AN INTEGRATION OF IOT AND MIXED REALITY USING FOG COMPUTING CONCEPTS

by

Jeschel Jabez Sugumar Sunthar Jesudian, B.E

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Engineering
December 2018

DEDICATION

To my parents and my brother,

for their endless love and support.

ACKNOWLEDGEMENTS

I would like to express my profound gratitude to my committee chair and thesis advisor, Dr. George Koutitas, for his guidance, support and encouragements during my entire Master of Science program. I am grateful for his trust, patience and constructive criticism, which helped me improve the quality of my research.

Special thanks to my committee members, Dr. Semih Aslan and Dr. William Stapleton for their time, guidance, helpful and critical reviews throughout the course of my research. I would also like to acknowledge all of my friends and colleagues and the department faculty and staff for making my time at the Ingram School of Engineering at Texas State University a wonderful experience.

Above all, I am truly indebted to my loving parents and my brother for their unfailing support and encouragements.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Description
IoT	Internet of Things
VR	Virtual Reality
AR	Augmented Reality
XR	Mixed Reality
API	Application Programming Interface
LED	Light Emitting Diode
HTML	Hyper Text Markup Language
JSON	JavaScript Object Notation
URL	Uniform Resource Locator
IP	Internet Protocol
UI	User Interface
ICMP	Internet Control Message Protocol
TCP	Transfer Control Protocol
UDP	User Datagram Protocol
IGMP	Internet Group Management Protocol
MQTT	Message Queuing Telemetry Transport
REST	Representational State Transfer

ABSTRACT

Internet of Things (IoT) applications require low latency for maintaining a steady, close to real-time environment. Latency sensitive IoT applications include monitoring, collecting sensor data, controlling actuators based on sensor data. The current trend for IoT computations depends on cloud computing. In this architecture data from IoT device is processed and controlled by service-based cloud platform providers. These providers are expensive and require high bandwidth for data heavy and low latency operations.

The proposed solution leverages Fog computing concepts; Fog computing is an extended cloud computing service brought closer to the edge of a company's network. Low latency for actuator control is achieved by processing data at the network edge before the data is uploaded to the cloud. This reduces the load on the cloud, which in turn, increases cost efficiency by utilizing fewer cloud services. This concept is applicable for a wide range of applications including industrial IoT, automation systems, and human-machine interface. One such use is Integration of IoT and Augmented Reality which is particularly sensitive to latency.

I. INTRODUCTION

Internet of Things (IoT) has become so familiar recently, and applications are so extensive that need for further research has become increasingly important [1]. However, with the recent deployment of Augmented Reality (AR) technologies in the market, new possibilities and new applications which integrate IoT and AR are now possible [2].

The most compelling research field in this new arena is the integration of the IoT and Mixed Reality (XR) by Microsoft HoloLens. It would result in the deployment of virtual interfaces for IoT through augmented 3D projections [2]. Moreover, with XR features providing a user interface over the projections to control the actions of the actual physical device.

Augmented Reality

It is the merging of the virtual world and the physical world; it is made possible through immersive headsets such as the Microsoft HoloLens, where digital objects co-exist with physical devices [2]. The Figure 1 shows how a virtual object is overlaid over the physical device. The user interface involved in controlling the augmented projection is by hand gestures and voice commands.

Augmented reality working in tantrum with IoT would provide the user with real-time and real-world information [2]. For instance, in the existing 3-D simulation system and virtual reality application, the user wears an immersive head-gear in which a work place scenario is demonstrated in 3-D simulation. During each session, the user can learn through graphically designed simulation with static data. However, these simulations could show case dynamic data from IoT device to help enhance the study and training, making them more engaging for the user [3]. If safety permits, the mixed reality devices could even act as control devices for less manual workflow.

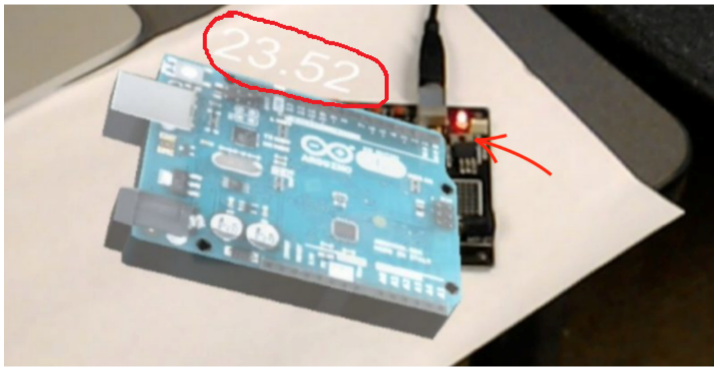


Figure 1. Hologram (AR) of a development board showing Sensor Data.

Temperature data (highlighted in red circle) taken from an IoT physical device which is augmented by a 3D avatar shown in Figure 1. The red arrow in the figure indicates the physical device below the augmented 3D projection. The application programming interface (API) which is responsible for hosting the temperature data on the augmented 3D projection from the IoT device is running on a Fog Computing environment.

Fog Computing

Fog computing is a term created by Cisco [4], which interprets as follows, an extended cloud computing service brought closer to the edge of a company's network. By utilizing this concept, we can handle the operation of computing, data storage and networking at the edge of the network [4]. The mentioned edge lies between the end devices and the cloud platform, thus resulting in faster response time for actuators which require direct local computation [5]. For example, an IoT device monitoring a temperature sensor of certain process needs to turn on an actuator to cool down when particular temperature is reached. This control instruction is processed and decided through cloud computing services [6]. However, this process will be faster if an instruction in-between IoT and Cloud took place [6]. This is where the Fog concept comes in play. The approach also ultimately reduces the utilization of the cloud platform and work in continuum with Cloud to explore new services. For apprehension, table 1 shows the difference between Cloud and Fog.

Industrial IoT

Business infrastructures utilizing machines from small scale to large scale have seen the advantages of field-ready IoT devices [7]. These IoT devices are also called Industrial IoT, which accumulate a vast amount of data from simple temperature sensors to multiple vibration sensors from machines optimized to work without stop [7]. These data are stored for monitoring the performance and reliability of the machines [7]. Later, the introduction of cloud computing made use of the extensive data in separate data centers where the data is used to analyze the performances of the machines and schedule calibration or maintenance breaks and other tasks which required computation [7]. The established system performs well for functions which do not need low latency services. However, it lags behind for operations requiring immediate attention. Moreover, the cloud platform is required to stay always active for live monitoring and data computation, this causes a significant constraint on the operational cost [8].

The scenarios will benefit from Fog concepts, which could provide a solution for storage, computing, and control by distributing the access to the current task by workload-based priority [8].

Thesis Outline

This thesis research will consider the above-mentioned scenarios and demonstrate the successful integration of the IoT and Augmented Reality using Fog concept and highlight its constancy in the real world. The proposed thesis would show case the following tasks and its research contributions:

- Creating an efficient Server-Client API based on Fog Computing Concept
- Creating a holographic application for HoloLens to make use of the Server-Client API
- Evaluate the conceptual system architecture.

II. LITERATURE REVIEW

After conducting a literature review to look into the prior effort made to address the task at hand, the following was found. Smart devices and IoT are in a well-developed stage [9]. Augmented reality has also blended smoothly into the real world [10]. Combining internet of things and augmented reality brings new possibilities and open attractive business propositions [11].

Research survey on Fog Computing

Fog computing is a recent development made for creating a distributed service for data storage, computation, and control which brings cloud services closer to the end user device thus reducing the response time [4]. Cloud computing has solved large data computation requirements in industrial application. However, during prolonged use, it applies significant strain on network bandwidth, making it more cost consuming [12]. Whereas Fog operates closer to the network edge and requires less bandwidth since most process is localized. A detailed comparison can be found in the paper [12] stating its reliability and scalability in the industrial environment.

Many researches point out different aspects of the network system to be the cause increase in latency. Most logical reason derived from the theoretical network delay is the distance between the provider and the user. As the distance increases the information has to travel much longer distance and it makes sense to arrive at that conclusion [13]. However, it is still unclear to pin point the issue of latency between the cloud and the end devices, simply due to insufficient research evidence [14].

Over the recent years, the growth in technology has provided ample computational room in systems thus reducing the possibility of the processing delay.

Which happened to be one of the reasons for the establishment of centralized data centers where heavy computational process was handled [15]. But that alone could not provide enough power to reduce latency when upload and download of data was considered. This unleashed a backlash on end users which depend on low latency functions which in turn had to wait in queue for its data to arrive from the cloud [16].

Therefore, it provided information for researches to come up with a decentralized data computational system which handled latency sensitive content at the edge of a network [12]. Fog plays a vital role in handling latency sensitive process such as emergency management where, immediate response is mandatory. Whereas, this does not mean it is cloud substitute but, rather be a complement to the cloud [13]. Its architecture shows keen leniency towards handling latency sensitive content closer to the end users Figure 2 showing results which answers the theoretical network delay factor of distance between cloud and the end-user.

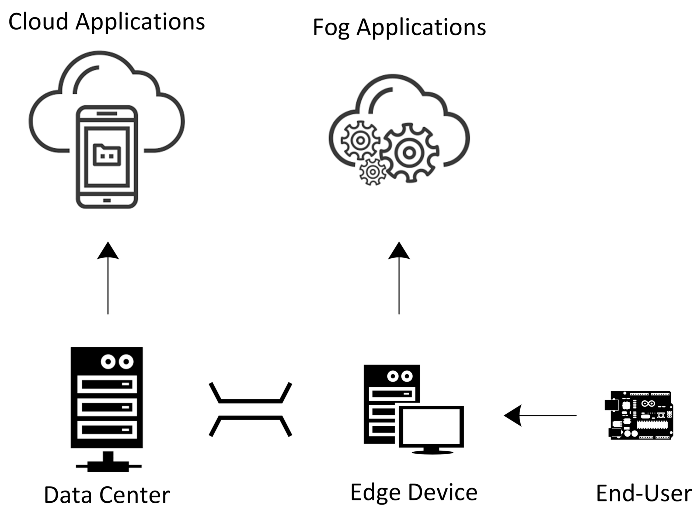


Figure 2. Fog Computing Architecture

Research survey on Industrial IoT

Most businesses utilize cloud service to maintain industrial devices such as heavy-duty machines [16]. In recent technological development, most automated machines require large data computational operations, which is primarily handled by Cloud services [16]. Cloud computing has provided the solution for problems related to massive data storage and as mentioned before for large-scale data computation [15].

In the recent years, many industrial organizations have accepted the use of embedded IoT in industrial environment to simply monitor and perform simple management tasks [17]. The utilization of cloud computing for production machines made valid contribution to data analytics which in turn provided information to strategize company operations. However, the embedded IoT which handles minor task and does not require data handling or complex computation where also integrated into the cloud [17]. This caused delays in latency sensitive operations because the process was simply in queue to be executed at the cloud [17].

The centralization of process at cloud has significant use when data analytics is considered but, not efficient enough for latency sensitive processes. In the recent years, it has been noted that network traffic of Internet of things devices has increased to disquieting numbers [18]. Ultimately resulting in adding more queue to the cloud systems. The factor responsible is due to increased use of smart devices [18]. The resource handling in the cloud is very sophisticated and complex, and during heavy network traffic, the information delivery to the end user is slow [19]. Fog computing could result in a handshake between the existing cloud. It would act as a filter to minimize the number of data entry to the cloud and handle latency sensitive process at the network edge.

Research survey on Augmented Reality

Now focusing on the augmented 3D projection to overlay on the internet of things device, a research paper [20] shows a scalable framework where the IoT informs the head-gear of its presence and makes it easier for tracking and proposes a control layer through an online connection. However, Microsoft HoloLens [2] eliminates the need for such framework by handling direct object tracking built into the actual head-gear which hosts a 32-bit computer [2]; the setup requires detail study of graphic designing software and calibration [2]. However, with support from third party application such as Vuforia the device tracking excels in pinpointing the physical device and overlays an augmented 3D projection with ease [2].

Recent research on augmented reality shows the numerous possibilities of its results namely, in education, in business, in lifestyle etc., [10]. In education, showing instructions and study guide for lab equipment. It could make study of complex subjects easier by visual representation in holograms and interact with them for my information [21]. In business environment, recent study in marketing techniques involve demonstrations of product and services through augmented reality has given rise in obtaining potential investors [3]. In lifestyle, it could obtain information about objects and street signs and indoor navigation from the internet and relay that information over an augmented reality for better comfort [20].

The number of possibilities rises each day as the technology behind its working is being perfected for matching specific use cases. The current awe factor in this field is the integration with IoT which would highlight the information visualization capabilities of augmented reality applications [22].

Problem statement and solutions

The work scenarios stated in research surveys, clearly shows the following are observed as the prime focus, need for storage and data handling capabilities for the IoT devices [23]. An adequate data computation, low latency system with a cost-effective solution for cloud service utilization is needed [24]. Need for further improvement in the training simulation where the user should be exposed to real-time data from the machines to utilize their training simulation better [25]. By comprehending the above statement and the initial introduction section, it would undoubtedly give an insight of what is to be done.

The solution for all the above problems combined is the integration of the IoT and XR [25]. Fog solves the delivery of real-time data to the immersive head-gear. And now the user has access to real-time data from the working industrial IoT during 3-D simulations [23]. The integration also provides virtual interaction with the actual physical device thus delivering an immersive mixed reality experience to the user.

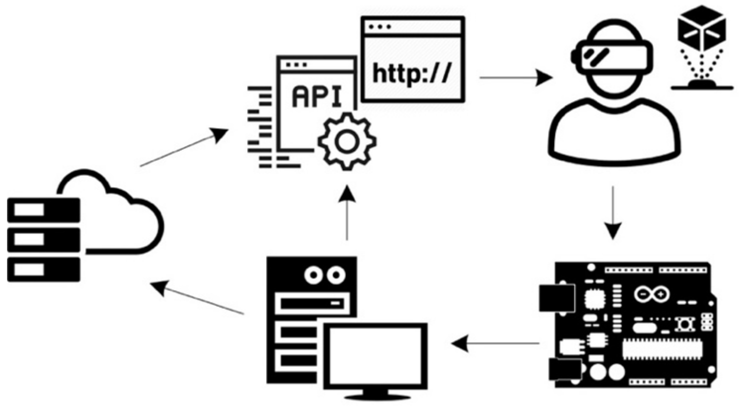


Figure 3. Proposed System Architecture.

The proposed architectural block from Figure 3 shows the overview of the stated solution. The back end of the IoT and XR integrating API is Fog ready which decreases the cloud to end device computational time and localizes the immediate time-dependent tasks to the network edge for real-time response [26]. The introduction of Fog concepts will reduce the need for always active cloud service, decreasing the operation cost of cloud computing and enable a controlled flow of required data to the cloud thus efficiently managing the cloud service [4].

The thesis finding which is to follow further in this report will show the Fog concepts in play. The programming scripts are designed and written in such way that it reveals the ground up optimization of the project and its research findings.

III. INTERNET OF THINGS DEVICE

Overview

The function of IoT device mainly consists of collecting data and processing data and storing data [19]. The development board selected for this thesis is Arduino Uno.

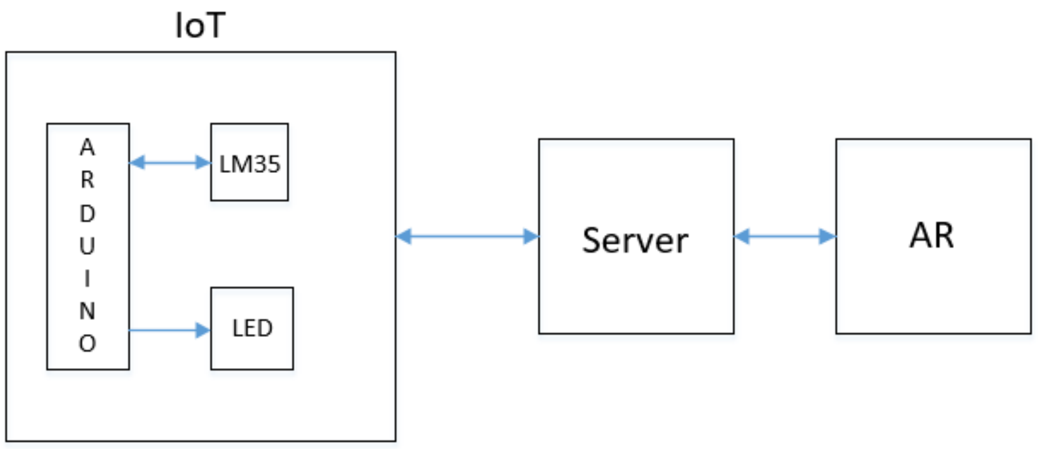


Figure 4. IoT Block Diagram

Arduino is an open source microcontroller development board based on Atmel micro controller chips and development by Arduino.cc [27]. It is programed using an Arduino IDE (Integrated Development Environment). It works on 5 volts powered through battery or USB when connected to computer [27]. It is perfect development board to run experiments and projects for beginners to experts. It is widely used in Internet of Things do-it-yourself project [28] for the reason that it is mainly open source and vast number of communities working with the device. It is clear to say that Arduino is the most popular embedded development board among hobbyists [28].

The reason behind choosing this board for this thesis is well understood. The support available for the board in means of firmware, software is within reach and we can move ahead with the thesis objectives with ease. The board we have chosen for the research is Arduino Uno which host an 8-bit AVR Atmel ATmega328 [27] with a clock speed of 16MHz. The Uno development board also host numerous input and output ports using which an external device such as sensors, actuators, etc., can be programed and added to the board with ease [27]. The technical specification of the Arduino Uno can be found in Table 1.

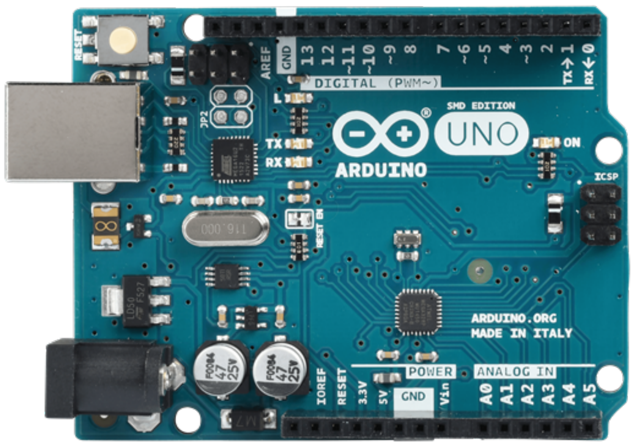


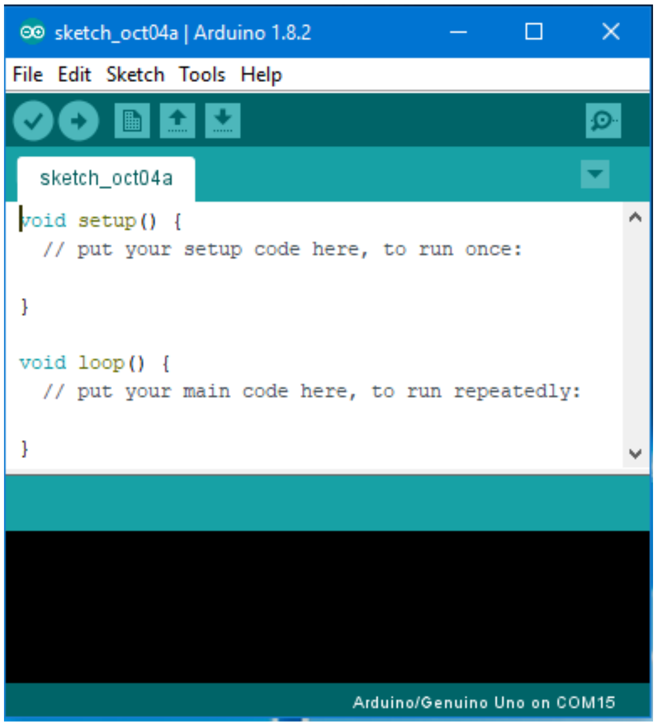
Figure 5. Arduino Uno Development Board

Table 1. Arduino Uno Technical Specifications [27]

Parameter	**Specification**
Input Voltage	7-12v (DC Jack) 5v (USB)
Max Output Current	40mA Per Pin
Processor	ATmega328P
Oscillator	16 MHz
Analog Pins	6
Digital Pins	14
PWM Pins	6
Other Specifications	4 LED’s (TX, RX, Pin 13, Power Indicator), Reset Switch
Flash	32 kB

Embedded Programming

The Arduino.cc community also provides user-friendly graphical user interface enabled software Figure 6 which make it all the more ideal for the research objective. The software highlights the keywords and functions which are in sync with the Arduino libraries in the backend automatically [28]. Thus, allowing the user to focus on the program scripts and its output results. The programming language used is embedded C++[27].


sketch_oct04a | Arduino 1.8.2
File Edit Sketch Tools Help
sketch_oct04a
Arduino/Genuino Uno on COM15

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 6. Arduino Integrated Development Environment

The IDE presets the programing environment with two major functions.

