Cost Effective Smart System using ESP8266

[The Inexpensive J.A.R.V.I.S]

Colin Manuel Fernandes
colinmanuel.fernandes@studenti.unimi.it
914777

Course Assesment Project for
Wireless and Mobile Networks
Prof. Simone Agazzi

Universita Degli Studi Di Milano

Sep 2018

Abstract

The field of IoT has been advancing with Smart systems turning out to be a norm for commercial and domestic use. A number of vendors have successfully commercialized these systems and are very prominent in the market. In this project an approach has been put forward to design a smart system by maintaining a 'standard' of keeping the cost as low as possible.

The project uses sensor network of **ESP8266** micro-controllers communicating http requests in a client-server mode. The sensor data is visualized on a web-server and further requests are processed based on the sensor data. Also in a another separate implementation the devices are controlled using voice commands.

All the softwares and services used are open source, and all the electronic bricks used in the project sum up to approximately 20€ .

Contents

1 Introduction

J.A.R.V.I.S.[1] (Just A Rather Very Intelligent System) is a popular frictional AI character in the Marvel comic world and also in Iron-man series of movies. The AI system as portrayed in the movie is a sophisticated smart home/building/suit/etc rather a universal system, which performs learning analysis, simulation, execution on a enormous scale. Although science of the system is questionable, the system still does make sense in most of its services. Such smart systems on a smaller scale have been developed by a number of vendors. The broader area of technology has been coined as **Internet of Things** rather abbreviated as **IoT**.

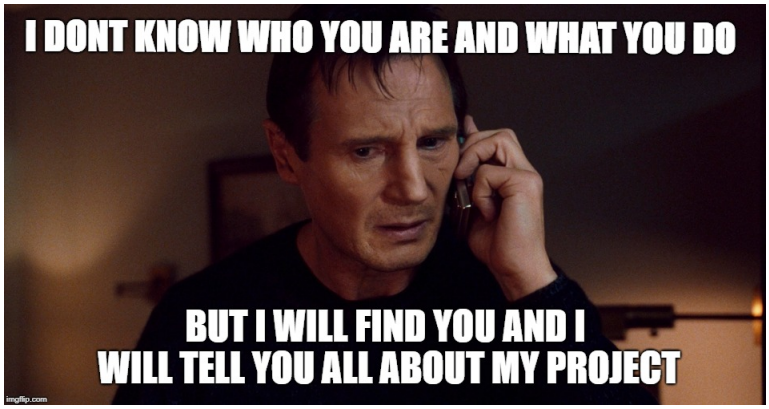


Figure 1: I have TAKEN Wireless and Mobile Networks

The current implementation is subset of my bigger project of which different parts are currently being developed and few already submitted for assessments with other subjects of my course. It is to be noted that the following implementation does not deploy missiles like the MARK-SERIES suites of Ironman, but it is a basic home automation system to analyze the environment and take appropriate actions based on the conditions provided. Cost effectiveness was the main 'standard' maintained throughout the design. "The nice thing about standards is that you have so many to choose from; furthermore, if you do not like any of them, you can just wait for next year's model" as quoted by Andrew S. Tanenbaum

The sensor network has been configured as star based on client-server communication. The clients collects the output of the sensors and post it on a collection sever. The collection server collects posts from multiple clients and post all the request to the web-server. The sensor data is visualized as a time series plot through the web-server. Also based on the sensor data appropriate operation are performed based on the given condition.

In the second implementation the relay are controlled based on commands which are input using speech.

2 Ingredients

2.1 Hardware Devices

2.1.1 ESP8266

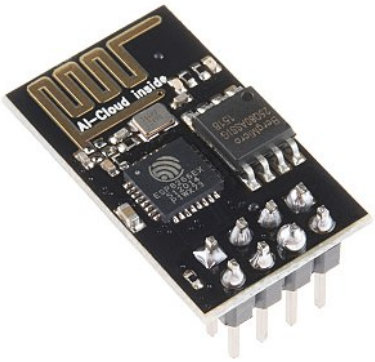


Figure 2: ESP8266 (ESP01)

ESP8266 [2] is a system on a chip (SoC) developed by a Chinese manufacturer Espressif. The price of one ESP-01 MCU is 1€, clones are even cheaper. It consists of a Tensilica L106 32-bit micro controller unit (MCU) and a Wi-Fi transceiver. It has 11 GPIO pins (General Purpose Input/Output pins), and an analog input as well. This means that you can program it like any normal Arduino or other microcontroller. And on top of that, you get Wi-Fi communication, so you can use it to connect to your Wi-Fi network, connect to the Internet, host a web server with real web pages, let your smartphone connect to it, etc .. The possibilities are endless!

