SMART SWITCH USING ANDROID APPLICATION AND NODEMCU ESP8266

A comprehensive project report has been submitted in partial fulfillment of the requirements for the three months of IoT internship

IOT RESEARCH LAB , ilabAfrica

This is to certify that the project titled SMART SWITCH USING ANDROID APPLICATION AND NODEMCU ESP82
carried out by:

NAME	PHONE NUMBER
Dancun Sikuku J	0799045392
Karen Kiprono	0721590695
Cynthia Njoki	0797181989

Under the supervision of:

NAME	PHONE NUMBER
Solomon Kamau	
Stephen Ng'etich	
Leonard Mabele	

DECLARATION

We declare that this is our own work conformed to the norms and guidelines given in the Ethical Code of Conduct .

1. Dancun Sikuku
2. Cynthia Njoki
3. Caren Kiprono

ABSTRACT

The main objective of this project is to develop a SMART SWITCH USING ANDROID APPLICATION AND NODEMCU ESP8266. Using a Nodemcu Esp8266 board being remotely controlled by any Android OS smart phone.As technology is advancing so houses are also getting smarter. Modern houses are gradually shifting from conventional switches to centralized control systems, involving remote controlled switches. Presently, conventional wall switches located in different parts of the house makes it difficult for the user to go near them to operate. Even more it becomes more difficult for the elderly or physically handicapped people to do so. Smart Switch system provides a most modern solution with smartphones. In order to achieve this, a Nodemcu board at the receiver end while on the transmitter end, a GUI application on the cell phone sends ON/OFF commands to the receiver where loads are connected. By touching the specified location on the GUI, the loads can be turned ON/OFF remotely through this technology.

LIST OF FIGURES

Project description

Developing a smart lighting system which is controlled by an android application over WiFi using HTTP protocol.

INTRODUCTION

Nowadays, we have remote controls for our television sets and other electronic systems, which have made our lives real easy. Have you ever wondered about smart switches which would give the facility of controlling tube lights, fans and other electrical appliances at home using a remote control? Off-course, Yes! But, are the available options cost-effective? If the answer is No, we have found a solution to it. We have come up with a new system called SMART SWITCH USING ANDROID APPLICATION AND NODEMCU ESP8266. This system is super-cost effective and can give the user the ability to control lights without even spending for a remote control. This project helps the user to control all the lights using his/her smartphone. Time is a very valuable thing. Everybody wants to save time as much as they can. New technologies are being introduced to save our time. To save people's time we are introducing a SMART SWITCH USING ANDROID APPLICATION AND NODEMCU ESP8266. . With the help of this system you can control your home lights from your android OS mobile phone. You can turn on/off your home lights over the internet.

COMPONENTS REQUIRED

1.1 Hardware requirements

- Nodemcu ESP8266
- Bread board
- Leds
- Jumper wires
- USB cable
- Resistor
- Bulbs(220V)
- Channel Relay
- Power supply
- Smart Phone

1.2 Software requirements

- Arduino IDE
- Android Studio
- Firebase (Real Time Database)

DESCRIPTION

NodeMCU ESP8266

NodeMCU is an open source firmware for which open source prototyping board designs are available. The name "NodeMCU" combines "node" and "MCU" (micro-controller unit).The term "NodeMCU" strictly speaking refers to the firmware rather than the associated development kits.

Both the firmware and prototyping board designs are open source.

The **NodeMCU ESP8266 development board** comes with the ESP-12E module containing ESP8266 chip having Tensilica Xtensa 32-bit LX106 RISC microprocessor. This microprocessor supports RTOS and operates at 80MHz to 160 MHz adjustable clock frequency. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects.

NodeMCU can be powered using Micro USB jack and VIN pin (External Supply Pin). It supports UART, SPI, and I2C interface.

