



COMP39X

2019/20

IoT Based Home Automation System using a Cloud server and Amazon Dot

DEPARTMENT OF
COMPUTER SCIENCE

University of Liverpool
Liverpool L69 3BX

Abstract

Commonly, electrical appliances in a home are controlled by switches that are responsible to allow electricity to flow on these devices. As the world evolves and the use of Internet is getting more and more frequent, we find new technology to match our needs. The Internet is becoming integral part of our lives. As the Internet keeps growing, home automation is getting ground and becoming a common practise for everyone. The process of Home Automation functions by making everything in a house automatically controlled using technology to monitor and change the status of a device rather than controlling it manually.

The project proposes an efficient implementation for IoT (Internet of Things) used for monitoring and controlling home appliances via World Wide Web. Home Automation System is an easy to use and friendly online dashboard, which communicates with the NodeMCU through an Internet gateway using low power communication protocol like WiFi. Through the web-application the user can manage home appliances like lights, speakers, fans etc. The server interacts with these devices which each of them is connected to a relay. This project also proposes and implementation for IoT with voice commands with the help of Sinric repository and user can manage every device by saying “Alexa turn on <device-name>”.

For this project the Waterfall model was followed, the first process model ever introduced. This model it is very simple to understand and use and it was introduced in Herbert D. Benington at the Symposium on Advanced Programming Methods for Digital Computers on 29 June 1956. This model is called waterfall because as the progress of a project continues it flows in largely one direction (going downwards like a waterfall).

Using this model, this project was divided in 6 sequel phases. One can occur after the previous has been done successfully. Each phase contains sub-categories and sub-elements which have to be implement so one phase can be considered as fully completed and then the project can move to the next phase.

Every phase of this model is analyzed in detail on the sections below.

Table of Contents

Abstract.....	5
Table of Figures.....	8
List of Tables	9
Abbreviations and acronyms	10
Chapter 1. Introduction	11
1.1 Aims & Objectives	12
1.2 Background Reading	12
1.2.1 IoT (Internet of Things)	13
1.2.2 IoT - Advantages	14
1.2.3 IoT- Disadvantages.....	15
1.2.4 IoT Software	16
1.2.5 Internet of Things - Technology and Protocols.....	17
1.2.6 Internet of Things - Common Uses	18
1.3 Systems of Home Automation	19
1.4 Worldwide Home Automation.....	19
1.5 Theory	20
1.5.1 What is a Database?	20
1.5.2 What is HTML?	20
1.5.3 What is SQLite?	21
1.5.4 What is JSON?	22
1.5.5 What is Python?	22
1.5.6 What is Django?	23
Chapter 2. Ethical Considerations.....	24
Chapter 3. System Analysis and Design	26
3.1 Development Process and Method	26
3.1.1 Requirements.....	26
3.1.2 Use cases.....	26
3.2 Design and development tools	27
3.2.1 Webserver and flow of Data	27
3.2.2 Web-application analysis	28
3.2.3 Adding relays to web-application	30
3.2.4 Arduino code Analysis.....	30
3.3 Interacting with Amazon Echo Dot	31
3.4 Hardware Specification	32

3.4.1 Hardware Components in Details	32
Chapter 4. Implementation	35
4.1 Connecting hardware components	35
4.2 Matching NodeMCU Pins with GPIO Pins	35
4.3 Connecting device to relay.....	36
4.4 Declaring libraries and constants.....	37
4.5 Compiling the program Uploading to ESP8266.....	37
4.6 Implementing the webserver locally Running python commands	39
4.7 Publishing web-server to cloud platform	40
4.8 Summary of Implementation (Waterfall model)	42
Chapter 5. Testing & Evaluation	43
5.1 Testing phase in Waterfall Model	43
5.2 Testing in run-time.....	43
5.2.1 Powering on a Lamp	45
5.2.2 Testing voice commands.....	45
5.3 Evaluation	46
Chapter 6. Conclusion	47
6.1 Results	47
6.2 Future Scope	47
Chapter 7. BCS Criteria & Self-Reflection	49
7.1 BCS Criteria	49
7.2 Self-Reflection	50
References	52
Appendices.....	53
Arduino IDE code	53
Web-application code.....	63

