EUROPEAN UNIVERSITY OF LEFKE

FACULTY OF ENGINEERING

Graduation Project 2

Voice Controlled Lighting System

Mafumu Mumba

174605

Lighting in buildings such as hotels, night clubs, dormitories, offices, studios and many more, is very essential for day to day activities to be carried out. Controlling of lights in these buildings is traditionally done manually. Controlling lights manually makes it difficult for people with disabilities and is also time consuming in buildings with multiple rooms.

In modern day society, with the advancement of technology this task can be automated and carried out remotely. In this writing, we explain how lights can be controlled remotely over the internet via voice commands using a Smartphone, tablet or personal computer. This project puts into consideration people with disabilities for it lifts off the burden of controlling lights manually.

Supervisor

Cem Kalyoncu

Publish Date

Table Of Contents

1. Introduction

1.1 Problem definition

Many industries in our society offer employment in most fields to individuals with full physical abilities. This is due to the fact that most tasks in these fields require manual input and because of this, disabled people are not considered for most employment opportunities for fear of them not being able their work efficiently. Performing tasks manually is also very time-consuming for it requires an individual to be at the designated location where the work is being done. This results in high running costs due to the heavy dependence on man power. In this document we specifically focus on the task of controlling lights. Traditionally the task of controlling lights is done by means of a switch which is installed or fixed in a particular location of a building such as a wall. The position at which this switch is placed only sides with the physical attributes and abilities of an average person and anyone else who does not fit in this bracket is disadvantaged. In an average set up such as an office, classroom or a households the functionality of the switch is to only turn a light on or off and does not facilitate other functions like brightness control(this helps with energy saving),fading, blinking and many more. Having mentioned everything above, the use of Automation systems is the solution. The automation system discussed in this document is a voice controlled lighting system. This system will enable a user to control lights remotely using a mobile device such as a smart phone, tablet or laptop over the internet through voice commands.

Voice controlled lighting is an embedded system that consist of a microcontroller, some LED'S and a voice recognition application installed on a smart mobile device. The system is designed to work over the internet and this is made possible by a microcontroller called the NODEMCU-ESP32S.In the system, the NODEMCUESP32S microcontroller acts as a station which connects to a local WIFI network. After connecting to the local WIFI network, the microcontroller generates the IP Address of the local WIFI network. Any device connected to the same WIFI network as the NODEMCU-ESP32S is able to access its web server through the IP Address generated. This enables a user to control lights using voice commands with the aid of the voice recognition application installed on the mobile device from any geographical location as long as the mobile device is connected to the same WIFI network as the NODEMCU-ESP32S.

Similar automation projects have been designed using different working logic and technology such as:

- **GSM based Home automation system.**

In this system an SMS containing a command for the appliance you want to control is sent to the GSM module by means of a mobile phone. When the GSM receives the message, it sends it to the Arduino microcontroller. The Arduino extracts the command, stores it in a variable as a string and compares it with the predefined string. If they match, a signal is sent to the appliance and it responds accordingly. This method is quiet costly since there is a fixed charge for every message sent.

- **Email based by using raspberry pie**

This system utilises Email technology and a microcontroller called a raspberry pie which in return is connected to an internet modem. The user is able to send a command to the email server which goes to raspberry pie which is connected to the appliance.

- **Cloud-based by using Zig Bee Microcontroller**

In this system, the mode of communication used is the zig bee based wireless network. The network is connected to a smart socket which is centrally controls the system by bringing the appliances and the user interface together which is provided by an application on an android phone.

1.2 Goals

This project simulates the controlling of lights in three separate rooms using voice commands over the internet without physically going there to tap the switch. The components involved in this simulation are a NODEMCU-ESP32S microcontroller, some LED lights and a Smartphone having a voice recognition application installed on it. The LED lights are connected to the pins of the NODEMCU-ESP32S microcontroller. The communication of the Smartphone and LED lights is made possible by connecting the NODEMCU-EPS32S and the Smartphone to the same local WIFI network. The Smartphone is able to access the NODEMCU-ESP32S web server via the local WIFI's IP Address. Once the communication is established, voice commands will be sent using a voice recognition application installed on a Smartphone. The voice commands are recorded on the smart phone and they are sent to the NODEMCU-ESP32S web server via its IP Address .The voice commands are stored on the NODEMCU-ESP32S microcontroller in a variable with a data type of string and they are compared with the predefined string of commands stored on the NODEMCU-ESP32S. If the voice commands sent from the smart phone match with the predefined commands on the NODEMCU-ESP32S, the NODEMCU-ESP32S sends a signal to the LED light connected to its pin and the LED light responds accordingly.

Some the benefits this project provides are:

- Lights can be remotely controlled over the internet, provided the device is connected to the same local WIFI network as the ESP32
- Lights can be controlled by users with various physical disabilities except the dumb without difficulties.

- Some of the light control features that this simulation provides such as brightness control can help with energy saving.
- Controlling lights remotely helps save up on time which maximizes the amount of tasks you can perform in a given instance.
- In a large industry or institution, this project can help cut down on labour costs.

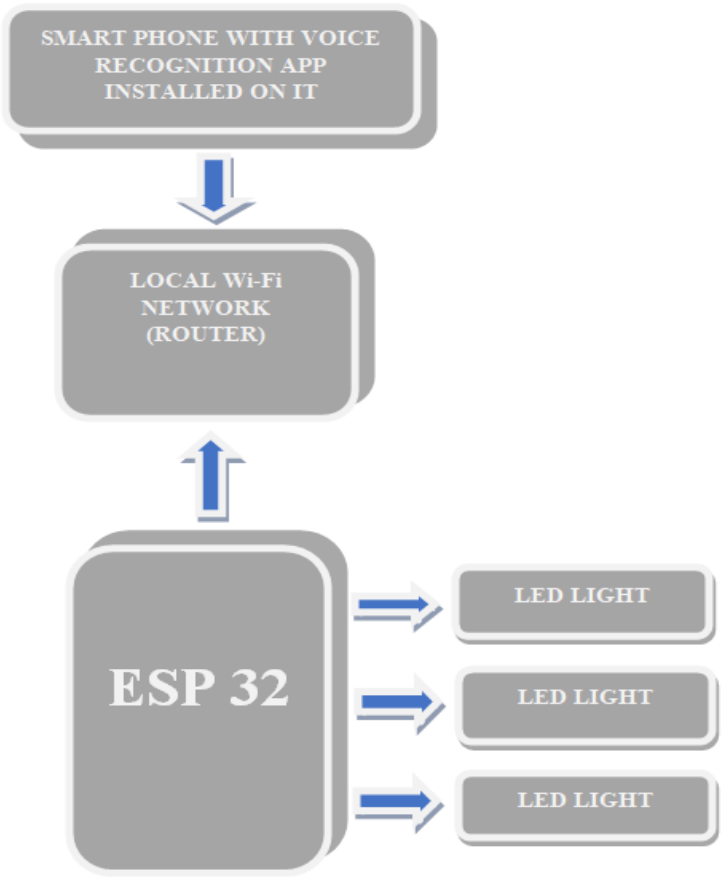


Figure 1

2. Literature Survey

Automation systems involve the use of machines instead of human physical effort to perform a task manually. Performing tasks with the aid of automation systems is empowering mankind by improving the rate at which work is done in households and businesses. Ever since the concept of automation was brought to fruition, scientists and researchers have been working tirelessly to make this more efficient, affordable and secure for people from all walks of life in order to increase productivity. Improvements and discoveries in the automation sector have been growing at an exponential rate. Currently, we live in a society where the internet has become an important tool and this was made clear during the COVID 19 pandemic. As of now,

automation systems are being incorporated with internet technologies such as WIFI. This system proves to be reliable since majority of the people in our society are familiar with the internet.

The chart below shows that sixty-three percent of the world's population use the internet.

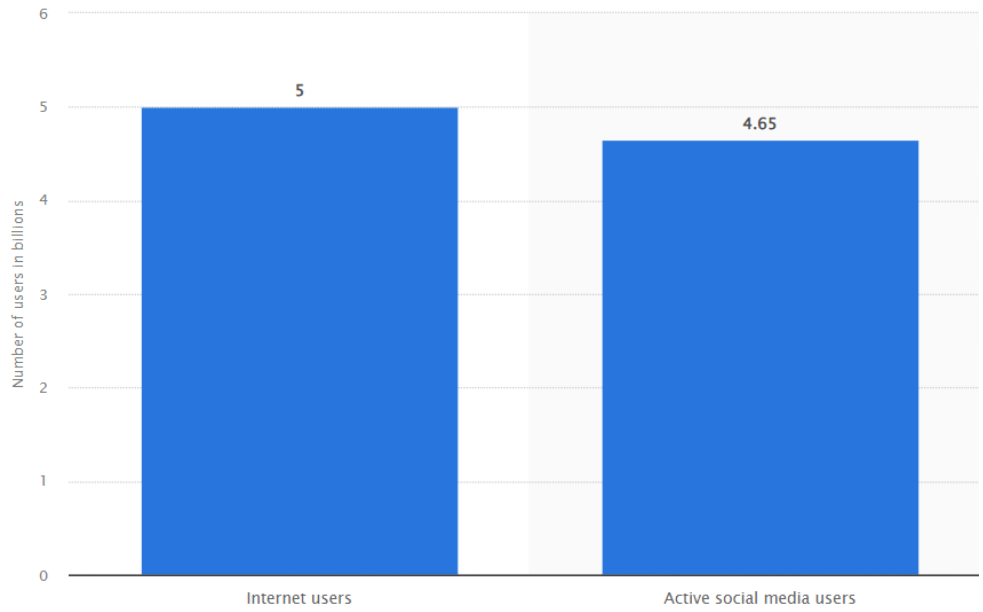


Figure 2

Wi-Fi based automation projects such as this one can be implemented in three major ways.

- Web-based server
- Interface module
- Wi-Fi connection

In a Wi-Fi based automation system, a user interface required for a client to log in to the web server using the interface module I-e Arduino, ESP32, and ESP8266 provided by the Wi-Fi network. The appliance connected to the interface module respond according to the commands processed by the web server. The user can access this web server remotely without any geographical constraints.

Cloud Based Automation System

Another form of technology that has been incorporated with automation systems is the use of a cloud to control appliances. With system a user can also control appliances remotely. The working logic behind this system is that it uses data that has been set to be controlled. The data fetched is referred to a cloud based server. The fetched data is stored into a Hadoop distributed file system (HDFS). The data is processed by a special framework that comes from HDFS using Map Reduce and it is later taken to the HDFS for storage. This makes it possible for the user to control the appliances remotely when the data is received.

GSM Based Automation System

Another automation project similar to the one discussed this document is one that is implemented using GSM (global system mobile communication).This system works in conjunction with an Arduino microcontroller, GSM modem and a mobile phone which is able to generate message alerts in accordance with user commands. The GSM modem is connected to the Arduino board and the commands are sent to the GSM modem through a smart phone. The appliances connected to the Arduino respond accordingly. A perfect example of a project that is implemented using GSM is multi-utility fire detector. This automation project uses the working logic explained above in reverse. A flame sensor and a GSM modem are connected to an Arduino board. In a case were a fire breaks out, the flame sensor sends a signal to the GSM modem via the Arduino board and the GSM modem send a message alert to mobile phone. This notifies the user that there is a fire and the advantage of this is that user will receive the message alert despite their geographical location. The major draw backs of GSM automation systems are that they are costly since every SMS is sent at a charge.

If we compare the structure of the voice controlled light system discussed in this document with the GSM based automation system, we see that the voice controlled lighting is interactive for it has voice recognition application installed on a smart phone that easy to navigate. The commands predefined on the ESP32 can be set in various languages; this makes it easy for people who are not able to read or spell words properly but are conversant in speak a particular language to use. As for GSM automation system a user has to be able to read and spell words correctly.

Cloud-based Automation System

This system uses a Zig Bee microcontroller. The Zig Bee is a microcontroller which supports mesh networking. A mesh network is a local area network (LAN), wireless local area network (WLAN) or virtual local area network (VLAN) which employs one of two decentralized connection arrangements: full mesh topology or partial mesh topology. Communication in this system happens through a network connected to a smart socket that centrally controls appliances. A user controls the appliances by means of an application installed on an android device or personal computer. This system is effectively for projects that involve temperature measurements, controlling entrance barriers and analysis of power consumption. The data collected from these projects is stored on a cloud and the user can access it from there. The major advantage of this system is it is cost effective because it has low energy consumption.

Konnex-Bus based automation system

This system was specifically designed for controlling lights and recording and monitoring temperatures. The system operates on a SIP (session initiation protocol) which acts as a connection provider. The user interacts with the system by means of a mobile application installed on a smart phone. The micro controllers that are used in this system are the Raspberry Pi and Konnex bus. For communication to be established, the Raspberry Pi acts as a link between the smart phone and the Konnex bus. The system is designed around the OSI reference model.