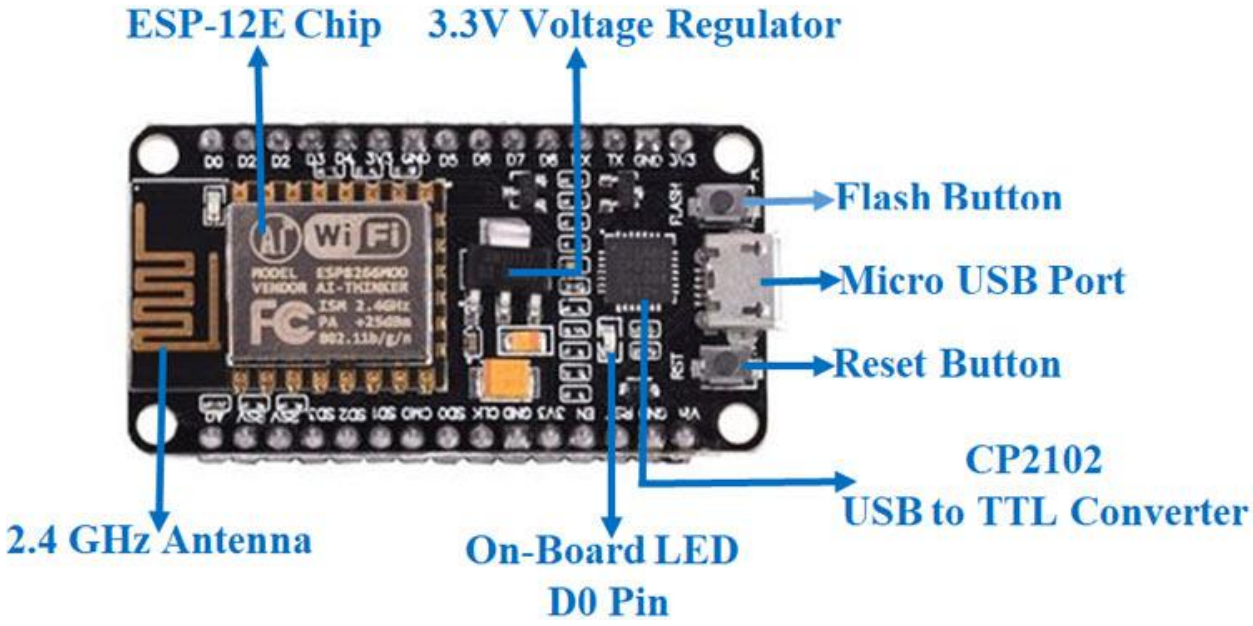

FIG 1 :NodeMCU ESP8266

Features of NodeMCU ESP8266

- Processor: L106 32-bit RISC microprocessor core based on the Tensilica Xtensa Diamond Standard 106Micro running at 80 MHz.
- Memory: ...
- External QSPI flash: up to 16 MiB is supported (512 KiB to 4 MiB typically included)

- IEEE 802.11 b/g/n Wi-Fi. ...
- 17 GPIO pins.
- SPI.
- I²C (software implementation)

Arduino IDE

The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. The source code for the IDE is released under the GNU General Public License, version 2. The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

Sketch

A program written with the Arduino IDE is called a sketch. Sketches are saved on the development computer as text files with the file extension .ino. Arduino Software (IDE) pre-1.0 saved sketches with the extension .pde. 16 A minimal Arduino C/C++ program consists of only two functions: setup(): This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch. loop(): After setup() has been called, function loop() is executed repeatedly in the main program. It controls the board until the board is powered off or is reset.

A minimal Arduino C/C++ program consist of only two functions:

setup(): This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch.

loop(): After setup() has been called, function loop() is executed repeatedly in the main program. It controls the board until the board is powered off or is reset.

RELAY:

A relay is an electrically operated switch. Many relays use an electromagnet to mechanically operate a switch, but other operating principles are also used, such as solid-state relays. Relays are used where it is necessary to control a circuit by a separate low-power signal, or where several circuits must be controlled by one signal. The first relays were used in long distance telegraph circuits as amplifiers: they repeated the signal coming in from one circuit and re-transmitted it on another circuit. Relays were used extensively in telephone exchanges and early computers to perform logical operations.