- Void setup ( )
- Void loop ( )

Void setup ( ): It marks the start of the programing script. It is a built-in function in which the initialization of the user variables and the input/output ports which will be in use are described [27].

Void loop ( ): It is a built-in function with-in which the execution of tasks take place. The task described inside the loop function run in an endless loop at the clock rate of 16MHz for Arduino UNO [27]. For example, take a look at a basic program script used to blink a light emitting diode.

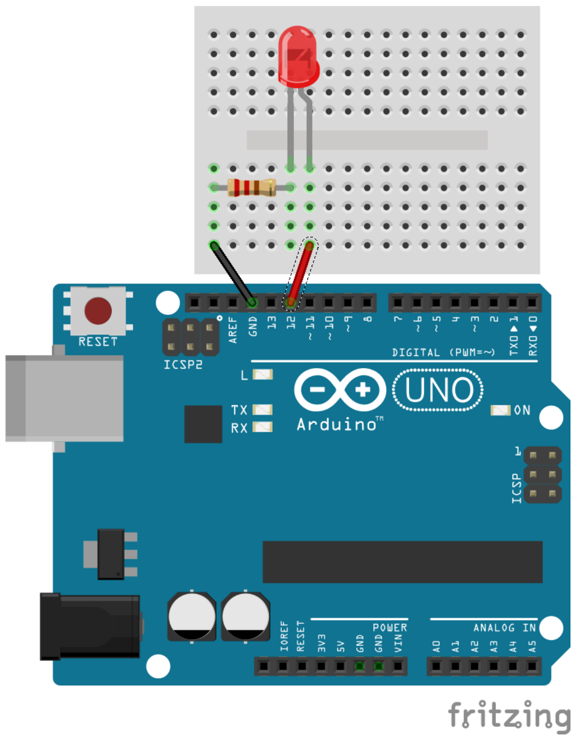


Figure 7. Circuit Design for LED Blink

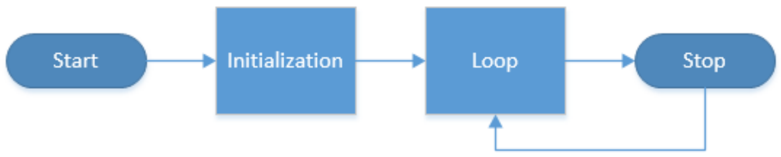
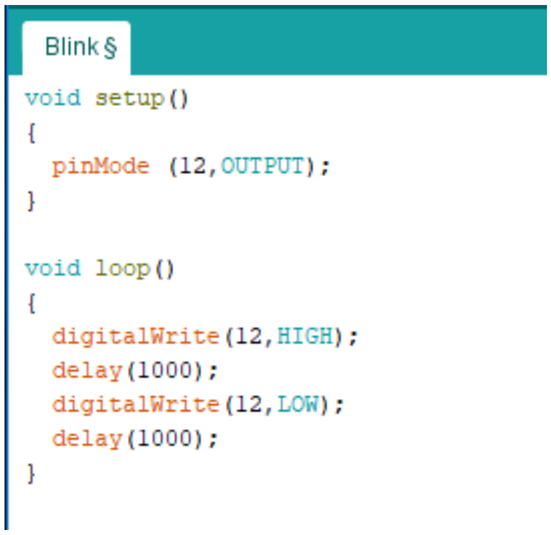


Figure 8. LED Blink flowchart



```
void setup()
{
  pinMode (12,OUTPUT);
}

void loop()
{
  digitalWrite(12,HIGH);
  delay(1000);
  digitalWrite(12,LOW);
  delay(1000);
}
```

Figure 9. LED Blink script

In the above example script “pinMode (12, OUTPUT);” as explained earlier, this line shows us that pin number 12 is initialized as output port. Further down the script “digitalWrite(12, HIGH);” is the task the Arduino is to perform, the script says the Arduino to make the digital port 12 to HIGH meaning set a digital voltage value to the port for “delay(1000);” delay of 1 second and the same to LOW state for a delay of 1 second to set it off. A detailed view of the circuit can be seen in Figure 7.

Sensor and Actuator

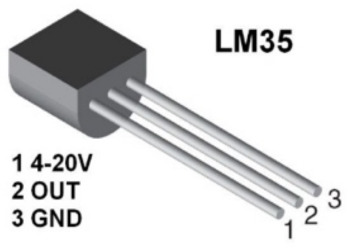


Figure 10. LM 35 with Pin Configuration [29]

The Arduino board hosts a temperature sensor and an LED to demonstrate the working of features available in an IoT device. This is one of the basic operations an IoT device could be programed to do which can clearly highlight the functions or tasks carried out by an IoT device over a graphical user interface enabled dashboard [9]. In our case it is through a basic HTML page over the internet.

The temperature sensor used in this research is the LM 35 it is an analog temperature sensor with three pins, as shown in the figure 8. It is a product of a company called Texas Instruments which specializes in making various types of semiconductors and integrated circuits [29]. LM 35 is calibrated to give temperature value in Celsius by default. It works on 5volt dc supply and has a range of Negative 55 to Positive 150 Celsius [29]. It is an idle sensor for research purpose and simple to use in an Arduino programming environment.

The actuator used is a simple red color light emitting diode. It performs as a substitute to any other actuators by simply being an output to indicate a response to an input task such as On/Off.

Data Collection

This section consists of the C++ program script uploaded to the Arduino Uno. The elaborate explanation of the code will follow below every significant part of the script which harbors the Fog concepts.

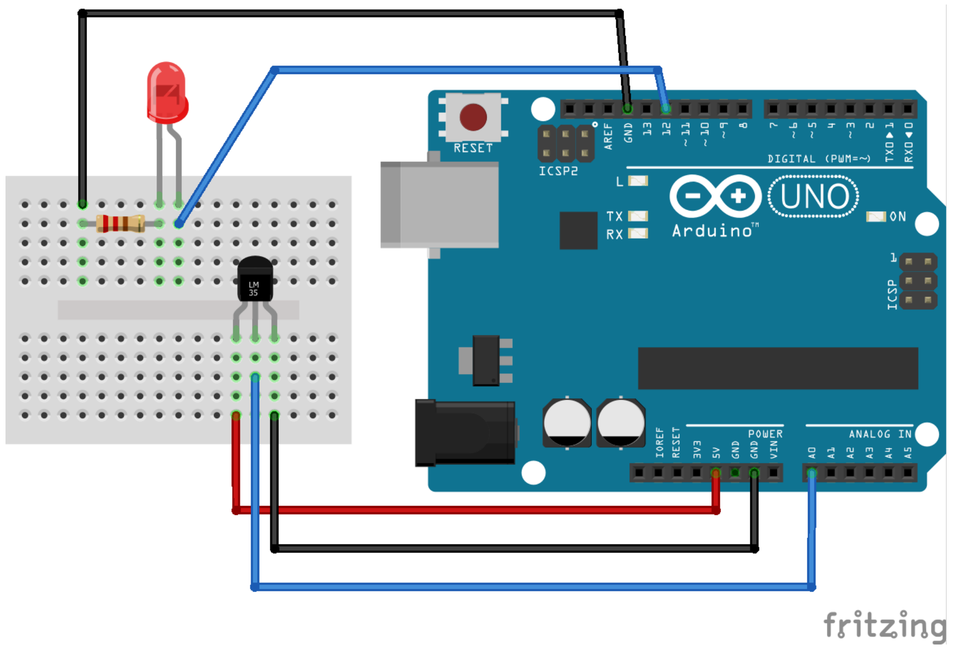
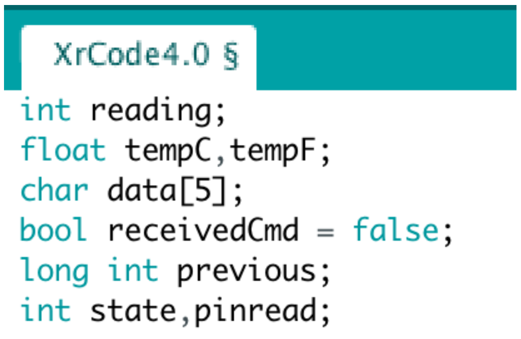


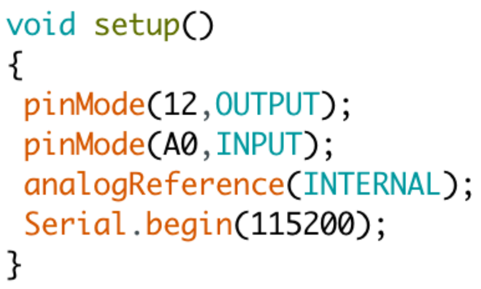
Figure 11. IoT device Circuit Design


XrCode4.0 §

```
int reading;
float tempC,tempF;
char data[5];
bool receivedCmd = false;
long int previous;
int state,pinread;
```

Figure 12. IoT script Data Initialization

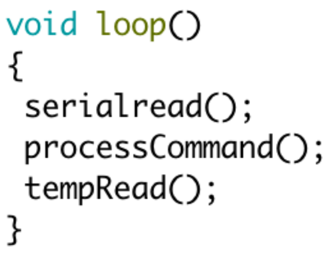
The above initialized global variables come in handy with the functions performed by the Arduino. Some of the variables will have stronger meaning when used in the functions which follows below.



```
void setup()
{
 pinMode(12,OUTPUT);
 pinMode(A0,INPUT);
 analogReference(INTERNAL);
 Serial.begin(115200);
}
```

Figure 13. Void Setup Pin Mode Initialization

The pin 12 is used as an output and pin A0 is used as input to receive data from the temperature sensor as seen in Figure 11. The functions that are to be performed by the Arduino are distributed into local functions within the scripts. This shows the concept of Fog can be implemented into the root level of programing an IoT board. The distribution of functions decentralizes the workload and guides the Arduino to perform the tasks when necessary rather than running in an endless loop of execution.



```
void loop()
{
 serialread();
 processCommand();
 tempRead();
}
```

Figure 14. Void Loop Functions

The serial read function Figure 15 and 16 looks for any input available at the port during every start of new execution loop. If the data is available at the port, then another loop starts making the port to stay open while it stores the incoming data in an array element called “data[]”.

After the data is stored the function stores an internal acknowledgement by saving a bool element into the variable *"receivedCmd = true"*. This will indicate whether the next function should execute or not based on the bool value of true or false.

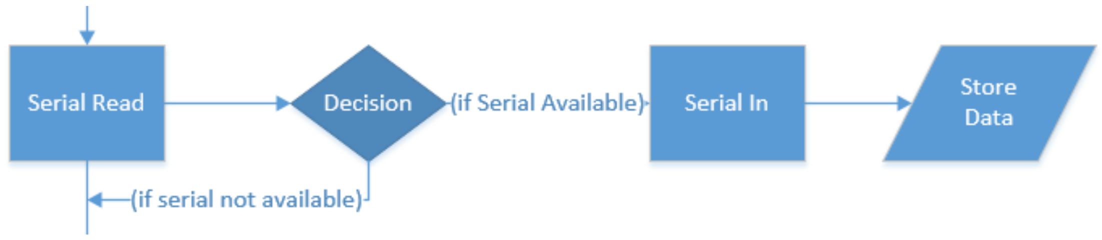
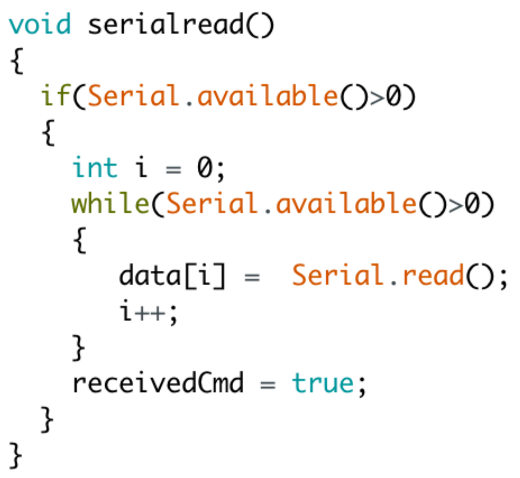


Figure 15. Serial Read Flowchart



```
void serialread()
{
  if(Serial.available()>0)
  {
    int i = 0;
    while(Serial.available()>0)
    {
       data[i] =  Serial.read();
       i++;
    }
    receivedCmd = true;
  }
}
```

Figure 16. Serial Read Function script

The "*processCommand()*" function Figure 17 and 18 compares the "*data[]"* array with specific set of commands to execute its related task. If the data received contain "h" then the function to turn LED On is executed and vice versa for Off. If the data received contain "t" then the function to read and update temperature variable is executed. In this way the scripts are organized for easy understanding and further development.

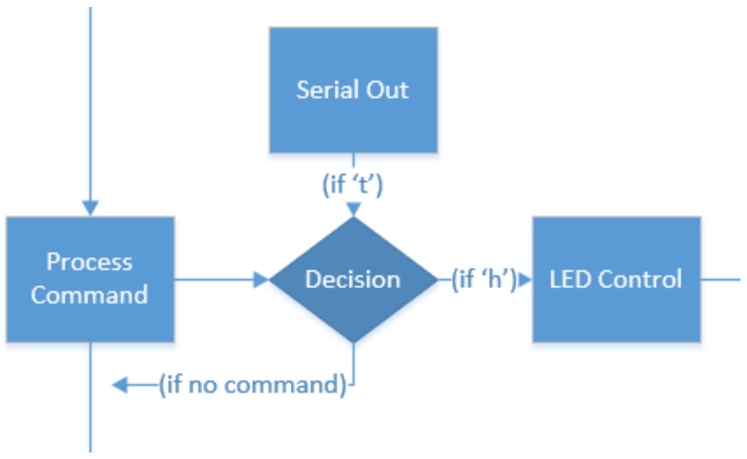
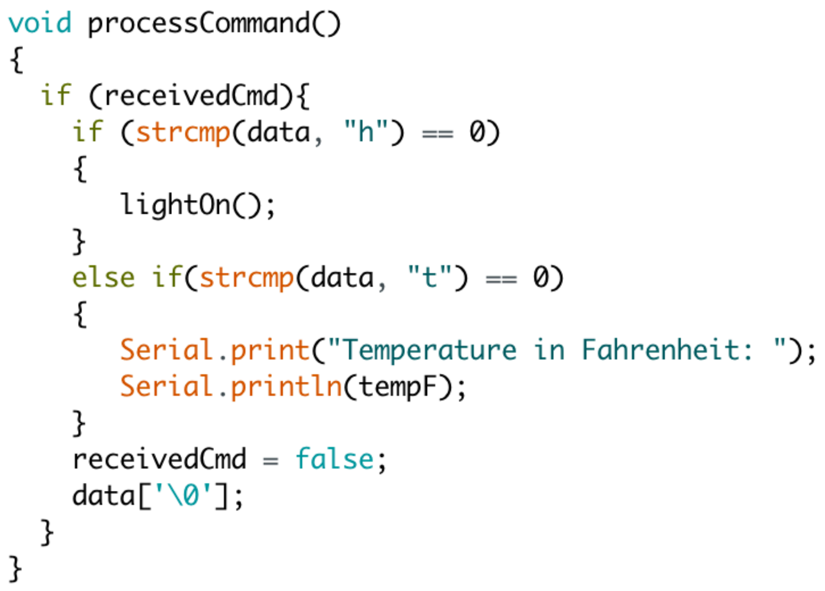


Figure 17. Process Command Function flowchart



```
void processCommand()
{
  if (receivedCmd){
    if (strcmp(data, "h") == 0)
    {
       lightOn();
    }
    else if(strcmp(data, "t") == 0)
    {
       Serial.print("Temperature in Fahrenheit: ");
       Serial.println(tempF);
    }
    receivedCmd = false;
    data['\0'];
  }
}
```

Figure 18. Process Command Function script

The lightOn() function Figure 19 and 20, performs as a Toggle switch for the LED added to the board as an actuator. When this function is invoked it performs a "*digitalread()*" on pin 12 to record the current state of the pin. Since pin 12 is a digital port, the only values you can get from it is 1 or 0. This is in turn used as an input to the *"if"* statement used in the function to either make the LED On/Off.

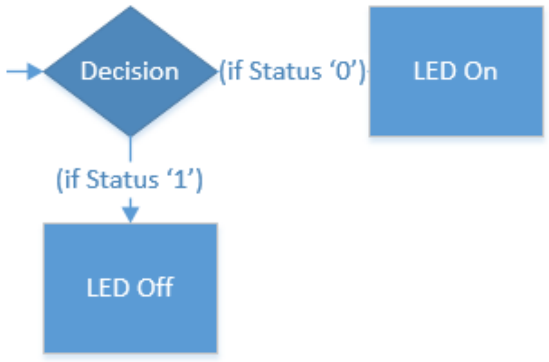
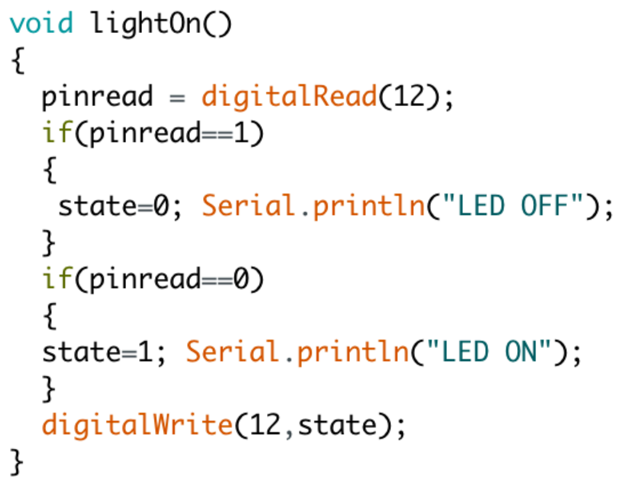


Figure 19. LightOn Function flowchart



```
void lightOn()
{
  pinread = digitalRead(12);
  if(pinread==1)
  {
   state=0; Serial.println("LED OFF");
  }
  if(pinread==0)
  {
  state=1; Serial.println("LED ON");
  }
  digitalWrite(12,state);
}
```

Figure 20. LightOn Function script

The tempRead function Figure 21 and 22, which is involved in posting the temperature value. This function is included in the main Loop function of the Arduino and it is executed in every Arduino process cycle. However, there is a Fog concept involved in optimizing the temperature read function[30]. At the start of the function a task *"long int current = millis();"* is performed. It is one of the Arduino in-built library function which measures the time it took for the complete execution of the program running in the Arduino. The time is measured to find if the temperature value has changed compared to the last execution. Using which we can control the run cycle of this function. A "If" statement is included in the function to see if the value has altered. If so, then the function would update the new value to the "*tempF*" variable, else it stays low. This way the task is optimized for efficiency.

