

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
RAMAPURAM CAMPUS
COLLEGE OF SCIENCE AND HUMANITIES



Department of Computer Applications

PRACTICAL RECORD

NAME :

REGISTERNO : **RA20312410200**

COURSE : **BCA**

SEMESTER/YEAR : **VI/ III**

SUBJECTCODE : **UCA20D09J**

SUBJECT NAME : **Internet of things**

April 2023



SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
RAMAPURAM CAMPUS
COLLEGE OF SCIENCE AND HUMANITIES



Department of Computer Applications

REGISTER NUMBER: RA20312410200

BONAFIDE CERTIFICATE

This is to certify that the bonafide work done by _____ in the subject **INTERNET OF THINGS (UCA20D09J)** at, SRM Institute of Science and Technology, Ramapuram Campus held on _____.

STAFF IN-CHARGE

HEAD OF THE DEPARTMENT

Submitted for the University Practical Examination held at S&H Computer Lab, Block V, College of Science and Humanities, Ramapuram Campus.

INTERNAL EXAMINER I

INTERNAL EXAMINER II

INDEX

EX NO	DATE	LIST OF EXPERIMENTS	PAGE NO	SIGNATURE
1	2.1.2023	Explain working of Raspberry Pi.		
2	10.1.2023	Controlling LED with raspberry pi board		
3	23.1.2023	Interfacing light sensor with raspberry pi board		
4	6.2.2023	Demonstrate a smart object API gateway service reference implementation in IoT toolkit.		
5	14.2.2023	Write and explain working of an HTTP- to-CoAP semantic mapping proxy in IoT toolkit		
6	21.2.2023	Describe gateway-as-a-service deployment in IoT toolkit		
7	6.3.2023	Application Frame work and Embedded Software Agents for IoT Toolkit		
8	14.3.2023	Explanation of Arduino with ESP8266		
9	20.3.2023	Home Automation level -1		
10	21.3.2023	Smart Irrigation System		
11	28.3.2023	Weather Reporting Systems		
12	3.4.2023	Air Pollution Monitor system		
13	3.4.2023	Remote surveillance System		

Ex.No:1

Name:

Date:

Reg No:

Explain working of Raspberry Pi.

Aim:

To explain Working of Raspberry Pi.

Explanation:

The Raspberry pi is a single computer board with credit card size, that can be used for many tasks that your computer does, like games, word processing, spreadsheets and also to play HD video. It was established by the Raspberry pi foundation from the UK. It has been ready for public consumption since 2012 with the idea of making a low-cost educational microcomputer for students and children. The main purpose of designing the raspberry pi board is, to encourage learning, experimentation and innovation for school level students. The raspberry pi board is a portable and low cost. Maximum of the raspberry pi computers is used in mobile phones.

Raspberry Pi Technology

The raspberry pi comes in two models, they are model A and model B. The main difference between model A and model B is USB port. Model a board will consume less power and that does not include an Ethernet port. But, the model B board includes an Ethernet port and designed in china. The raspberry pi comes with a set of open source technologies, i.e. communication and multimedia web technologies. In the year 2014, the foundation of the raspberry pi board launched the computer module that packages a model B raspberry pi board into module for use as a part of embedded systems, to encourage their use.

Raspberry Pi Hardware Specifications

The raspberry pi board comprises a program memory (RAM), processor and graphics chip, CPU, GPU, Ethernet port, GPIO pins, Xbee socket, UART, power source connector. And various

interfaces for other external devices. It also requires mass storage, for that we use an SD flash memory card. So that raspberry pi board will boot from this SD card similarly as a PC boots up into windows from its hard disk.

Essential hardware specifications of raspberry pi board mainly include SD card containing Linux OS, US keyboard, monitor, power supply and video cable. Optional hardware specifications include USB mouse, powered USB hub, case, internet connection, the Model A or B: USB WiFi adaptor is used and internet connection to Model B is LAN cable.

Memory

The raspberry pi model aboard is designed with 256MB of SDRAM and model B is designed with 512MB. Raspberry pi is a small size PC compare with other PCs. The normal PCs RAM memory is available in gigabytes. But in raspberry pi board, the RAM memory is available more than 256MB or 512MB

CPU (Central Processing Unit)

The Central processing unit is the brain of the raspberry pi board and that is responsible for carrying out the instructions of the computer through logical and mathematical operations. The raspberry pi uses ARM11 series processor, which has joined the ranks of the Samsung galaxy phone.

GPU (Graphics Processing Unit)

The GPU is a specialized chip in the raspberry pi board and that is designed to speed up the operation of image calculations. This board designed with a Broadcom video core IV and it supports OpenGL

Ethernet Port

The Ethernet port of the raspberry pi is the main gateway for communicating with additional devices. The raspberry pi Ethernet port is used to plug your home router to access the internet.

GPIO Pins

The general purpose input & output pins are used in the raspberry pi to associate with the other electronic boards. These pins can accept input & output commands based on programming raspberry pi. The raspberry pi affords digital GPIO pins. These pins are used to connect other electronic components. For example, you can connect it to the temperature

sensor to transmit digital data.

XBee Socket

The XBee socket is used in raspberry pi board for the wireless communication purpose.

Power Source Connector

The power source cable is a small switch, which is placed on side of the shield. The main purpose of the power source connector is to enable an external powersource.

UART

The Universal Asynchronous Receiver/ Transmitter is a serial input & output port. That can be used to transfer the serial data in the form of text and it is useful for converting the debugging code.

Display

The connection options of the raspberry pi board are two types such as HDMI and Composite. Many LCD and HD TV monitors can be attached using an HDMI male cable and with a low-cost adaptor. The versions of HDMI are 1.3 and 1.4 are supported and 1.4 version cable is recommended. The O/Ps of the Raspberry Pi audio and video through HDMI, but does not support HDMI I/p. Older TVs can be connected using composite video. When using a composite video connection, audio is available from the 3.5mm jack socket and can be sent to your TV. To send audio to your TV, you need a cable which adjusts from 3.5mm to double RCA connectors.

Model of a Raspberry Pi Board

The Raspberry Pi board is a Broadcom (BCM2835) SOC (system on chip) board. It comes equipped with an ARM1176JZF-S core CPU, 256 MB of SDRAM and 700 MHz. The raspberry pi USB 2.0 ports use only external data connectivity options. The board draws its power from a micro USB adapter, with min range of

2. Watts (500 MA). The graphics, specialized chip is designed to speed up the operation of image calculations. This is in built with Broadcom video core IV cable that is useful if you want to run a game and video through your raspberry pi.



Features of Raspberry PI Model A

- The Model A raspberry pi features mainly includes
- 256 MB SDRAM memory
- Single 2.0 USB connector
- Dual Core Video Core IV Multimedia coprocessor
- HDMI (rev 1.3 & 1.4) Composite RCA (PAL and NTSC) Video Out
- 3.5 MM Jack, HDMI, Audio Out
- SD, MMC, SDIO Card slot on board storage
- Linux Operating system
- Broadcom BCM2835 SoC full HD multimedia processor
- 8.6cm*5.4cm*1.5cm dimensions

Result:

Thus the program has been verified and completed successfully.

Ex.No:2

Name:

Date:

Reg No:

Controlling LED with raspberry pi board

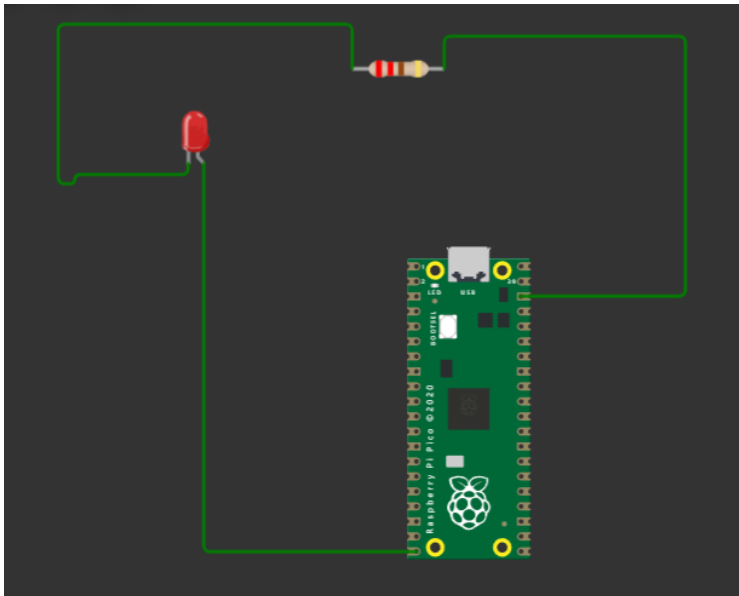
Aim:

To control LED with raspberry pi board

Program:

```
Main.py
from machine import Pin
import utime
led = Pin(15,Pin.OUT)
while True:
    led.toggle()
    utime.sleep(1.5)
```

Output:



Result:

Thus the program has been verified and completed successfully.