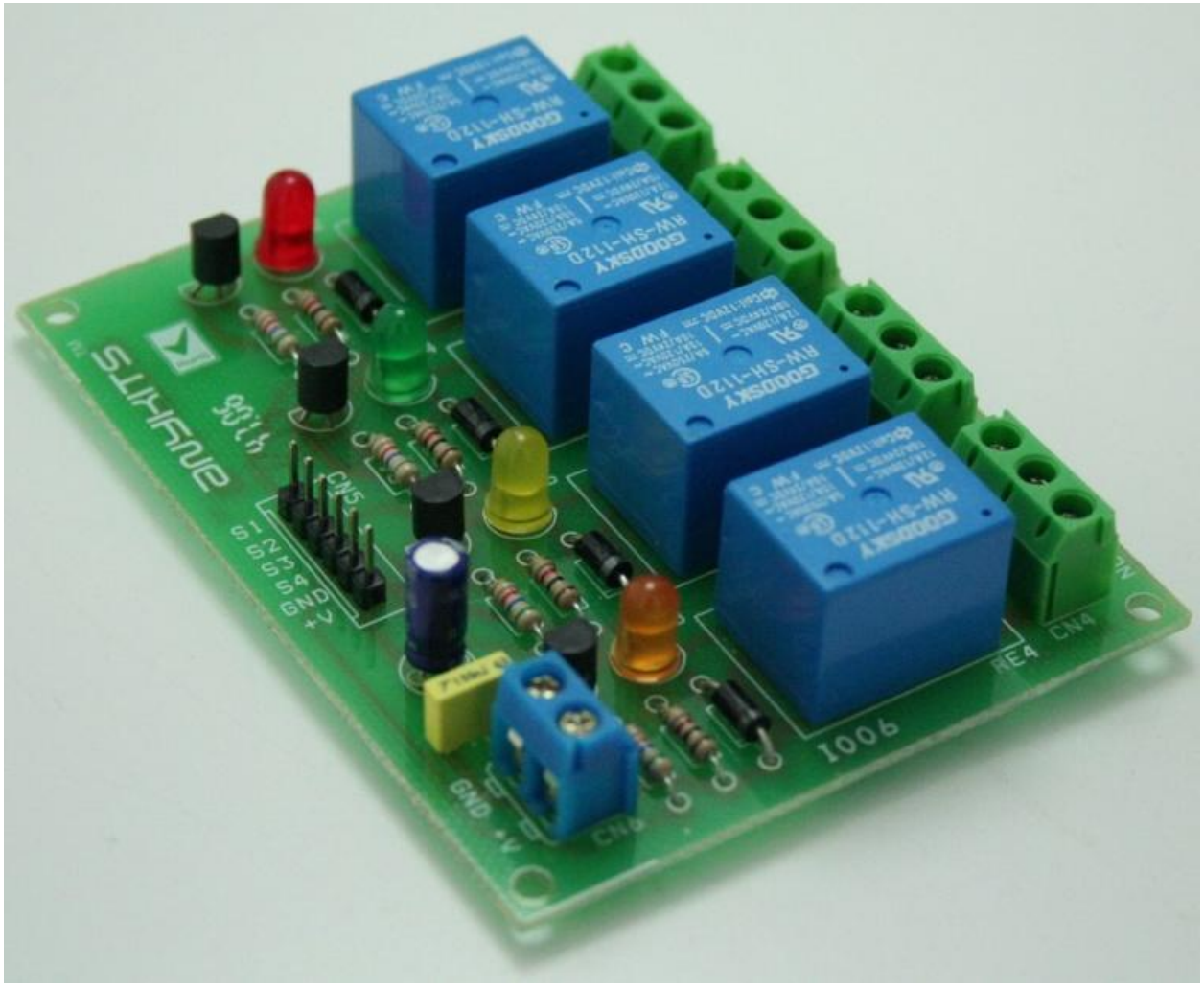

FIG 2 :4 CHANNEL RELAY

Features

- Input supply 12 VDC @ 170 mA
- Output four SPDT relay
- Relay specification 5 A @ 230 VAC

- Trigger level 2 ~ 5 VDC
- Berg pins for connecting power and trigger voltage
- LED on each channel indicates relay status
- Power Battery Terminal (PBT) for easy relay output and aux power connection
- Four mounting holes of 3.2 mm each
- PCB dimensions 88 mm x 68 mm

Design an Electronic Circuit using Fritzing

Fritzing is a completely free Circuit Design program available across all operating system
This project will take you through the process of creating a circuit from the initial design to a complete circuit.
This includes building a prototype on a breadboard, creating a schematic diagram and then designing a printed circuit board (PCB) which is then sent off for fabrication.
This also includes the soldering of the board along with the software.