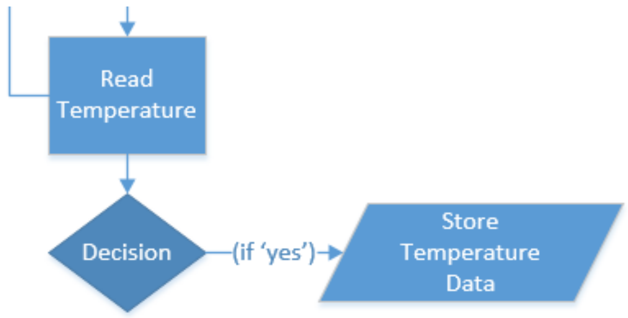
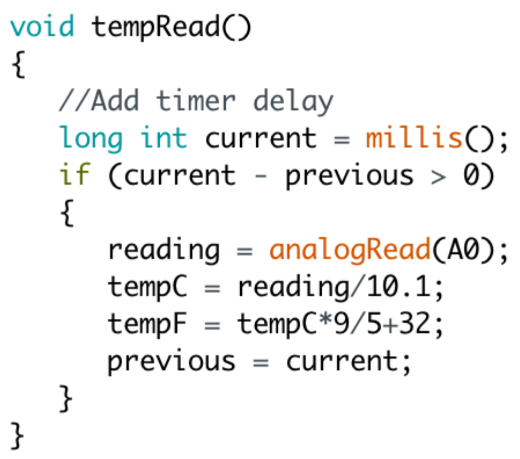


Figure 21. TempRead Function flowchart



```
void tempRead()
{
    //Add timer delay
    long int current = millis();
    if (current - previous > 0)
    {
        reading = analogRead(A0);
        tempC = reading/10.1;
        tempF = tempC*9/5+32;
        previous = current;
    }
}
```

Figure 22. TempRead Function script

IV. INTERNET OF THINGS SERVER

Overview

Server is a computer program, or a computer designated to provide functionality for other computers or programs [30]. The general idea behind the server architecture is server-client relation, the use of global platform to run applications which shares the interfaces between devices [30]. Centralization of applications to run on a single communication bus, lend to the current systems which run in local and global network [30]. Which brought the rise of cloud servers and portability by enabling people to access content over Internet. Later in the technological development, the work load on servers began increasing high causing severe traffic in the network causing application interfaces to fail [15]. Then the term of decentralization brought about new ways of transporting data from one point to another [15].

The pendulum of centralization and decentralization has been an ongoing problem faced even in today's technological era [32]. The term Fog computing was recently coined by CISCO who are the lending pioneers in networking [26]. The fundamentals of the server-client come into play but, this time the process is made flexible so that the pendulum can swing back and forth. Meaning, the devices can change their functionalities based on all the necessities involved in handling a task [12]. For example, a person using a smart phone to view some navigational content, can be moved over to a car's stereo which also utilizes similar features synced from phone. However, the content has moved from phone to stereo and this shows the portability in platform. Both devices running in a synced network but now the stereo has taking over the task of showing the person the data and the phone has simply become the communication channel through which the stereo is synced [13].

Fog concepts dispels the idea of distributing the computation, communication, control and storage thus, creating a decentralized process [31]. However, it also works in sync with the cloud servers creating the smooth transition to centralization for computation heavy tasks and moving back to distributed system for priority and less computational tasks enabling a fast-responsive system [1].

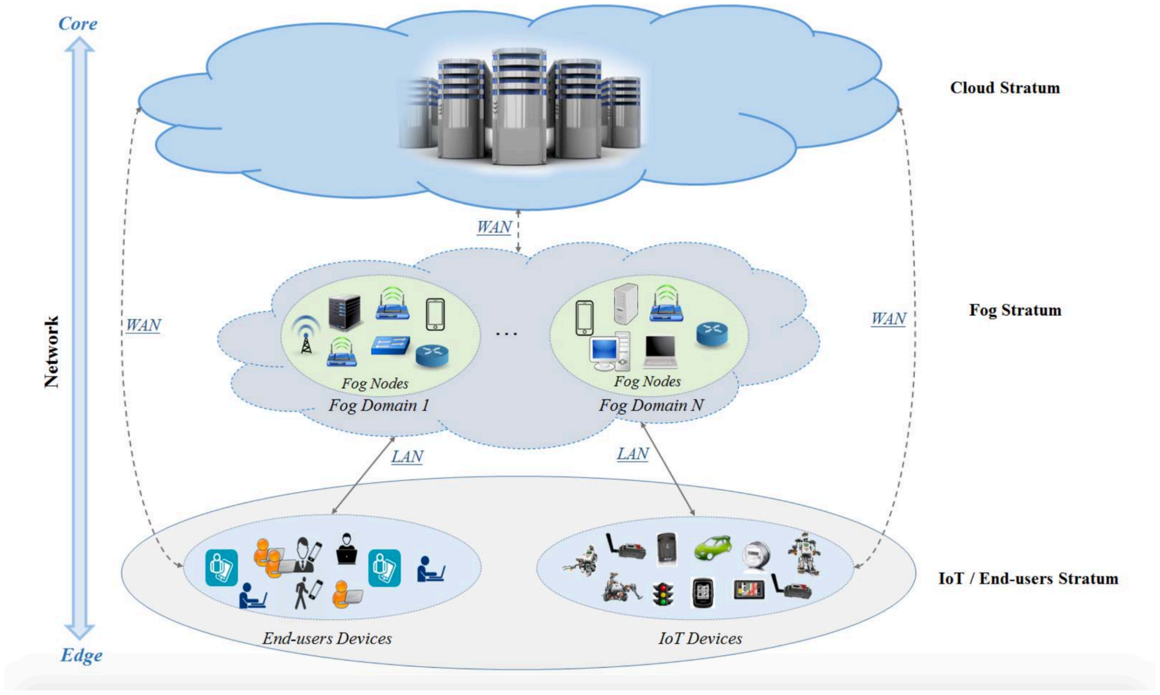


Figure 23. Topology showing the Fog is closer to the End-Users [1]

Server Design

The elaborated server section showing the internal structures involved in the server program can be seen in Figure 24. The main part will be the server code which interfaces the other application such as communication, user interface and storage to provide client with complete service expected from the server at hand [15]. The current task at hand is to view the real-time data from an IoT device which is in sync with this server and provide that data to any number of clients accessing this content in the local network as well as in the Internet.

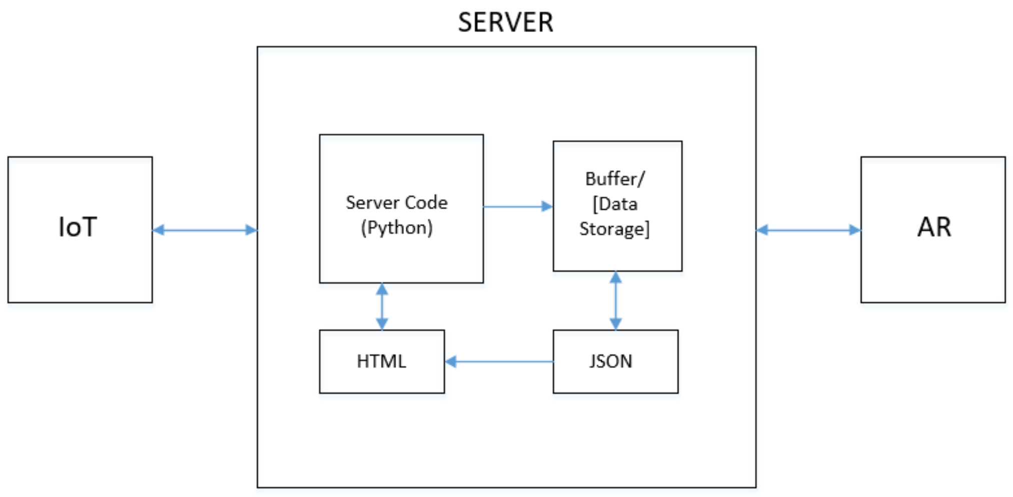


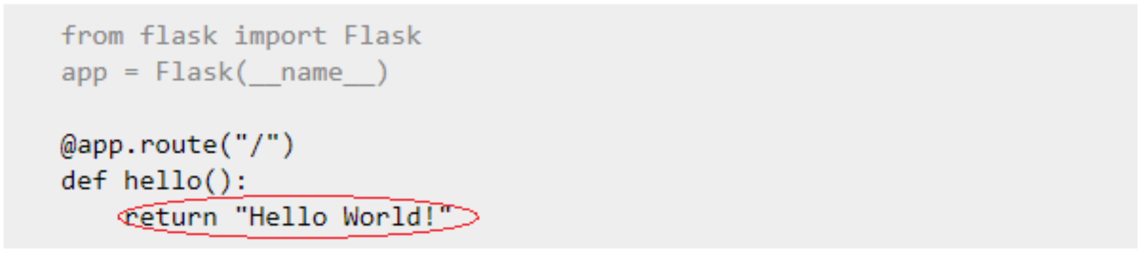
Figure 24. Server Block Diagram

Server is the integrator in this solution where it is the API between the IoT and AR device. The major research done is reflected upon the system when the data logging to a database is eliminated in the process. This shows a valid proof of concept in play showing the unnecessity of database for the above task. The idea is derived from the fog computing concepts and simply by experimentation and use of database in the proposed system [7]. It is found after multiple test and results that the use of database to log the temperature is not required for the integration. Making it slightly fast in response.

Flask Programming

The server script is written in python as it is one of the basic programing languages using which a server can be formed and does not involve complicated libraries [34]. This setup is an idle place to test the fog concepts since every line of script can be optimized specially for the task at hand of viewing the IoT device data over internet close to real-time [33].

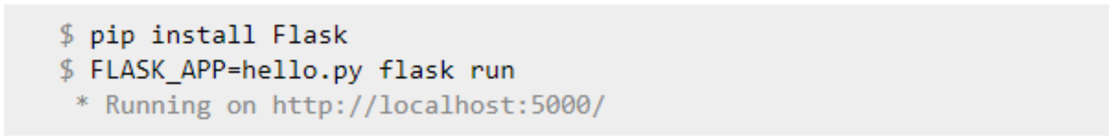
The structure involves Flask which is a micro framework for python based on Werkzeug web server gateway interface (WSGI), Jinja2 template [34] and are completely open source. It opens a path to research and development, with ample opportunities for experiment and learning and gather new research data. For example, Flask provides a basic template to show how simple it is to run a server program [34]. The Figure 25 and 26 shows a simple server script to display “Hello World!” in the web browser.



```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"
```

Figure 25. Flask Sample Script [34]



```
$ pip install Flask
$ FLASK_APP=hello.py flask run
 * Running on http://localhost:5000/
```

Figure 26. Flask Execution Command [34]

The output is seen on a web browser when the above-mentioned http link is accessed. The output would display a simple “Hello World!” text in the webpage.

Similarly, the below script is also designed to execute and give output on the webpage. However, here we have designed a HTML page to be displayed and the HTML page works in sync with the server script in the backend acting as input and output portal [35] for the server script.

We are using the following libraries,

- Flask – It forms the basic platform to use and execute server functions
- Render_template – It is a built-in library which helps execute the HTML page and make it available as input and output portal for the server script.
- Serial – To establish a serial link to obtain information from the IoT device
- JSON – To refresh the HTML page with new data from the IoT device.

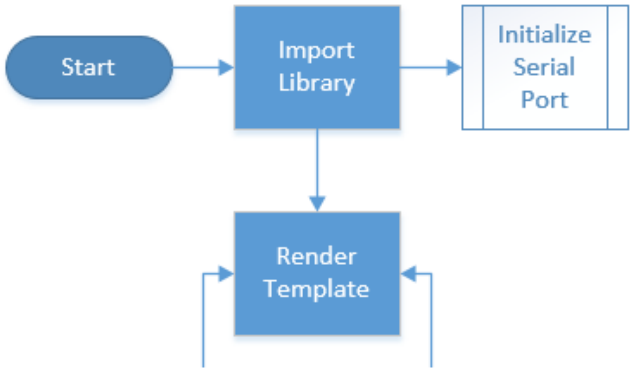
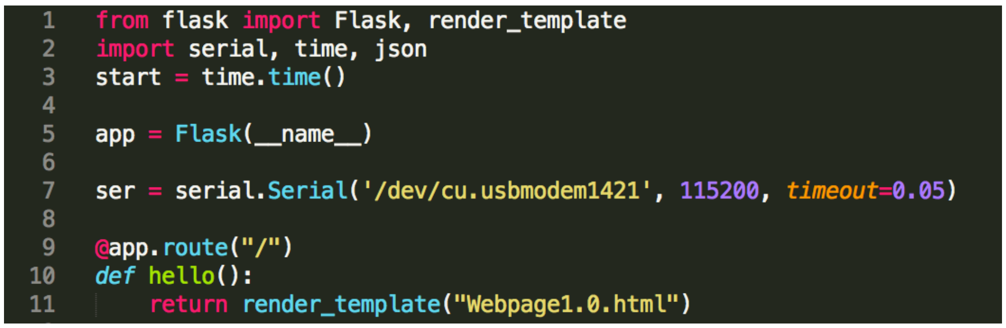


Figure 27. Flask Initialization flowchart



```
from flask import Flask, render_template
import serial, time, json
start = time.time()

app = Flask(__name__)

ser = serial.Serial('/dev/cu.usbmodem1421', 115200, timeout=0.05)

@app.route("/")
def hello():
    return render_template("Webpage1.0.html")
```

Figure 28. Flask Initialization script

The script in Figure 28, executes the required libraries first and load them to start the server process. At first the server loads the HTML page "*Webpage1.0.html*" on request from the web browser when the http link is established.

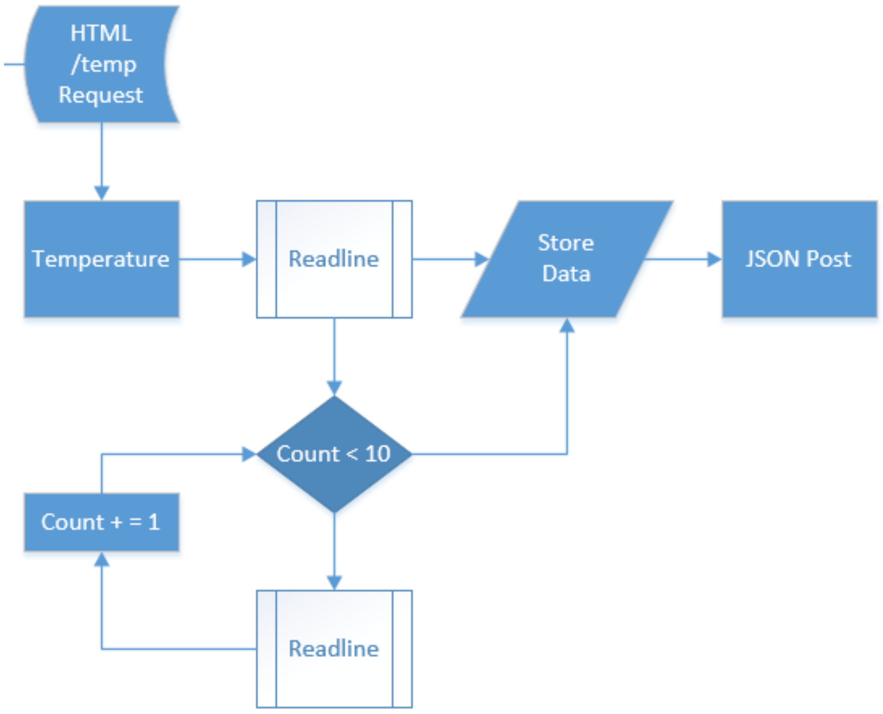
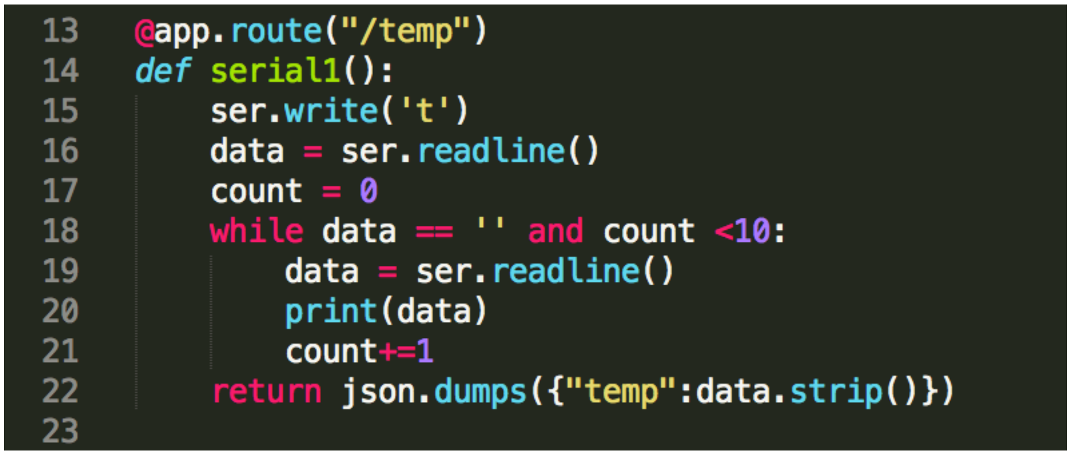


Figure 29. Flask Temperature Update Function flowchart



```
@app.route("/temp")
def serial1():
    ser.write('t')
    data = ser.readline()
    count = 0
    while data == '' and count <10:
        data = ser.readline()
        print(data)
        count+=1
    return json.dumps({"temp":data.strip()})
```

Figure 30. Flask Temperature Update Function script

The script in Figure 30, loads in response to the request from the HTML page asking for "*/temp*". When executed, it opens the serial port on the connected IoT device and asks for the temperature data and loads the value on to JSON buffer storage script which in turn refreshes the previous data with the new temperature data. The while statement is a failsafe protocol designed to make sure the serial portal is ready for use when requested from the HTML page [8]. This level of control and temporary storage on the JSON buffer is a Fog concept, where control to take data from the IoT device is given to server rather than IoT device [9].

The traditional way is that the IoT updates to a database and then server retrieves from the database [36]. This process is eliminated to reduce the number of executions needed to update a temperature data on to a HTML page. The protocol is designed as such to facilitate the task at hand of simply viewing the current data. The Fog concept come in play here where the storage for the vast data update from the IoT device is cut short.

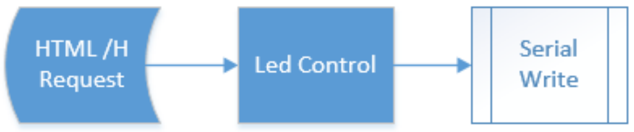
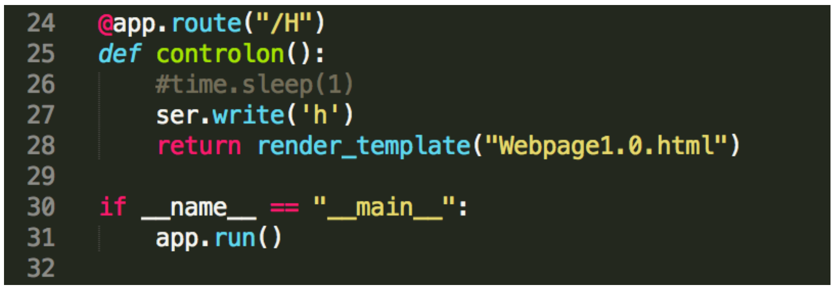


Figure 31. Flask LED Function flowchart



```
@app.route("/H")
def controlon():
    #time.sleep(1)
    ser.write('h')
    return render_template("Webpage1.0.html")

if __name__ == "__main__":
    app.run()
```

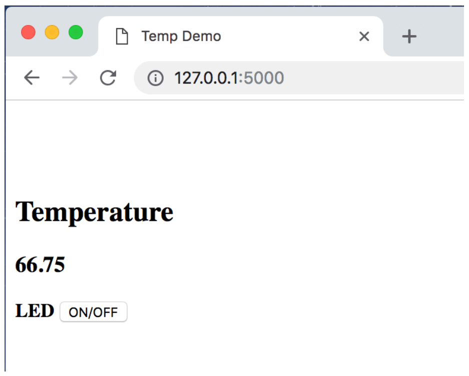
Figure 32. Flask LED Function script

The script in Figure 32, loads in response to the request from the HTML page asking for "*/H*". When the function "*/H*" is requested from the HTML page, the server opens the serial port and gives a command to turn On/Off an actuator on the IoT device.