3. Background Information

The voice controlled lighting system is an embedded system. An embedded system is a microprocessor system that consists of computer hardware that is controlled by software to perform a specific task independently or as a component of a complex system mostly in real-time.

This project comprises of a circuit which is built of hardware and the software part which is the voice recognition application, which consists a front end and the back end.

3.1 Software Required

Back End

This part of the application consists of the logic responsible for running it. In the voice recognition application, the backend ensure that all the features such as the input of voice commands on the client also known as the front-end are valid and functional.

Below are the software used in the backend development:

- **MIT APP Inventor** is a web based integrated development environment that helps software developers who are not conversant with programming languages such as flutter to create android applications. It provides a block based code editor which has all the necessary operations.MIT APP Inventor was used to develop the back end of the voice recognition application. The backend of this application has the logical operations that collect the voice from the user, converts it to a string and sends it over to the microcontroller. The figure below show the part of the IDE responsible for the back end development.

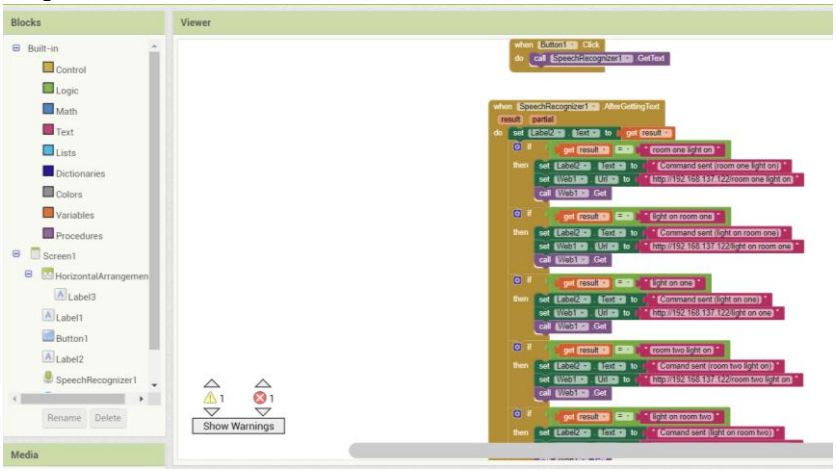


Figure 3

- **Arduino IDE** (integrated development environment) is an open source software, meaning that it can be downloaded and installed to my personal computer free of charge. It can also be modified to your own specification without any copyright infringement issues. The software is user friendly for it is easy to navigate. It has a code editor which supports high-level languages such as C, C++ and other languages such as HTML and JavaScript can be

incorporated. In this project the Arduino IDE was used to write codes and commands on to ESP32 microcontroller.

Front End

The front end also known as the client side is the part of the application that the user gets to interact with. In this project, the Voice recognition application has a graphical user interface (GUI), meaning that the platform uses icons and graphics for navigation.

- **MIT App Inventor** was also used in the designing of the user interface. MIT App Inventor was picked for this task for it is faster and easier to use due to its icon based code editor. The figure below shows the front end developer of the IDE.

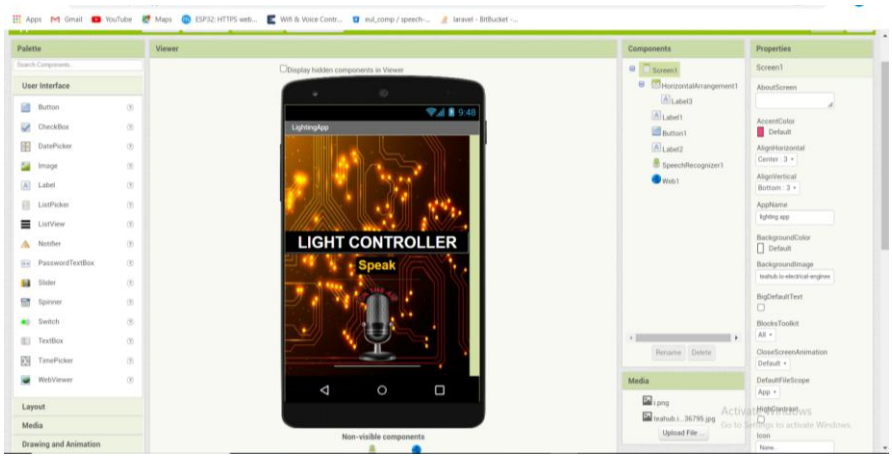


Figure 4

After developing the back end and front end of the software MIT App Inventor compiles the code and builds it into an apk file.

3.2 Other software

- **Bit Bucket**

This a cloud that is git based. It was used as a repository for all the project files.

3.3 Hardware

- **Small LED Lights**

Small light-emitting diode (LED) lights operate on the basis of a semiconductor light source. Light emission in LEDs occurs when current flows through the semiconductor's electrons which recombine with the electron holes resulting in the release of energy in the form of protons. LED lights are available in a variety colours such as red, green, blue and white. The energy required for the electrons to cross the band gap of the semiconductor is what determines the colour of the LED lights.
In this project the small LED lights are used to emulate household or industrial bulbs that we use in our day to day lives.
Below are the advantages of using Small LEDs for this project.

- They are cheap and easy to access
- They are able to light from very low voltage sources e.g.(3.3v)
- There a wide range of colours available as compared to traditional light bulbs

- They have legs that can easily be inserted on a breadboard or the female end of a jumper wire

Figure 5

- **Resistors**

A resistor is a two terminal semi electrical conductor which helps regulates current flow, adjusts signals levels, divide voltages and many other uses. In a circuit, it is the bridge between the voltage supply and the device or component receiving the current hence preventing the device from getting damaged from high current.
Since the small LEDs in this project operate on a low power supply, the resistors will prevent them for blowing up.

Figure 6 Resistor

- **Breadboard**

A breadboard is a plastic board having several tiny holes in it. The holes on it help with inserting electrical components temporarily when building a circuit. The holes are arranged in the form of rows and columns this helps prevent short circuit since current flows in a singular direction.
In this project it is used as a platform for all the circuitry involved.

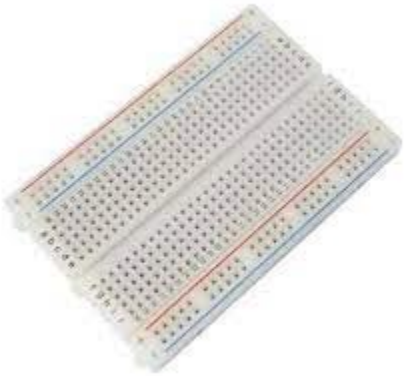


Figure 7 small Breadboard

- **Jumper Wires**

Wires are metallic which makes them good conductors of electricity. Jumper wire exist in three main types; male to male wires, female to male wires and female to female wires. Jumper Wires are bread boarding friendly hence in this project they will be used for all the connection in the circuit.

Figure 8 Jumper Wires

- **NODEMCU-ESP32S**

The NODEMCU development board is a microcontroller that contains a chip called the ESP32.

The ESP32 chip was developed and released on the 6th of September, 2016 by a company called Espressif Systems. Its predecessor is the ESP8266 which was

developed by the same company and it was released on the 30th December 2013. Both chips are very affordable and they are commonly used in DIY projects as Wi-Fi providers.

Figure 9 ESP32 chip

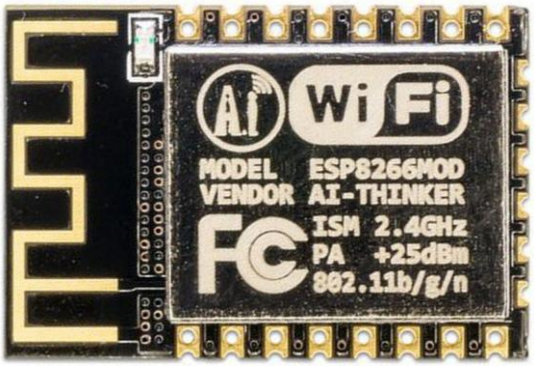


Figure 10 ESP8266

Below is a comparison in features between the ESP32 and ESP8266 chip.

Features	ESP32	ESP8266
Microcontroller Unit(MCU)	32-bit Dual-core	32-bit Single-core
Frequency	160 MHZ to 240MHZ	80MHZ
Wi-Fi	HT40	HT20
Communication Protocols	Supports:SPI,12C,UART,ADC,DAC,PWM	Supports:SPI,12C,UART,ADC,DAC,PWM
Bluetooth	Supports Bluetooth 4.2 and Bluetooth Low Energy	Not available
Input Power Supply	3.3 volts	3.3 volts

GPIO PINS	34	17
Built-In Temperature Sensor	Available	Not available
Built-In Hall Effect Sensor	Available	Not available
Touch Sensitive PINS	Available and can be used to wake it up from deep sleep mode	Not available
SRAM	Available	Not available
Flash	Available	Not available
PWM Hardware	None	None
PWM Software	16 channels	8 channels
ADC	12-bit	10-bit
CAN	Available	Not available
Ethernet MAC Interface	Available	Not available
Working Temperature	-40^0C to 125^0C	-40^0C to 125^0C

From the table above we see that the ESP32 as a successor, has more built in and improved features than the ESP8266 hence it is more convenient to use for automation projects.

NODEMCU-ESP32S Architecture

The NODEMCU ESP32-S development board has two push buttons, one which is used to flash the board whenever code is uploaded on it and the other is used to reset it once the code is done uploading and has to be executed in the serial monitor of the Arduino IDE. The board has an ESP32 chip attached to it and a total of 38 pins, majority being GPIO pins.

Figure 11 NODEMCU-ESP32S

When using the GPIO pins, extra precautions have to be taken for the pin out are grouped into different categories according to their function respectfully. So certain pins would not work depending on what you decide to use them for.

The pins are categorised as follows:

Input only pins

The pins that fall under this category are GPIOs 34 to 39.They are referred to as input only pins because of the absence of internal pull-down or pull-up resistors hence cannot be used as outputs.

- GPIO 34
- GPIO 35
- GPIO 36
- GPIO 39

SPI flash pins

It is not advisable to use these pins for the connected to the integrated SPI flash on the ESP32 chip. These pins range from GPIO 6 to GPIO 11.

- GPIO 6 (SCK/CLK)
- GPIO 7 (SDO/SD0)
- GPIO 8 (SDI/SD1)
- GPIO 9 (SHD/SD2)
- GPIO 10 (SWP/SD3)
- GPIO 11 (CSC/CMD)

Capacitive touch GPIOs

Capacitive pins are pins that are able to sense variations of objects that carry an electric charge. A good example of such an object is the human skin. In practice, this can be tested by touching the GPIOs with a finger and that is usually used to wake the ESP32 from deep sleep.

There are 10 GPIOs with this property, namely:

- T0 (GPIO 4)
- T1 (GPIO 0)
- T2 (GPIO 2)
- T3 (GPIO 15)
- T4 (GPIO 13)
- T5 (GPIO 12)
- T6 (GPIO 14)
- T7 (GPIO 27)
- T8 (GPIO 33)
- T9 (GPIO 32)

ADC PINS

There are 18 pins that can be used as Analogue to Digital Converter (ADC) input channels, these are:

- ADC1_CH0 (GPIO 36)
- ADC1_CH1 (GPIO 37)
- ADC1_CH2 (GPIO 38)
- ADC1_CH3 (GPIO 39)
- ADC1_CH4 (GPIO 32)
- ADC1_CH5 (GPIO 33)
- ADC1_CH6 (GPIO 34)
- ADC1_CH7 (GPIO 35)
- ADC2_CH0 (GPIO 4)
- ADC2_CH1 (GPIO 0)
- ADC2_CH2 (GPIO 2)
- ADC2_CH3 (GPIO 15)
- ADC2_CH4 (GPIO 13)
- ADC2_CH5 (GPIO 12)

- ADC2_CH6 (GPIO 14)
- ADC2_CH7 (GPIO 27)
- ADC2_CH8 (GPIO 25)
- ADC2_CH9 (GPIO 26)

DAC PINS

The ESP32 has 2 pins which can be used to convert digital signals into analogue signal outputs. These pins are:

- DAC1 (GPIO25)
- DAC2 (GPIO26)

RTC PINS

These pins are routed to a low-power system and they can be used when the ESP32 is in deep sleep hence they can also be used as an external wake up source. The GPIOs that fall under this category are:

- RTC_GPIO0 (GPIO36)
- RTC_GPIO3 (GPIO39)
- RTC_GPIO4 (GPIO34)
- RTC_GPIO5 (GPIO35)
- RTC_GPIO6 (GPIO25)
- RTC_GPIO7 (GPIO26)
- RTC_GPIO8 (GPIO33)
- RTC_GPIO9 (GPIO32)
- RTC_GPIO10 (GPIO4)
- RTC_GPIO11 (GPIO0)
- RTC_GPIO12 (GPIO2)
- RTC_GPIO13 (GPIO15)
- RTC_GPIO14 (GPIO13)
- RTC_GPIO15 (GPIO12)
- RTC_GPIO16 (GPIO14)
- RTC_GPIO17 (GPIO27)

PWM PINS

16 independent channels on the ESP32 posses the ability to control LEDs using Pulse Width Modulation (PMW). PMW is used in lighting projects for features such as controlling brightness. Any GPIO that can be used as an output can act as PWM pin and these are:

- GPIO 2
- GPIO 4
- GPIO 5
- GPIO 12
- GPIO 13
- GPIO 14
- GPIO 16
- GPIO 15
- GPIO 17
- GPIO 18
- GPIO 19
- GPIO 20
- GPIO 21
- GPIO 22
- GPIO 23
- GPIO 24
- GPIO 25
- GPIO 26
- GPIO 27
- GPIO 32
- GPIO 33

I2C PINS

Only two pins can be set as I2C channels. On the ESP32,any pin can be set but by default, these are the pins:

- GPIO 21 (SDA)
- GPIO 22 (SCL)

Interrupt PINS

All GPIOs on the ESP32 qualify to be configured as interrupts.