Ex.No:3

Name:

Date:

Reg No:

Interfacing light sensor with raspberry pi board

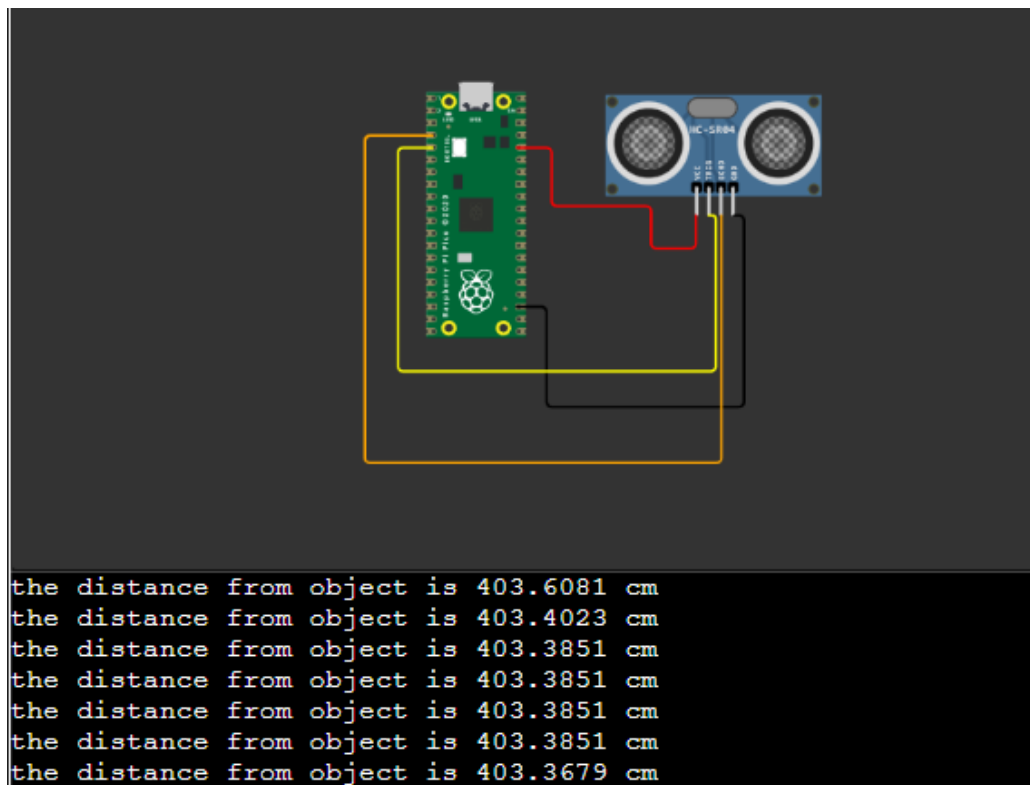
Aim:

To interface light sensor with raspberry pi board

Program:

```
Main.py
from machine import Pin
import utime
trigger = Pin(3, Pin.OUT)
echo = Pin(2, Pin.IN)
def ultra():
    trigger.low()
    utime.sleep_us(2)
    trigger.high()
    utime.sleep_us(5)
    trigger.low()
    while echo.value()==0:
        signaloff = utime.ticks_us()
    while echo.value() ==1:
        signalon = utime.ticks_us()
    timepassed = signalon - signaloff
    distance = (timepassed * 0.0343) / 2
    print("the distance from object is", distance,"cm" )
while True:
    ultra()
    utime.sleep(1)
```

Output:



Result:

Thus the program has been verified and completed successfully.

Ex.No:4

Name:

Date:

Reg No:

Demonstrate a smart object API gateway service reference implementation in IoT toolkit.

Aim:

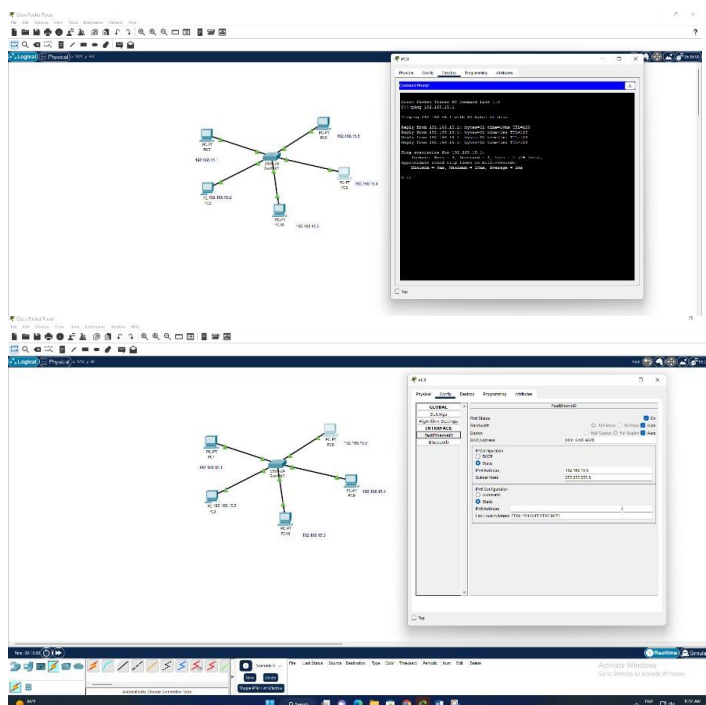
To Demonstrate a smart object API gateway service reference implementation in IoT toolkit.

Explanation:

The Smart Object API is a Semantic Web Linked Data application for the Internet of Things (IoT). It consists of a URI-Object encapsulation of semantic and real-time data properties associated with features of interest. The Smart Object architecture roughly conforms to the Virtual Entity, the Information Model, and the Channel Model set out in the IoT-A Architecture Reference Model (IoT-A ARM). Supports direct interaction between smart sensors, smart gateways, cloud/internet services, and user devices. Interaction uses standard web protocols and formats and is semantically a superset of the CoAP protocol.

Service framework is to include object creation from semantic metadata, semantic database, discovery, and linkage, API capability keys, and threaded server.

Output:



Result:

Thus the program has been verified and completed successfully.

Ex.No:5

Name:

Date:

Reg No:

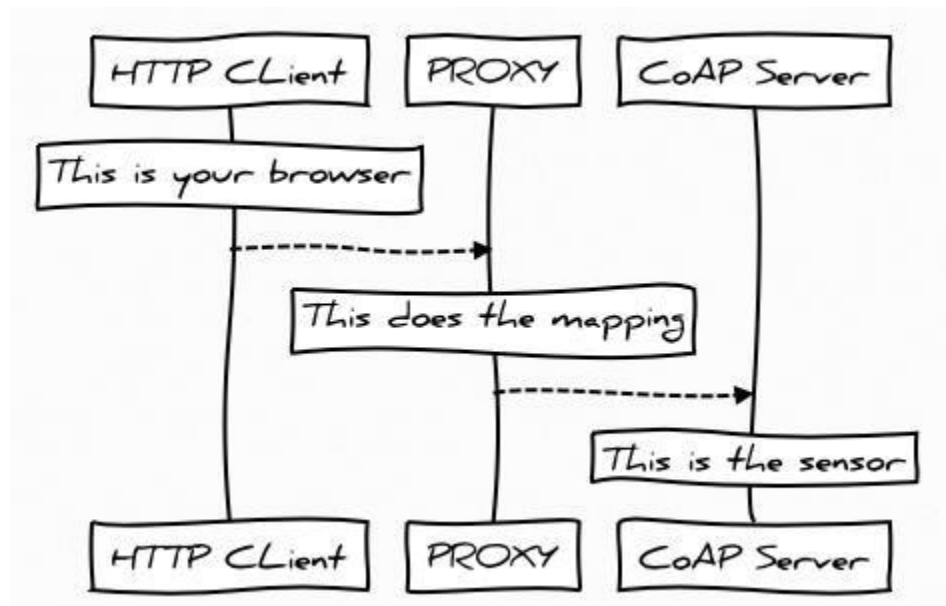
Write and explain working of an HTTP- to-CoAP semantic mapping proxy in IoT toolkit

Aim:

To Write and explain working of an HTTP- to-CoAP semantic mapping proxy in IoT toolkit

Explanation:

The point of the draft (soon RFC) is to describe how to do HTTP-to-CoAP Mapping to allow HTTP clients to access a CoAP Server via a proxy. This works under the assumption that, despite the similarities between CoAP and HTTP, not all browsers will implement support for it and that some legacy devices will need proxying. Another assumption is that users will like to use their smartphones with their home sensors. The set up would look like this:

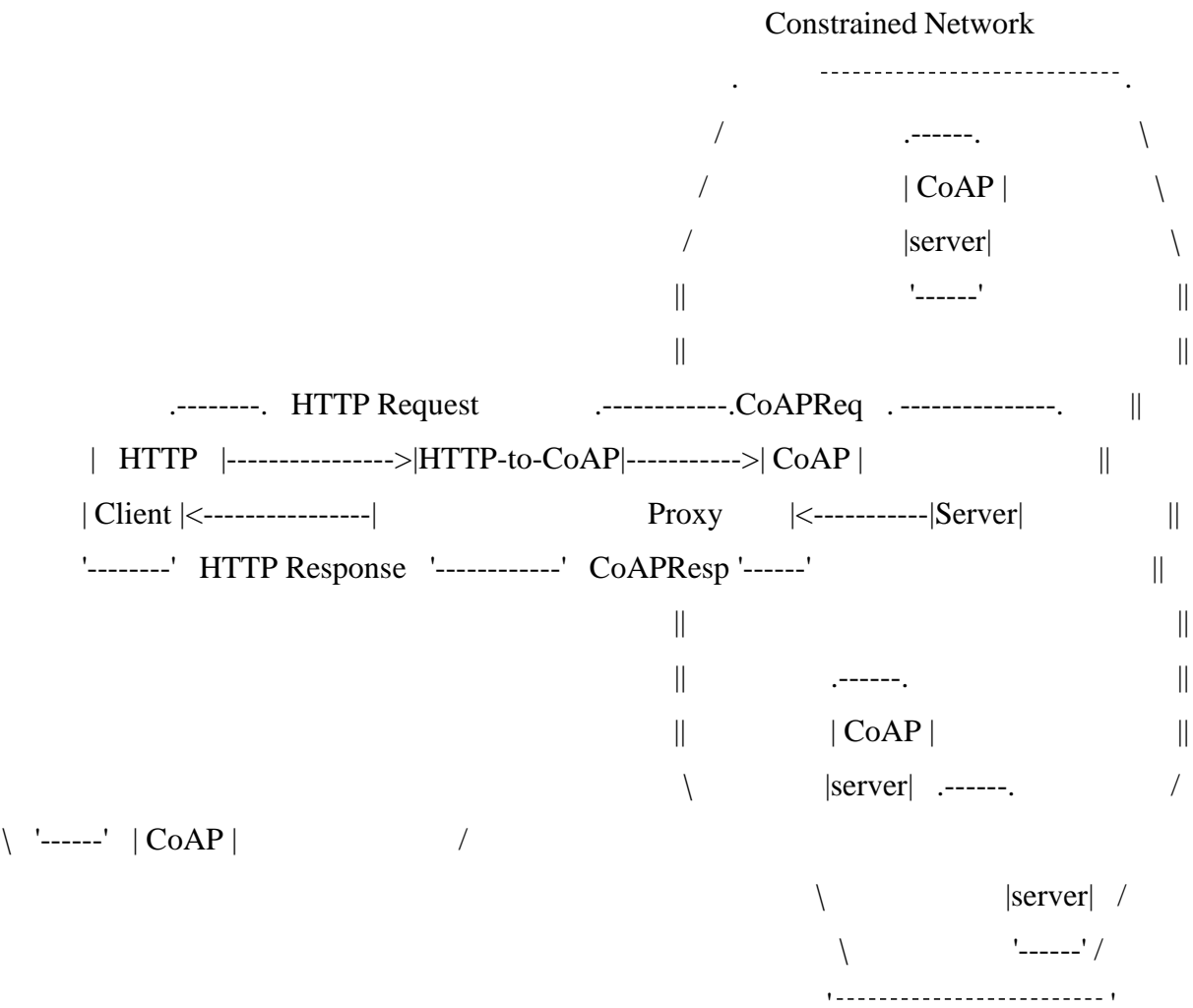


HTTP-to_CoAP Mapping Scenario with world-class graphics

HTTP-to-CoAP Proxy

A HC proxy is accessed by an HTTP client which wants to access a resource on a CoAP server. The HC proxy handles the HTTP request by mapping it to the equivalent CoAP request, which is then forwarded to the appropriate CoAP server. The received CoAP response is then mapped to an appropriate HTTP response and finally sent back to the originating HTTP client.

See Figure 1 for an example deployment scenario. Here a HC proxy is located at the boundary of the Constrained Network domain, to avoid sending any HTTP traffic into the Constrained Network and to avoid any (unsecured) CoAP multicast traffic outside the Constrained Network. A DNS server (not shown) is used by the HTTP Client to resolve the IP address of the HC proxy and optionally also used by the HC proxy to resolve IP addresses of CoAP servers.



Normative requirements on the translation of HTTP requests to CoAP requests and of the CoAP responses back to HTTP responses are defined in the basic mapping of request methods and simple response code mapping between HTTP and CoAP, and leaves many details of the cross- protocol HC proxy for future definition. This document provides additional guidelines and more details for the implementation of a HC Proxy, which should be followed in addition to the normative requirements. Note that the guidelines apply to all forms of an HC proxy (i.e. Reverse, Forward, Intercepting) unless explicitly otherwise noted.

Result:

Thus the program has been verified and completed successfully.

Ex.No:6

Name:

Date:

Reg No:

Describe gateway-as-a-service deployment in IoT toolkit

Aim:

To Describe gateway-as-a-service deployment in IoT toolkit.

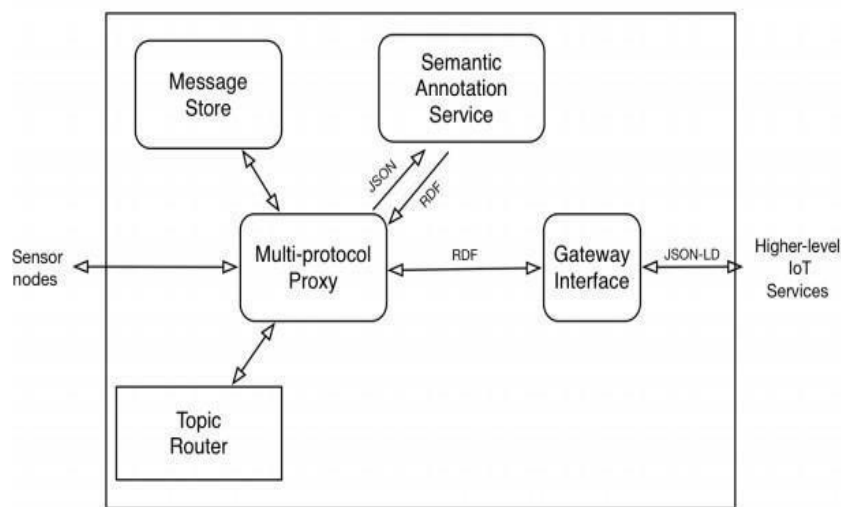
Explanation:

The Internet of Things (IoT) is set to occupy a substantial component of future Internet. The IoT connects sensors and devices that record physical observations to applications and services of the Internet. As a successor to technologies such as RFID and Wireless Sensor Networks (WSN), the IoT has stumbled into vertical silos of proprietary systems, providing little or no interoperability with similar systems. As the IoT represents future state of the Internet, an intelligent and scalable architecture is required to provide connectivity between these silos, enabling discovery of physical sensors and interpretation of messages between things. This paper proposes a gateway and Semantic Web enabled IoT architecture to provide interoperability between systems using established communication and data standards. The Semantic Gateway as Service (SGS) allows translation between messaging protocols such as XMPP, CoAP and MQTT via a multi-protocol proxy architecture. Utilization of broadly accepted specifications such as W3C's Semantic Sensor Network (SSN) ontology for semantic annotations of sensor data provide semantic interoperability between messages and support semantic reasoning to obtain higher-level actionable knowledge from low-level sensor data.

While developing IoT solutions we come across common tasks of connecting “things” to the cloud. For devices that are not connected to the cloud directly, gateways are used. Such gateway can be a system on chip (SoC) device: cable modem, set top box, home or industrial automation gateway, smart phone, home entertainment system, laptop or PC. All these gateway should have some sort of interface (BLE/ZigBee/etc radios, adapter, digital or analog inputs) that can connect to the actual device: lamp, controller, sensor, appliance.

Semantic Gateway as Service (SGS)

The heart of the semantic IoT architecture is the SGS, which bridges low level raw sensor information with knowledge centric application services by facilitating interoperability at messaging protocol and data modeling level. The description below is complemented by Open Source code available at <https://github.com/chheplo/node-sgs> which is further being enhanced and evaluated in the context of CityPulse, a large multi-institutional EU FP7 supported project along with an effort for additional community engagement and development.