The server script concludes with the “*app.run()*” which makes the complete python script into an active executable file [34].

HTML Design

Hypertext Markup Language is a standard markup language which is syntactically distinguishable from text for creating web pages and web applications [37]. It is the front end of the server through which requests and tasks are performed. It is programed using the REST API protocol, where the client and server follows GET data and POST data [37]. The HTML file is rendered by the Flask script and feed to the web browser for display when called using the host http as shown in Figure 33.



Temperature

66.75

LED

Figure 33. Rendered HTML Webpage

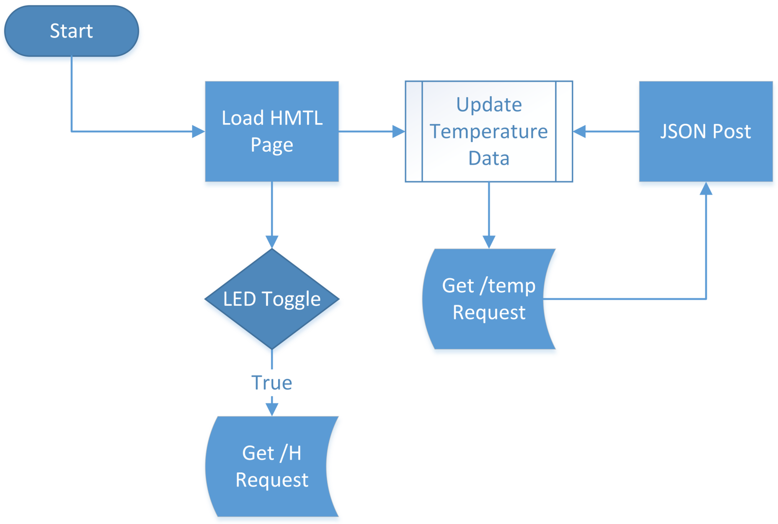
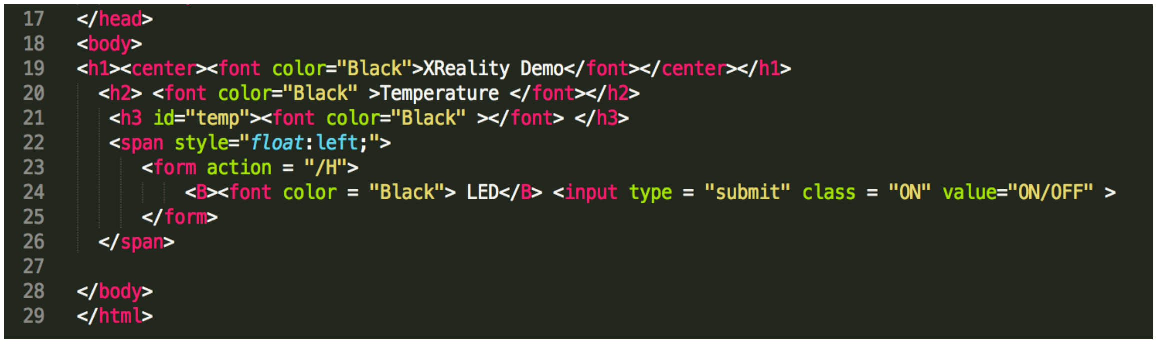


Figure 34. HTML Webpage flowchart



```
</head>
<body>
<h1><center><font color="Black">XReality Demo</font></center></h1>
  <h2> <font color="Black" >Temperature </font></h2>
   <h3 id="temp"><font color="Black" ></font> </h3>
   <span style="float:left;">
      <form action = "/H">
          <B><font color = "Black"> LED</B> <input type = "submit" class = "ON" value="ON/OFF" >
      </form>
  </span>

</body>
</html>
```

Figure 35. HTML Webpage script

Buffer and JSON

JavaScript Object Notation is a lightweight data-interchange format [38]. It enables a web browser to communicate with the server to update content which does not require user intervene [38]. It is a backend script (shown in Figure 37), to HTML which enables the dynamic refresh of data from buffer to the front end periodically when front end is active.

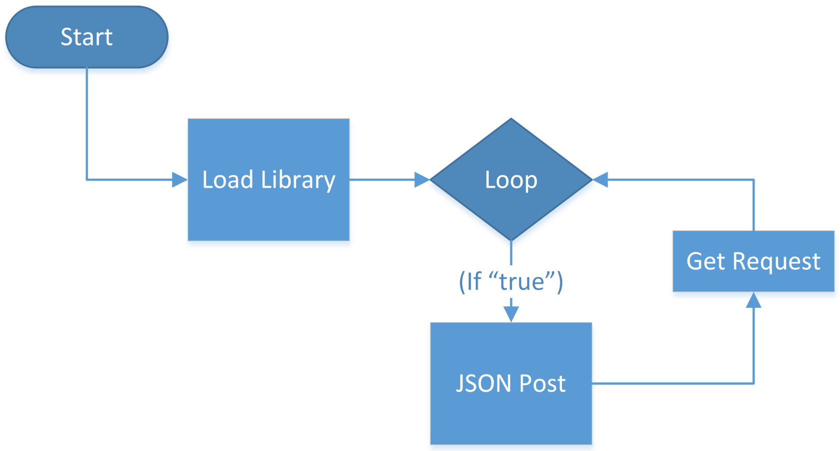


Figure 36. JavaScript Update Function flowchart

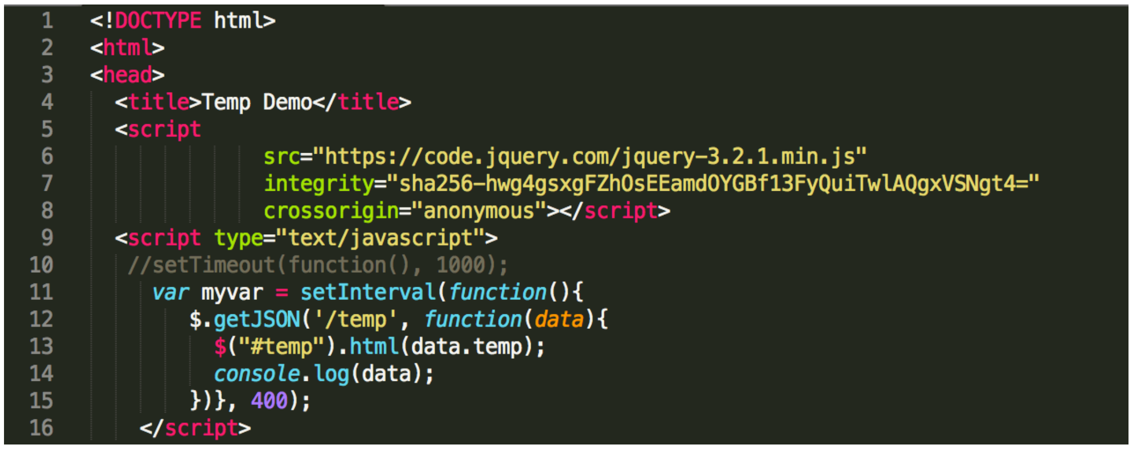

```
<!DOCTYPE html>
<html>
<head>
  <title>Temp Demo</title>
  <script
            src="https://code.jquery.com/jquery-3.2.1.min.js"
            integrity="sha256-hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4="
            crossorigin="anonymous"></script>
  <script type="text/javascript">
   //setTimeout(function(), 1000);
     var myvar = setInterval(function(){
        $.getJSON('/temp', function(data){
          $("#temp").html(data.temp);
          console.log(data);
        })}, 400);
    </script>
```

Figure 37. JSON embedded in HTML Webpage script

V. AUGMENTED REALITY

Overview

AR is a user experience through immersive headset technology such as Microsoft HoloLens [2], were the real-world objects are over-laid by computer generated 3D objects [2]. The major value of the AR experience is to visualize digital data in a user's perception of the real world [20]. Most current trend of use is in the area of knowledge sharing, education, data visualization etc.,[3][20][22]. However there are many more advancements yet to be seen in this field. The difference between VR, AR and XR can be seen in Figure 38.

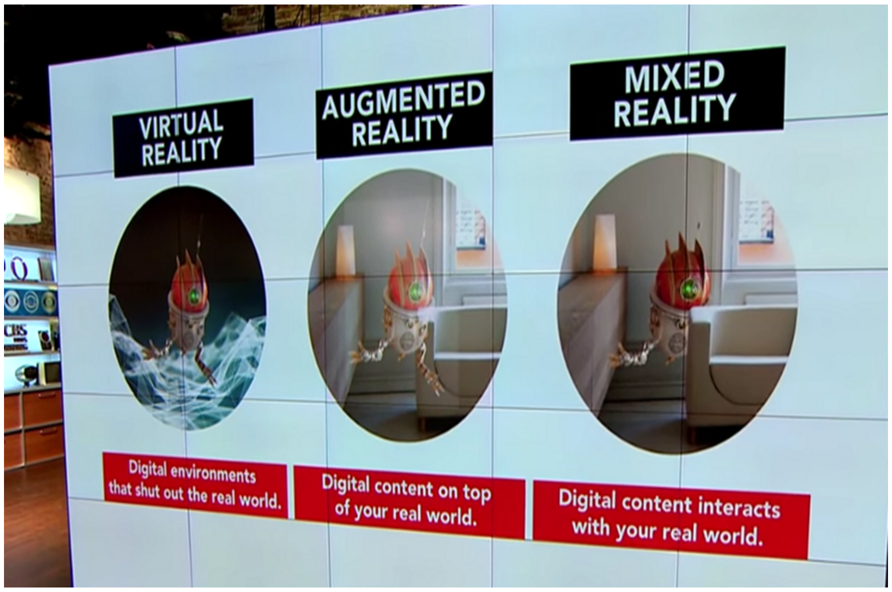

VIRTUAL REALITY
AUGMENTED REALITY
MIXED REALITY
Digital environments that shut out the real world.
Digital content on top of your real world.
Digital content interacts with your real world.

Figure 38. Digital Reality [39]

The holographic applications are made using game development software such as Unity 3D, Unreal engine, etc. [2]. The user can make use of AR development tools such as ARCore, ARKit, Vuforia etc. [40][41[42], which integrate smoothly into Unity 3D to help create AR applications with ease.

The recent deployment of Microsoft HoloLens in the market has given ample room for developers to conduct research to find new possibilities. This device can overlay digital objects onto real-world object and also stay in sync while the user interactions are translated to realize their actions in the digital world [21]. Thus, creating a bridge between the real and digital world.

Holographic Application Design

The main objective of this proposed thesis is to integrate the IoT with the XR experience to visualize the data from the IoT in real-time. The Figure 39 elaborates the architectural block of the AR application where it clearly shows that the UI is the major part which helps facilitate the integration.

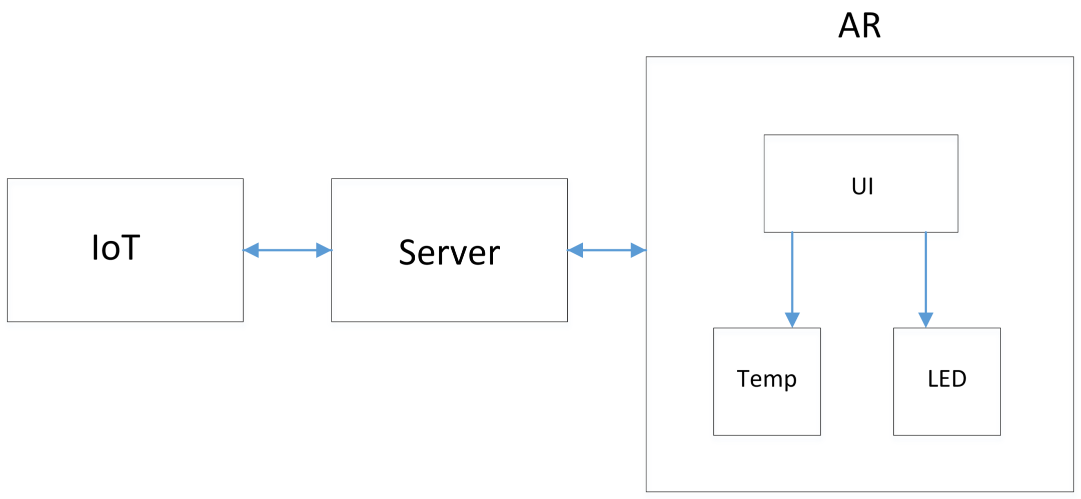
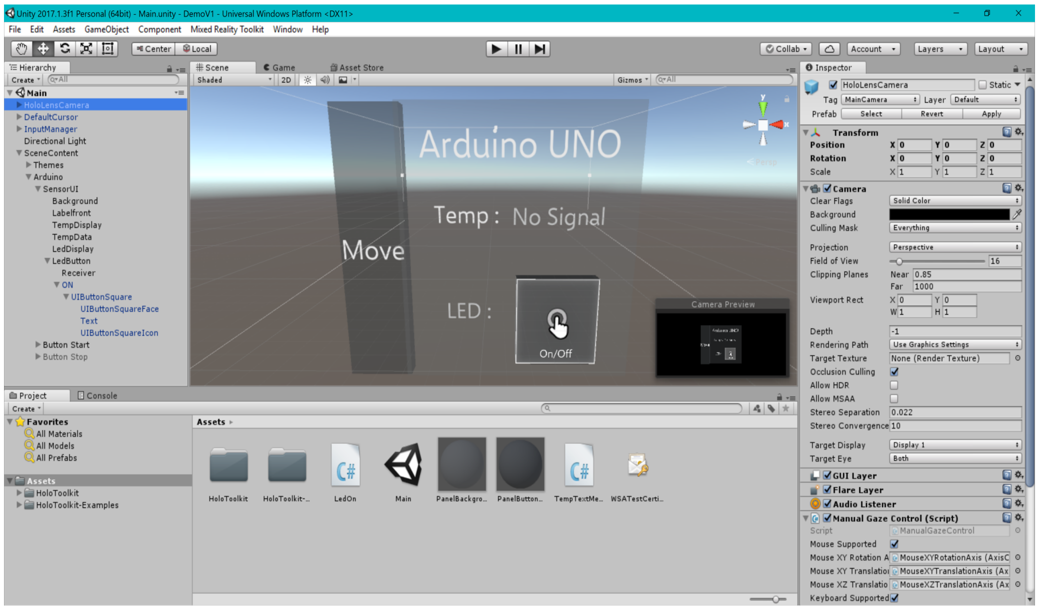


Figure 39. AR Block Diagram

The IoT data is visualized through Microsoft HoloLens which enables Augmented Reality to project data in 3D space. The preview of the design from the Unity 3D software can be seen in Figure 40.



HoloLensCamera ☐ Static
Tag MainCamera Layer Default
Prefab Select Revert Apply

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

✓ Camera
Clear Flags Solid Color
Background ■
Culling Mask Everything
Projection Perspective
Field of View 16
Clipping Planes Near 0.85 Far 1000
Viewport Rect X 0 Y 0 W 1 H 1
Depth -1
Rendering Path Use Graphics Settings
Target Texture None (Render Texture)
Occlusion Culling ✓
Allow HDR ☐
Allow MSAA ☐
Stereo Separation 0.022
Stereo Convergence 10
Target Display Display 1
Target Eye Both

✓ GUI Layer
✓ Flare Layer
✓ Audio Listener
✓ Manual Gaze Control (Script)
Script ManualGazeControl
Mouse Supported ✓
Mouse XY Rotation A MouseXYRotationAxis (AxisC
Mouse XY Translatio MouseXYTranslationAxis (Ax
Mouse XZ Translatio MouseXZTranslationAxis (Ax
Keyboard Supported ✓

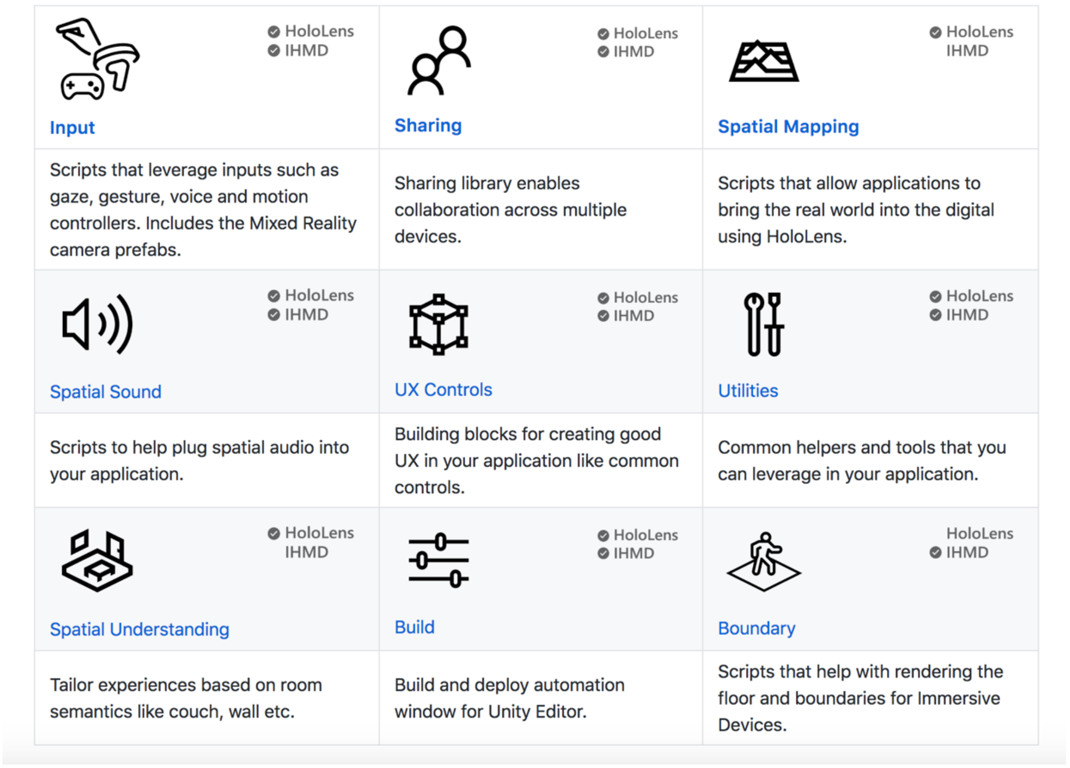
Figure 40. Unity 3D Version 2017.1.3 Preview

Unity 3D is a software which can be executed on different computing platforms and it is designed for the development of video games [22]. Basically, it is a game engine which provides essential tools for developers, enabling them for rapid deployment.

Development Tools

It is to help AR users to access and interact with the IoT device. The created AR experience should synchronize with the real-world object. Therefore, we make use of the development tools made for HoloLens by Microsoft [2][41]. The development tools work as a plug and play when added to the Unity project to create the desired game view.

Mixed Reality toolkit from Microsoft is required to setup the initially settings for the Unity project. Which contains pre-made tools such as Holographic buttons, application templates and basic scripts involved in general UI design [41]. The features available from the Mixed Reality tools can be seen in Figure 41.