Strapping PINS

These GPIOs used to boot or flash the ESP32.

- GPIO 0
- GPIO 2
- GPIO 4
- GPIO 5 (must be HIGH during boot)
- GPIO 12 (must be LOW during boot)
- GPIO 15 (must be HIGH during boot)

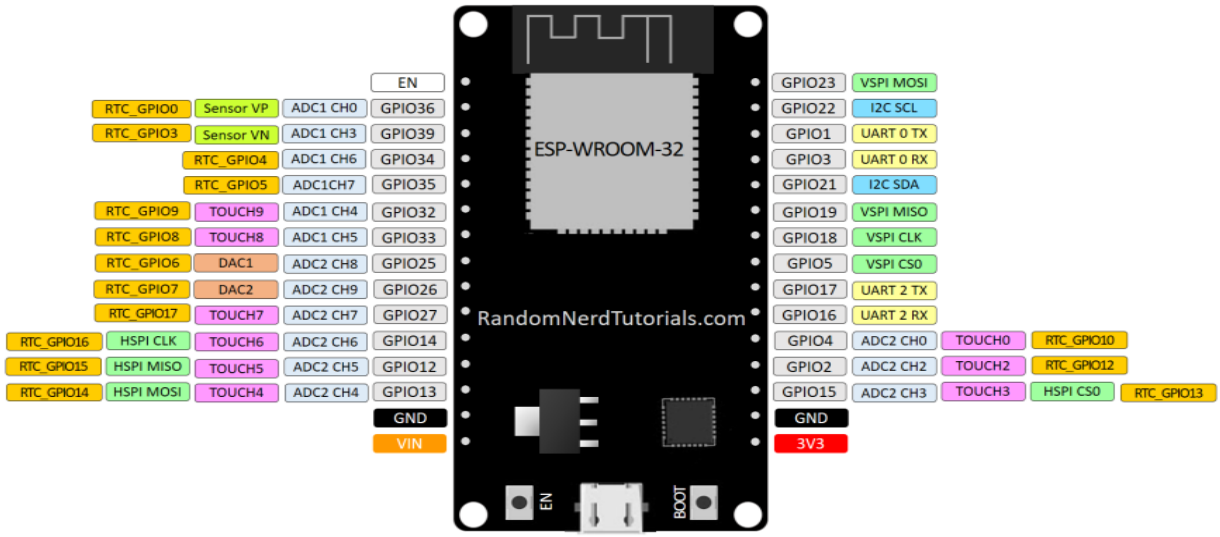


Figure 12 in depth view of NODEMCU-ESP32S pins

The NODEMCU ESP32-S web server can be accessed by two methods, station mode, soft access point mode or both at the same time.

Station (STA) Mode

In this mode the NODEMCU ESP32-S is able to connect to an existing local Wi-Fi network and act as station. It gets the IP address from the Wi-Fi network and it's from this IP address that it is able to set up a web server. Once the web server is set, it delivers the web pages to all the devices connected to the local Wi-Fi network.

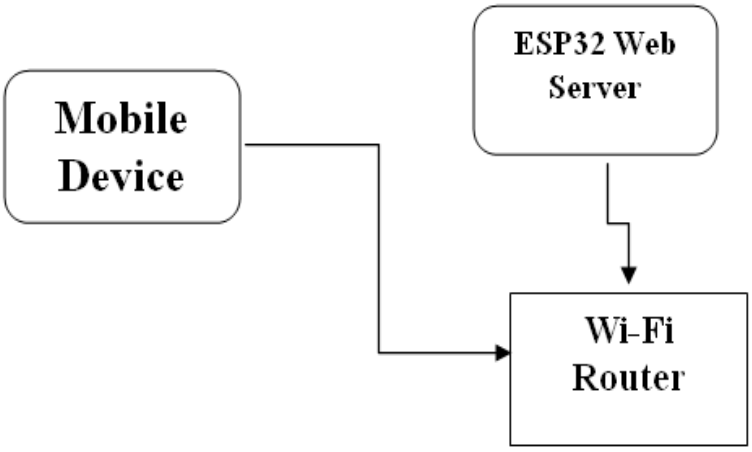


Figure 13 STA mode of NODEMCU ESP32 -S

Soft Access Point (AP) Mode

In this mode, the NODEMCU ESP32-S acts as Wi-Fi router and it is called a Soft Access Point (AP). This is due to the fact that it does not have interface to a wired network hence a maximum of device can be connected to it. Soft access point mode is archived by the NODEMCU ESP32-S creating a new Wi-Fi network and setting its own Wi-Fi network name (SSID), password and IP address. Once this is done, web pages are delivered to devices connected to it.

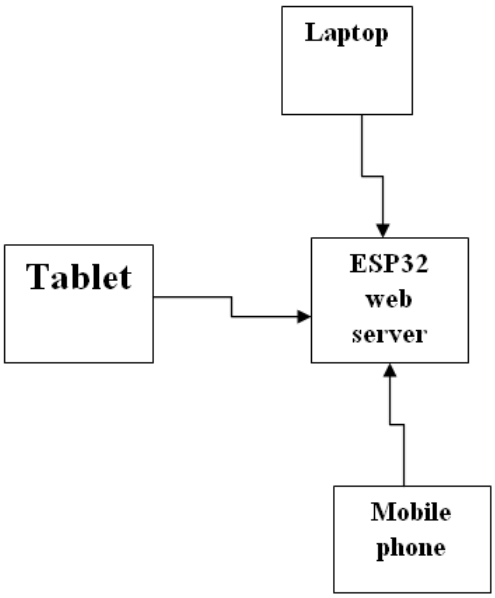


Figure 14

In this project we use the NODEMCU ESP32 development board for the following reasons:

- The NODEMCU ESP32s development board is very popular and that makes it easy to access on online sites such as Amazon, Aliexpress, and EBay.
- It has on it a micro USB port which we connect to the computer in order to power it up and program it.
- The microcontroller is programmable using open source Internet of Things (IOT) Software, in this case, the Arduino IDE (Integrated Development Environment).

4.0 Design Documents

4.1 Structure of the system

The user first opens up the voice recognition application on the mobile android device. On the application there is a button having a microphone icon. The user will click on the button and the Google speech recognition tool will pop up. When the Google speech recognition application pops up, the user can speak through it and say the specific voice commands needed to control the LEDs. After the user is done speaking, the Google speech recognition tool will convert the analogue signal (the users' voice) to a string/text that is easily understood by the computer (Android device). The voice recognition application then sends the string of voice commands to the ESP32 web server via the internet through the local Wi-Fi networks IP address. When the web server receives the string commands, they are compared with the predefined commands of type string stored on the microcontroller. If the commands sent match with the predefined commands on the microcontroller, the LEDs will respond accordingly and the command will be sent back to the device which converts it to an audio output. For example if the user says "lights on" the exact same string will be searched for on the web server during execution and if a matching one is found, all three LEDs will go on. If it is not found, the user will have to say a different command.

4.2Data flow diagram

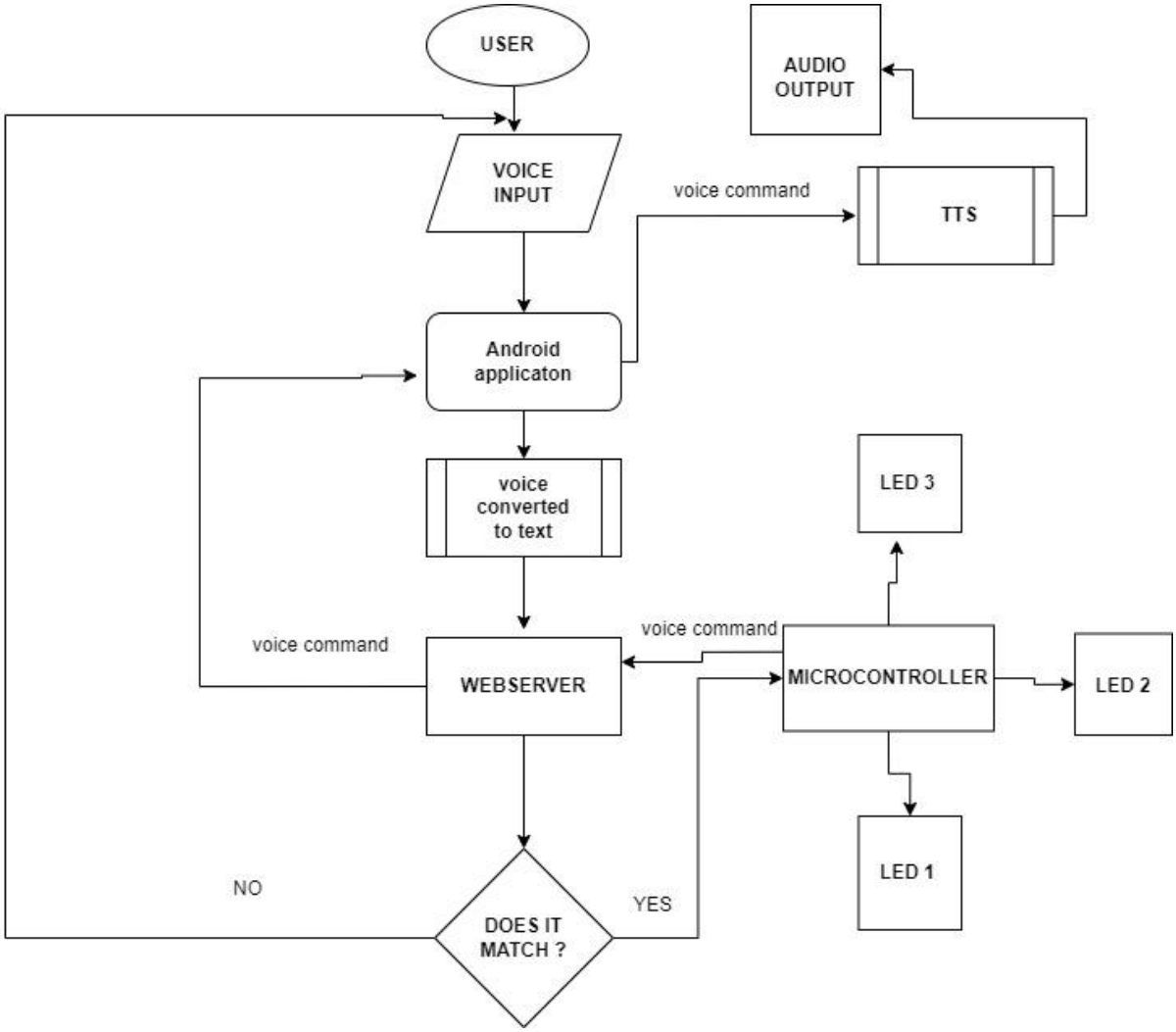


Figure 15 flow diagram

Above an illustration on how the system works starting from the android application until the LED lights.

Functional Requirements

In the case of the voice controlled lighting system, we are using a voice recognition application installed on an android device which in turn sends the voice commands provided by the user to the ESP32 web Server over the internet. So for this reason the android device that hosts the application has to poses certain criteria.

For this particular system, the voice recognition application functions on a mobile device that runs on an Android Operating system version 4.4 (Kitkat), 5.0 (Lollipop), 4.3.1(jellybean) and below. Any devices with a latest Android Operating system will display the interface of the application but the features will barely be functional. Also, the application will not work on mobile devices that operate on systems other than Android such as Apple.

When the user opens the application, a button having a microphone application will appear. To send a voice command, the user will have to press the button and speak through it. The voice is sent to the Google speech recognition application which takes on the process of converting it to a string.

The processed input will then be sent to the ESP32 web server via the internet. Once this is sent, the string is compared with the predefined commands on the microcontroller during code execution and the LED's shall respond accordingly.