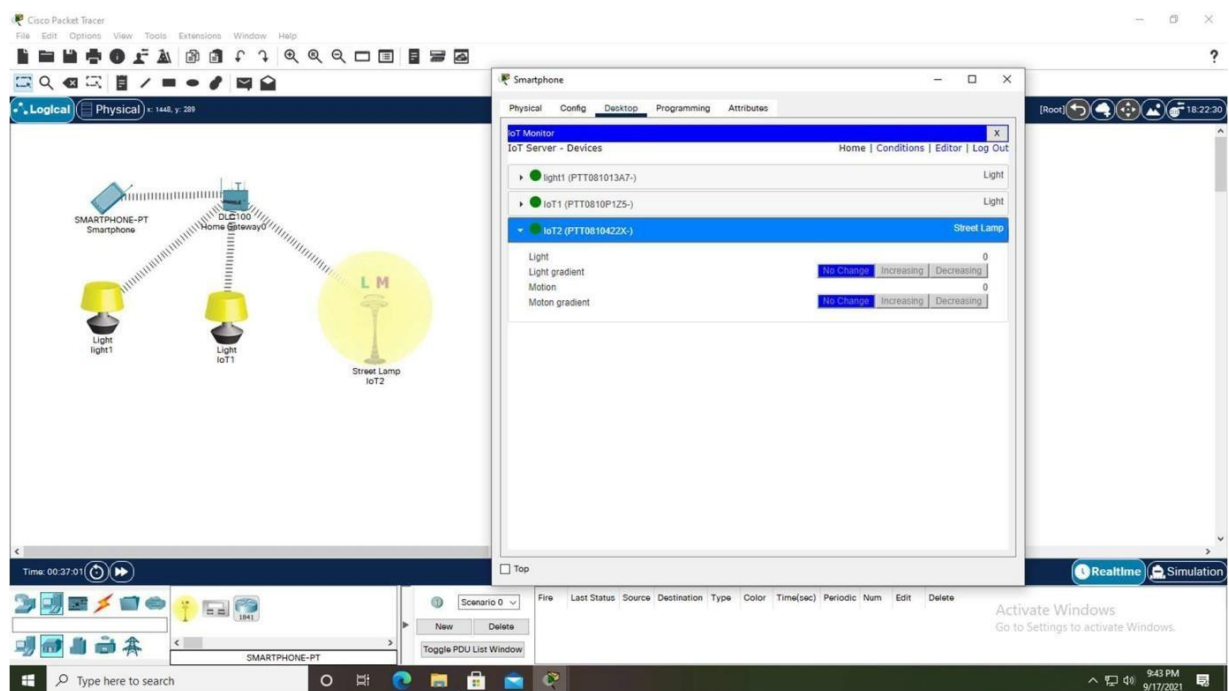


The SGS has three core components as described in Figure:

- (1) multi-protocol proxy,
- (2) semantic annotation service,
- (3) Gateway service interface.

The SGS also has components for required capabilities such as message store and topics router, which assist multi-protocol proxy and gateway service interface in translation between messaging protocol. At a high level, SGS architecture connects external sink nodes to the gateway component using primitive client agents, which support MQTT, XMPP or CoAP. In contrast, the gateway service interface connects cloudservices or other SGSs via REST or pubsub protocol. Before raw sensor data is forwarded from proxy to gateway interface, it is annotated using SSN and domain specific ontologies. Although the semantically annotated data is in RDF format at the multi-protocol proxy, the gateway interface converts the data into JSON, specifically linked data (JSON-LD) format to support RESTful protocols.

Output:



Result:

Thus the program has been verified and completed successfully.

Ex.No:7

Name:

Date:

Reg No:

Application Framework and Embedded Software Agents for IoT Toolkit

Aim:

To Explain application framework and Embedded software agent for IOT Toolkit.

IoT Frameworks

For an IoT framework to be reliable and dependable, some minimal set of measures should be satisfied to achieve integration and interoperability in IoT. These frameworks span across the IoT research communities ranging from academic research to organizational research which focus on integrating things in IoT. Since IoT paradigm itself is still in evolving state, we propose a set of minimal measures to be satisfied by IoT frameworks for integration. These are:

- **Contract decoupling:** An IoT system contains heterogeneous devices with disparate communication protocols. An integration framework should be competent enough to efficiently handle contract decoupling. Contract decoupling is the ability of service consumers and service producers to independently evolve without terminating the contract between them [19]. For example, a service might be in a JSON format and the service consumer needs an input in XML. The framework should provide support to transform the message to the format that fulfils the contract between them.
- **Scalability:** Given the evolving nature of IoT and the predictions and calculations by [3] and [2], an efficient integration framework should be scalable and evolvable enough to support the billions of things soon to be connected to the internet.
- **Ease of testing:** An integration framework should support ease of testing and debugging. It should provide support for debugging defects and failures, integration testing, component testing, system testing, compatibility testing, installation test, functional and non-functional testing, performance testing and security testing.
- **Ease of development:** An IoT integration framework should provide a means of easy development for developers. The framework should exclude all complexities and provide proper documentation for non-developers and developers with basic programming knowledge to easily understand the internals of the framework.

- **Fault tolerance:** An IoT system has to be dependable and resilient. An intelligent integration framework should effectively handle faults as IoT devices can eventually toggle between offline and online states. The framework should provide self-healing mechanisms for transient faults (network faults, node level faults, etc.), unauthorised access error, server crash failure, omission failure (when the server does not receive incoming requests from client), timing fault, etc.
- **Lightweight implementation:** Integration frameworks should have a lightweight overhead both in its development and deployment stage. It should be lightweight and easy to install, uninstall, activate, deactivate, update, versioning and adaptable.
- **Service coordination:** Service coordination is the orchestration and choreography of services. Service orchestration is the coordination of multiple services by a mediator acting as a centralised component. Service choreography on the other hand, is the chaining of services together to execute a particular transaction. Integration frameworks should support at least either or both to achieve reliability.
- **Inter domain operability:** The framework should further be extensible to support inter domain communication. For example, in a smart car domain, an integration framework should also provide support for communication and interaction with traffic lights, road closure, etc. belonging to a smart city domain.

Application Framework for IOT

IoT applications have been developed and deployed in several domains such as transportation and logistics, healthcare, retail and supply chain, industry and environment [5, 6]. Despite their pervasiveness, developing IoT applications remains challenging and time-consuming. This is because it involves dealing with several related issues, such as lack of proper identification of roles of various stakeholders, as well as the lack of appropriate frameworks to address the large-scale and heterogeneity in IoT systems [7]. Another major challenge is the difficulty in achieving effective programming abstractions at different technology layers, ranging from device software to middleware services and end-user applications [8]. These difficulties increase the development time, resources and delay the deployment of the IoT applications. The complexity of IoT applications implies that it is inappropriate to develop one in an ad hoc manner and as a result, a framework is required. An IoT application framework can simplify the difficult process of coping with heterogeneous devices and

software components, overcoming the complexities of distributed system technologies, handling a high volume of data, designing an architecture for the application, implementing it in a program, writing specific code to validate the application, and finally deploying it. A number of researchers have proposed several IoT application frameworks with each having its own strength and weakness. A study of the various application development frameworks for IoT is an important step for designing and developing a high-quality IoT application.

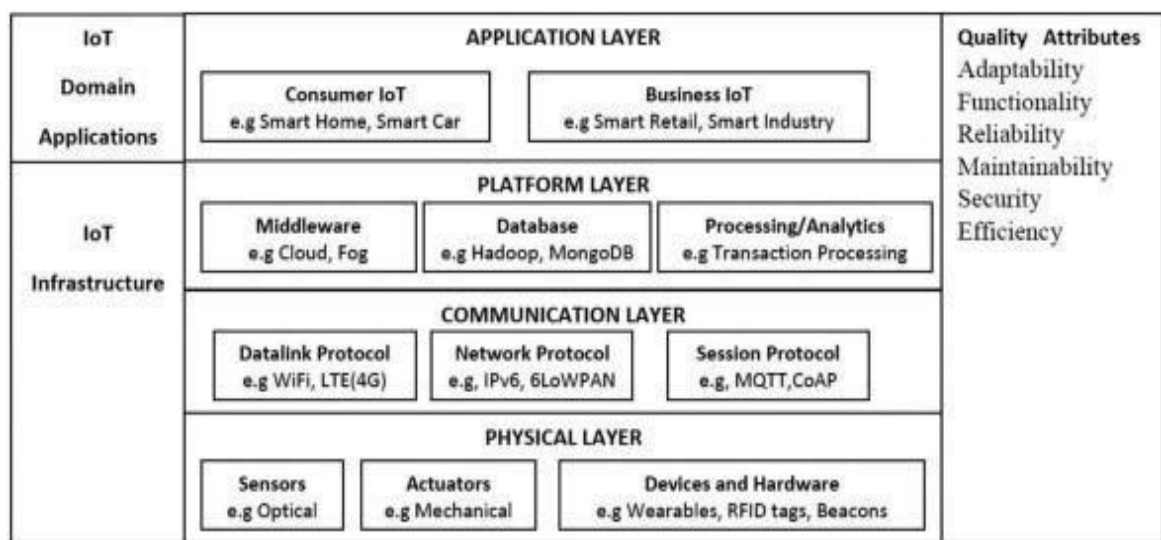


Fig. IoT technology Architecture

Agent Toolkits

Currently there are a variety of toolkits available on the market ranging from general agent development platforms, like AgentBuilder developed by Reticular Systems, to highly specialized tools, like Excalibur developed by the Technical University of Berlin which allows for the creation of autonomous agents in a complex computer-game environment. The AgentBuilder web site² identifies numerous agent toolkits available on the market.