✓ HoloLens ✓ IHMD **Input**	✓ HoloLens ✓ IHMD **Sharing**	✓ HoloLens IHMD **Spatial Mapping**
Scripts that leverage inputs such as gaze, gesture, voice and motion controllers. Includes the Mixed Reality camera prefabs.	Sharing library enables collaboration across multiple devices.	Scripts that allow applications to bring the real world into the digital using HoloLens.
✓ HoloLens ✓ IHMD Spatial Sound	✓ HoloLens ✓ IHMD UX Controls	✓ HoloLens ✓ IHMD Utilities
Scripts to help plug spatial audio into your application.	Building blocks for creating good UX in your application like common controls.	Common helpers and tools that you can leverage in your application.
✓ HoloLens IHMD Spatial Understanding	✓ HoloLens ✓ IHMD Build	HoloLens ✓ IHMD Boundary
Tailor experiences based on room semantics like couch, wall etc.	Build and deploy automation window for Unity Editor.	Scripts that help with rendering the floor and boundaries for Immersive Devices.

Figure 41. Mixed Reality Toolkit [41]

To help overlay the digital object over the IoT device, we make use of Vuforia augmented reality tools. Vuforia tools work in sync with the AR device's in-built visionary sensors to align the digital object with the real-world object [3][43]. There are mainly two types of object recognition, they are

- Marker design
- Marker less design

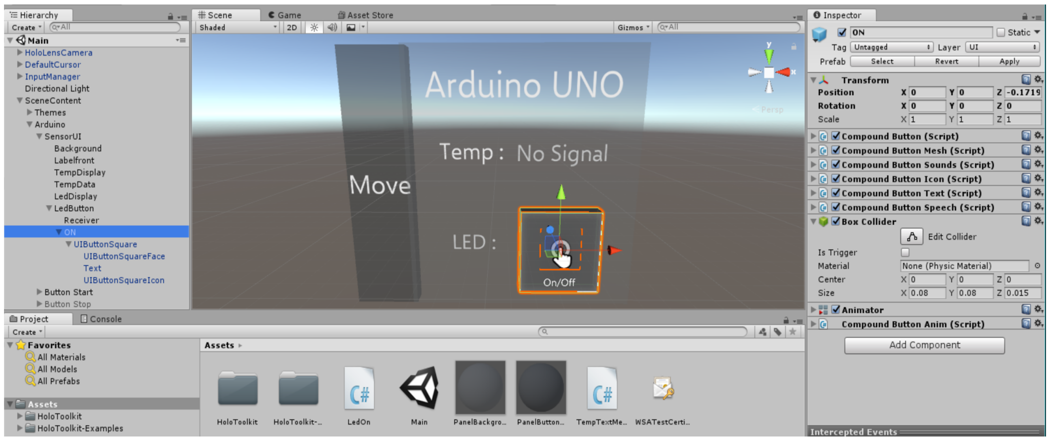
In the Marker design, a pre-processed 2D image is loaded into the build files of the holographic application. The Vuforia tools help process a selected image as a marker and when detected through the camera it projects the digital object over the physical image in the real-world [3][43].

In the Marker less design, the 3D object which is to be recognized is scanned step by step using a Vuforia mobile application. The scanning process of the 3D object is set in bright environment and have to follow precise instructions from the Vuforia scanning guidelines [43]. The scanned file is then uploaded on to the Vuforia dev portal, where the scanned file is processed with Vuforia technology and returned as a Unity Asset which can then be added to the Unity project.

The proposed Unity project has been built with no Markers. The use of Vuforia to recognize the device is still widely researched and are not stable [43]. Though the marker less design has slight advantage over the marker less design, the detection of the object is sometimes lost due to lighting conditions and appearance of the object itself [3]. Whereas, the marker design can be selected with high focus points, such as whites, greys, and black, which are major detection points of Vuforia's recognition algorithm [3].

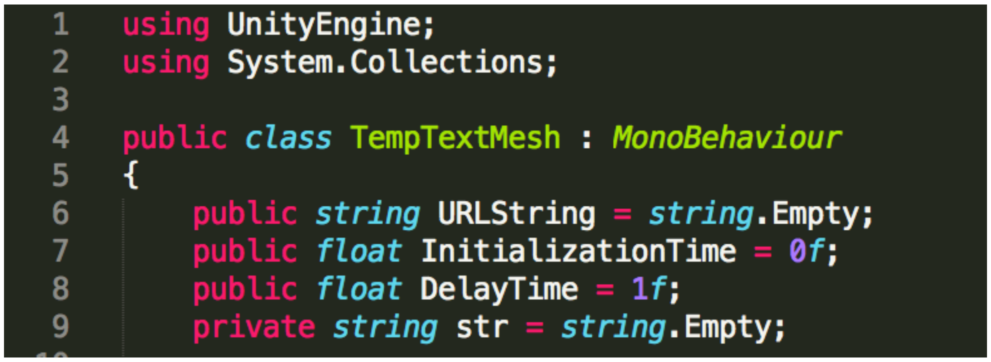
User Interface Scripts

Every object designed in the Unity using mixed reality tools has pre-built scripts which contains the codes for its operations. For example, if a holographic button is added to the project for on/off, to assign the on/off function to it, one must edit its pre-built scripts to match the required execution [22]. These are the backend scripts which enables or performs the operations assigned to its counterpart UI. In the Figure 42, we can see the UI designed to show the information received from the IoT device. The "Temp:" displays the temperature value beside it. The Holographic button is programed to turn on/off the LED hosted on the IoT device. The scripts which performs the data retrieval from the IoT is explained in detail below.



```
ON                                   Static
Tag Untagged      Layer UI
Prefab  Select  Revert  Apply
Transform
Position   X 0   Y 0   Z -0.1719
Rotation   X 0   Y 0   Z 0
Scale      X 1   Y 1   Z 1
Compound Button (Script)
Compound Button Mesh (Script)
Compound Button Sounds (Script)
Compound Button Icon (Script)
Compound Button Text (Script)
Compound Button Speech (Script)
Box Collider
           Edit Collider
Is Trigger
Material   None (Physic Material)
Center     X 0      Y 0      Z 0
Size       X 0.08   Y 0.08   Z 0.015
Animator
Compound Button Anim (Script)
```

Figure 42. UI Design with TextMesh and Holographic Panel



```
using UnityEngine;
using System.Collections;

public class TempTextMesh : MonoBehaviour
{
    public string URLString = string.Empty;
    public float InitializationTime = 0f;
    public float DelayTime = 1f;
    private string str = string.Empty;
```

Figure 43. Unity script Data Initialization

The Figure 43 shows the initialization of functions and variables involved in the scripts. The function "*GetDataFromWeb*" Figure 45, is invoked at the beginning of the application and the UI panel is designed to be manually moved by the user and align over the physical IoT device. Retrieving data from the server is done by performing a GET request to the webpage hosted but the IoT device server [44]. The request to the webpage will return a string value, which is then processed and only the required information is displayed in the UI. The output of the GET request can be seen in Figure 46.

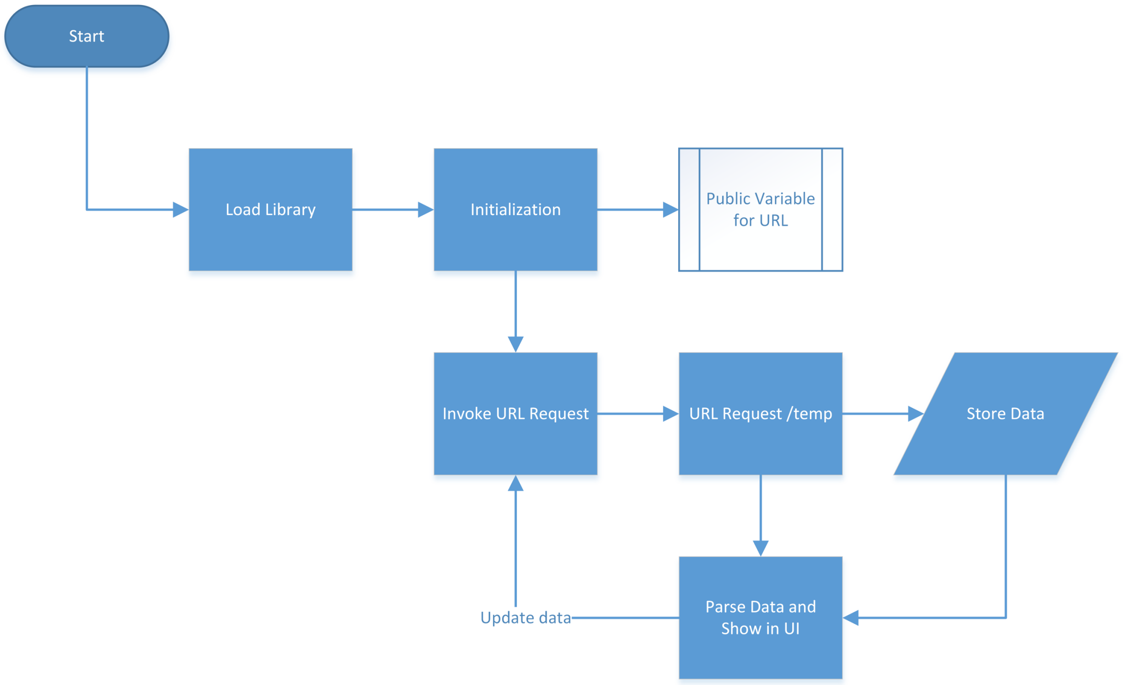
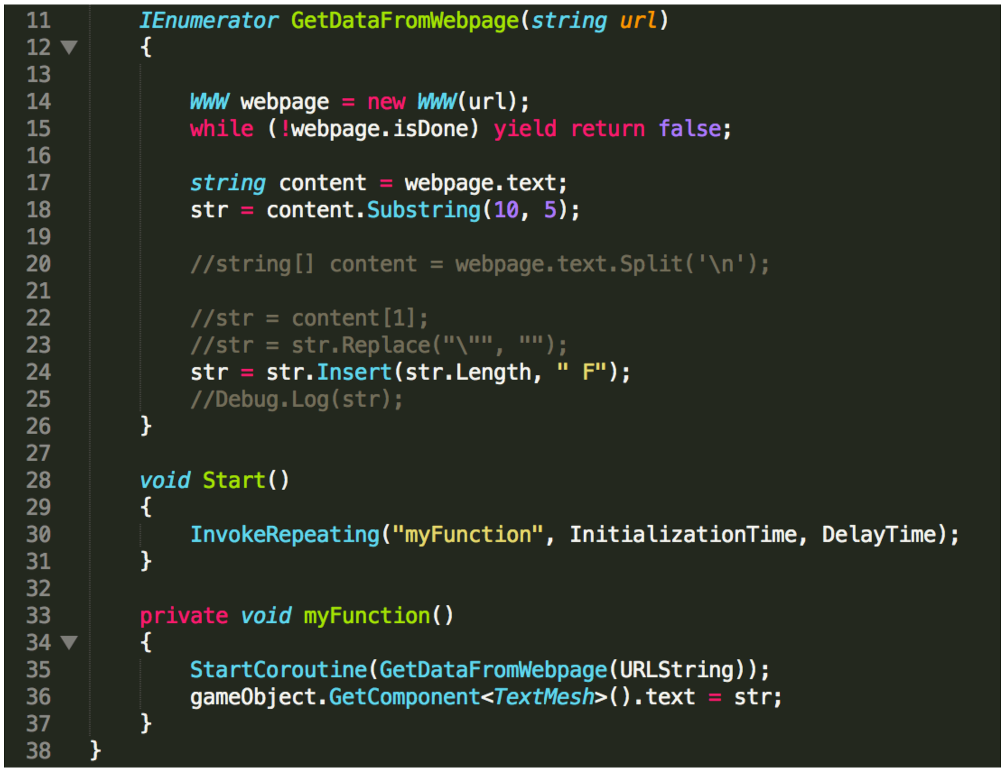


Figure 44. TextMesh TempRead Function flowchart


```
    IEnumerator GetDataFromWebpage(string url)
    {

        WWW webpage = new WWW(url);
        while (!webpage.isDone) yield return false;

        string content = webpage.text;
        str = content.Substring(10, 5);

        //string[] content = webpage.text.Split('\n');

        //str = content[1];
        //str = str.Replace("\"", "");
        str = str.Insert(str.Length, " F");
        //Debug.Log(str);
    }

    void Start()
    {
        InvokeRepeating("myFunction", InitializationTime, DelayTime);
    }

    private void myFunction()
    {
        StartCoroutine(GetDataFromWebpage(URLString));
        gameObject.GetComponent<TextMesh>().text = str;
    }
}
```

Figure 45. TextMesh TempRead Function script

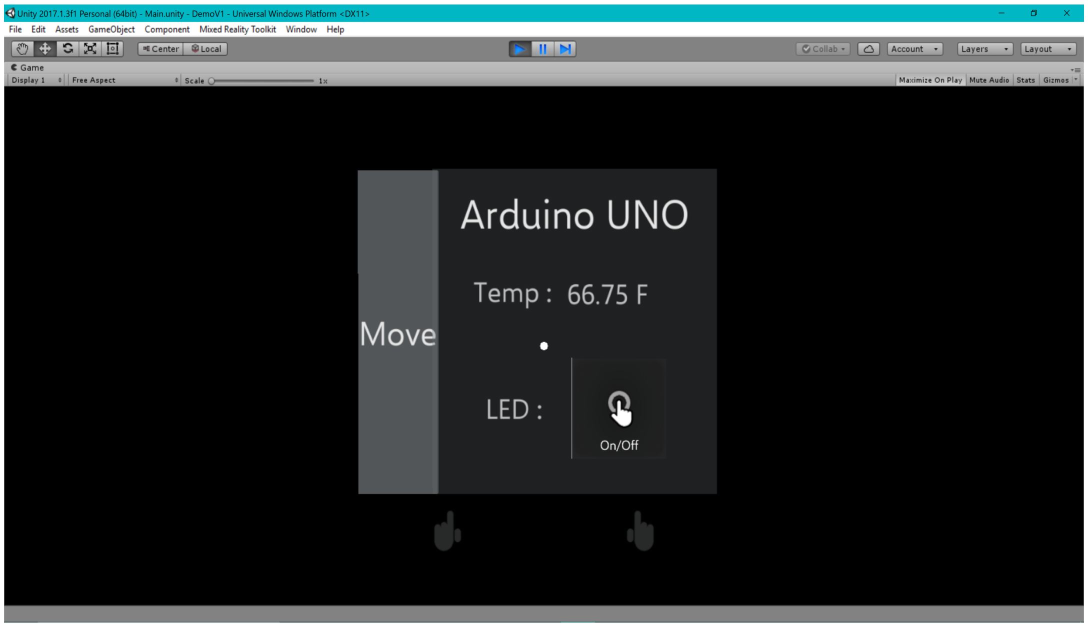
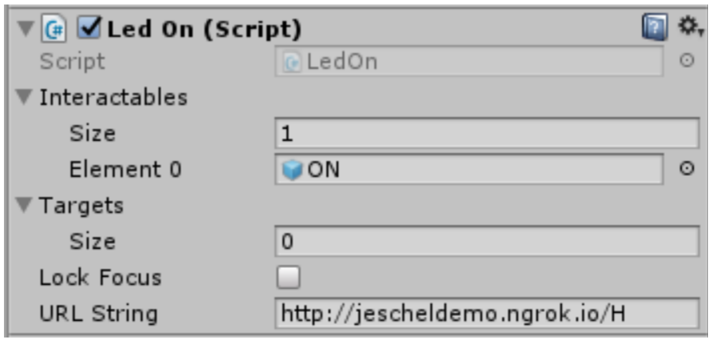


Figure 46. Rendered UI with IoT Data

The script below performs in the same manner as the temperature data retrieval script. However, there are few more libraries which are involved in the Holographic button operations. The operation of the holographic button is linked with the HoloLens pointer. Therefore, when interacted with the button, it executes two functions, one to make the click interaction and the other to make a URL GET request to the LED function on the IoT hosting server [44]. The URL address required for obtaining the data is stored is a public variable Figure 47. Thus, making the update of URL address in the backend easier on the unity 3D editor.



Led On (Script)

Script	LedOn
Interactables	
Size	1
Element 0	ON
Targets	
Size	0
Lock Focus	☐
URL String	http://jescheldemo.ngrok.io/H

Figure 47. URL feed for LED Control

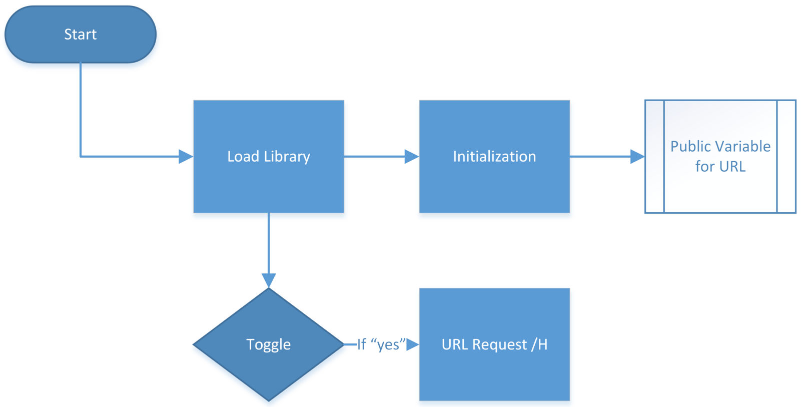
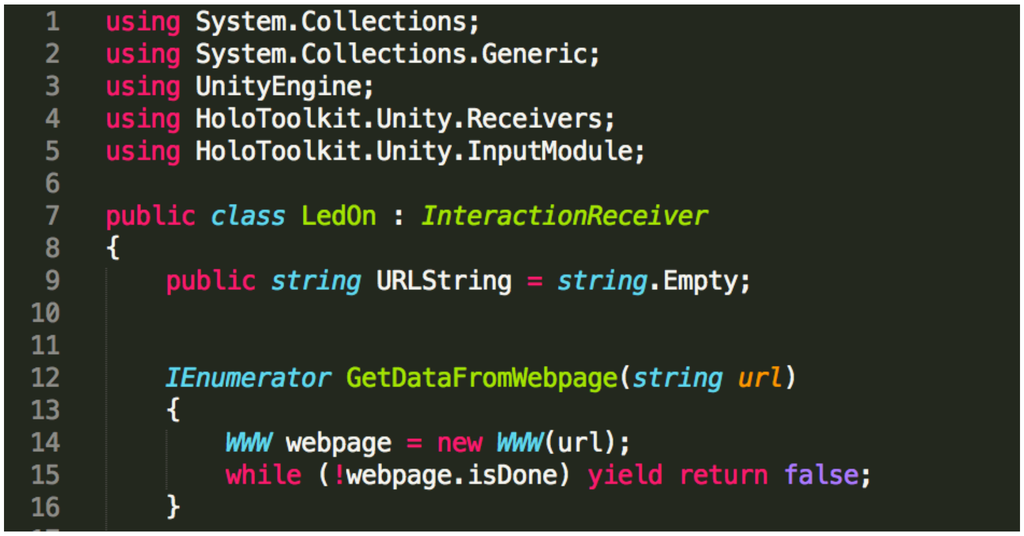


Figure 48. LED Control flowchart



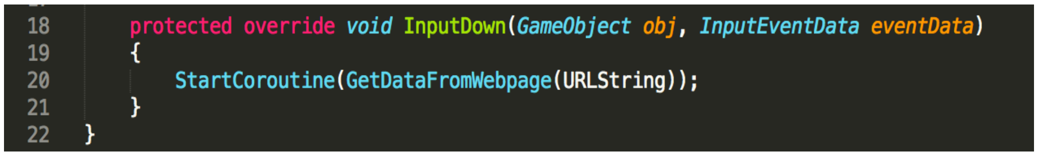
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using HoloToolkit.Unity.Receivers;
using HoloToolkit.Unity.InputModule;

public class LedOn : InteractionReceiver
{
    public string URLString = string.Empty;


    IEnumerator GetDataFromWebpage(string url)
    {
        WWW webpage = new WWW(url);
        while (!webpage.isDone) yield return false;
    }
```

Figure 49. LED Control script

The function below is the action script which will be activated by the click gesture on the HoloLens pointer. When activated the function "*GetDataFromWebpage*" Figure 49, is invoked and the URL call is made to perform the on/off of the LED hosted on the IoT device.