Core parts of fritzing project

- Breadboard
- Resistor
- Jumper wires
- LEDs
- Esp 8266 module

Process of designing the PCB

- You take your working prototyped circuit setup, your bird's nest of jumper cables and your attached peripherals and head right into the breadboard view of a new fritzing sketch.
- Once there you can drop-n-drop any number of parts from the "bins" of parts e.g LEDs, resistor & readboard. If the exact part you are using doesn't exist in the parts bins, In my case I added esp module in my mine part because it was not there.
- When you have all your parts placed in your workspace, I imported an Arduino import microcontroller.
- Connect up the relevant parts with jumper wires by clicking and dragging on the component legs.
- When you create a replica of your circuit in the breadboard view of Fritzing you can transition between breadboard, schematic and PCB views.
- Jumper cable and breadboard connections will be represented as faint traces within these two views and the AutoRoute function can be used to fill them in.

- You can now export your 'sketch' in a number of ways:
- As an Image (PNG, JPG, SVG, PDF)
- As an Etchable PDF, SVG or Gerber file for production
- Generate a List of Materials (.html file)

Getting around the PCB View

- You will see a dark gray grid workspace, the black outline of your Arduino and a gray box peppered with labeled yellow holes. The gray box is your PCB workspace; you can resize it by dragging the corners in/out.

- If you look at any component's 'mask' we notice it has a thin, dashed line connecting its legs to where it was connected on your breadboard prototype. These are known as rats-nest connections within circuit design systems. They represent a physical connection between components, in this case, they are imported from our breadboard view.

- Get started rotating and maneuvering your components around your PCB. Try to 'untangle' your rats' nest traces as much as possible to make later steps faster.