Non-Functional Requirements

Other than the technicalities and the working logic of the system, other factors have to be put in place for the operation to be complete.

Below are some of the major non-functional requirements that must be taken into consideration:

- Since information on the system is sent over the internet, it is very important that the android device has Wi-Fi connectivity (internet connection).
- The application will only process inputs that are identified, so any voice commands sent in a language other than English will not be recognized. So the language settings for the Google speech recognition application should be set to English.
- The data received on the ESP32 web server sent by the mobile device is only executed if there is communication between the mobile device and the microcontroller. For this to be made possible, the IP address of the local Wi-Fi network generated by the microcontroller must be the same as the one on the mobile device.
- The system can only perform one task at a time; when a user sends the voice command he/she will have to wait for the LED to respond to it before sending another one. Tasks cannot be carried out simultaneously.

4.2Your Context Diagram

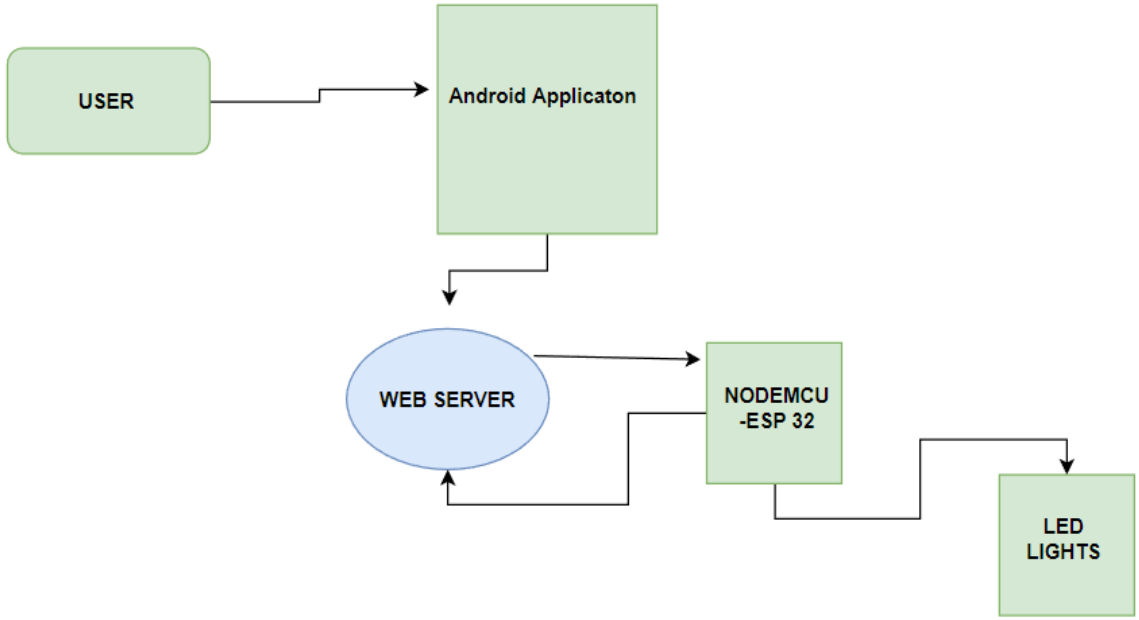


Figure 16

The context diagram above illustrates how the user interacts with the whole system that involves mobile device, web server, the microcontroller and the LED lights. When the user provides the voice input, the android application converts it to a string and sends it over to the web server. The microcontroller receives the commands and if they match, the sting is sent back to the android through web server and is converted to audio output. The LED's then respond accordingly.

5.0Methodology

The voice controlled lighting system is an embedded system; this means that it involves hardware and software. The system simulates how lights in various rooms of a building can be controlled remotely using voice commands sent over the internet with the aid of a mobile device. To achieve this we use a circuit comprised of a microcontroller and small LED lights, and a custom application installed on a mobile device. The microcontroller will have to act as a station which connects to the local WIFI network. Once connected, it collects the IP address and this will make other devices connected to the same network as microcontroller be able to access the web server. When the microcontroller is in station mode, more the one device on the network can access the web server through the application.

The software part of the system involves the android application and the hardware part involves the microcontroller and the LED light circuit connections. The hardware and the android app controlled by the user are supposed to coordinate. They will coordinate in such a way that when the user sends a voice command through the application, the hardware (LED lights) responds in accordance with the input received on the web server.

So according to this, a step by step approach has to be followed on how the project design is implemented.

5.1Development Cycle

This part of the document presents the phases followed in the development of the voice recognition application as well as the hardware.

Phase 1: Hardware Development

In the previous chapters of this document, it is mentioned that the NODEMCU-ESP32S development board will be used for the web server (Wi-Fi) and will also have LED lights attached to its pins.

From the features of the microcontroller discussed in the previous chapters, the decision made is to use it in station mode so that multiple devices can access the web server. Setting up the NODEMCU-ESP32S as a station will be done using the Arduino IDE.

Once the microcontroller has been configured as a station, the circuit can be set up with the understanding of what each pin on the board is used for. Furthermore the codes containing the commands of how the LEDs should respond will be written in the Arduino IDE and uploaded on the development board.

Phase 2: Software Development

The voice recognition application has a graphical user (GUI) interface which collects voice commands from a user, converts them to a string and sends them over to the NODEMCU-ESP32S web server over the internet. The development of this application specialises in being able to convert an analogue signal which in this case is a voice, into a format the computer is able to comprehend (text) and once that is archived, it is sent over the internet to

the NODEMCU-ESP32S development board and the LED lights connected to it respond accordingly.

So this phase of the cycle involves the development of the voice recognition application and the tool that is used to build the application is MIT App Inventor.

Phase 3: Testing the coordination of the hardware (circuit) and software (Android application)

In this phase of development we get to see the working status of the whole system and it is off this analysis that improvements or adjustments will be made. To test the coordination of the system, the user will have to send some voice commands through the voice recognition application installed on the smart and from there we see if the LEDs connected to the microcontroller respond accordingly. If the LEDs do not respond accordingly, both codes on the microcontroller and the application will have to be checked for bugs. Other troubleshooting methods such as checking the circuitry of the hardware will also be applied.

Phase 4: System Review

In this phase, the product would and been formally completed. The product will be ready to be presented to all the necessary stakeholders. The stakeholders get to use the product and the feedback based of their experience will be taken note of. The data collected from the stakeholders is cardinal for future updates and improvements of the system hence making it more sustainable.

5.2 IMPLEMENTATION

Setting up the NODEMCU ESP32-S in Arduino IDE

Before programming the NODEMCU-ESP32S we first have to download and install the Arduino IDE. After installing the Arduino IDE, we notice that the only development boards available in the software by default are Arduino microcontrollers such as Arduino Uno, Arduino ATMEGA etc. So to get started, we have to download and install the ESP32 development boards in the Arduino IDE.

1. To download and install the ESP32 development boards we go to the preferences section and insert the ESP32 development boards URL.

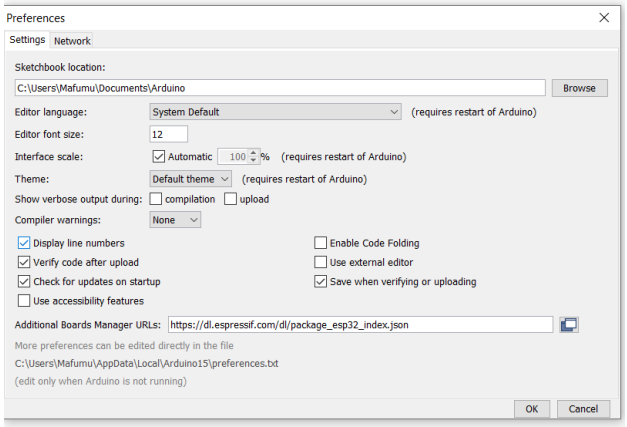


Figure 17 Preferences section

2. The next step is to go to the boards' manager, search for ESP32 and install.

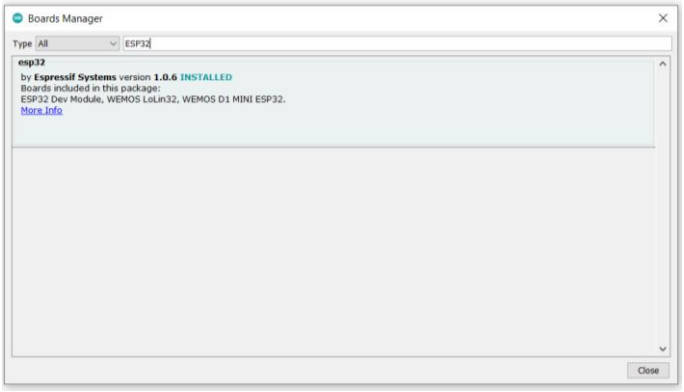


Figure 18 Boards manager

3. Select the development board you are using, which in our case is the NODEMCU-ESP32S

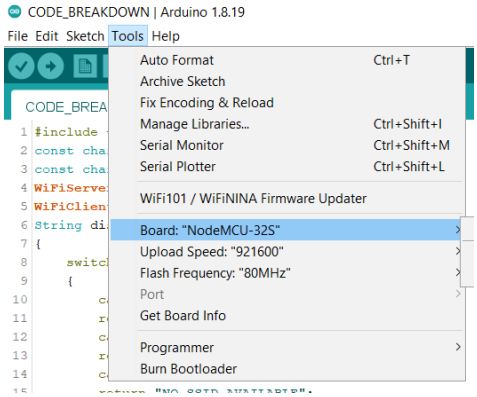


Figure 19

After setting up the Arduino IDE and installing the ESP32 boards, the next phase is connecting the ESP32 to Wi-Fi

ESP32 Wi-Fi Configuration

Since the voice controlled lighting system will allow more than one device to have access connecting to the web server we are going to use the NODEMCU-ESP32S in station mode. In this mode the NODEMCU-EPS32S acts as a station by connecting to the local Wi-Fi network and collects the IP address. All other mobile devices in the system have to connect to the same Wi-Fi network that NODEMCU-ESP32S is connected to and will access the web server using the IP address collected by the microcontroller.

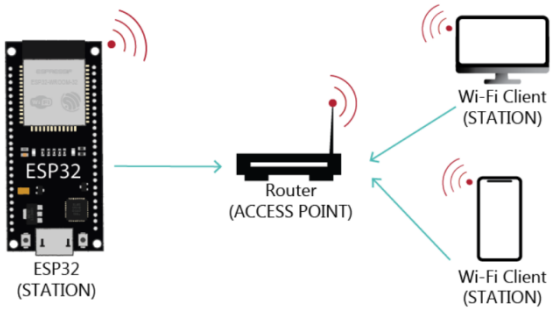


Figure 20

In order to configure the ESP32 for Wi-Fi, we have to include the needed Wi-Fi library in the code editor and this case we use "<WiFi.h" >. After that we have to initialize two characters variables which will store the username and password for the Wi-Fi network that microcontroller and mobile devices will connect to. The ESP32 microcontroller must then begin searching for the Wi-Fi network that corresponds with the credentials initialised in the code. When the network is found, the microcontroller will connect to it and print the IP address in the serial monitor of the Arduino IDE. To avoid an endless loop of trying to connect to the Wi-Fi network, we incorporate a function that checks the Wi-Fi status, scans it, checks if the network is available, connects and disconnects until the microcontroller connects to the network. The behaviour of the ESP32 connectivity can be seen from the serial monitor of the Arduino IDE.

Figure 21

Figure22 Code for wifi configuration

Figure23 Behaviour of wifi connectivity in the serial monitor

Circuit Design

After configuring the NODEMCU-ESP32S for Wi-Fi we will have to interface it with the LEDs. Like earlier mentioned this project emulates a building having lights in three separate rooms. So for this project we are going to use three LED lights connected in parallel. The other materials that will be used in this project are jumper wires; for all necessary connections in the circuit, a small breadboard and resistors for each LED light.

Selecting the right resistor

Small LED lights operate on very low currents so using them without resistors risks them blowing up. The voltage output supply of the NODEMCU-ESP32S development board is around 3.3V and each LED operates on a forward voltage 2V and a current of 20ma. Since

we are connecting the resistors in series with the LEDs respectively, we first subtract the power supply voltage from the forward voltage of the LED (3.3V – 2V) and the difference (1.3V) is used as the voltage in the calculation of the resistance using ohms law; Resistance = voltage/current (1.3V / 0.02A) which gives us 65 ohms. So a 65 ohms resistor is the most ideal but anything above that will work except it will affect the brightness of the LED negatively. In our case we use a 220 ohms resistor for each LED.

Interfacing the LEDS with the NODEMCU

The pins on the microcontroller that will be used are output pins that support pulse width modulation (PWM) to allow for features such as brightness control. So for this reason we use GPIO pins 5, 14 and 15. The LED has to legs, the longer one is the anode; this is the one which is connected to the resistor in series and the female to female jumper wire is connected to the resistor end and later connected to the GPIO pin. The other leg is connected to the ground pin in order to complete the circuit.