There are different ways to communicate and program the ESP8266. The ESP MCU comes with a AT firmware out of the box. So we can use AT commands to communicate with the ESP using a serial port like putty or Arduino serial port. Programming can also be accomplished by using the onboard UART by connecting it to a Arduino or using USB to Serial converter. Another most important feature of the ESP is it supports **OTA** or Over The Air updates.Uploading over Serial is fine during development, when you have access to the Serial pins and the USB port. But once your project is finished, and you put it inside an enclosure, it not that easy to upload updates with bug fixes or new features.A solution to this problem is Over The Air updating, or OTA for short. As the name implies, this technology allows you to upload new code over a Wi-Fi port, instead of Serial.

2.1.2 DTH11

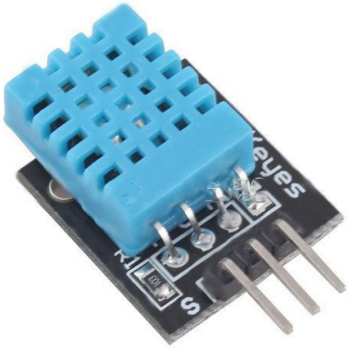


Figure 3: DTH11

DHT 11 is a low cost temperature and humidity sensor. It uses 5V DC for operation and the digital output can be read using a micro controller.The sensor includes a resistive sense of wet components

and an NTC temperature measurement devices, and connected with a high performance 8 bit MCU.

2.1.3 Relay

Figure 4: Magnetic Relay

A relay is an electrically operated switch. Relays come in two types either magnetic or solid state relay. Magnetic relay consists of moving parts activated using electromagnet to make or break a connection. Solid State Relay on the other hand has no moving parts rather uses thyristors or transistors for switching.

2.2 Programming Environment and Programming Language

2.2.1 Arduino IDE

2.2.1.1 Libraries

- **ESP8266WiFi** - Setting up an Access point, or connecting to Access point/Router in Station Mode
- **ESP8266HTTPClient** - send and receive http requests between client/server
- **ESP8266WebServer** - set up a webserser with port
- **DHT** - Reading input from DTH sensors
- **Adafruit_MQTT** - communication using MQTT protocol between Adafruit server and ESP

2.3 Web Services

2.3.1 ThingSpeak

ThingSpeak open IoT platform by The Mathworks Inc[3]. It enables us to aggregate, visualize and analyze live data streams in the cloud.It can perform online analysis and processing of the data as it comes in. Often used for prototyping and proof of concept IoT systems that require analytics.

2.3.2 Adafruit IO

Similar to thingsSpeak Adafruit handles requests using MQTT communication protocol.

Adafruit Industries is an open-source hardware company founded in 2005 by MIT hacker & engineer, Limor "Ladyada" Fried. Her goal was to create the best place online for learning electronics and making the best designed products for makers of all ages and skill levels. For further interest to know more about Limor and Adafruit a video link has been provided in the reference citeAdafruit.

2.3.3 IFTTT

IFTTT abbreviated from **If Then Then That** is a free web based service to create chains of simple conditional statements called applets. Eg. IFTTT lets you connect your voice assistant with around 600 applets[4].

2.4 Communication Protocols

2.4.1 HTTP

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, and hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web. Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

2.4.2 MQTT

MQTT (Message Queuing Telemetry Transport) is a publish-subscribe-based messaging protocol. It works on top of the TCP/IP protocol. It is designed for connections with remote locations where a "small code footprint" is required or the network bandwidth is limited. The publish-subscribe messaging pattern requires a message broker. In the current project the Adafruit libraries enable us to communicate via MQTT requests.