Table of Figures

Figure 1: Use cases Diagram	27
Figure 2: Flow of Data on Webserver	28
Figure 3: Flow of Data on Webserver and Logic	29
Figure 4: Django administration dashboard	29
Figure 5: Adding Relays through Admin Dashboard	30
Figure 6: Relay information in JSON	30
Figure 7: Function to determine relay status	31
Figure 8: Function to check if voice command is given	31
Figure 9: NodeMCU	33
Figure 10: 8-channel Relay Module	33
Figure 11: Breadboard	34
Figure 12: Jumper wires	34
Figure 13: Connecting NodeMCU with Relays	35
Figure 14: Connecting a device to a relay	36
Figure 15: Including libraries in Arduino IDE	37
Figure 16: Declaring constants	37
Figure 17: Declaring web-server address	37
Figure 18: Uploading the program to ESP8266	38
Figure 19: Writing program to ESP memory	38
Figure 20: Python commands	39
Figure 21: Command to run server locally	39
Figure 22: Server running locally	40
Figure 23: Git command_1	40
Figure 24: Git command_2	40
Figure 25: Python application on Heroku Server	41
Figure 26: Git command_3	41
Figure 27: Git command_4	41
Figure 28: Serial Monitor showing Relay Status	43
Figure 29: Turning Relays on through Webserver	44
Figure 30: Serial Monitor showing new Relay Status	44
Figure 31: Hardware Components in run-time	45
Figure 32: Serial Monitor showing Relay activated with Alexa	45

List of Tables

Table 1: Hardware Components	32
Table 2: NodeMCU \leftrightarrow GPIO mapping.....	36

Abbreviations and acronyms

IOT – Internet of Things

IC - Integrated Circuit

NO - Normally Open

NC - Normally Closed

COM – Common

LCD – Liquid Crystal Display

LED - Light Emitting Diode

USB – Universal serial bus

GPIO – General Purpose Input Output

API – Application Program Interface

VCC – Voltage Common Collector

GND - Ground

Chapter 1. Introduction

An IoT - based Home Automation System focuses on controlling home appliances either locally while the user is home, or remotely through a cloud-server. Home Automation enables the ability for each person to remotely or automatically control home devices around the house. A home appliance is a device designed to perform specific tasks, and by doing these tasks remotely the user's life is easier and less-worrying. The devices that can be controlled locally or remotely can be lights, speakers, TVs or even garage doors. The words home appliance and device can be both used since they have the same meaning.

Home Automation is becoming quite common in our days since it makes our lives easier and more effective. Automation in general already occurs in many fields such as, Computers, Industries and companies. The idea of remote management of home appliances over the Internet from anywhere, anytime, and under any conditions can be a reality in today's world. In extension, assume the user is at work and it's a really cold day outside, the user can simply turn the Heat on at home anytime he wants, and by the time he is back home the temperature is at the desired level. This ability, to be able to control the state of specific devices through the Internet at any time only provides the user a comfortable and pleasant home.

The invention of computers and computer networks has revolutionized communication in this contemporary world. The Internet of Things has a close relationship with computers and computer networks.

Looking at these aspects, we have developed an IoT based Home Automation System using a Cloud server where our web-application is stored and Amazon Dot is used for voice commands.

IOT: The Internet of Things (IoT) is a network of physical devices embedded with electronics, software, actuators, sensors and connectivity which gives the ability on these devices to be able to connect and exchange data. (Rouse, 2020)

The principle of IoT permits physical objects to be controlled and sensed remotely via network infrastructures by creating links for the direct integration of physical devices into computerized systems.

When the IoT is merged with actuators and sensors this technology is categorized into cyber-physical systems that include advanced technologies such as Smart Homes, Smart





Cities, Smart grids and Smart World. The results of using IoT is improved efficiency, accuracy and economic benefit because of reduced human intercession.

As the number of controllable home appliances increases rapidly, the need to control and interact with them increases too. This problem can be solved with the use of IoT, by giving the ability to most important home appliances to connect to the Internet. This includes both smart and non-smart devices, which generalises our appliances to one category since all appliances required a power source to work. This Home Automation System gives the ability for non-smart devices to be as effective as the smart devices are, by controlling them through a webserver.

1.1 Aims & Objectives

The main aim of this project was to create an Arduino-based circuit that will control home appliances and also check their status. To create that circuit and achieve IoT by connecting devices to it, a microcontroller called NodeMCU was used. On this microcontroller, an 8-module relay is connected directly which is responsible for turning on/off devices attached to it. A power source of 5V is also needed to power on the NodeMCU and relays, as well as a power source for every device connected to a relay.

To classify this project as successfully made, the aims stated below had to be fulfilled:

-  Connecting Arduino-based circuit with Relays
-  Connecting Relays with home appliances
-  Data sending from NodeMCU to web-server
-  Web-server controlling and storing data for these devices

Each of these aims was approached and implemented step-by-step, starting from the first up to the last one. The difficulty to fulfil these aims was that they are interconnected and every step has to be done before the next one can start.