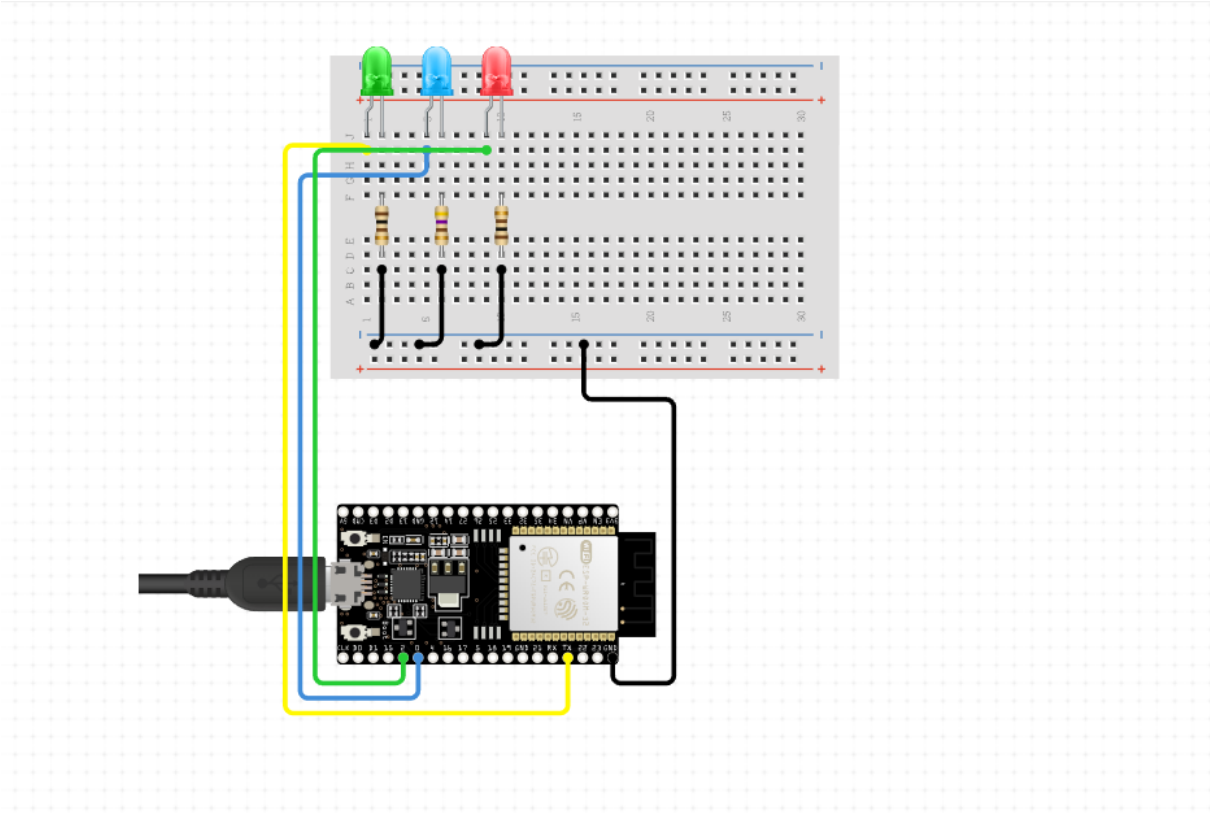


Figure 24 circuit design

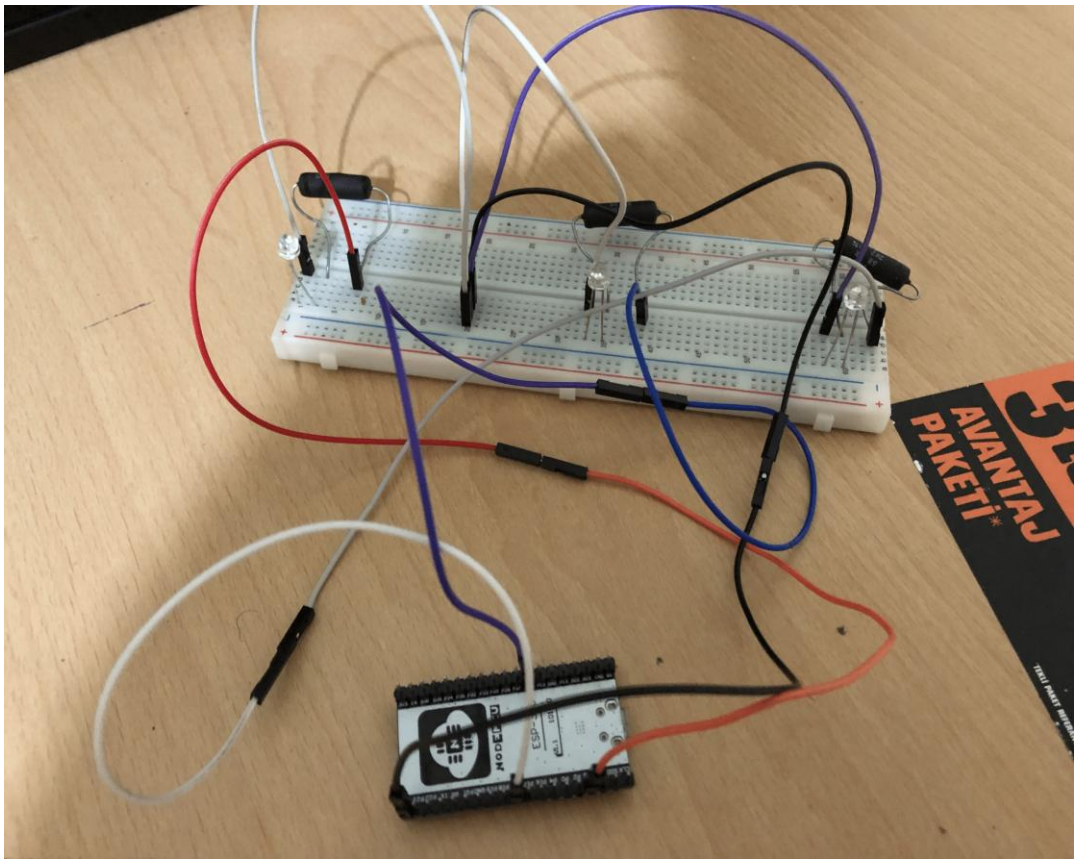


Figure 25 actual cicuit

LED Control Features

Before we can start adding light controlling features to the system it is important the GPIO pins that the LED pins are connected are initialised and declared in the Arduino IDE.

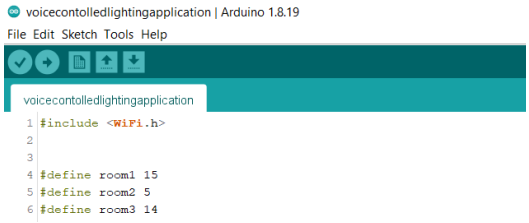


Figure 26

1. Turning On and Off the LEDs

This is the feature that determines the state of the LED light, whether it will be on or off. To archive this, since we are using digital output pins, the function called digitalWrite is used. For us to determine if the LED is going to be on, the arguments we use in the function is the pin number and the keyword ‘HIGH’. If the LED is going to be off, the argument we use in the function is the pin number and a keyword ‘LOW’.

Figure 27

2. Dimming of the LEDs

For this feature, the pulse width modulation (PWM) property of the pins comes into play. PWM is the notion that data can be encoded within a single signal chest pattern. How this happens is that the data is included in a signal using the duration that the signal is high, relative to the duration that the signal is low. So relating to the dimming of lights gradually goes HIGH and LOW over a specific time period which in this case is the delay in milliseconds.

When making an LED dimmable on NODEMCU-ESP32S development board, we have to put into consideration three important variables; the set frequency, duty cycle resolution which ranges from 1 to 16 bits and PWM channel which ranges from 0 to 15. In our case we use PWM channel 0, frequency of 5000Hz and duty cycle resolution of 8bits which means we can control the brightness from a range 0 to 255 units. To configure the LED with the PWM functionalities, we use the function called **ledSetup** which takes in the frequency, resolution and channel as arguments. Next we have to attach the channel to the GPIO pin carrying the signal and to do this we use **ledAttach** pin which takes in the channel and pin number as arguments. We then create two loops; one which increases the brightness from 0 to 255 and another which reduces it from 255 to 0. These loops will interchange between delay times of twenty milliseconds. The **ledcWrite** function changes the brightness and it takes duty cycle and the channel as arguments.

```
#define room3 14
const int c =0;
const int f = 5000;
const int res = 8;
```

Figure 28

```
void fade(int roomnumber)
{

    ledcSetup(c,f,res);
   ledcAttachPin(roomnumber,c);
  for(int Cycle = 0; Cycle <= 255; Cycle++){

    ledcWrite(c, Cycle);
    delay(20);
  }
  for(int Cycle = 255; Cycle >= 0; Cycle--){
    ledcWrite(c, Cycle);
    delay(20);



  }



}
```

Figure 29

3. Blinking of the LEDs

The logic that is used to make an LED blink is applied by setting the initial state of the LED light as 'HIGH', which means that the LED is on and then later changing it to 'LOW' (off) after a set time interval and vice versa respectively. In our case, the LEDs will switch between states over an interval of a 1000 milliseconds. For the amount of times we want the LEDs to blink, we use a loop which later terminates when the desired number of iterations is reached.

voicecontolledlightingapplication | Arduino 1.8.19

File Edit Sketch Tools Help

voicecontolledlightingapplication

```
  }
  void led_blink(int roomnumber)
  {

    for(int i=0;i < 5;i++)
    {
   digitalWrite(roomnumber,HIGH);
delay(1000);
digitalWrite(roomnumber,LOW);
delay(1000);
digitalWrite(roomnumber,HIGH);


      }



    }
    void all_led_blink()
    {
      for(int i= 0; i< 5;i++)
      {
digitalWrite(room3,HIGH);
digitalWrite(room1,HIGH);
digitalWrite(room2,HIGH);
delay(1000);
digitalWrite(room3,LOW);
digitalWrite(room2,LOW);
digitalWrite(room1,LOW);
delay(1000);


        }
```

Figure 30

Predefining the voice commands

The voice commands in this project are very cardinal since they are responsible for controlling the lights. How does this work? The voice input fed into the android application from the user is converted to a string (text/sentence) by the voice recognition application (Google speech) which then sends this to the Web server over the internet through the local WIFI networks IP address. For the user to expect a positive response from the system, the voice input is supposed to match the commands predefined on the microcontroller. Random voice inputs will not work since they will not match with the predefined commands on the system. The choice of commands predefined on the microcontroller matters, below are some of the factors we have to put into consideration:

- **Language** is an important factor to consider since the voice recognition application is set to comprehend English. So the string of commands should be in English.
- We should avoid the use of **homophones.** A homophone is a word that has the same pronunciation as another word but has a different meaning and/or spelling e.g. “two “and “to”. When a homophone is used, the recognition application may convert the voice input from user into a sting that does not match the one predefined on the microcontroller.
- The string of commands should be short and precise so that the voice recognition application can convert the voice input from the user in a short space of time.
- The spacing used when predefining the string of commands should be no more than 20% or one space .Anything more than that would mean that the user should be pause for some time after every word uttered. This would cause confusion and the input would not match the predefined commands, hence LEDs won’t respond.

