very strongly related to new emerging technology. This refutes the common saying that very little of the materials taught in university engineering courses is used by engineers working in the labor market. This dependency (relationship) is clearest in engineering design. I may count the technical skills that I learned or improved at the training site, other than those gained at college. At last, I hereby conclude that I have successfully completed my industrial training in Smart Bridge and gained knowledge.