1.2 Background Reading

Home Automation concept started back in the 1980s. Home automation refers to the idea of controlling home appliances while at work or at any other place apart from home. This technology during these years was used to provide help to disabled people and to make their lives easier. From 1980 until nowadays, technology has made huge improvements and steps.

Many devices we have now are called “smart”, a term used to describe a device that can connect to other devices or networks via different wireless protocols such as Bluetooth, Wi-Fi, NFC, LTE, and 5G. The devices that are called smart, interact with each other making things easier and more-effective.

The idea of Home Automation systems has advantages to many aspects of our lives such as safety and children management (while children are staying home) (Fritz, 2020).

A well-organised home automation system not-only helps us to interact with devices and manage their state, but also, ensures safety with sensors and cameras that are the “eyes” monitoring the house at any time.

Home Automation and Energy Saving: These terms often go hand by hand since with home automation the power consumption on a house can be managed and tracked. This means that the cost of electricity can significantly be reduced by setting limits or timers for specific appliances. Devices that often run all the time during the day, can set to sleep state or to only drain power at specific times of the day. This leads to an eco-friendly home, not only power saving but also friendly to the environment.

1.2.1 IoT (Internet of Things)

IoT (**Internet of Things**) is a progressive automation and analytics system which accomplishes to interact with networking, big data, sensing and Artificial Intelligence in order to deliver complete systems for a product or a service. These systems are surprisingly more effective when applied to an industry or a company. IoT Systems have many applications in industries because of their ability to suit to any environment. These systems send and receive data, they are automated and they are usefully to many operations of an industry.

IoT Key Features:

The key features of IoT include Artificial Intelligence, Connectivity between devices, Sensors, Active engagement and Small Devices.

Artificial Intelligence: IoT is capable of making anything “smart” meaning it enlarges every aspect of our life with the power of data collection, artificial intelligence algorithms and networks. This can simply mean, that IoT can be responsible for controlling the heating and the lights home.

Connectivity: New technologies for networking and specifically for IoT networking, mean that networks are not exclusively tied to major providers. Networks can exist on a smaller and cheaper scale, yet practical and useful. These smaller-networks come along with small devices.

Sensors: IoT loses its power without the use of sensors. They act as determining instruments which transforms the IoT from a standard passive network of devices into an active system, necessary for real-world integration.

Active Engagement: Today's interaction with connected technology mostly happens with passive engagement. IoT introduces an activate engagement directly using a product, a service or content in general.

Small Devices: Over the years, the devices have become smaller, cheaper and more compact. IoT handles small-built devices to deliver its precision, scalability and versatility.

1.2.2 IoT - Advantages

IoT – Advantages

The IoT advantages range across every area of lifestyle and business. Below there are stated some of the advantages that IoT has.

Strengthen Data Collection – The Modern Data collection in our days is unfortunately limited while its design is for passive use. IoT breaks these limitations and enables humans to analyse the world as they see it. It can be called as an actual image of everything of our world.

Improved Customer Engagement – Data analytics suffer from unseen-errors and significant flaws in efficiency of a system; and as a result, engagement remains passive. IoT completely transforms this to achieve wealthy and more effective engagement with audiences.

Technology Optimization – The same technologies and data which improve the customer experience also improve device use, and aim for more potent improvements to technology. IoT unlocks a world of critical functional and field data.

Reduced Waste – IoT makes areas of improvement clear. It provides real-world information with least waste leading to more effective management of resources.

1.2.3 IoT- Disadvantages

IoT – Disadvantages

Although IoT delivers an extremely well to mention set of benefits, it also presents a significant set of disadvantages. Below there are stated those disadvantages.

Security – IoT creates an ecosystem of interconnected devices communicating over networks. The system offers little control despite any security measures. This leaves users exposed to various kinds of attackers.

Privacy – The sophistication of IoT provides substantial personal data in extreme detail without the user's active participation.

Complexity – Some people find IoT systems complicated in terms of design, deployment, and maintenance given their use of multiple technologies and a large set of new enabling technologies.

Flexibility – Many are concerned about the flexibility of an IoT system to integrate easily with another. They worry about finding themselves with several conflicting or difficult to manage systems.

Compliance – IoT, like any other technology in the realm of business, must comply with regulations. Its complexity makes the issue of compliance seem incredibly challenging when many consider standard software compliance a battle.