3 Project Architecture

3.1 Sensor Network

3.1.1 Data Flow Diagram

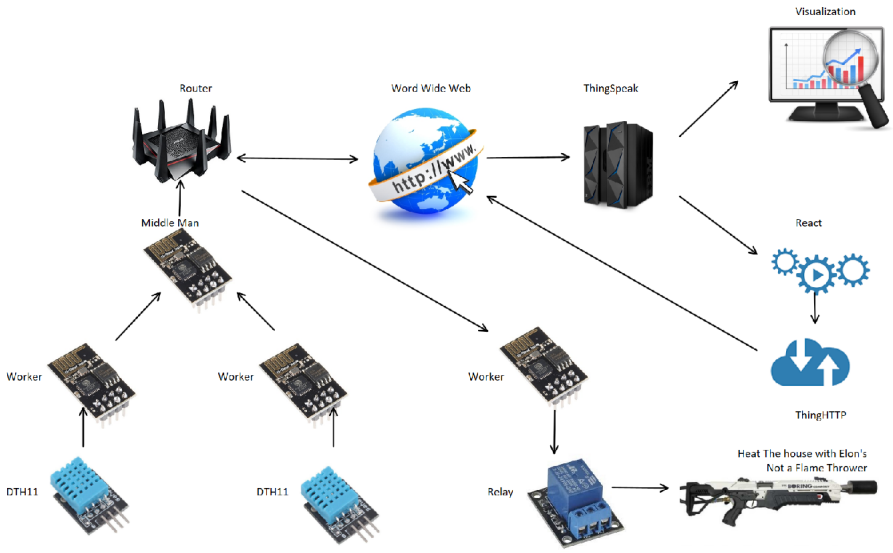


Figure 5: Sensor Network

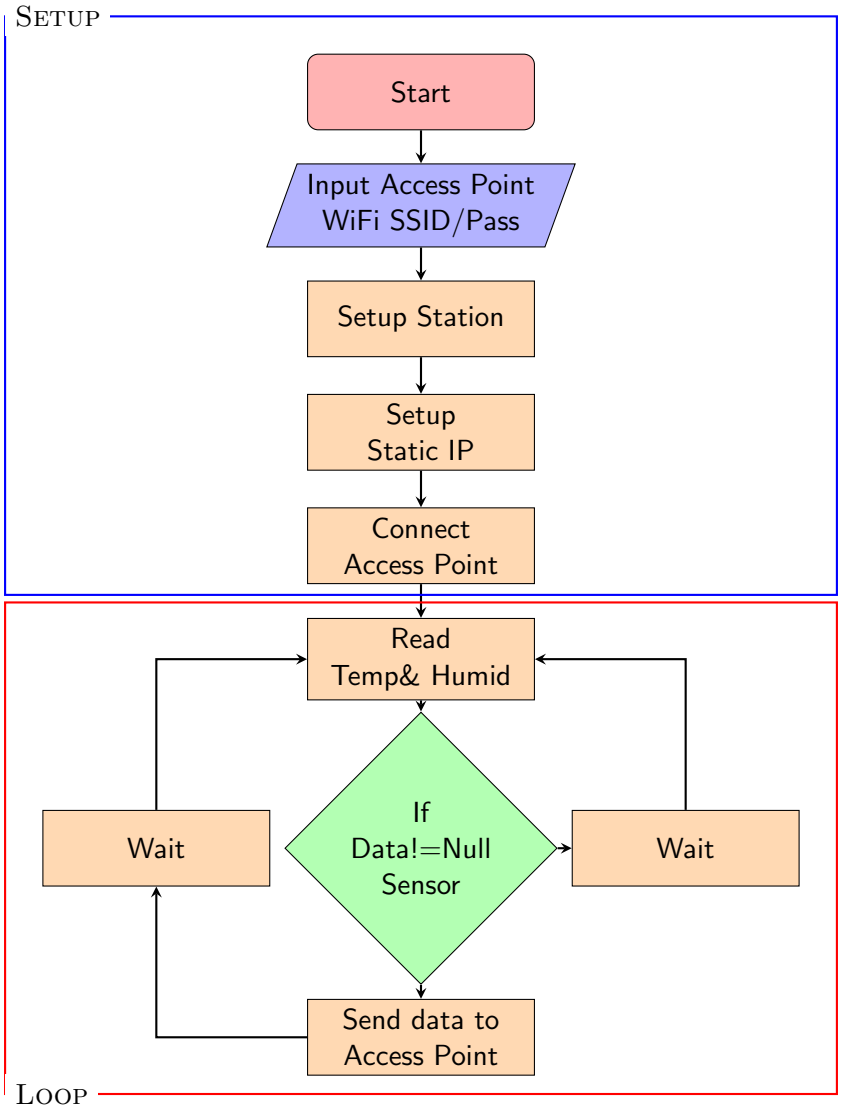
To maximize the availability the ESP are connected in Client-Server-Webserver fashion. The client ESP's or as referred to as 'worker ESP's' are setup for station mode. The Worker ESP's are connected as nodes to the Server ESP which is referred to as 'middle man' ESP. The data from DHT11 humidity and temperature sensor is collected by the worker ESP's at regular time intervals and posted to default access point IP of the middle man in the form of http url parameters.