BREADBOARD DIAGRAM

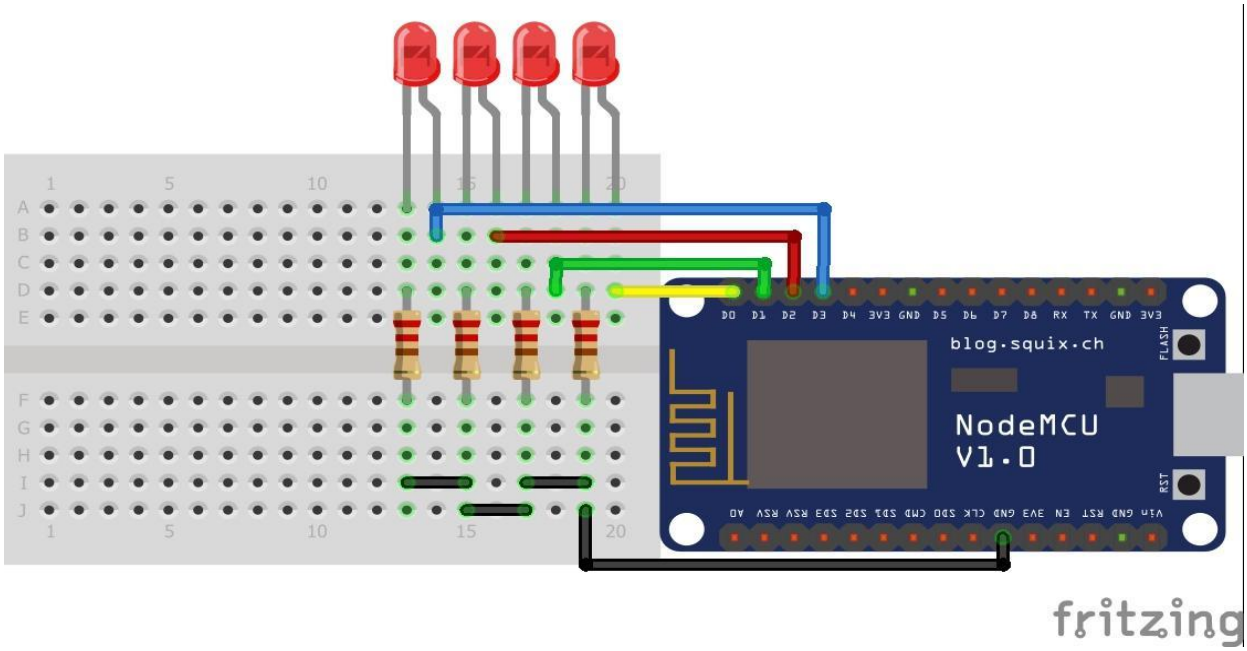

FIG 3:BREADBOARD DIAGRAM

SCHEMATIC DIAGRAM

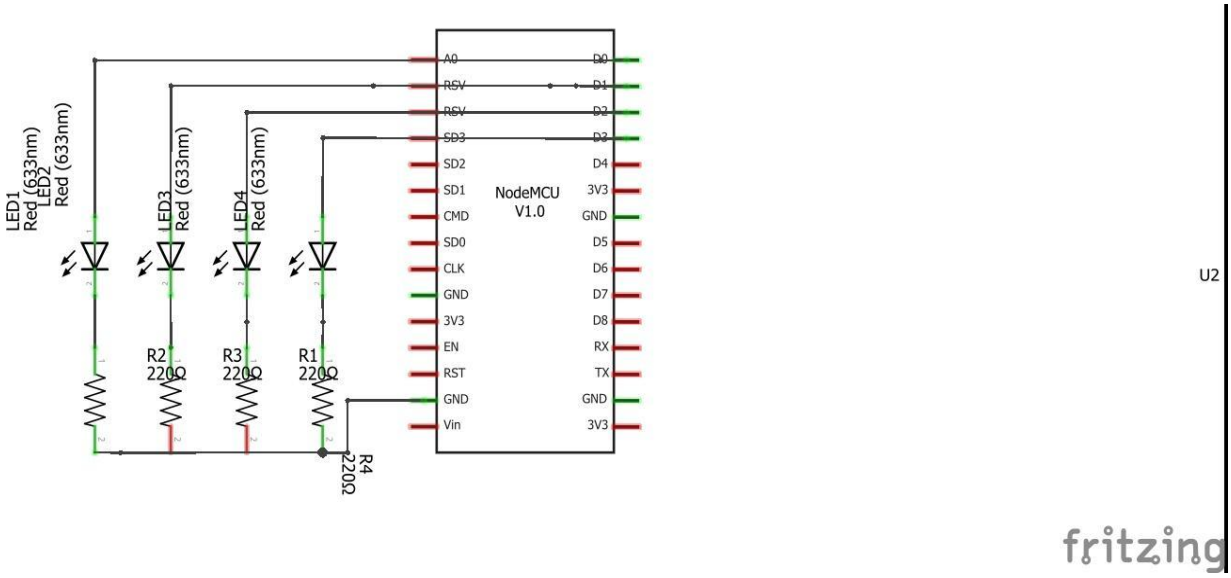

FIG 4:SCHEMATIC DIAGRAM

PCB DIAGRAM

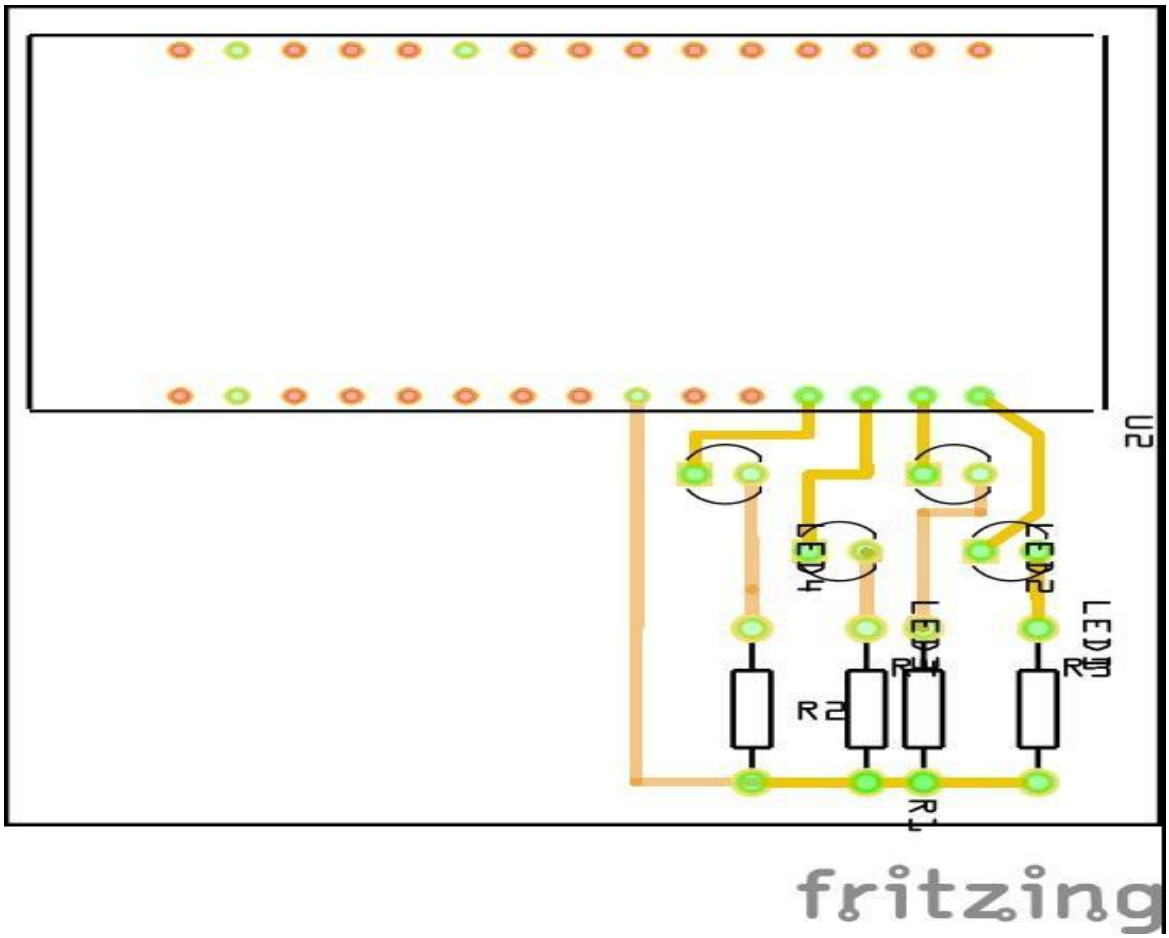

FIG 5:PCB DIAGRAM

SYSTEM DESIGN

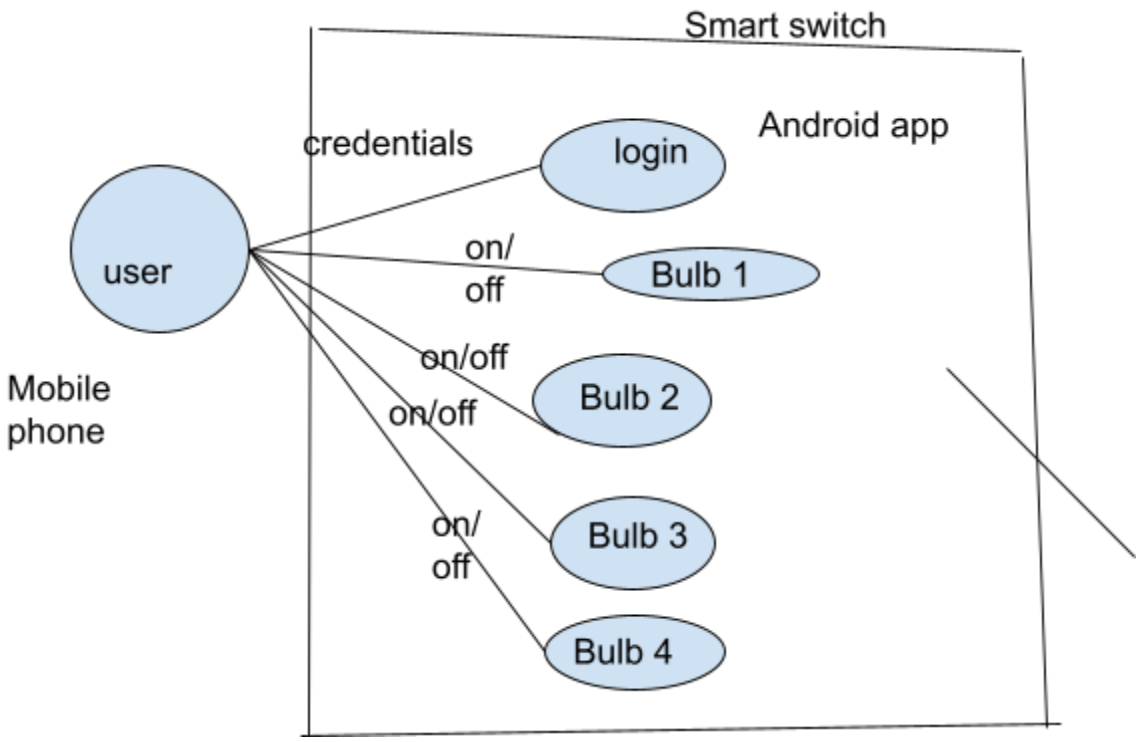

FIG 6 USE CASE DIAGRAM

Use case

Here we have explained how the users can interact with the system. This will help users to troubleshoot

some of the important things like login, new user registration. Below is our user manual:

User registration

For the first time users need to register themselves to access the system.

After registering the user needs to login to access the credentials.Below is how the login panel looks like

After the successful login the system will direct the user to the bulb menu.

Bulbs control

Challenges Faced During Implementation

Challenge	Solution.
Android application crashes when an attempt to send data to the realtime database is made.	Allowing read and write on the Firebase realtime database.
kFirebaseFingerprint error.	Updating Arduino IDE to version 1.8.9 and downgrading the firebase arduino library to version 1.0.3
bool begin(String host, uint16_t port, String uri, const uint8_t httpsFingerprint[20])	Updating the Firebase arduino fingerprint using the link: https://www.grc.com/fingerprints.htm By copying the firebase realtime database url and extracting the fingerprint using the above link.