```
    protected override void InputDown(GameObject obj, InputEventData eventData)
    {
        StartCoroutine(GetDataFromWebpage(URLString));
    }
}
```

Figure 50. Holographic Button Interface

VI. SYSTEM ARCHITECTURE

Overview

The following presents the complete conceptual model of the IoT, Server, AR integrated together to show the final structure and pattern of the proposed system shown in Figure 51.

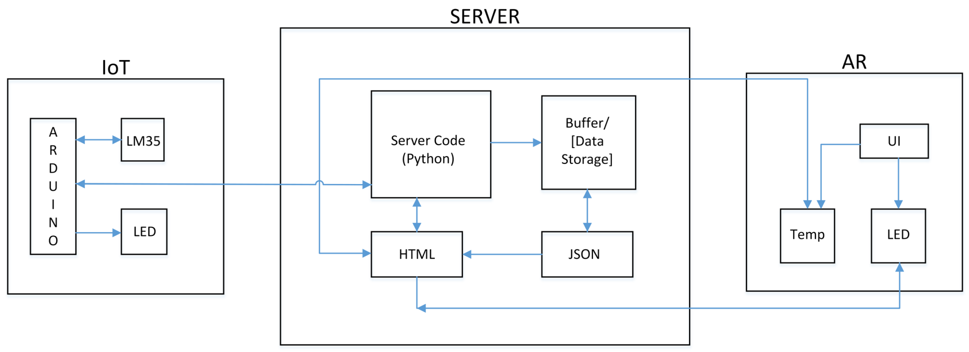


Figure 51. Block diagram of the Proposed System Architecture

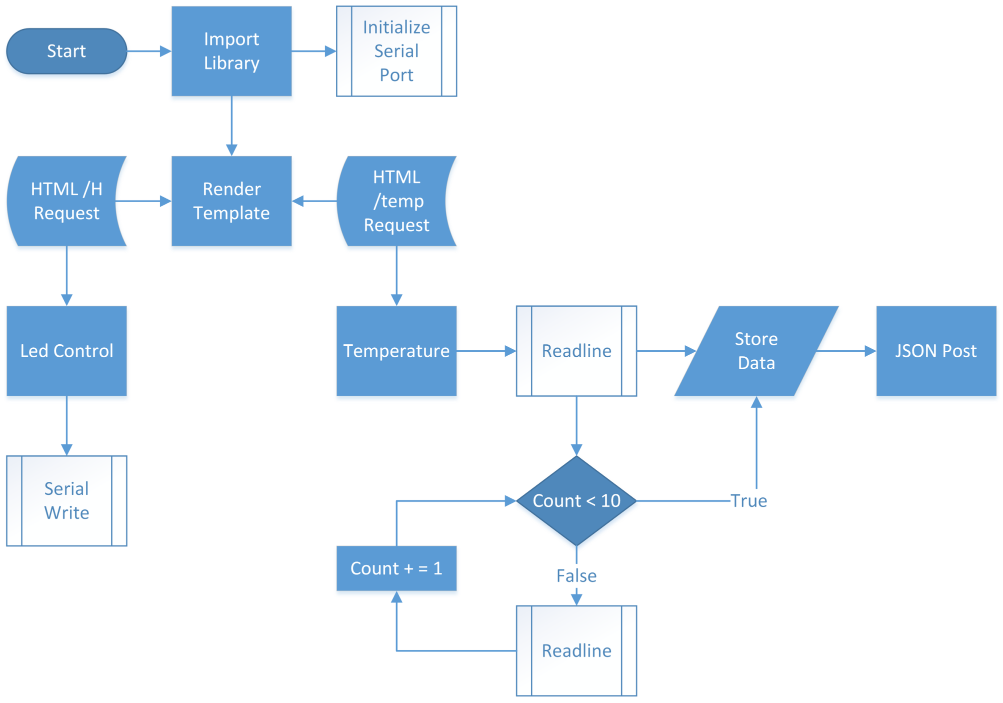


Figure 52. Complete flowchart of the Server Script

IoT Device Test

The Arduino script is uploaded to the Arduino Uno and manually tested if they respond with the desired result. The below flowchart shows the functions which need to be processed. The functions which are to be tested is as follows,

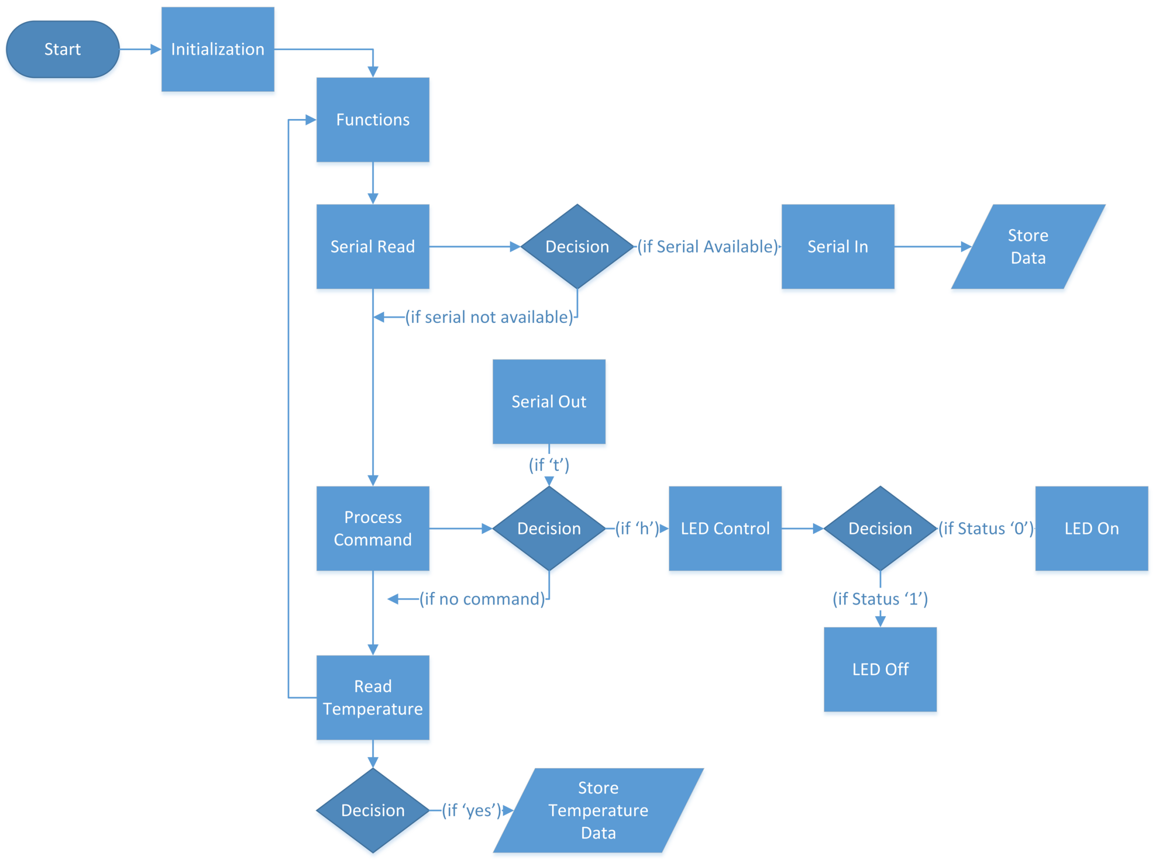
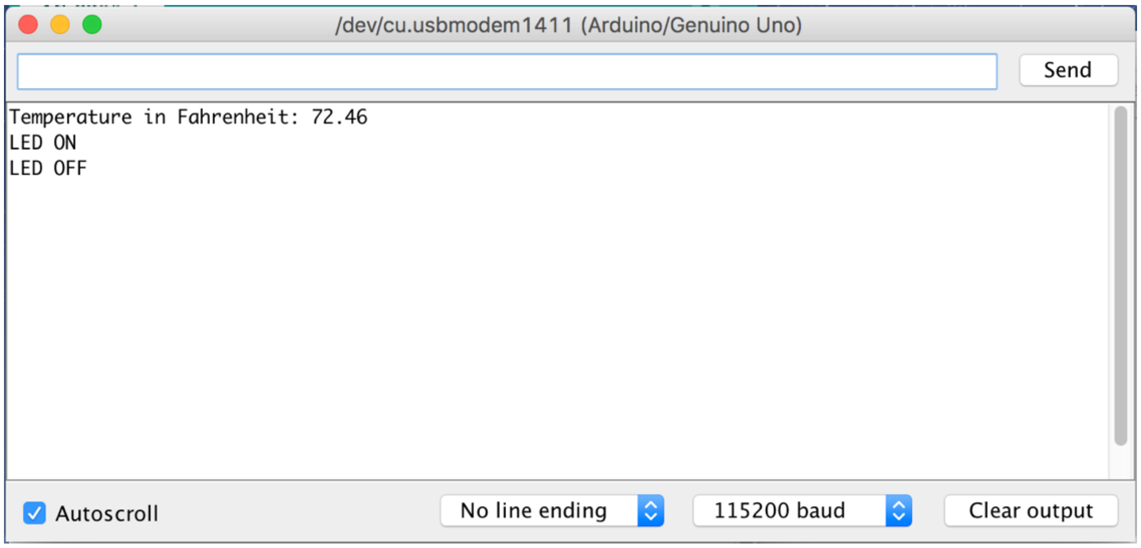


Figure 53. Complete flowchart of the Arduino script



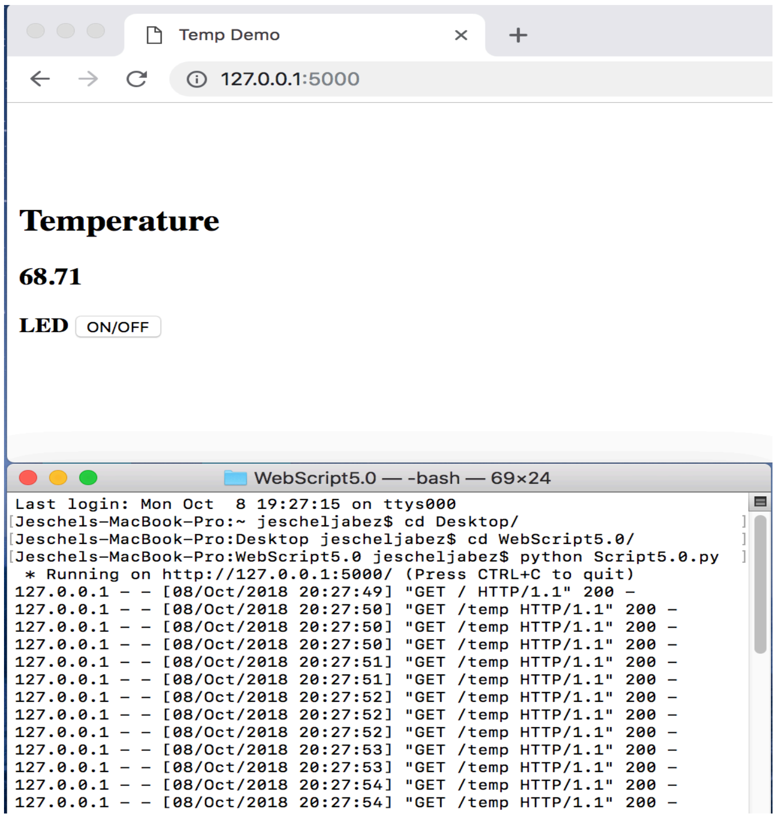
```
Temperature in Fahrenheit: 72.46
LED ON
LED OFF
```

Figure 54. Serial Output from the IoT Device

The tests are executed successfully. The embedded program loaded on to the Arduino has performed as expected. The serial output of the Arduino can be seen in Figure 54.

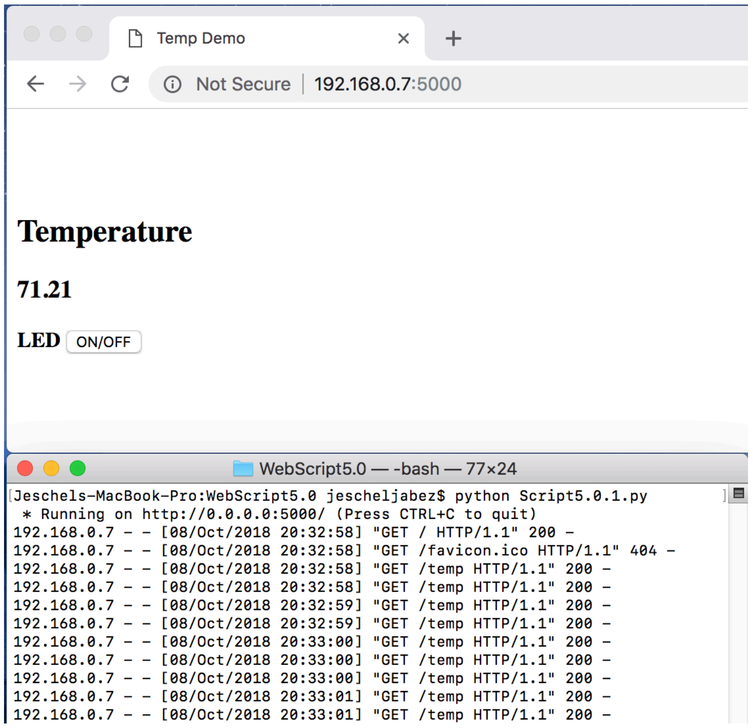
Server Operation Test

The Flask program in executed in the system through terminal window commands. Initially the server is set to run and host data in the localhost of the computer. Next the server is set to host data onto the edge of the network by running on the computer's local IP. At first the server script is set to run in localhost and observed for any errors and then set to run in local IP of the computer (edge of the network) [44]. The output of both the test can be seen in Figure 55 and 56 respectively.



```
Last login: Mon Oct  8 19:27:15 on ttys000
Jeschels-MacBook-Pro:~ jescheljabez$ cd Desktop/
Jeschels-MacBook-Pro:Desktop jescheljabez$ cd WebScript5.0/
Jeschels-MacBook-Pro:WebScript5.0 jescheljabez$ python Script5.0.py
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [08/Oct/2018 20:27:49] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 20:27:50] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 20:27:50] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 20:27:50] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 20:27:51] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 20:27:51] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 20:27:52] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 20:27:52] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 20:27:52] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 20:27:53] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 20:27:53] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 20:27:54] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 20:27:54] "GET /temp HTTP/1.1" 200 -
```

Figure 55. Server running in Localhost.



```
Jeschels-MacBook-Pro:WebScript5.0 jescheljabez$ python Script5.0.1.py
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
192.168.0.7 - - [08/Oct/2018 20:32:58] "GET / HTTP/1.1" 200 -
192.168.0.7 - - [08/Oct/2018 20:32:58] "GET /favicon.ico HTTP/1.1" 404 -
192.168.0.7 - - [08/Oct/2018 20:32:58] "GET /temp HTTP/1.1" 200 -
192.168.0.7 - - [08/Oct/2018 20:32:58] "GET /temp HTTP/1.1" 200 -
192.168.0.7 - - [08/Oct/2018 20:32:59] "GET /temp HTTP/1.1" 200 -
192.168.0.7 - - [08/Oct/2018 20:32:59] "GET /temp HTTP/1.1" 200 -
192.168.0.7 - - [08/Oct/2018 20:33:00] "GET /temp HTTP/1.1" 200 -
192.168.0.7 - - [08/Oct/2018 20:33:00] "GET /temp HTTP/1.1" 200 -
192.168.0.7 - - [08/Oct/2018 20:33:00] "GET /temp HTTP/1.1" 200 -
192.168.0.7 - - [08/Oct/2018 20:33:01] "GET /temp HTTP/1.1" 200 -
192.168.0.7 - - [08/Oct/2018 20:33:01] "GET /temp HTTP/1.1" 200 -
```

Figure 56. Server running in Network Edge.

Holographic Application Test

The unity project is compiled and built for the Windows platform which is the platform for Microsoft HoloLens. The application is started in the HoloLens and observed for the desired result. The Figure 57 shows the image taken from the HoloLens display.

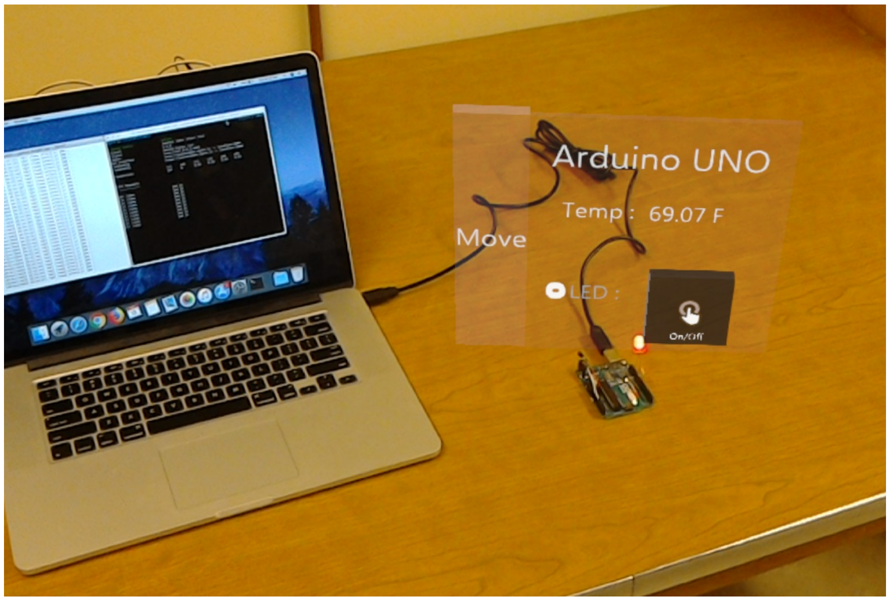


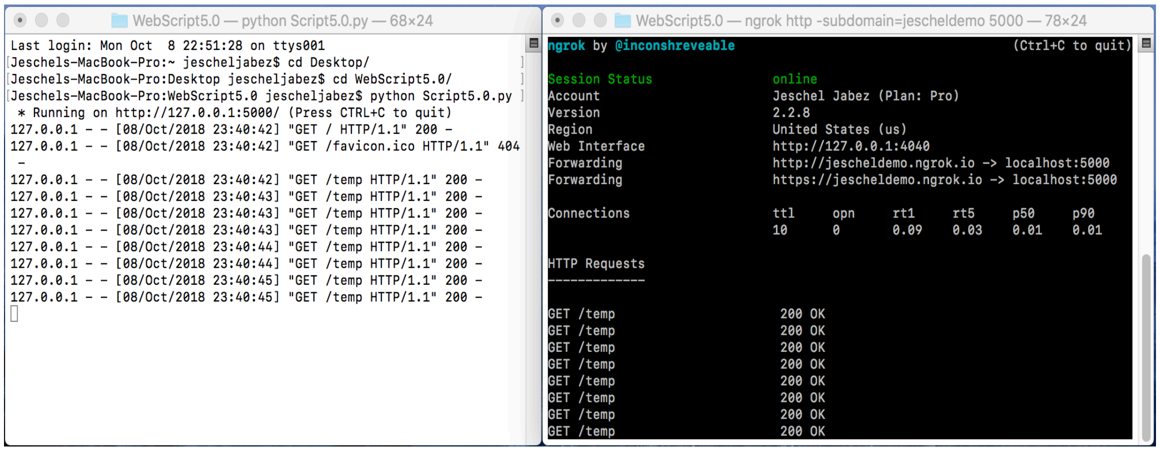
Figure 57. Mixed Reality Visualization

VII. RESULTS

Ngrok Tunnels

It is a third-party tunnel broker [45]. It helps expose local server running in a computer to the internet through secure tunnels [45]. It is lightweight executable script designed to take the port value of the running server as input and tunnels it through to a ngrok public server and expose it to the internet [46]. It is very useful feature where we can deploy web sites for demo purposes and for testing server-client scripts under development.

The Ngrok also provides paid plans, with which a user can reserve a subdomain for public URL usage. This feature comes in handy for testing the proposed server for Latency and also feed the hosted URL to HoloLens application for testing demo purposes. The Figure 58 shows the server running in the Localhost (left) and Ngrok Tunnel (right) with the reserved subdomain in use for retrieving data from IoT. The command line instructions are "*ngrok http -subdomain=jeschelxr 5000*" subdomain name is jeschelxr and port number is 5000.


WebScript5.0 — python Script5.0.py — 68×24
WebScript5.0 — ngrok http -subdomain=jescheldemo 5000 — 78×24