The middle man listens to the parameters on a default port. When a post request arrives at the middleman, connection is established with internet router and subsequently to the ThingSpeak web-server. After establishing the connection, the client data is forwarded to ThingSpeak, The temperature is visualized as time series on a web browser.

To perform any action based on the sensor data, the React framework is used. We can specify conditions and respective action is performed. In the project the React is set to monitor changes in temperature and depending on the request we forward a action using thingHTTP. The thingHTTP requests an corresponding url based on the temperature values. The ESP listens to service port set open on the router and turns on or off to the relay connected to the ESP. The relay is a switch hence can be used to connect the devices to mains.

3.1.2 Process Flow

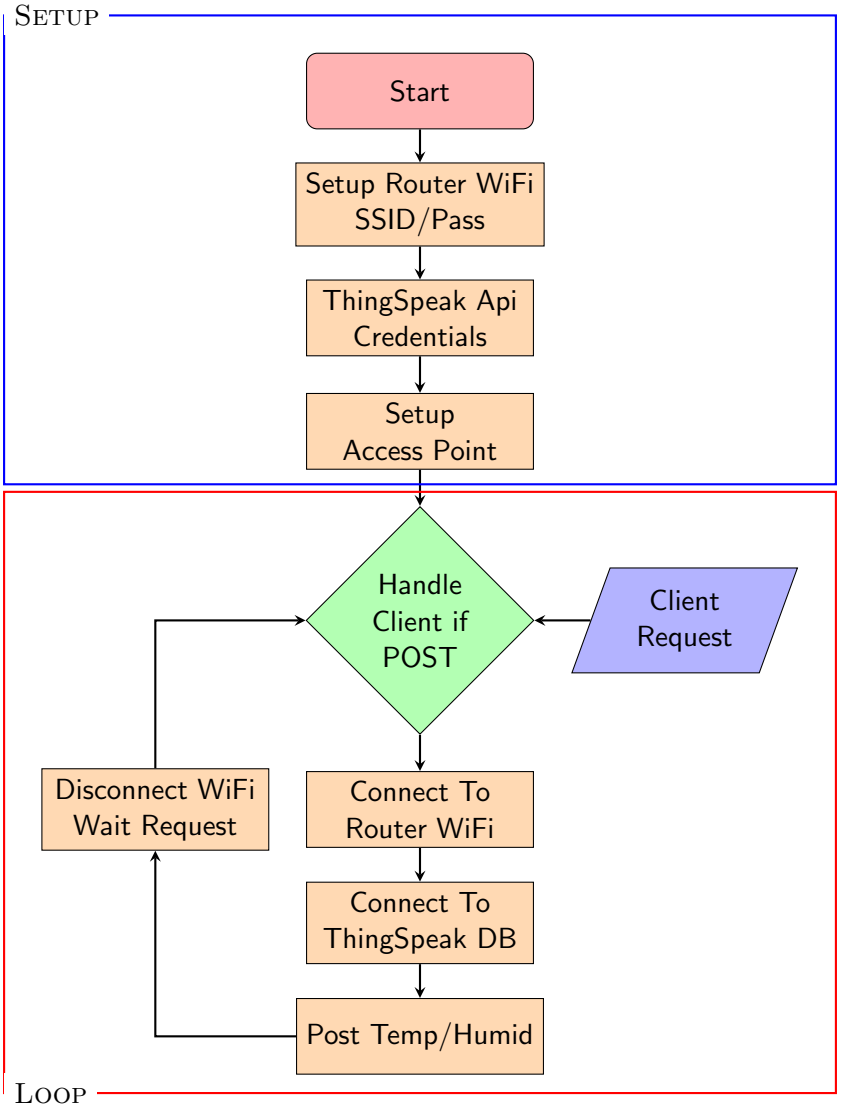
3.1.2.1 Client



The client ESP's or the worker ESP's are set up in station mode. To avoid conflict of IP the workers are set to static IP's. Data from DTH11 is collected at regular intervals. The collected data is built into a string and then posted to the intermediate middle man using http requests. The http library is used for the purpose. We could directly connect the clients to the Internet. But to avoid discontinuity during outage the data is moved through an internal server.