Changing finger prints visit https://github.com/FirebaseExtended/firebase-arduino/pull/507/commits/4407bf1a8a0d68a186fbfb96274204ae345ceb81

Copy the //2021-01 version value of the fingerprint and replace it in firebasehttpclient.h file in your arduino library.

CODE

The code for this project can be found on github
https://github.com/ilabafrica-IoTlab/SMART-SWITCH

CONCLUSION

The system as the name indicates, ‘SMART SWITCH USING ANDROID APPLICATION AND NODEMCU ESP8266’ makes the system more flexible and provides an attractive user interface compared to other home automation systems. In this system we integrate mobile devices into our smart switch. A novel architecture for a smart switch system is proposed using relatively new communication technologies. The system consists of mainly three components: **a nodemcu esp8266**, **bulbs** and **relay circuits**. WIFI is used as the communication channel between android phone and the Nodemcu esp8266 microcontroller. We hide the complexity of the notions involved in the home automation system by including them into a simple, but comprehensive set of related concepts. This simplification is needed to fit as much of the functionality on the limited space offered by a mobile device’s display. This report proposes a low cost, secure, ubiquitously accessible, auto-configurable, remotely controlled solution. The approach discussed in the report is novel and has achieved the target to control home lights remotely using the WiFi technology to connect system parts, satisfying user needs and requirements. WiFi technology capable solution has proved to be controlled remotely, provide home security and is cost effective as compared to the previously existing systems. Hence we can conclude that the required goals and objectives of the smart switch system have been achieved. The system design and architecture were discussed, and the prototype presents the basic level of smart switch control and remote light control has been implemented. Finally, the proposed system is better from the scalability and flexibility point of view than the commercially available home automation systems.

REFERENCE

1.Wikipedia
2. Wireless Sensor Networks: Concepts, Applications, Experimentation and Analysis. 2016. p. 108. ISBN 9811004129. The use of standardized, open standards over proprietary protocols provides the industry with the freedom to choose between suppliers with guaranteed interoperability. Standardized solutions usually have a much longer lifespan than proprietary solutions.