What are Agent Toolkits?

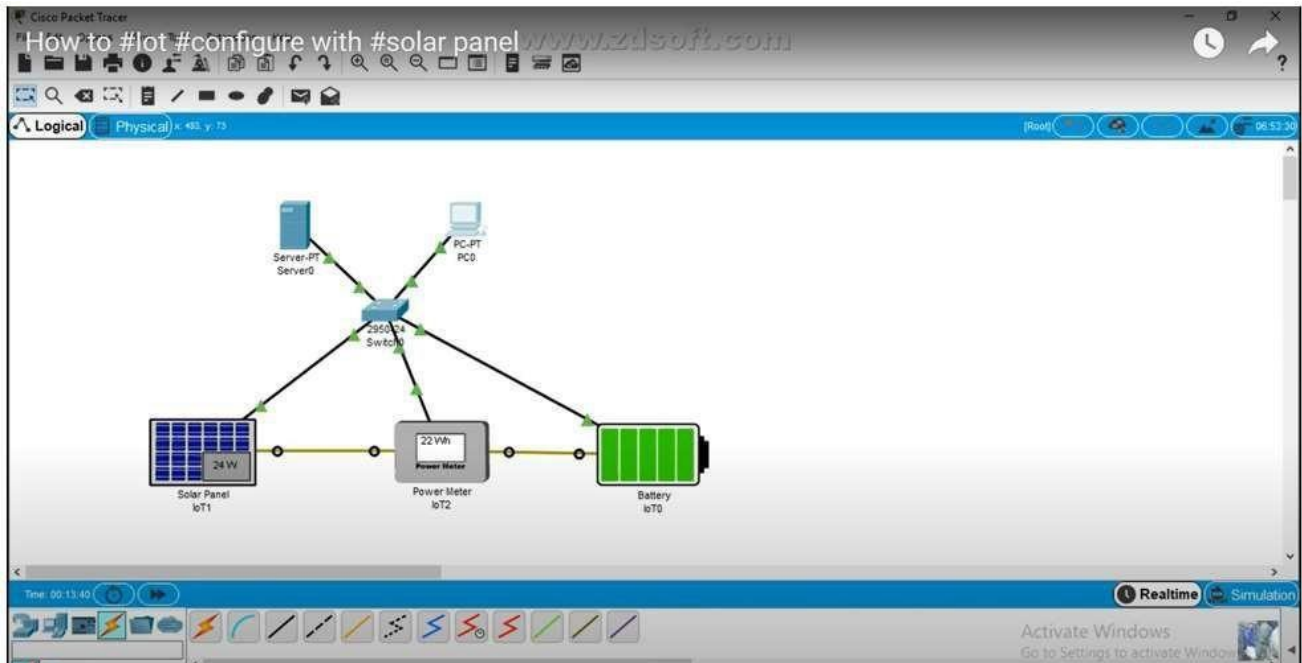
There is no universal definition of agent toolkits. Each vendor uses its own explanation of the term. For example, Reticular Systems states that its AgentBuilder toolkit application “is an integrated tool suite for constructing intelligent software agents”. Authors of the Java Agent Development Environment (JADE) define their toolkit as “a software framework to make easy the development of agent application for interoperable multi-agent systems”

An agent toolkit is defined as any software package, application or development environment that provides agent builders with a sufficient level of abstraction to allow them to implement intelligent agents with desired attributes, features and rules. Some toolkits may offer only a platform for agent development, whereas others may provide features for visual programming. Agent toolkits may also provide an environment for running, monitoring, analyzing and testing agents, which is very important for both researchers and students learning about agent technologies. For example, in case of multi-agent systems, an agent development environment provides a context for agent interaction and sets of governing rules.

Why are Agent Toolkits needed?

The reasons why agent developers use agent toolkits is similar to those reasons why software developers who deal with object-oriented programming (OOP) prefer to use special development environments like Java VisualAge or Microsoft Visual Basic. First, they provide a certain level of abstraction in which programmers can develop their objects. Second, they incorporate some features of visual programming, which saves much time and makes development easier, more attractive and enjoyable. Third, they offer run-time testing and debugging environments. Finally, they allow programmers to reuse classes (definitions of objects) created by other programmers. Unfortunately, existing OOP development platforms and compilers do not support all facets of agent development. For example, they do not address the implementation of agent features, agent interaction rules, communication language, and a common knowledge base. This is why a new suite of agent toolkits has appeared on the market in the last few years: to create a development environment that fully supports agent creation, testing, and reuse.

Output:



The interface shows the configuration of the Solar Panel (IoT1) using a Python script. The script is named "LED (Python) - main.py" and is located in the "Programming" tab. The script includes the following code:

```
LED (Python) - main.py
Open New Delete Rename Import
main.py
pyjs.py
3 from gpio import *
4 from time import *
5 from pyjs import *
6
7 MAX_LIGHT_PERCEPT = 1
8 VOLUME_AT_RATE = 100000
9 value = 0
10
11 def setup():
12     setComponentOpacity("black", 1)
13     add_event_detect(0, isr)
14     isr()
15
16 def isr():
17     global value
18     value = 0
```

The script is running, and the output shows "Starting LED (JavaScript)...". The interface also shows the "Advanced" tab with a "Run" button and a "Clear Outputs" button. The status bar at the bottom right shows "Realtime" and "Simulation" modes.

Result:

Thus the program has been verified and completed successfully.

Ex.No:8

Name:

Date:

Reg No:

Explanation of Arduino with ESP8266

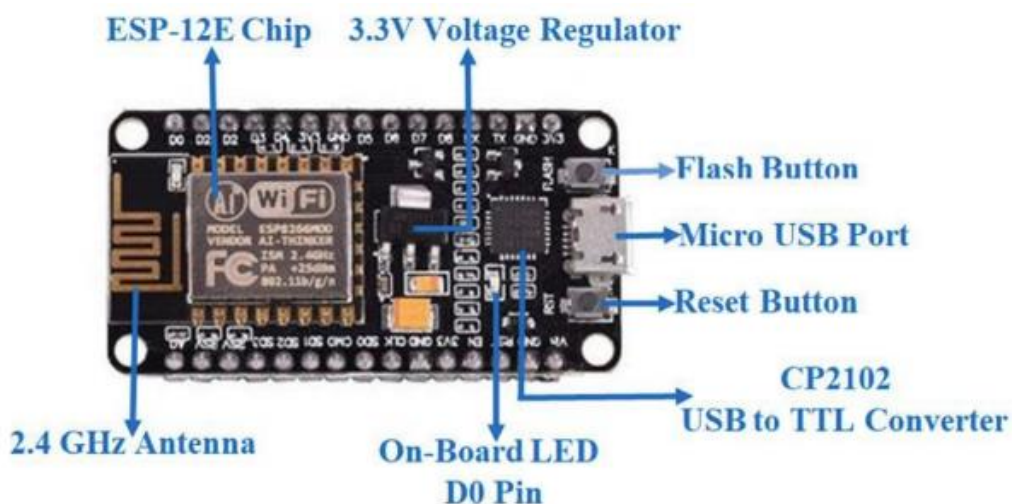
Aim:

To explain Arduino with ESP8266.

Explanation:

ESP8266 - Node MCU This is an open source development board with a firmware that runs on ESP8266 module. The ESP-8266 module is a wireless programmable microcontroller board. The ESP8266 WiFi board is a SOC with integrated TCP/IP protocol stack that can give any secondary microcontroller access to a WiFi network.