Figure 6: Always have a Plan B

3.1.2.2 Server

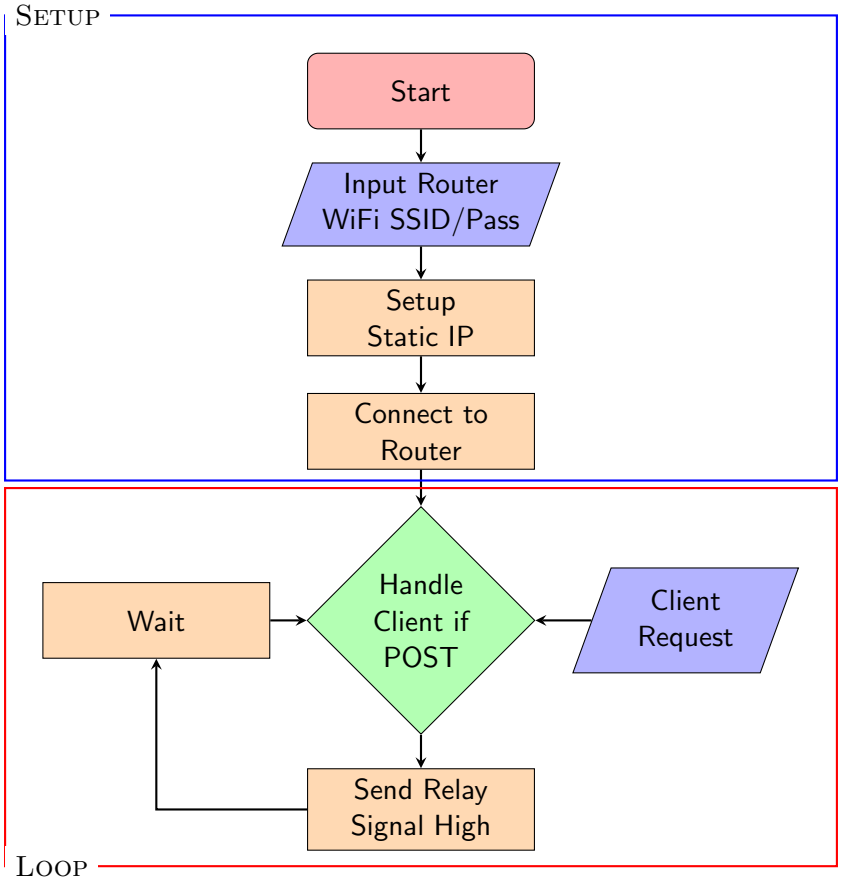


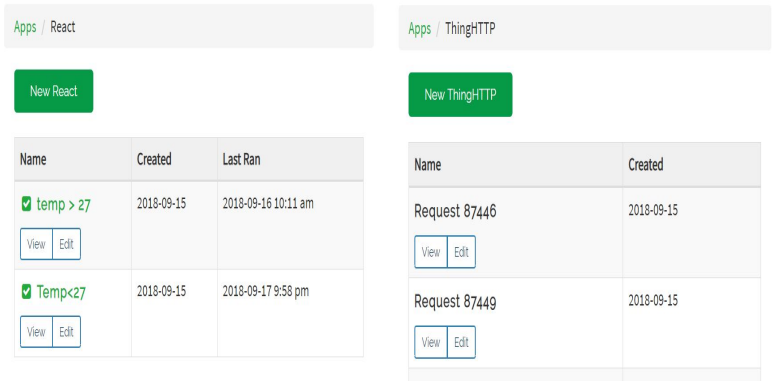
'The middleman ESP is an intermediate server between all sensor devices to the Web. The ESP is connected in both Station mode(connecting to the internet) and access point mode (enabling workers to connect to it) We could directly connect the clients to the network but this is a layer of fail safe mechanism. If the internet is not available then the data can be accessed through this ESP. the middleman listens to incoming http requests on the default port. Once the request is made the readings available is the request url are forwarded to ThingSpeak webserver through the router.

The time series data is visualized from the ThingSpeak server.

Figure 7: Visualization on the Channel

3.1.2.3 Controlling Relay using port forwarding on router





(a) React (b) Lateral

Figure 8: Images with Non frontal Faces

To perform actions on the server we use the 'React' platform of thingSpeak. Conditions provided in the react is checked against the data when it arrives. When the conditions are met the http request is sent to the server in the sensor network on the default port via thingHTTP. ThingHTTP is another application of ThingSpeak to communicate http requests. This is accomplished by enabling the port forwarding mechanism in the internet router. The service port of ESP is enabled on the router to listen to http request.

Once the action is requested signal high/low is sent through the GPIO pins of ESP, subsequently triggering the relay.

Alternative to avoiding set up of port forwarding on the router is to monitor the url using IFTTT applets. An applet is created in 'IFTTT' using 'Webhooks' and 'Adafruit IO' services. The webhooks application monitors for HTTP requests from ThingsSpeak. Once request is available the Adafruit IO

is triggered. The Adafruit IO publishes a request through MQTT-Broker(Message Queuing Telemetry Transport). The ESP in the home network is subscribed to the MQTT - Broker request from Adafruit IO.