```
Last login: Mon Oct  8 22:51:28 on ttys001
Jeschels-MacBook-Pro:~ jescheljabez$ cd Desktop/
Jeschels-MacBook-Pro:Desktop jescheljabez$ cd WebScript5.0/
Jeschels-MacBook-Pro:WebScript5.0 jescheljabez$ python Script5.0.py
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [08/Oct/2018 23:40:42] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 23:40:42] "GET /favicon.ico HTTP/1.1" 404
 -
127.0.0.1 - - [08/Oct/2018 23:40:42] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 23:40:43] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 23:40:43] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 23:40:43] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 23:40:44] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 23:40:44] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 23:40:45] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [08/Oct/2018 23:40:45] "GET /temp HTTP/1.1" 200 -
```

```
ngrok by @inconshreveable                                     (Ctrl+C to quit)

Session Status                online
Account                       Jeschel Jabez (Plan: Pro)
Version                       2.2.8
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://jescheldemo.ngrok.io -> localhost:5000
Forwarding                    https://jescheldemo.ngrok.io -> localhost:5000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              10      0       0.09    0.03    0.01    0.01

HTTP Requests
-------------

GET /temp                      200 OK
GET /temp                      200 OK
GET /temp                      200 OK
GET /temp                      200 OK
GET /temp                      200 OK
GET /temp                      200 OK
GET /temp                      200 OK
GET /temp                      200 OK
```

Figure 58. Ngrok Tunnel

MQTT Broker

The latency of the proposed server is compared with the commonly used IoT protocol called MQTT. Message Queue Telemetry Transport (MQTT) is a type of protocol based on subscribe and publish lightweight communication protocol. It makes use of a middle server which handles the interface between subscriber and publisher [47]. In short, the server is a cloud-based system where a user can program the IoT device to use a log in credentials to publish data to the server and use the same to retrieve data in any device. The server model shown in the Figure 59 uses the Adafruit IO Broker [47].

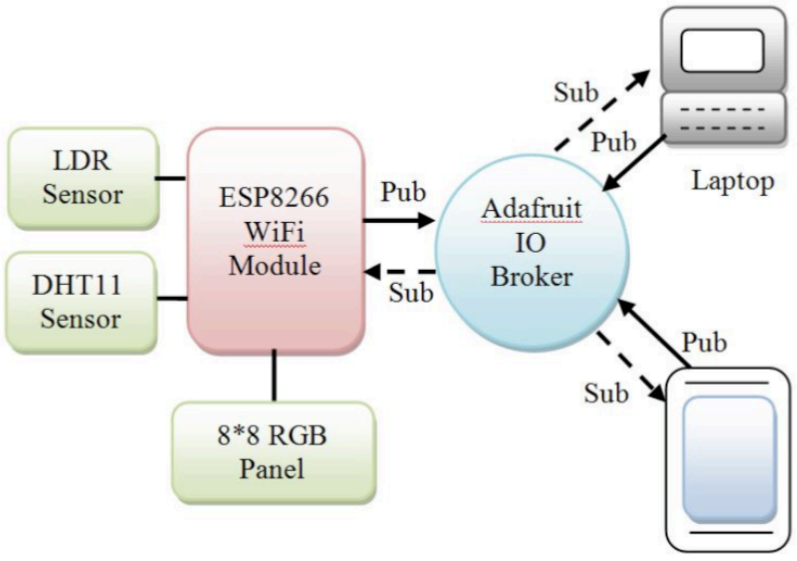


Figure 59. MQTT Broker Model [47]

After performing a research survey, it is found that the MQTT uses the similar configuration as the proposed server [47]. However, a dedicated database is established to store and retrieve data for users from which the data is obtained by RESTful method [47]. This makes it ideal for comparison, and so a similar setup is designed and run through ping test with same Ngrok credentials as seen in Figure 58.

A Test server [48] Figure 60, designed with database logging is selected to compare and evaluate the working of the proposed server.

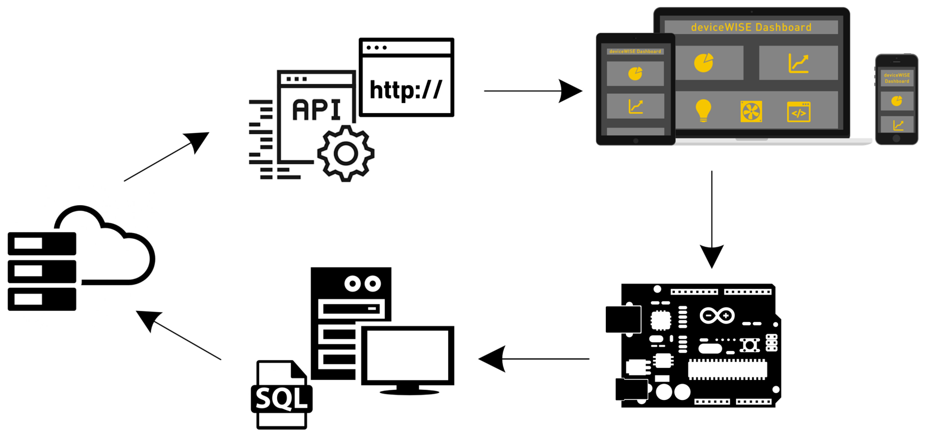
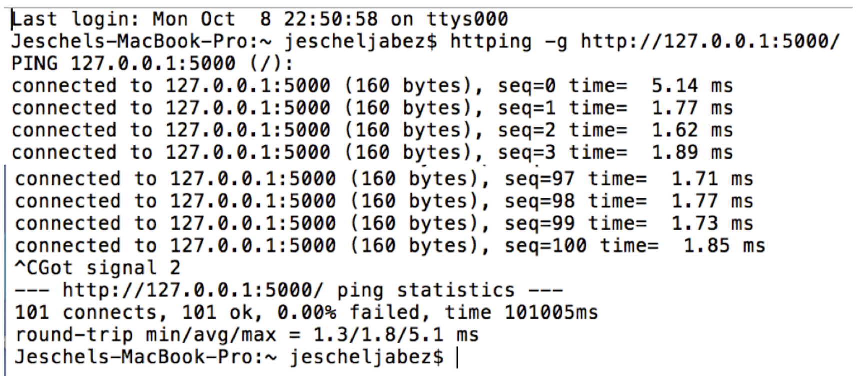


Figure 60. Test Server Architecture

Latency Evaluation

The time taken for data to travel from one endpoint to another over the internet protocol is called Network Delay. The major factors involved in a network delay are, Transmission Delay which is the time taken to upload the data packet on the link. Propagation Delay which is the time taken for the data to reach its destination (physically). Queuing Delay which is the time spend by the data packet in queue during node hop. Processing Delay is the time taken by the processor to process the data packet. Most Research point out that Queuing causes the major delay [49].

The mode of test is done using **ping**. It is a computer network utility used to analyze the reachability of a host over an Internet Protocol network [50]. Ping sends an Internet Control Message Protocol (ICMP) echo request packets to the required host and waits for a response [50]. In this evaluation test, a ping network utility program called **httping** is used [51]. It is an open source network utility freeware which works on the Ping methodology [51]. In operation during test, the program measures the time taken to connect to the given server, and also, measure time taken for response when a request is sent to the server.

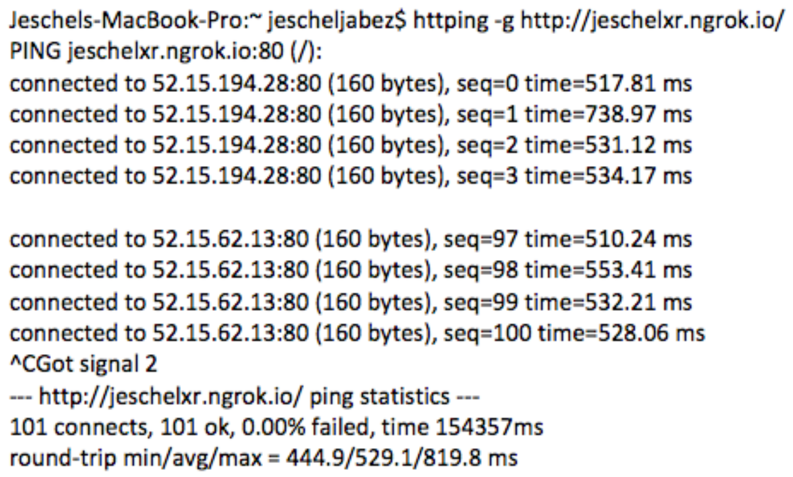


```
Last login: Mon Oct  8 22:50:58 on ttys000
Jeschels-MacBook-Pro:~ jescheljabez$ httping -g http://127.0.0.1:5000/
PING 127.0.0.1:5000 (/):
connected to 127.0.0.1:5000 (160 bytes), seq=0 time=  5.14 ms
connected to 127.0.0.1:5000 (160 bytes), seq=1 time=  1.77 ms
connected to 127.0.0.1:5000 (160 bytes), seq=2 time=  1.62 ms
connected to 127.0.0.1:5000 (160 bytes), seq=3 time=  1.89 ms
connected to 127.0.0.1:5000 (160 bytes), seq=97 time=  1.71 ms
connected to 127.0.0.1:5000 (160 bytes), seq=98 time=  1.77 ms
connected to 127.0.0.1:5000 (160 bytes), seq=99 time=  1.73 ms
connected to 127.0.0.1:5000 (160 bytes), seq=100 time=  1.85 ms
^CGot signal 2
--- http://127.0.0.1:5000/ ping statistics ---
101 connects, 101 ok, 0.00% failed, time 101005ms
round-trip min/avg/max = 1.3/1.8/5.1 ms
Jeschels-MacBook-Pro:~ jescheljabez$ |
```

Figure 61. Httping Result of Proposed Server running in Localhost

The packet size is determined by the program by default using the Internet control message protocol (ICMP) [52]. ICMP adds error corrections data to the datagram's (it is the data packet sent to the server during ping) source address if TTL (time to live) is elapsed to attempt on the link [52]. The datagram size also varies from IPv4 to IPv6 and also include various IP fragments such as TCP, UDP, and IGMP etc. [52]. The Internet Engineering Task Force introduces new protocols from time to time which also affects network delays when running through open source freeware [52]. If root of network latency has to be determined the current research would dive deeper into micro technical details [33][49][52].

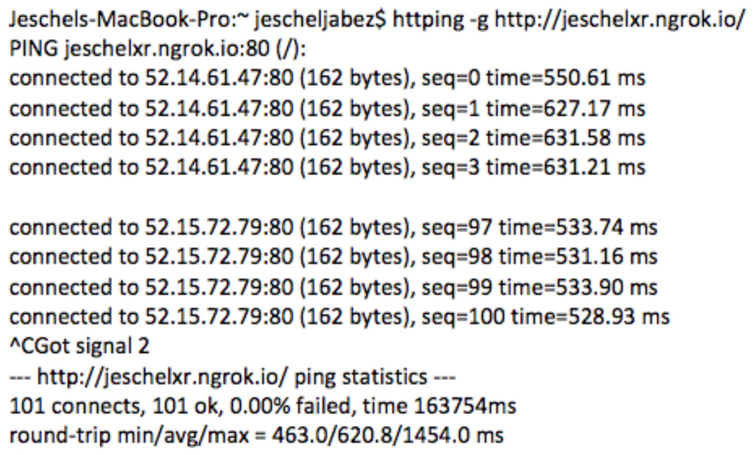
The results of the Ping test can be seen in the following Figures 61,62,63,63,65 with proposed server running on localhost, over the internet and the test server over the internet through Ngrok tunnel and also over the network edge to show case the latency estimation of the Fog computing concepts respectively.



```
Jeschels-MacBook-Pro:~ jescheljabez$ httping -g http://jeschelxr.ngrok.io/
PING jeschelxr.ngrok.io:80 (/):
connected to 52.15.194.28:80 (160 bytes), seq=0 time=517.81 ms
connected to 52.15.194.28:80 (160 bytes), seq=1 time=738.97 ms
connected to 52.15.194.28:80 (160 bytes), seq=2 time=531.12 ms
connected to 52.15.194.28:80 (160 bytes), seq=3 time=534.17 ms

connected to 52.15.62.13:80 (160 bytes), seq=97 time=510.24 ms
connected to 52.15.62.13:80 (160 bytes), seq=98 time=553.41 ms
connected to 52.15.62.13:80 (160 bytes), seq=99 time=532.21 ms
connected to 52.15.62.13:80 (160 bytes), seq=100 time=528.06 ms
^CGot signal 2
--- http://jeschelxr.ngrok.io/ ping statistics ---
101 connects, 101 ok, 0.00% failed, time 154357ms
round-trip min/avg/max = 444.9/529.1/819.8 ms
```

Figure 62. Httping Result of Proposed Server running over Internet

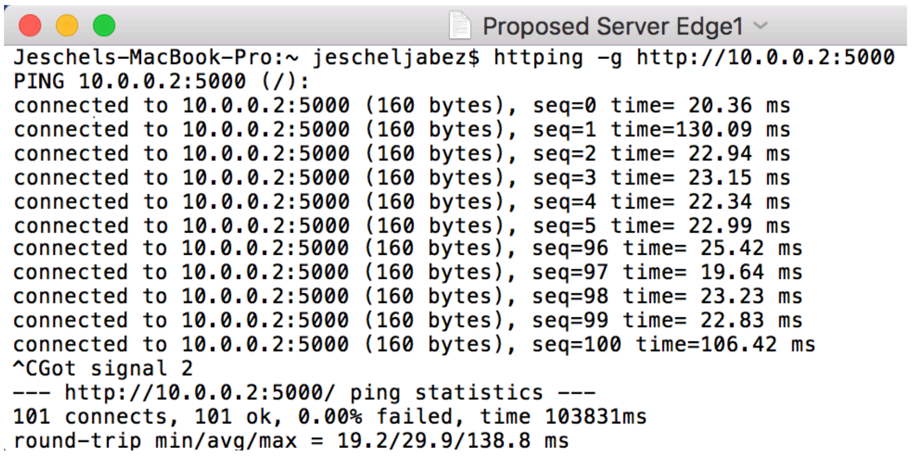


```
Jeschels-MacBook-Pro:~ jescheljabez$ httping -g http://jeschelxr.ngrok.io/
PING jeschelxr.ngrok.io:80 (/):
connected to 52.14.61.47:80 (162 bytes), seq=0 time=550.61 ms
connected to 52.14.61.47:80 (162 bytes), seq=1 time=627.17 ms
connected to 52.14.61.47:80 (162 bytes), seq=2 time=631.58 ms
connected to 52.14.61.47:80 (162 bytes), seq=3 time=631.21 ms

connected to 52.15.72.79:80 (162 bytes), seq=97 time=533.74 ms
connected to 52.15.72.79:80 (162 bytes), seq=98 time=531.16 ms
connected to 52.15.72.79:80 (162 bytes), seq=99 time=533.90 ms
connected to 52.15.72.79:80 (162 bytes), seq=100 time=528.93 ms
^CGot signal 2
--- http://jeschelxr.ngrok.io/ ping statistics ---
101 connects, 101 ok, 0.00% failed, time 163754ms
round-trip min/avg/max = 463.0/620.8/1454.0 ms
```

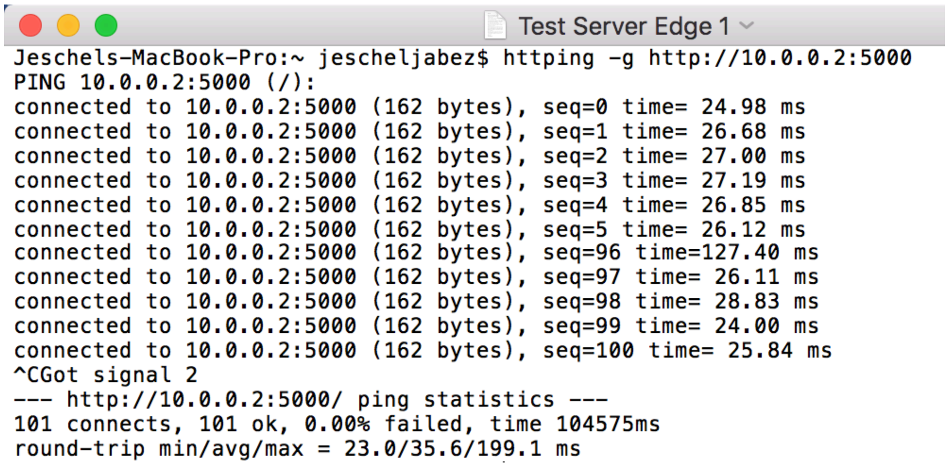
Figure 63. Httping Result of Test Server running over Internet

The above ping test was conducted with network bandwidth of 400Mbps download and upload, at the university research lab. The same setup is done for the network edge latency test Figure 64 and 65 showing the significant improvement in the latency. The ping test on proposed server running on the network edge with the same network configuration as the test performed on the servers over the internet using Ngrok tunneling.