The ESP8266 board is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor and therefore this is more suitable to be used as a sensing node that is capable to sense the data from various wirelessly connected IoT sensor nodes and send data to the central server like



	Arduino	Raspberry Pi	ESP8266 Node MCU
Developer	Arduino	Raspberry Pi Foundation	ESP8266 open source community
Type	Single board microcontroller	Mini computer	Single board microcontroller
Operating System	None	Linux	XTOS
CPU	Atmel, ARM, Intel	ARM Cortex	LXT106
Clock Speed	16 MHz	1.2GHz	26 MHz – 52 MHz
Memory	32KB	1-4GB	Upto 128MB
Storage	1KB	MicroSDHC Slot	4MB
Power	USB, Battery, Power Supply	USB, Power Supply	USB
Operating Voltage	5V	5V	3.3V
I/O Connectivity	SPI I2C UART GPIO	SPI DSI UART SDIOCSI GPIO	UART, GPIO

Table 1: A comparison of development boards

Result:

Thus the program has been verified and completed successfully.

Ex.No:9

Name:

Date:

Reg No:

Home Automation level -1

Aim:

To Demonstrate the Home Automation using Cisco Packet Tracer.

Procedure:

Step1: Open Cisco packet tracer and select the wireless devices in that select home gateway icon.

Step2: Select the home icon from the end devices.

Step3: From the home icon select fan, light and window.

Step4: From the device option select the smart phone to enable the connectivity with the smartphone configure the following step:

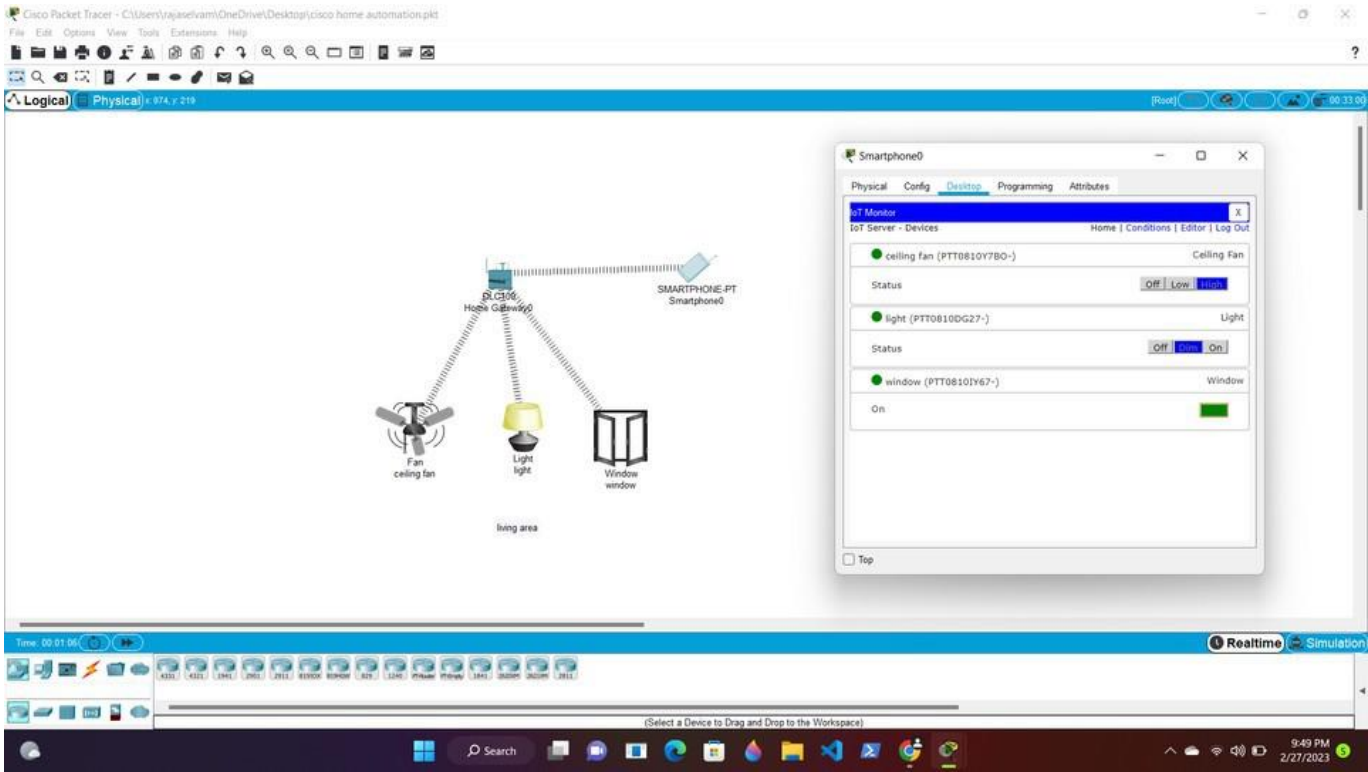
- i.) From the menu, select config option, in that go to wireless option, copy the SSID of the home gateway and paste it in the smartphone SSID.

Step5: Enable the Home gateway in IOT server on config from all the connected end devices fan, light and window.

Step 6: select smartphone, choose desktop in that IOT monitor, login with IP address of gateway, Username: admin and the Password: admin.

Step 7: Now you can monitor the status of the end devices and change the status of the connected devices.

Output:



Result:

Thus the program has been verified and completed successfully.

Ex.No:10

Name:

Date:

Reg No:

Smart Irrigation System

Aim:

To Demonstrate the Smart Irrigation System using Cisco Packet Tracer.

Procedure:

Step1: select home gateway from the option network device -select wireless devices in that select home gateway.

Step2: From the icon end device select home icon in that select Humiture monitor, lawn sprinkler and water level monitor.

Step3: Automatically these device will be connected with Home gateway .

Step4: copy the SSID of the home gateway and paste it in smart phone, now smart phone has been connected with home gateway.

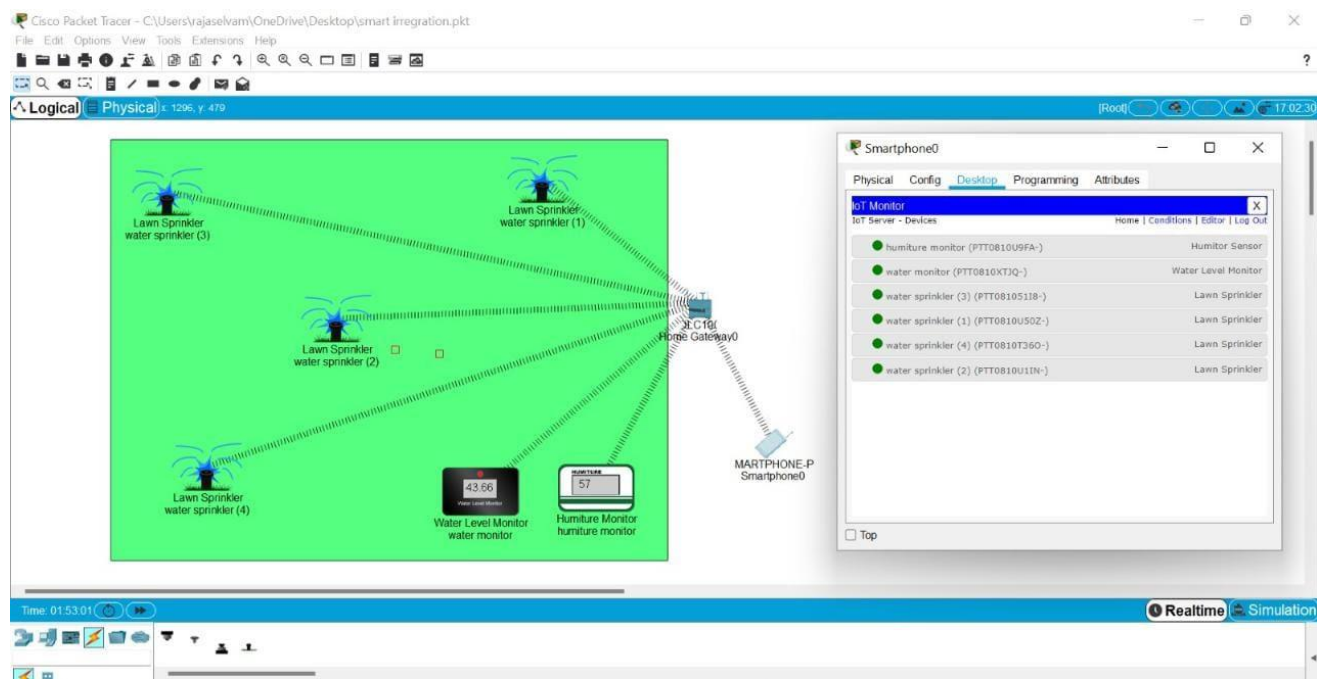
Step5: Enable all the device such as lawn sprinkler, Humiture and water level monitor with Iot server, select home gateway

Step6: Login using Iot server ip address, select condition option in that, click add in the turn off sprinkler by fixing the threshold value of greater than 3.0, then select lawn sprinkler and set the status false.