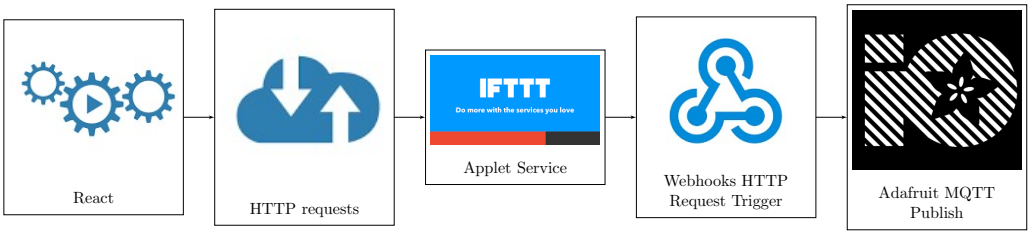


Figure 14: HTTP Trigger Applet

3.2 Voice controlled Device

3.2.1 Data Flow Diagram

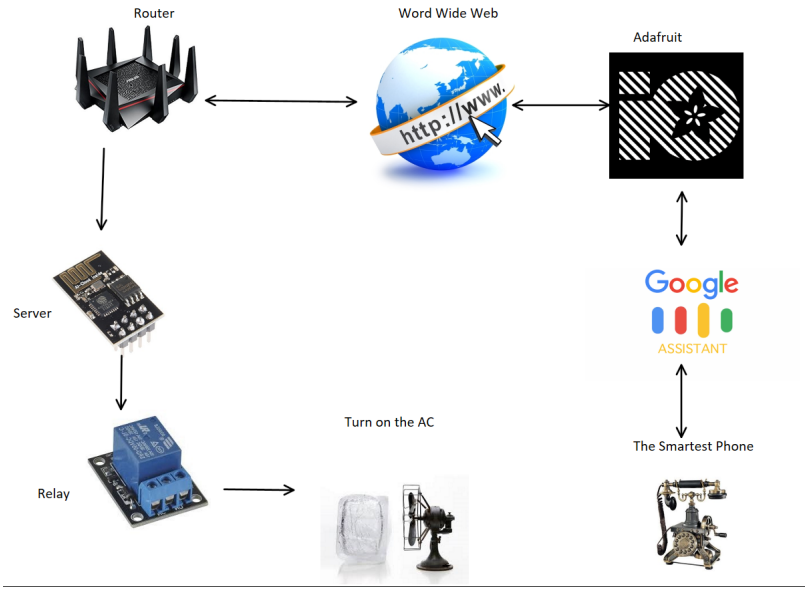
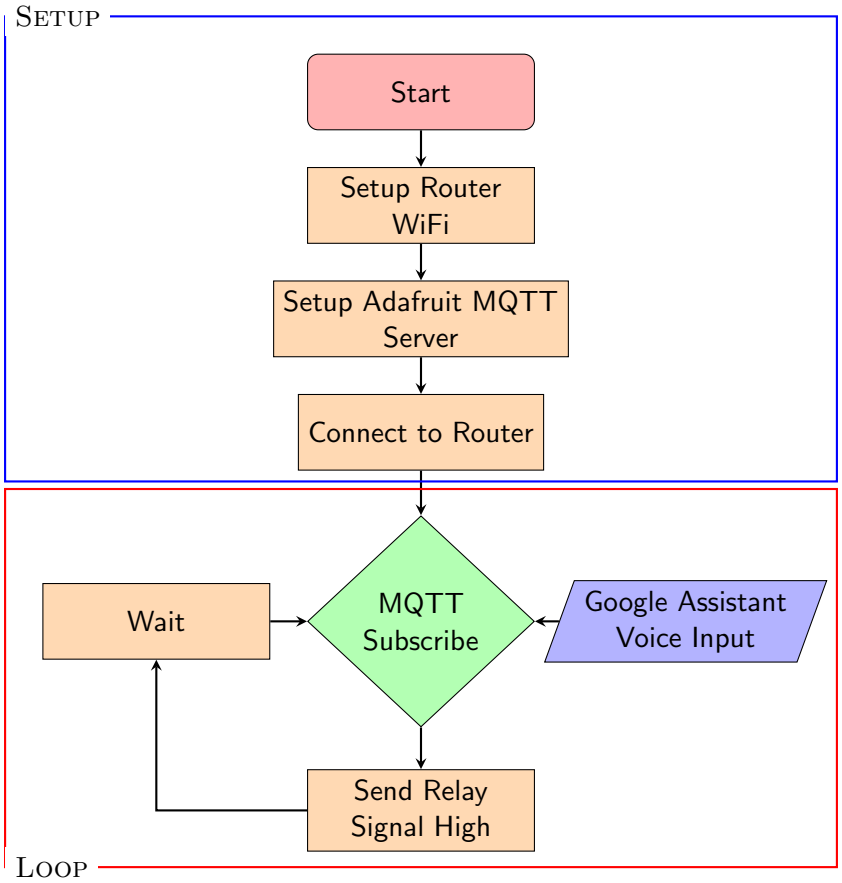


Figure 15: Voice controlled Device Architecture

3.2.2 Process Flow

3.2.2.1 Controling Relay Using MQTT



Devices can be set to trigger using the voice commands. Another applet is created on the IFTTT service platform. This applet publishes an Adafruit MQTT request when a voice command is triggered. The subscribed ESP waits for requests from Adafruit MQTT Broker and performs the appropriate action when the requests arrive.