To get started with setting up the voice commands, we first have to initialise variable of data type string. This variable stores the voice commands.

```
11 const char* password = "12345678";
12 String voicecommand;
13 WiFiServer server(80);
14 WiFiClient client;
15
```

Checking and comparing the voice commands

We have to make a function that checks for incoming requests from the android device. This function reads incoming string commands that are only in one sentence and the sentence should only be in one line. This is because the voice recognition application converts the voice from the user into a string without adding any newline characters to it such as ‘\r’, ‘\n’ or ‘endl’. So the function is designed to read the string only up to the end of the line. Once that check is passed, the HTTP request of the command is checked. The web server only receives the commands having the index HTPP/1.1 in them. After that string returned in a

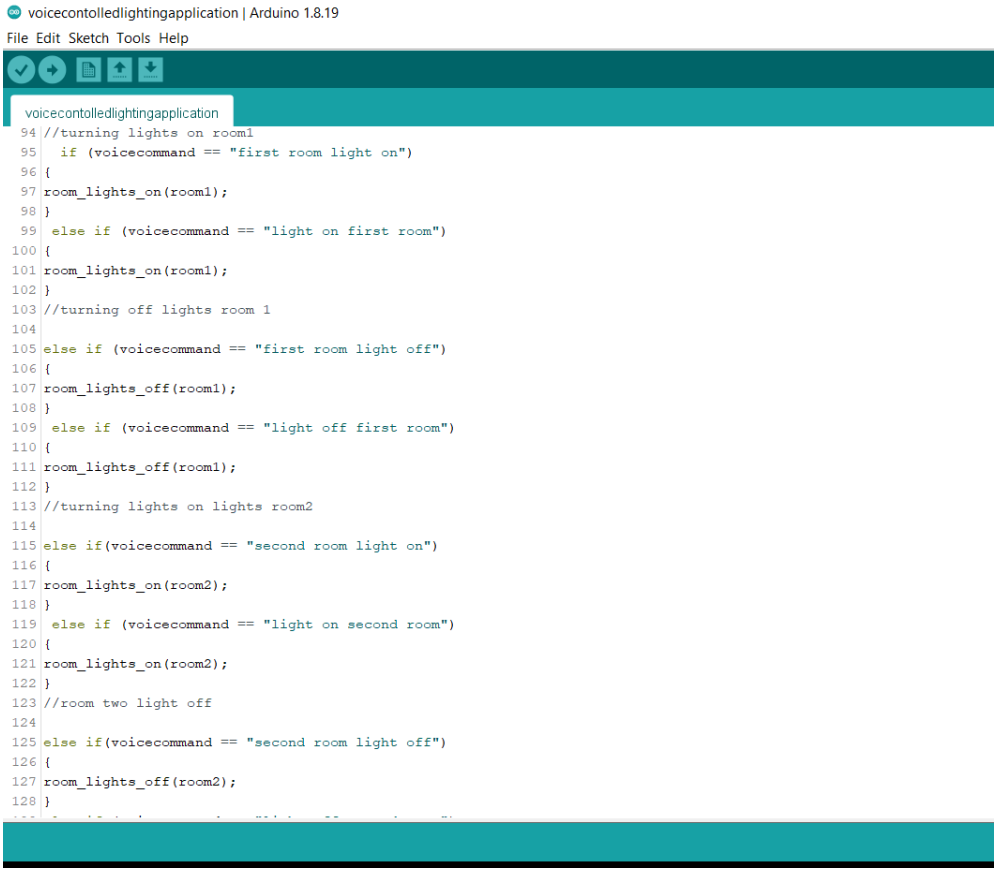
variable and the variable storing it is what will be used to compare with the predefined commands on the microcontroller.

```
15
16 String result;
17
18 String IncomingRequest()
19 {
20
21 while(client.available())
22 {
23 voicecommand = (client.readStringUntil('\r'));
24 if ((voicecommand.indexOf("HTTP/1.1")>0))
25 result = voicecommand;
26 }
27
28 return result;
29 }
30
```

In the loop function of the Arduino IDE, the first thing which is done is to check if there is a client (mobile device) connected to the server. If there is a client connected to the server, the string variable that stores the voice commands is initialised and equal to the string coming from the client; in this case we call the function that receives incoming requests. For as long as the device is connected to the server and the user keeps sending commands, the loop will keep running and executing. The loop only terminates when the client disconnects from the server.

```
84 void loop() {
85
86 client = server.available();
87 if (!client)
88 return;
89 while(!client.available())
90 delay(1);
91 voicecommand = (IncomingRequest());
92 voicecommand.remove(0, 5);
93 voicecommand.remove(voicecommand.length()-9,9);
```

After the string holding the voice commands from the client is initialised, it is compared with the predefined strings on the microcontroller. The comparison is done using IF ELSE statement. If the strings in the condition are equal or in this case match, a function is called to perform the users' desired task. If the string from the client does not match the string in the conditional statement, it is sent to the next conditional statement and this will go on until all of them are checked. The user will not see any response from the LED's if voice command sent does not much with the predefined strings in the conditional statements.



To aid users who may not have the ability to see, an echo of the predefined string that matches the users voice input is sent back to the mobile device from the server. A function to send back the echo is called and sends the string through the index HTPP/1.1.

```
    }
    void Echofeedback(String echo)
{
  client.println("HTTP/1.1 200 OK ");
  client.println("Content-Type: text/html");
  client.println("");
  client.println("<!DOCTYPE HTML>");
  client.println("<html>");
  client.println(echo);
  client.println("</html>");
  client.stop();
  delay(1);
}

```

And for those that have the ability to see, an error message will be sent to the app if there is a problem in connection between the application and the server. The codes responsible for sending this text are shown below.

```
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println("");
client.println("");
client.println("");
client.println("OK");
client.println("");
client.flush();
client.stop();
delay(1);
}
```

```
Done compiling.

Sketch uses 655246 bytes (49%) of program storage spac
Global variables use 38012 bytes (11%) of dynamic memo
```

The code can now be uploaded on the microcontroller. When uploading the code on the microcontroller, caution must be taken that nothing should be connected to its GPIO pins.

Voice recognition application development

After all the circuit connections and codes have been uploaded to the microcontroller, the voice recognition application will have to be developed for the system to be complete. Like earlier mentioned, the voice recognition application has a front end and the back end.

Front End development

The front end of the voice recognition application is the part of the application that the user interacts with and it is called the User interface. The voice recognition application operates on a graphical user interface.

The graphical user interface will have a simple design that the user can easily navigate and perform the task in the shortest possible time. This is to emulate the use of a physical switch in a regular building. For a physical switch in a building, what is important to know the location and you can perform the desired task.

Navigation of the user interface

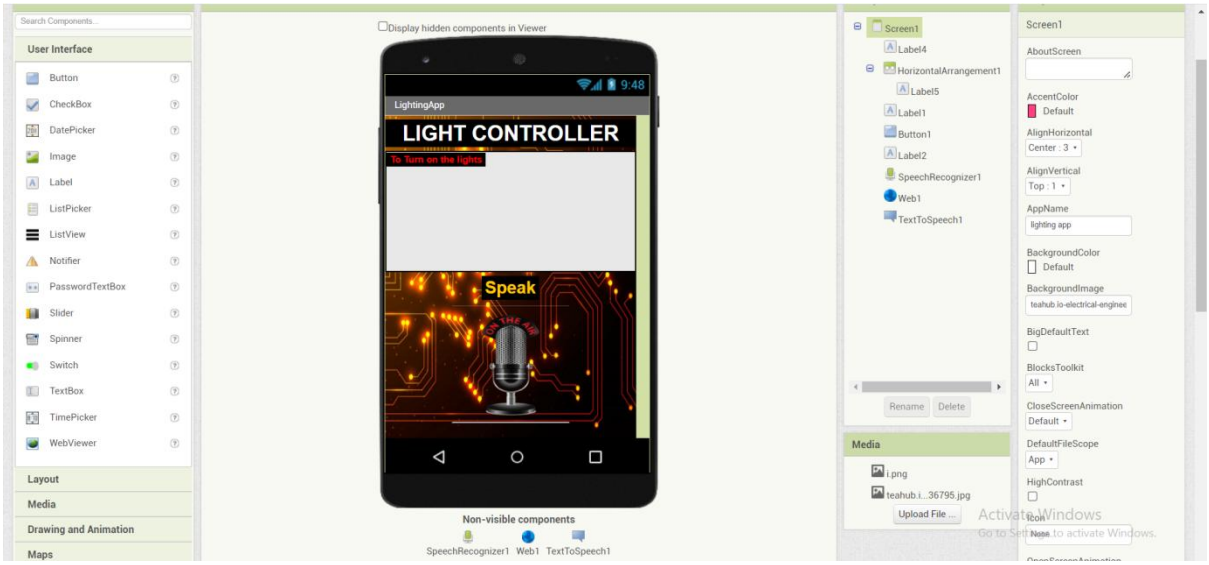
- The application will have a list of commands for specific tasks displayed and the user can pick from that list what command they will use as voice input.
- The application will have a button that when the user clicks on, the Google voice recognition pops up.

The user interface of this application is designed using MIT App Inventor. To add features to the interface, you simply just pick a component from the palette and place it on the screen of the android device emulated.

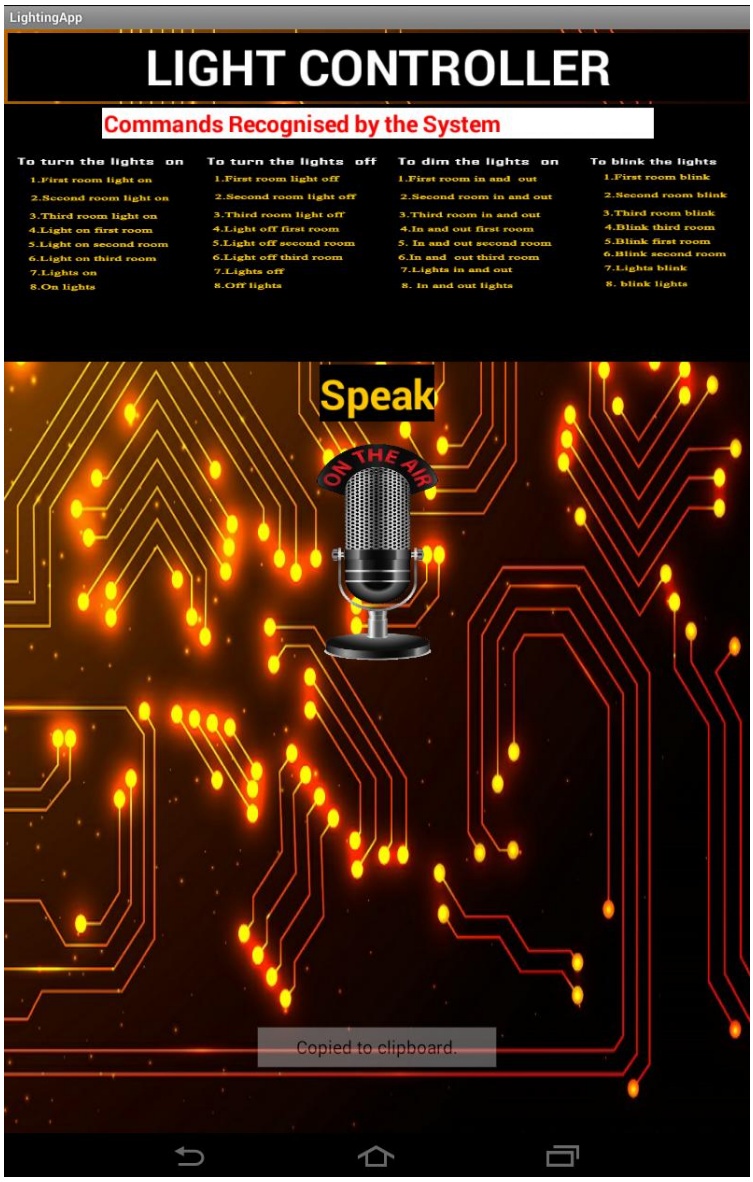
The components used to build the interface are:

- A background image
- A label that is used to display applications heading or welcome Title

- A text box in read only format that displays the list of commands
- A button with a microphone icon on it



Development of user interface



User interface of the application

Back End development

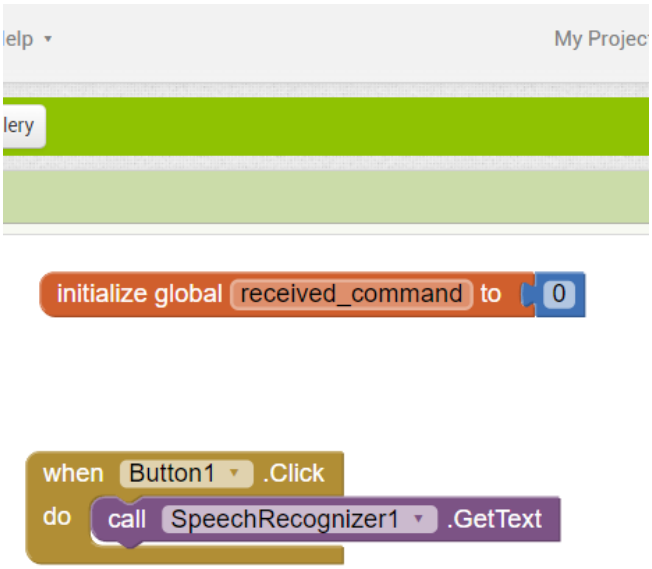
The backend of the application is responsible for running the operations that make it functional. In this application, the operations that need to be handled are:

- The conversion of the voice input to a string
- The application being able to use WIFI connectivity of the mobile device
- The application should be able to send the string of commands to Esp32 web server over the internet
- The application should be able to receive strings from the web server
- The application should be able to convert strings received from the web server to a audio output
- When a button is clicked the speech recognizer should pop up