Step7: Now set the condition lawn sprinkler enable on by fixing the condition water level less than Or equal 5.0 and set lawn sprinkler status as true.

Step8: Now go to home option and check the status of the connection device.

Output:



Result:

Thus the program has been verified and completed successfully.

Ex.No:11

Name:

Date:

Reg No:

Weather Reporting Systems

Aim:

To Demonstrate the Weather Reporting Systems using Cisco Packet Tracer.

Procedure:

Step-1: open the cisco packet tracer. Select home gateway from network device.

Step-2: Select tablet, pc from end device and temperature monitor from home icon.

Step-3: Select SSID of home gateway and paste it in SSID tablet pc.

Step-4: Select mcu-pt and connect heater and air cooler and temperature sensor from home icon and connect to mcu.

Step-5: Generate the code in the programming option of node mcu.

Step-6: The heater and air cooler is disable at the room temperature 20 to 30 degree. when temperature goes below 20 degree, heater is switched on automatically. when temperature rises above 30 degree, ac is on automatically.

CODE:

```
var pata=A0;var
pdac=1; var
pdhe=2; var
pdd=3;

function setup()
{
    pinMode(pdac,OUTPUT);
    pinMode(pdhe,OUTPUT);
}

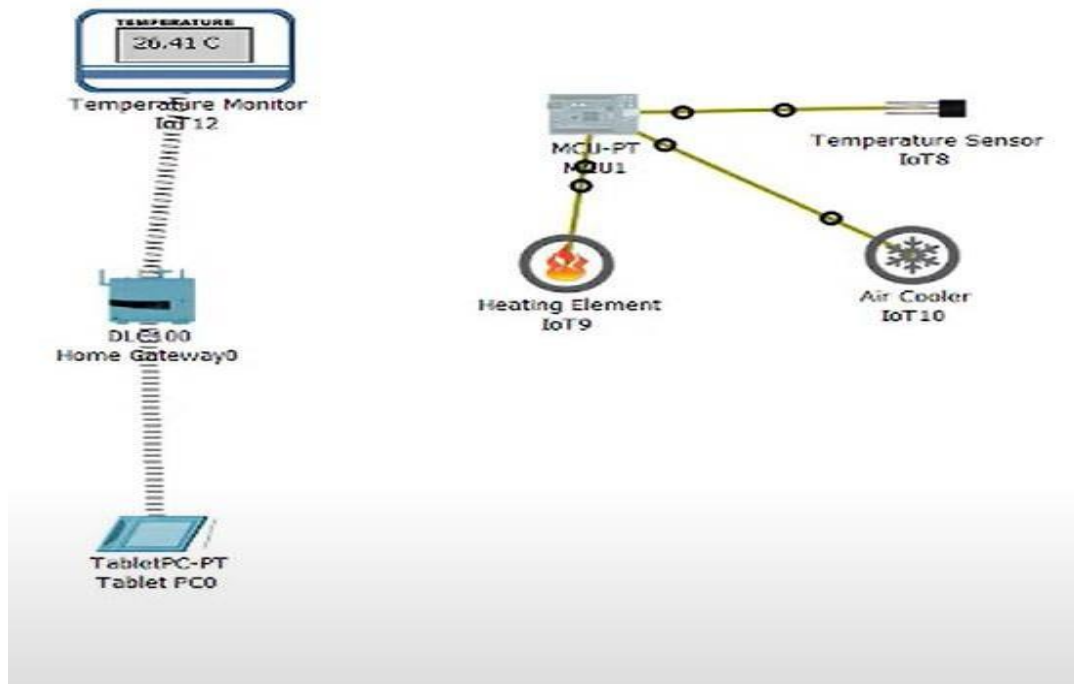
function loop()
```

```
{  
    var temperature=Math.round([(analogRead(A0)-0)*[100- -100])/(1023 -0)]  
+ -100);
```

```
Serial.println("Temperature:"+temperature); if(temperature>35)
```

```
{  
    analogWrite(pdac,1);  
    analogWrite(pdac,HIGH);  
    analogWrite(pdhe,0);  
}  
else  
    if(temperature<20){ analogW  
rite(pdhe,1);  
    analogWrite(pdhe,HIGH);  
    analogWrite(pdac,0);  
}  
else{ analogWrite(pdac,0)  
    ; analogWrite(pdhe,0);  
}  
}
```

Output:



Result:

Thus the program has been verified and completed successfully.

Ex.No:12

Name:

Date:

Reg No:

Air Pollution Monitor system

Aim:

To Demonstrate the Air Pollution Monitor system.

Procedure:

Step1: open the cisco packet tracer. Select home gateway from network device.

Step-2: Select smartphone, window, siren from end devices home icon.

Step3: Select old car from cities of end devices.

Step-4: Select SSID of home gateway and paste it in SSID of smartphone.

Step-5: Enable the devices connected to home gateway.

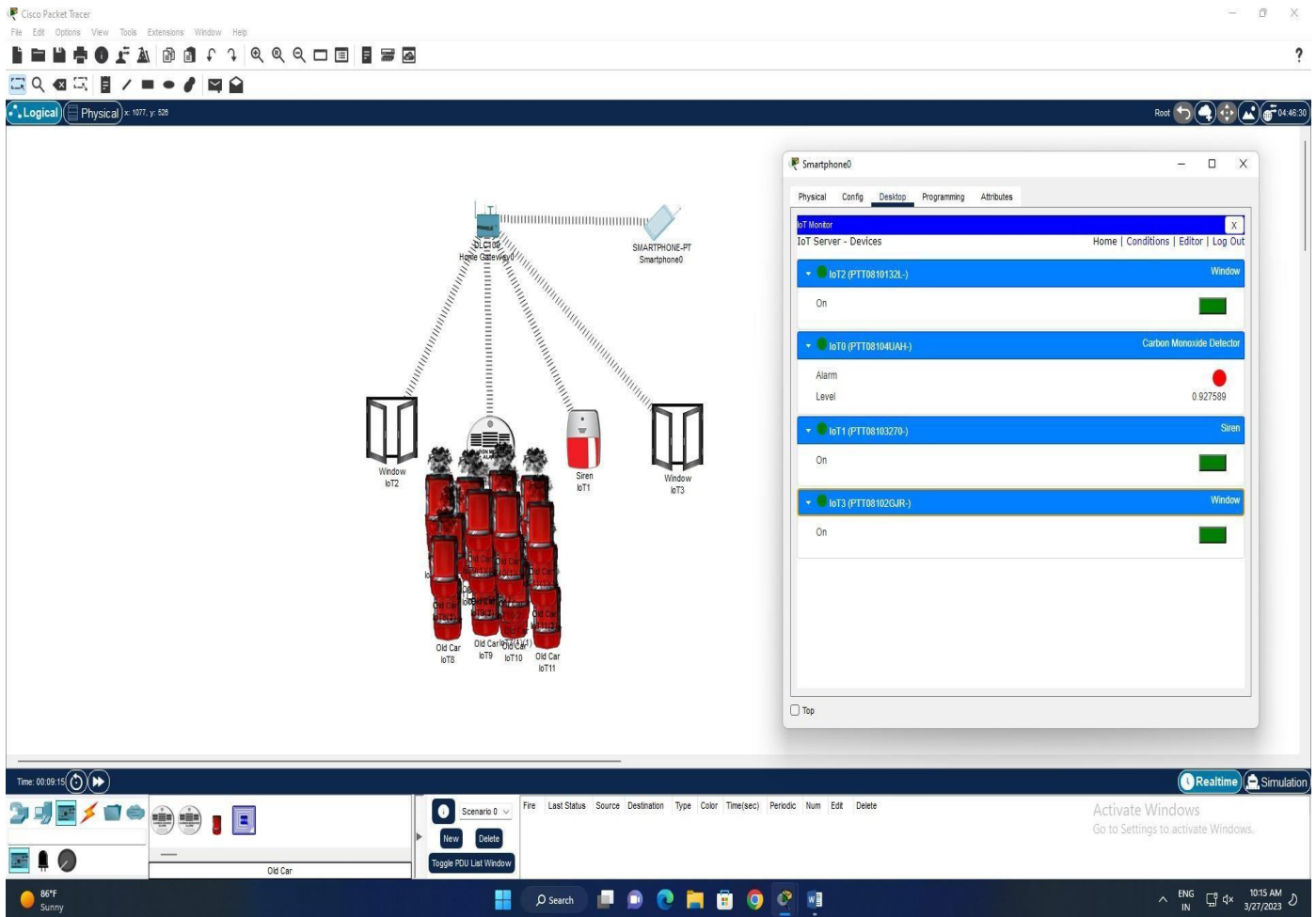
Step-6: Select smartphone login with IP address, user name and password.