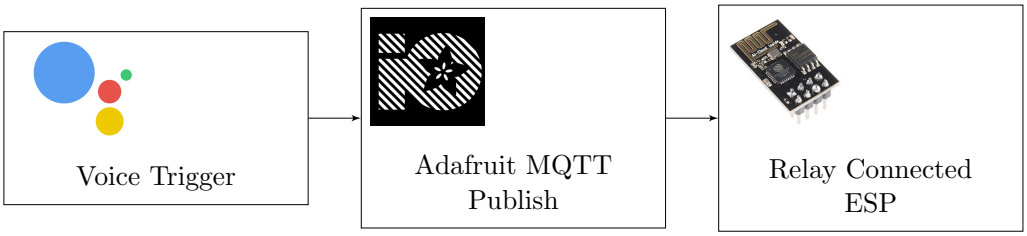


Figure 19: Voice Assistant Trigger Applet

4 Future Work

4.1 Custom Webserver

ThingSpeak is a good platform for quick implementations. Its useful when making a prototype, A dedicated common webserver can provide for a better service and reduce the working cost further.

4.2 Security

Security is an important aspect for IoT as the devices or the webservers is venerable to spam requests .

4.3 MQTT

Move HTTP requests to MQTT publish subscribe method for less pay load and power consumption.

4.4 Additional Features

Inclusion of image capture and analysis for user authentication and build preference profiles.

4.5 Reinforcement learning

Using neural network sequence models for authentication and trigger detection. Also once authenticated update profiles to user preferences over time.

5 Gantt Chart

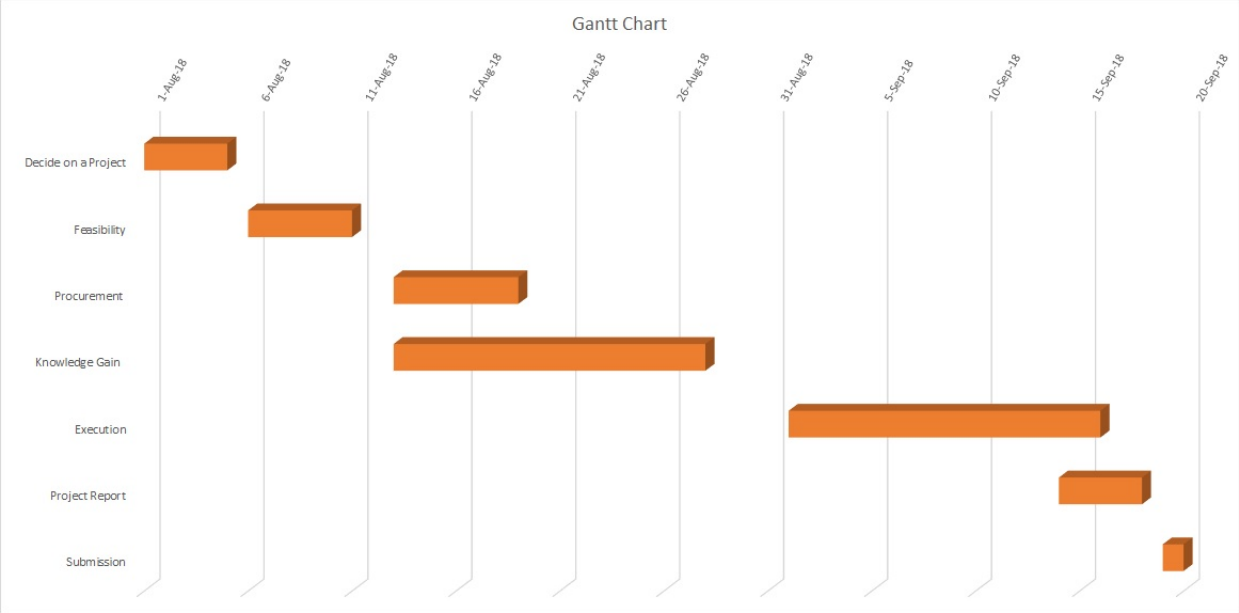


Figure 20: Project Planing

References

[1] S. Lee, "J.A.R.V.I.S," 2017. [YouTube Source].

[2] P. P, "A Beginner's Guide to the ESP8266."

[3] T. M. Inc, "ThingSpeak."

[4] I. Inc, "IFTTT."