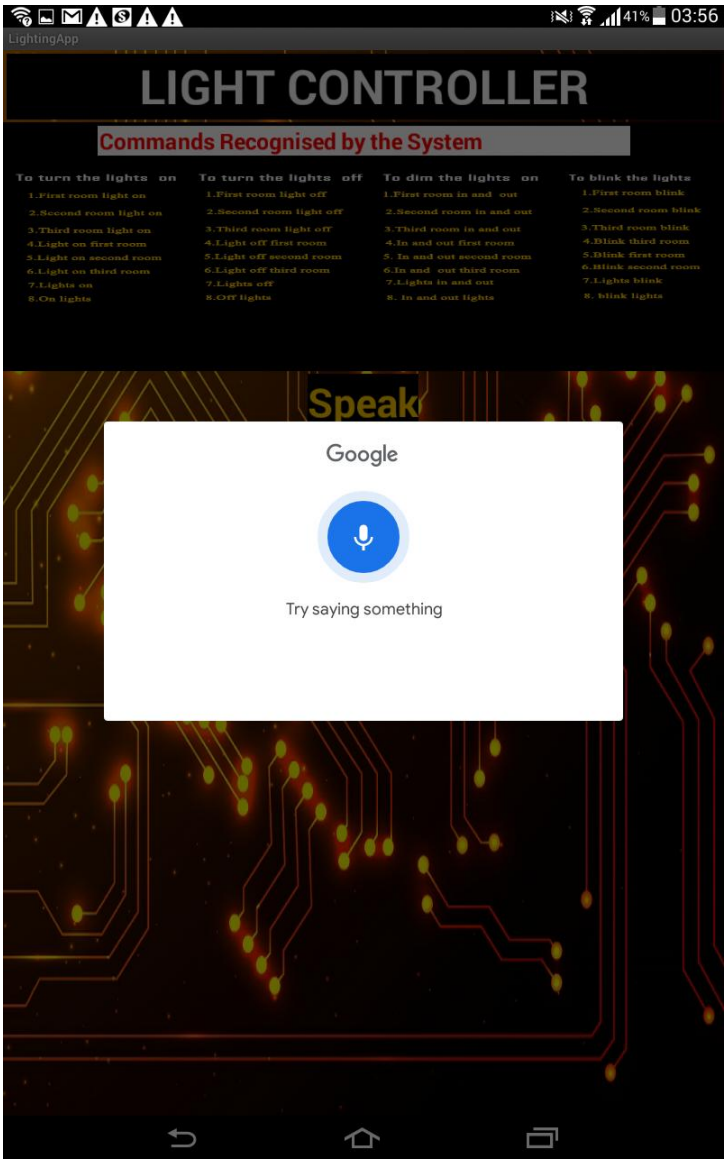
MIT App Inventor was used to develop the backend of this application.

Giving functionality to the button

When the button is clicked the Google speech recogniser should pop up. The non visible component that is responsible for carrying out this action is the speech recogniser. To achieve this we use a control structure which has a do while loop in the bock. So we place the button in the block which calls the speech recogniser component. The figure below represents how the code is executed (when the button is clicked, open Google speech recognition).



The code that gives the button functionality



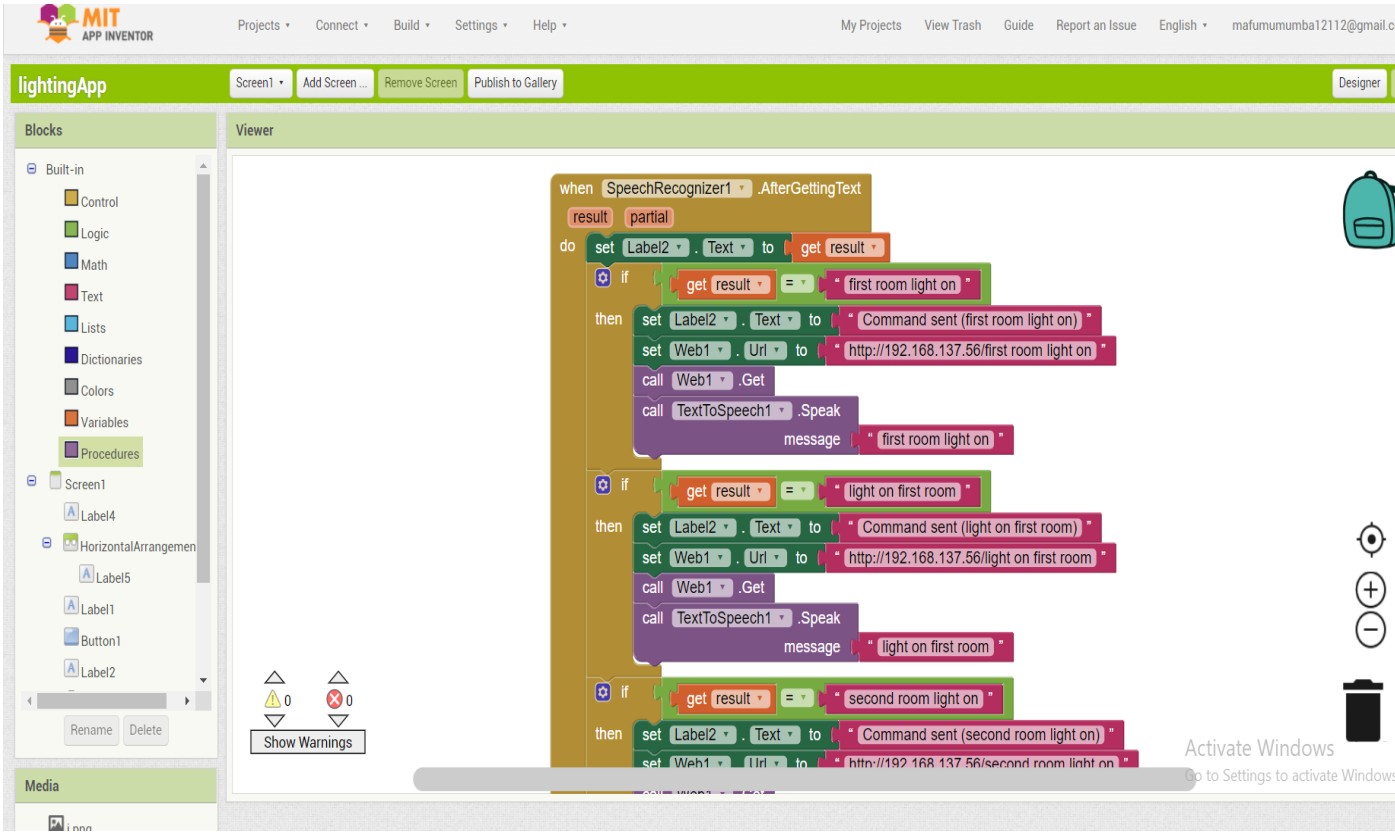
When button is clicked

Converting voice to string

When the user speaks through the voice recognition app, the speech is going to be converted to a string by the speech recognizer. How this works is, the speech recogniser will pick up the voice from the user which is an analogue signal and coverts it to a digital signal using the ADC (analogue to digital converter).The analogue to digital converter measures the waves of the sound in the audio file in great detail and filters them out only living relevant sounds. The sounds are then divided into multiple segments and then matched to units of sounds that distinguish one word from another in the English language. These units are called phonemes. The phonemes are then run through a network containing a list of well-known phrases, sentences and words, and then compared. Voice recognition applications installed on mobile devices operate on technology called speaker-independent.

Sending the string to web server

Once the speech recogniser has successfully converted the voice input to a string, the string is stored in a variable called result the when do block. The result put in a conditional statement which when true, the string is set to a text held in a label which then triggers a component called web. The web component gets access to the WIFI connection of the mobile device and connects the application to the internet. It is in this part were set the IP address of the local WIFI network that the router is connected to. The web component sends the string of commands to the web server using the IP address as the URL. Each string converted is run through this procedure by a couple of 'if then else statements'.



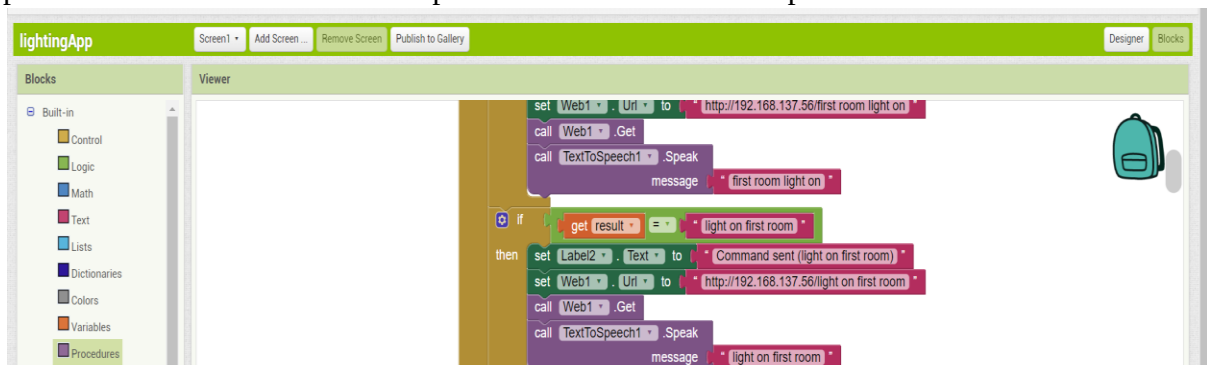
Converting the string to audio output

So once the string of voice commands from the user is sent to the web server and it matches with the predefined strings on the microcontroller, the microcontroller sends the matched string back to the voice recognition application as confirmation. When the application receives the string, it is supposed to alert the user in the form of audio output, which in this case an automated voice is reading the string out loud. The component responsible for carrying out this task is called (TTS) TextToSpeech.

For the string to audio conversion to take place, theses are processes performed:

- **Normalisation-** this process involves narrowing down many different methods of reading text to the most appropriate one. This done because unlike humans, computers do not have the sense to figure out pronunciations. So they arrive at the most likely pronunciation using neural networks.
- After normalisation, the words are converted to English phonemes.

- The phonemes are then matched with multiple similar sounding human voices pre-recorded and installed on the pc and sent out as audio outputs



All the blocks containing the commands in MIT App inventor are connected to each other in the form of a jigsaw puzzle. After all the codes have been written, they are compiled and executed into android APK file. This file can then be downloaded and installed on the mobile device.

Testing

After hardware and software development process is done, how the two coordinate has to be checked and if there are any faults in the system, it is off this analysis that they will be corrected. The three testing techniques that will are:

- White box testing
- Black box testing
- Functionality testing

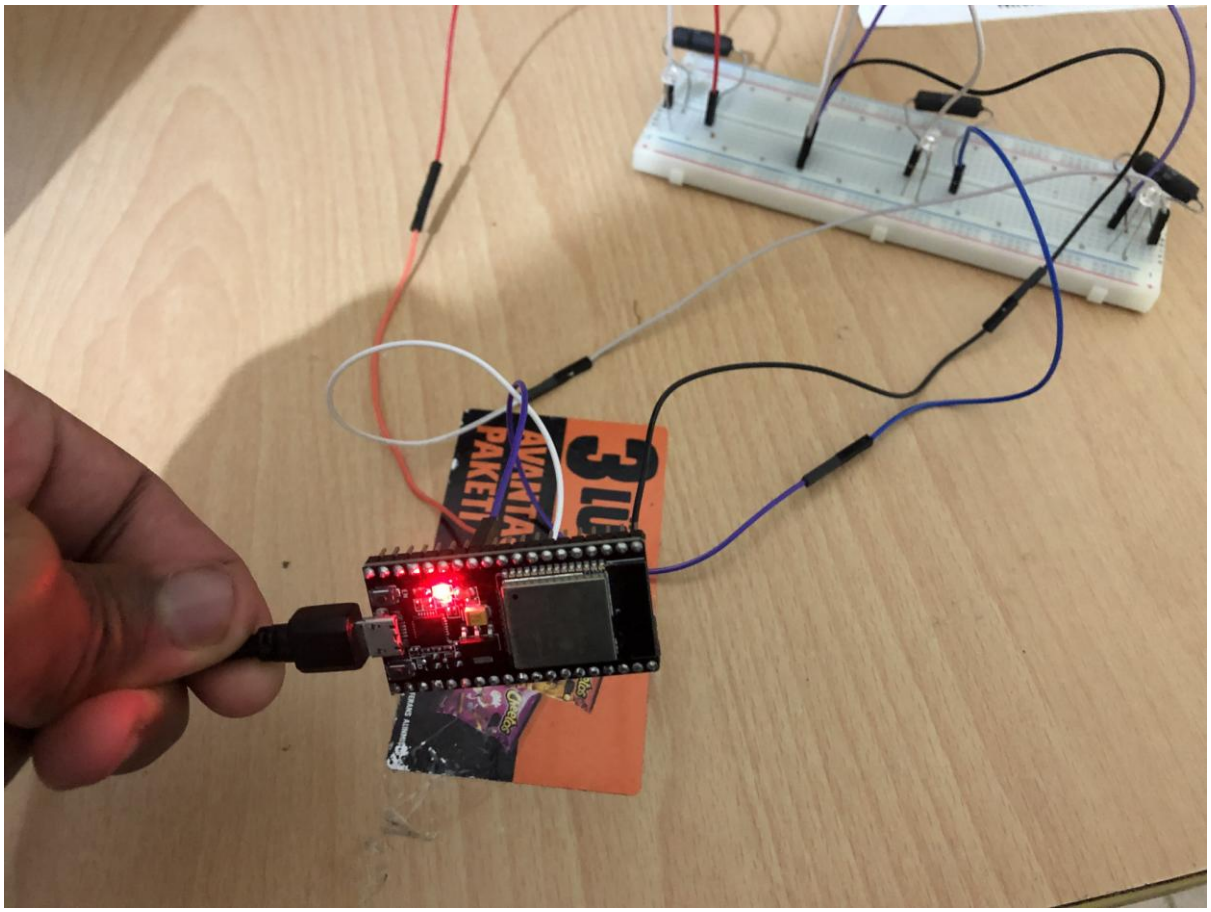
White Box Testing

When testing the system, the major error that was encountered was that the voice recognition application was having trouble with classifying homophones hence the LEDs did not respond because the commands were not matching. For example the user says the word ‘two’ it was being converted to a string as ‘to’ which is not predefined on the microcontroller. So the internal structure of the code stored on the micro controller had to be inspected and all strings containing homophonic words had to be replaced non-homophonic words.