Step-7: Select the smartphone, go to programming option set the conditions for Window and siren.

Conditions:

IoT Monitor					
IoT Server - Device Conditions					
Home Conditions Editor Log Out					
Actions		Enabled	Name	Condition	Actions
Edit	Remove	Yes	on window	IoT0 Level > 0.7	Set IoT2 On to true Set IoT3 On to true
Edit	Remove	Yes	on siren	IoT0 Level > 0.9	Set IoT1 On to true
Edit	Remove	Yes	no risk	IoT0 Level < 0.7	Set IoT2 On to false Set IoT3 On to false Set IoT1 On to false
Add					

Output:



Result:

Thus the program has been verified and completed successfully.

Ex.No:13

Name:

Date:

Reg No:

Remote surveillance System

Aim:

To demonstrate with Remote surveillance system in Cisco packet tracer.

Procedure:

Step-1: open the cisco packet tracer. Select home gateway from network device.

Step-2: Select smartphone, webcam, motion detector siren from end devices home icon.

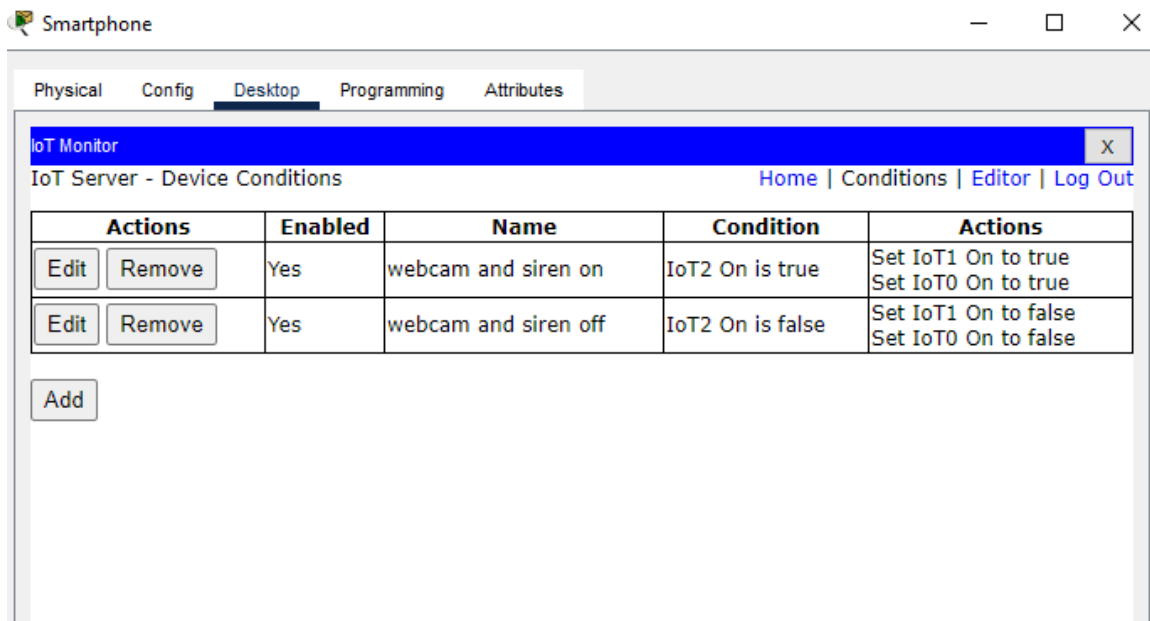
Step-3: Select SSID of home gateway and paste it in SSID of smartphone.

Step-5: Enable the devices connected to home gateway.

Step-6: Select smartphone login with IP address, user name and password.

Step-7: Select the smartphone, go to programming option set the conditions for webcam and siren.

Conditions:

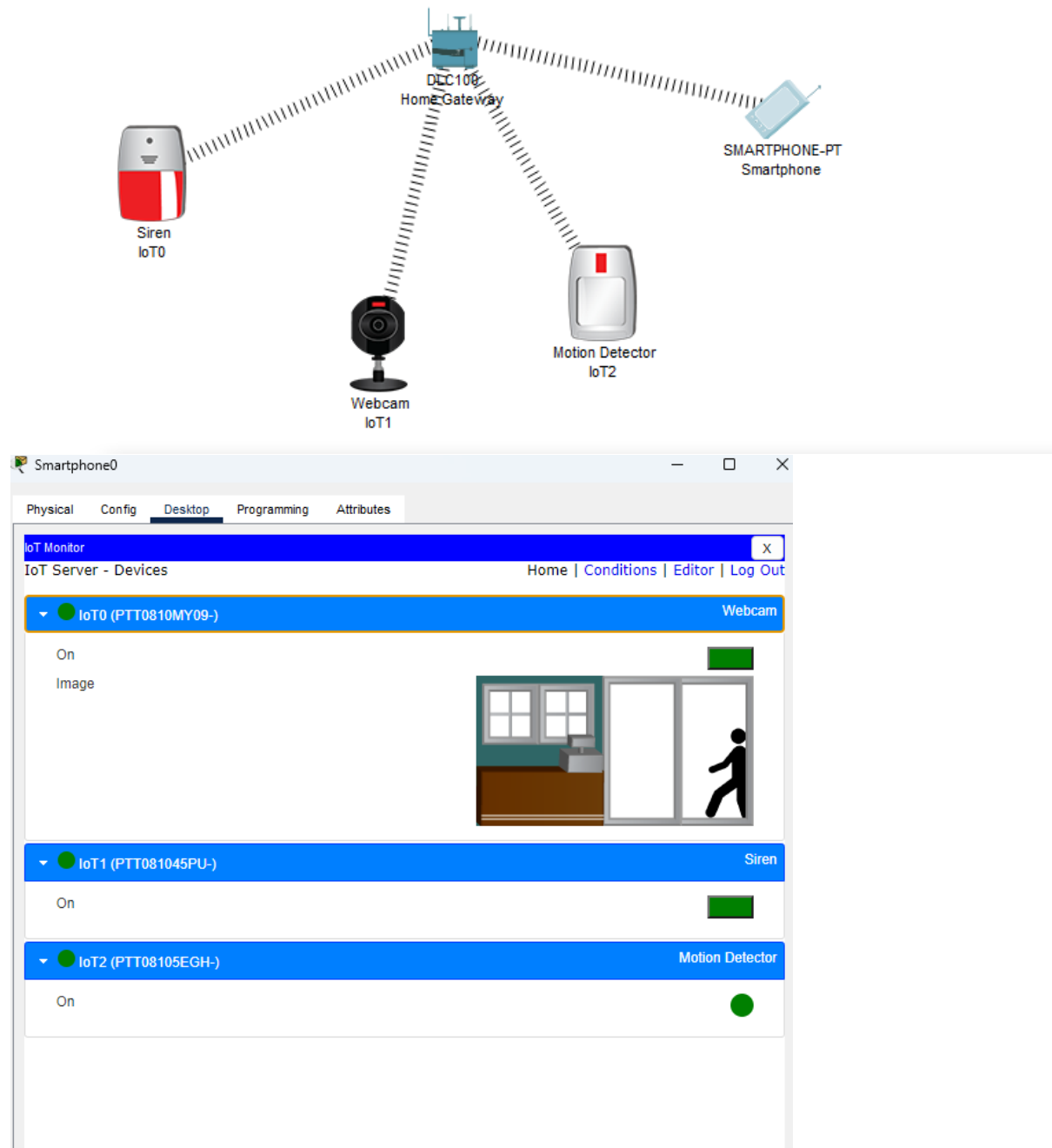


The screenshot displays the 'Smartphone' configuration window in Cisco Packet Tracer, with the 'Programming' tab selected. The interface shows the 'IoT Monitor' section with a table of device conditions. The table has columns for 'Actions', 'Enabled', 'Name', 'Condition', and 'Actions'. Two conditions are listed: one for 'webcam and siren on' and another for 'webcam and siren off'. Each condition has an 'Edit' and 'Remove' button. Below the table is an 'Add' button.

Actions	Enabled	Name	Condition	Actions
Edit Remove	Yes	webcam and siren on	IoT2 On is true	Set IoT1 On to true Set IoT0 On to true
Edit Remove	Yes	webcam and siren off	IoT2 On is false	Set IoT1 On to false Set IoT0 On to false

[Add](#)

Output:



Result:

Thus the program has been verified and completed successfully.