```
Jeschels-MacBook-Pro:~ jescheljabez$ httping -g http://10.0.0.2:5000
PING 10.0.0.2:5000 (/):
connected to 10.0.0.2:5000 (160 bytes), seq=0 time= 20.36 ms
connected to 10.0.0.2:5000 (160 bytes), seq=1 time=130.09 ms
connected to 10.0.0.2:5000 (160 bytes), seq=2 time= 22.94 ms
connected to 10.0.0.2:5000 (160 bytes), seq=3 time= 23.15 ms
connected to 10.0.0.2:5000 (160 bytes), seq=4 time= 22.34 ms
connected to 10.0.0.2:5000 (160 bytes), seq=5 time= 22.99 ms
connected to 10.0.0.2:5000 (160 bytes), seq=96 time= 25.42 ms
connected to 10.0.0.2:5000 (160 bytes), seq=97 time= 19.64 ms
connected to 10.0.0.2:5000 (160 bytes), seq=98 time= 23.23 ms
connected to 10.0.0.2:5000 (160 bytes), seq=99 time= 22.83 ms
connected to 10.0.0.2:5000 (160 bytes), seq=100 time=106.42 ms
^CGot signal 2
--- http://10.0.0.2:5000/ ping statistics ---
101 connects, 101 ok, 0.00% failed, time 103831ms
round-trip min/avg/max = 19.2/29.9/138.8 ms
```

Figure 64. Httping Test on Proposed Server running on Network Edge



```
Jeschels-MacBook-Pro:~ jescheljabez$ httping -g http://10.0.0.2:5000
PING 10.0.0.2:5000 (/):
connected to 10.0.0.2:5000 (162 bytes), seq=0 time= 24.98 ms
connected to 10.0.0.2:5000 (162 bytes), seq=1 time= 26.68 ms
connected to 10.0.0.2:5000 (162 bytes), seq=2 time= 27.00 ms
connected to 10.0.0.2:5000 (162 bytes), seq=3 time= 27.19 ms
connected to 10.0.0.2:5000 (162 bytes), seq=4 time= 26.85 ms
connected to 10.0.0.2:5000 (162 bytes), seq=5 time= 26.12 ms
connected to 10.0.0.2:5000 (162 bytes), seq=96 time=127.40 ms
connected to 10.0.0.2:5000 (162 bytes), seq=97 time= 26.11 ms
connected to 10.0.0.2:5000 (162 bytes), seq=98 time= 28.83 ms
connected to 10.0.0.2:5000 (162 bytes), seq=99 time= 24.00 ms
connected to 10.0.0.2:5000 (162 bytes), seq=100 time= 25.84 ms
^CGot signal 2
--- http://10.0.0.2:5000/ ping statistics ---
101 connects, 101 ok, 0.00% failed, time 104575ms
round-trip min/avg/max = 23.0/35.6/199.1 ms
```

Figure 65. Httping Test on Test Server running on Network Edge

Graphical comparison

Now, after the httping test is successfully executed. The results are exported to an excel sheet and the following graphs Figure 66, 67, 68 are obtained.

Table 2. Latency Comparison over the Internet

Servers	Minimum Time(ms)	Average Time(ms)	Maximum Time(ms)
Proposed Server Latency in Internet	444.9	529.1	819.8
Test Server Latency in Internet	463.0	620.8	1454.0

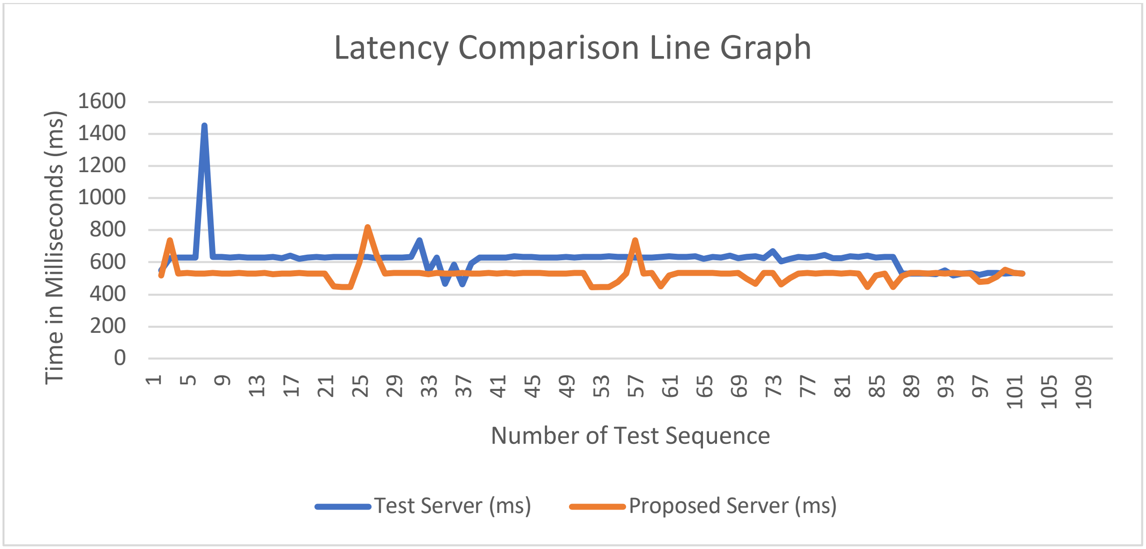


Figure 66. Latency Graph of the Servers running over the Internet

As seen from the above tests the Proposed Server has a round-trip latency of roughly 4% lower in minimum time, 14.7% decrease in average time and 43.65% less maximum time when compared to the Test Server. The same test is conducted on the server while running over the network edge figure 64, 65.

Table 3. Latency Comparison over the Network Edge			
Servers	Minimum Time(ms)	Average Time(ms)	Maximum Time(ms)
Proposed Server Latency	19.2	29.9	138.8
Test Server Latency	23.0	35.6	199.1

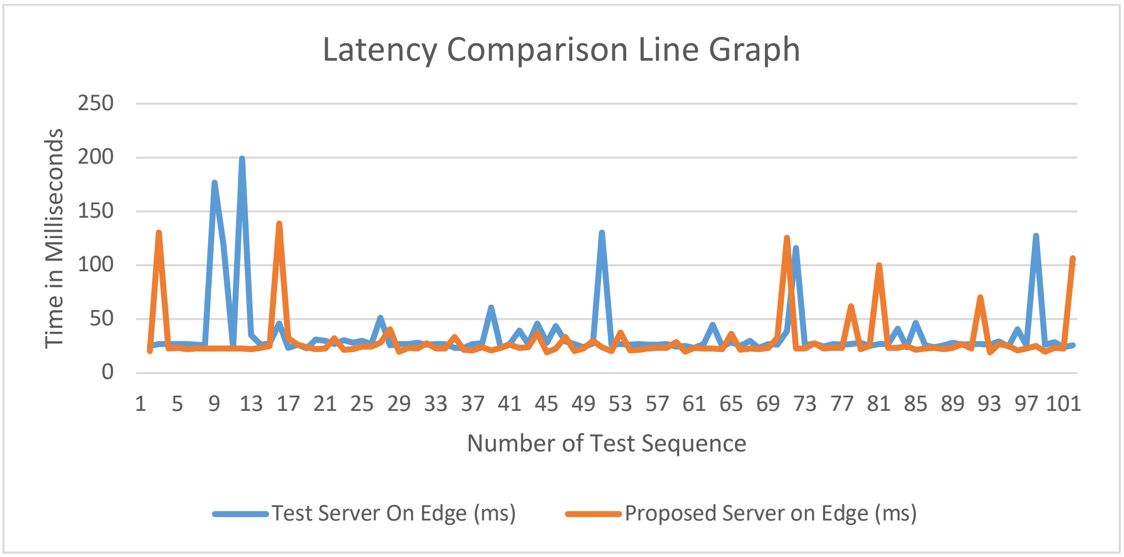


Figure 67. Latency Graph of the Servers running over the Network Edge

As seen from the above tests the Proposed Server has a round-trip latency of roughly 16.5% lower in minimum time, 16% decrease in average time and 30.6% less maximum time when compared to the Test Server.

The graph in Figure 68 shows the latency comparison done between the servers running in the network edge and the existing MQTT broker available for DIY IoT projects [9]. It will clearly show the significance of running a server at network edge. The Table 4 shows the roundtrip time taken for making a request and receiving a response from the respective servers.

Table 4. Latency Comparison of the Servers with MQTT Brokers

Servers	Minimum Time(ms)	Average Time(ms)	Maximum Time(ms)
Proposed Server Latency over Network Edge	19.2	29.9	138.8
Test Server Latency over Network Edge	23.0	35.6	199.1
Cayenne Dashboard Latency over Internet	202.9	221.8	389.8
Adafruit Dashboard Latency over Internet	194.6	240	2573

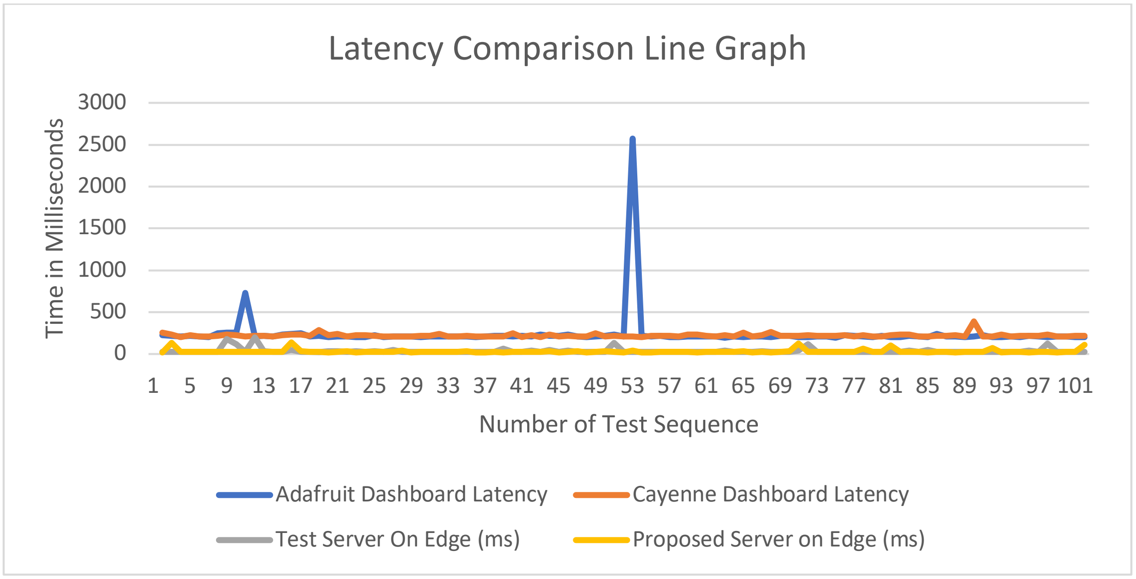


Figure 68. Latency Graph of the IoT Servers

As bluntly seen on Figure 68, the peak in latency at middle of the sequence could be hop errors and re-routing of data due to unknown error at network nodes in the internet. The latency in Table 4 is well justified with conclusion arrived from the Fog concept. All pointing towards the distance between networks nodes are one of the major latency inducing element in a network.

VIII. CONCLUSION

After looking closely into the above conceptual model, it is seen that going through cloud data resources which are stored in a compound allocation and time-consuming index logs, can be avoided using Fog computing. Localizing the essential data process by usage frequency of the user is found competent. The task requiring immediate attention is processed at the edge of the enterprise network rather than from cloud service. Thus, reducing the cost involved in handling cloud service by efficiently utilizing the cloud only for large data entries and index log updates.

The Fog computing process could result in a handshake between the Cloud service and the end-user devices. Thus, eliminating the possible lag in real-time data visualization and also establish a control layer at the enterprise network edge. An optimized application programming interface running using fog concepts demonstrated high efficiency in data handling latency.

The combined solution with Microsoft HoloLens gives the user lag free augmented 3D experience, and the mixed reality function utilizes the control layer at the enterprise network edge to add virtual interaction to the physical device.

REFERENCES

[1] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, “A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges,” IEEE Communications Surveys Tutorials, vol. 20, no. 1, pp. 416–464, Firstquarter 2018.

[2] Microsoft, “Microsoft HoloLens,” Microsoft HoloLens. [Online]. Available: https://www.microsoft.com/en-us/hololens/commercial-overview. [Accessed: 10-Oct-2018].

[3] D. Adrianto, M. Hidajat, and V. Yesmaya, “Augmented reality using Vuforia for marketing residence,” in 2016 1st International Conference on Game, Game Art, and Gamification (ICGGAG), 2016, pp. 1–5.

[4] “Edge computing vs. fog computing: Definitions and enterprise uses,” Cisco. [Online]. Available: https://www.cisco.com/c/en/us/solutions/enterprise-networks/edge-computing.html. [Accessed: 23-Oct-2018].

[5] “IoT, from Cloud to Fog Computing,” blogs@Cisco - Cisco Blogs, 25-Mar-2015. [Online]. Available: https://blogs.cisco.com/perspectives/iot-from-cloud-to-fog-computing. [Accessed: 10-Oct-2018].

[6] A. Yousefpour, G. Ishigaki, and J. P. Jue, “Fog Computing: Towards Minimizing Delay in the Internet of Things,” in 2017 IEEE International Conference on Edge Computing (EDGE), 2017, pp. 17–24.

[7] J. Fu, Y. Liu, H. Chao, B. K. Bhargava, and Z. Zhang, “Secure Data Storage and Searching for Industrial IoT by Integrating Fog Computing and Cloud Computing,” IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4519–4528, Oct. 2018.

[8] H. S. Raju and S. Shenoy, “Real-time remote monitoring and operation of industrial devices using IoT and cloud,” in 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), 2016, pp. 324–329.

[9] K. Grgić, I. Špeh, and I. Heđi, “A web-based IoT solution for monitoring data using MQTT protocol,” in 2016 International Conference on Smart Systems and Technologies (SST), 2016, pp. 249–253.

[10] S. Lanka, S. Ehsan, and A. Ehsan, “A review of research on emerging technologies of the Internet of Things and augmented reality,” in 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2017, pp. 770–774.

[11] M. Schneider, J. Rambach, and D. Stricker, “Augmented reality based on edge computing using the example of remote live support,” in 2017 IEEE International Conference on Industrial Technology (ICIT), 2017, pp. 1277–1282.

[12] J. B.-M. Numhauser, “Fog Computing- Introduction to a new Cloud evolution. Proceedings from the CIES III Congress, January 2012,” Escrituras silenciadas: paisaje como historiografía / José Francisco Forniés Casals (ed. lit.), Paulina Numhauser (ed. lit.), Proceedings from the CIES III Congress, January 2012.

[13] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” in Proceedings of the first edition of the MCC workshop on Mobile cloud computing - MCC ’12, Helsinki, Finland, 2012, p. 13.

[14] C. Puliafito, E. Mingozzi, and G. Anastasi, “Fog Computing for the Internet of Mobile Things: Issues and Challenges,” in 2017 IEEE International Conference on Smart Computing (SMARTCOMP), 2017, pp. 1–6.

[15] “IBM Knowledge Center - The client/server model.” [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/SSAL2T_8.2.0/com.ibm.cics.tx.doc/concepts/c_clnt_sevr_model.html. [Accessed: 09-Oct-2018].

[16] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, “Industrial Internet of Things: Challenges, Opportunities, and Directions,” IEEE Transactions on Industrial Informatics, pp. 1–1, 2018.

[17] H. P. Breivold and K. Sandström, “Internet of Things for Industrial Automation – Challenges and Technical Solutions,” in 2015 IEEE International Conference on Data Science and Data Intensive Systems, 2015, pp. 532–539.

[18] H. Hejazi, H. Rajab, T. Cinkler, and L. Lengyel, “Survey of platforms for massive IoT,” in 2018 IEEE International Conference on Future IoT Technologies (Future IoT), 2018, pp. 1–8.

[19] M. B. Yassein, M. Q. Shatnawi, S. Aljwarneh, and R. Al-Hatmi, “Internet of Things: Survey and open issues of MQTT protocol,” in 2017 International Conference on Engineering MIS (ICEMIS), 2017, pp. 1–6.

[20] D. Jo and G. J. Kim, “ARIoT: scalable augmented reality framework for interacting with Internet of Things appliances everywhere,” IEEE Transactions on Consumer Electronics, vol. 62, no. 3, pp. 334–340, Aug. 2016.

[21] A. Srivastava and P. Yammiyavar, “Augmenting tutoring of students using Tangible Smart Learning Objects: An IOT based approach to assist student learning in laboratories,” in 2016 International Conference on Internet of Things and Applications (IOTA), 2016, pp. 424–426.

[22] E. Kucera, O. Haffner, and R. Leskovský, “Interactive and virtual/mixed reality applications for mechatronics education developed in unity engine,” in 2018 Cybernetics Informatics (K I), 2018, pp. 1–5.

[23] "computing-solutions.pdf." https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-solutions.pdf

[24] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in 2017 Global Internet of Things Summit (GIoTS), 2017, pp. 1–6.

[25] J. Guhl, S. Tung, and J. Kruger, "Concept and architecture for programming industrial robots using augmented reality with mobile devices like microsoft HoloLens," in 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2017, pp. 1–4.

[26] "computing-overview.pdf" https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf

[27] A. Nayyar and V. Puri, "A review of Arduino board's, Lilypad's amp; Arduino shields," in 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), 2016, pp. 1485–1492.

[28] P. V. Vimal and K. S. Shivaprakasha, "IOT based greenhouse environment monitoring and controlling system using Arduino platform," in 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), 2017, pp. 1514–1519.

[29] "LM35 ±0.5°C Temperature Sensor with Analog Output and 30V Capability | TI.com." [Online]. Available: http://www.ti.com/product/LM35. [Accessed: 10-Oct-2018].

[30] "Client/Server Computing." [Online]. Available: http://www.scis.nova.edu/~lloydjam/week08.html. [Accessed: 09-Oct-2018].

[31] C. Matt, "Fog Computing," Bus Inf Syst Eng, vol. 60, no. 4, pp. 351–355, Aug. 2018.

[32] M. Chiang and T. Zhang, "Fog and IoT: An Overview of Research Opportunities," IEEE Internet of Things Journal, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[33] J. Li, T. Zhang, J. Jin, Y. Yang, D. Yuan, and L. Gao, "Latency estimation for fog-based internet of things," in 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), 2017, pp. 1–6.

[34] "Welcome | Flask (A Python Microframework)." [Online]. Available: http://flask.pocoo.org/. [Accessed: 09-Oct-2018].

[35] D. Pavithra and R. Balakrishnan, “IoT based monitoring and control system for home automation,” in 2015 Global Conference on Communication Technologies (GCCT), 2015, pp. 169–173.

[36] A. Sahadevan, D. Mathew, J. Mookathana, and B. A. Jose, “An Offline Online Strategy for IoT Using MQTT,” in 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), 2017, pp. 369–373.

[37] tedhudek, “Windows Network Architecture and the OSI Model.” [Online]. Available: https://docs.microsoft.com/en-us/windowshardware/drivers/network/windows-network-architecture-and-the-osi-model. [Accessed: 10-Oct-2018].

[38] “JSON.” [Online]. Available: https://www.json.org/. [Accessed: 10-Oct-2018].

[39] “(23) The Mixed Reality Revolution is here, and it’ll change your world forever | LinkedIn.” [Online]. Available: https://www.linkedin.com/pulse/mixed-reality-revolution-here-itll-change-your-world-forever-lucas/. [Accessed: 09-Oct-2018].

[40] “Integrating Apps and Content with AR Quick Look - WWDC 2018 - Videos,” Apple Developer. [Online]. Available: developer.apple.com/videos/play/wwdc2018/603/. [Accessed: 10-Oct-2018].

[41] MixedRealityToolkit-Unity uses code from the base MixedRealityToolkit repository and makes it easier to consume in Unity.: Microsoft/MixedRealityToolkit-Unity. Microsoft, 2018.

[42] “Quickstart for Android | ARCore,” Google Developers. [Online]. Available: https://developers.google.com/ar/develop/unity/quickstart-android. [Accessed: 10-Oct-2018].

[43] “Vuforia Developer Portal |.” [Online]. Available: https://developer.vuforia.com/. [Accessed: 29-Oct-2018].

[44] H. G. C. Ferreira, E. D. Canedo, and R. T. de Sousa, “IoT architecture to enable intercommunication through REST API and UPnP using IP, ZigBee and arduino,” in 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2013, pp. 53–60.

[45] J. Lau, M. Townsley, and I. Goyret, “Layer two tunneling protocol-version 3 (L2TPv3),” 2005.

[46] “ngrok – documentation.” [Online]. Available: https://ngrok.com/docs. [Accessed: 10-Oct-2018].

[47] R. K. Kodali and K. S. Mahesh, “A low cost implementation of MQTT using ESP8266,” in 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), 2016, pp. 404–408.

[48] G. Koutitas, J. Jabez, C. Grohman, C. Radhakrishna, V. Siddaraju, and S. Jadon, “Demo/poster abstract: XReality research lab — Augmented reality meets Internet of Things,” in IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2018, pp. 1–2.

[49] R. Ramaswamy, N. Weng, and T. Wolf, “Characterizing network processing delay,” in IEEE Global Telecommunications Conference, 2004. GLOBECOM ’04., 2004, vol. 3, pp. 1629-1634 Vol.3.

[50] Y. Chen and T. Kunz, “Performance evaluation of IoT protocols under a constrained wireless access network,” in 2016 International Conference on Selected Topics in Mobile Wireless Networking (MoWNeT), 2016, pp. 1–7.

[51] “httping(1) - Linux man page.” [Online]. Available: https://linux.die.net/man/1/httping. [Accessed: 09-Oct-2018].

[52] J. Postel, “User datagram protocol,” 1980.