The other error encountered was that predefined strings having more than one space in the sentence where not matching with the strings coming from android device. So the called on the microcontroller had to be adjusted.

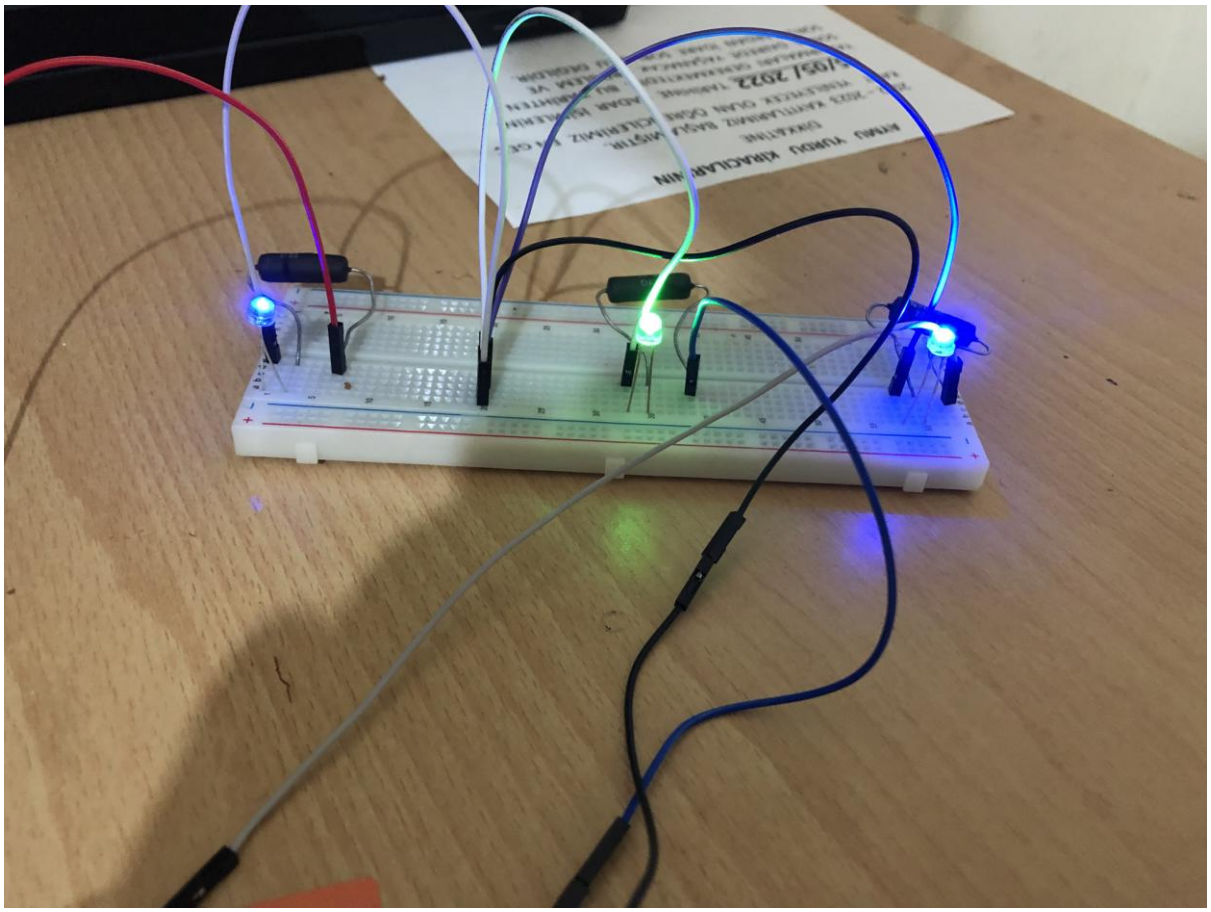
Black Box testing

In this stage, the hardware of the system was inspected to see if everything is working fine. For example, when NODEMCU-ESP2 is connected to the power source a red light must flush to represent that it is working. The applications communication with hardware was also tested, one way of carrying out this test was by saying something like “lights on first room” and the led light went on.



Functionality Testing

Each and every light controlling feature of the system such as blinking, dimming and turning on and off was tested to see if they are all working. For example in the image below shows that the lights went on the user said lights on through the voice recognition application.



6. Conclusion

6.1Summary

This report represent represents the practical and theoretical success of the system. The design of the system clearly illustrates how lights can be controlled using a users voice input with the aid of a mobile device. This project highlights how the concept of embedded systems can be applied and useful in our modern day society. This project also showcases how the internet is a multidimensional tool and shows how it is not only limited to activities such as social media.

6.2Benefits

a. Benefits to the user:

Below shows how the voice controlled lighting system is beneficial to users.

1. **Time saving:** In large buildings having a lot of rooms and lights, the process of controlling lights is quite tedious ad taxing for it requires a user to navigate where every switch for each light is located. This results in a lot of time being spent to perform this task. This system makes that process easier because the user can perform this task remotely over the internet. For example, in the case it is an emergency this task will be performed in the shortest possible time.
2. **Cost effective:** In large setups such as hotels that operate in large like skyscrapers, if the task of controlling lights is carried out manually, it would require a lot of

manpower which in this case is employees. Having a lot of employees in a business means a large sum of funds has to be allocated towards wages. The fact the voice controlled lighting system is automated, firms can cut down on having a large number of employees hence saving costs on wages.

3. **Disabled people:** People with certain disabilities usually encounter challenges with performing most tasks and controlling lights is one of them. Traditional switches are placed in locations were only an average person can reach. There are some instances where the position of the switch may be too high and someone who is shorter than usual cannot reach it, the switch maybe located in such a confined place that someone who blind will have difficulties and there are also households that only occupied by people of old age. So to some of the types people described, performing the task of controlling lights would be very difficult or near to impossible. The voice controlled lighting application puts into consideration user that have disabilities but can speak. This would be very beneficial to people with disabilities.
4. **Job creation for disabled people:** Most job descriptions only favour people with full physical abilities due to the manual input required to perform a task. But since the voice controlled lighting system is automated, disabled people would also stand chance in getting jobs that involve automated input.
5. **Energy saving:** Certain feature of this system such as dimming can be used to reduce on power consumption. Sometimes the light in a room maybe too bright but usually the traditional switch rarely provides the option to reduce the light intensity. This may also be convenient for users who are sensitive to very bright light.

b.Benefits to me:

Below are some of the ways in which taking part in this projected has benefited me.

1. **Mind opening:** A few years ago the concept of embedded systems was very strange to me. I could not even fathom the working concept behind systems such as automated doors, flame sensors, washing machines etc. The thorough research involved in making this project come to life has made it all make sense to me and has laid a strong foundation on my knowledge on embedded systems. The idea of this project came to me when I realised how most of the world population owns a mobile device and has internet access. So if a mobile device is so close and dear to us, why not use it to perform daily tasks such as controlling lights. The coming of COVID 19 also made me realise how the world can stay functional with the use of the internet.
2. **Career path:** Working on this project has made me develop an interest in automation systems and I am even more fascinated to know how other automated systems work. I see myself working on more automated projects and this has made me realise the role that I can play in the technology industry. As a professional, I would prefer that this field is my specialty.
3. **Satisfaction:** The fact this project can make difference in people's live such as disabled people proves that engineering is used for the right reasons, it can make the world a better place by improving the quality of living in societies.

6.3Ethics

1. Since this project involves hardware, safety precautions are followed very carefully. The circuit will be designed and connected correctly to avoid

accidents. Accidents would occur as are result of wrong connections which may cause short circuits. To avoid short circuits, we ensure that all the wires that are used a properly insulated and tightly connected. We will also ensure that that the area in which the system is being tested is free and clear from foreign electrical conductor that may come into contact with circuit and cause sparks. We have to bear in mind that short circuits very dangerous in the sense that the sparks produced may cause a fire which can inflict physical harm on the user and also damage components. This is according to the Engineering Code of Ethics, under the Fundamental cannons section which states, Engineers, in the fulfilment of their professional duties, shall: Hold paramount the safety, health, and welfare of the public.

2. All the information present about this project is well researched and is within my knowledge areas. Extra effort was put in to consult other professionals such as professors and lecturers on issues that i felt uncertain, hence i can confidently sign of this project. This was done in accordance with the Code of Ethics for Engineers, Rules of Practice, part two; Engineers shall perform services only in the areas of their competence and furthermore section 'a' states; Engineers shall undertake assignments only when qualified by education or experience in the specific technical fields involved.
3. We also ensure that the system does not pollute the environment. One way it could do so is through noise pollution from the mobile device. To avoid this, we ensure that audio output from the mobile device be kept to a normal standard using volume controls. This is according to the Code of Ethics for Engineers, Professional obligations number 3, section 2; Engineers shall at all times strive to serve the public interest, part d; Engineers are encouraged to adhere to the principles of sustainable development in order to protect the environment for future generations .
4. Since the system works with the internet, personal or private information will not be collected from the user and the application will be made secure enough that it does not redirect the user to illicit sites on the internet. This is because the safety of the user is first priority. This strengthened by the Engineering Code of Ethics, under the Fundamental cannons section which states, Engineers, in the fulfilment of their professional duties, shall: Hold paramount the safety, health, and welfare of the public.

Why did I choose this project?

In the year 2020 the world was hit by a pandemic called COVID 19. This pandemic put the world to a standstill as physical interactions or activities would speed up the spread of the virus. What kept most of these institutions such as schools, firms and banks running was the internet. The internet proved to be a very vast network that could connect the world in real time without individuals being physically present at a specific geographical location. Students attended classes online, employees were able to work and attend meetings remotely online. It is off these circumstances that I took keen interest to research on how automation systems can be incorporated with the internet and to be specific, how lights could be controlled remotely using voice commands.

6.4Future works

Working on this project has made me realise how the concept of automation systems interests me hence my career path is headed towards this direction. So in future, from the experience and knowledge I will acquire from the technology industry, I will add more features to the system such as controlling home appliances and since it only considers people who can speak, improvements will be made so dumb people can also use it. The goal will be to make it a fully functional home automation system. So the answer is yes I will continue working in this project.

7. REFERENCES

Books:

[1] NSPE Code of Ethics for Engineers, ©2022 National Society of Professional Engineers | 1420 King St., Alexandria, VA 22314 | 888-285-NSPE (6773)

[2] A.M. K. Rupam Kumar Sharma, "Android interface based GSM home security system," in IEEE, Ghaziabad, India, 7-8 Feb. 2014.

[3] R. H. R. Bikash Agrawal, "SD-HDFS: Secure Deletion in Hadoop Distributed File System," in IEEE, San Francisco, CA, USA, 06 October 2016.

Webs:

[4] https://www.itwissen.info/en/konnex-KNX-117263.html#gsc.tab=0

[5] https://techtutorialsx.com/2019/04/07/esp32-https-web-server/

[6] https://randomnerdtutorials.com/esp32-pinout-reference-gpios/

[7] https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/

[8]https://www.instructables.com/Blinking-an-LED-With-ESP32/#:~:text=Connect%20the%20negative%20pin%20(cathode,shown%20as%20the%20red%20wire.

[9]https://aws.amazon.com/what-is/speech-to-text/#:~:text=Let's%20take%20a%20closer%20look,an%20analog%20to%20digital%20converter.

[10] https://www.explainthatstuff.com/how-speech-synthesis-works.html

[11] https://electropeak.com/learn/speak-to-arduino-control-all-parts-by-google-assistant/