1.2.4 IoT Software

IoT software addresses its key areas of networking and action through platforms, embedded systems, partner systems, and middleware. These individual and master applications are responsible for data collection, device integration, real-time analytics, and application and process extension within the IoT network. They exploit integration with critical business systems (e.g., ordering systems, robotics, scheduling, and more) in the execution of related tasks.

Data Collection

This software manages sensing, measurements, light data filtering, light data security, and aggregation of data. It uses certain protocols to aid sensors in connecting with real-time, machine-to-machine networks. Then it collects data from multiple devices and distributes it in accordance with settings. It also works in reverse by distributing data over devices. The system eventually transmits all collected data to a central cloud server.

Device Integration

Software supporting integration binds (dependent relationships) all system devices to create the body of the IoT system. It ensures the necessary cooperation and stable networking between devices. These applications are the defining software technology of the IoT network because without them, it is not an IoT system. They manage the various applications, protocols, and limitations of each device to allow communication.

Real-Time Analytics

These applications take data or input from various devices and convert it into viable actions or clear patterns for human analysis. They analyse information based on various settings and designs in order to perform automation-related tasks or provide the data required by industry.

Application and Process Extension

These applications extend the reach of existing systems and software to allow a wider, more effective system. They integrate predefined devices for specific purposes such as

allowing certain mobile devices or engineering instruments access. It supports improved productivity and more accurate data collection.

1.2.5 Internet of Things - Technology and Protocols

IoT mainly exploits standard protocols and networking technologies. However, the major technologies and protocols that IoT is interacting with *are RFID, NFC, low-energy Bluetooth, low-energy wireless, low-energy radio protocols, LTE-A, and WiFi-Direct.*

These technologies support the specific networking functionality required in an IoT system in contrast to a standard uniform network of common systems.

NFC and RFID

NFC (near-field communication) and RFID (radio-frequency identification) provide simple, low-energy, and adaptable options for identity and access tokens, connection bootstrapping, and payments.

- 🌐 **NFC** consists of communication protocols for electronic devices, typically a mobile device and a standard device.

- 🌐 **RFID technology** enables 2-way radio transmitter-receivers to identify and track tags associated with specific objects.

Low-Energy Bluetooth This technology supports the low-power, long-use need of IoT function while exploiting a standard technology with native support across systems. **Low-Energy Wireless** This technology replaces the most power-hungry aspect of an IoT system. Though sensors and other elements can power down over long periods, communication links (i.e., wireless) must remain in listening mode.

Low-energy wireless not only reduces power consumption, but also extends the lifetime of the device because of its less use. Radio Protocols ZigBee, Z-Wave, and Thread are radio protocols for creating low-rate private area networks. These technologies are low-power, but offer high throughput unlike many similar options. This increases the power of small local device networks without the typical costs.

LTE-A

LTE-A, or LTE Advanced, delivers an important upgrade to previous LTE technology by increasing not only its coverage, but also reducing its latency and raising its coverage. It gives IoT a tremendous power through expanding its range, with its most significant applications being vehicle, UAV, and similar communication.

WiFi-Direct

WiFi-Direct significantly decreases the need for an access point. It allows P2P (peer-to-peer) connections with the speed of WiFi, but with lower latency. WiFi-Direct eliminates an element of a network that often slows it down, and it does not compromise on speed or throughput.

1.2.6 Internet of Things - Common Uses

IoT has applications across all industries and markets. It spans user groups from those who want to reduce energy use in their home to large organizations who want to streamline their operations. It proves not just useful, but nearly critical in many industries as technology advances and we move towards the advanced automation imagined in the distant future.

Engineering, Industry, and Infrastructure

Applications of IoT in these areas include improving production, marketing, service delivery, and safety. IoT provides a strong means of monitoring various processes; and real transparency creates greater visibility for improvement opportunities. The deep level of control afforded by IoT allows rapid and more action on those opportunities, which include events like obvious customer needs, nonconforming product, malfunctions in equipment, problems in the distribution network, and more.

Government and Safety

IoT applied to government and safety allows improved law enforcement, defines, city planning, and economic management. The technology fills in the current gaps, corrects many current flaws, and expands the reach of these efforts. For example, IoT can help city planners have a clearer view of the impact of their design, and governments have a better idea of the local economy.

Home and Office

In our daily lives, IoT provides a personalized experience from the home to the office to the organizations we frequently do business with. This improves our overall satisfaction, enhances productivity, and improves our health and safety. For example, IoT can help us customize our office space to optimize our work.

Health and Medicine

IoT pushes us towards our imagined future of medicine which exploits a highly integrated network of sophisticated medical devices. Today, IoT can dramatically enhance medical research, devices, care, and emergency care. The integration of all elements provides more accuracy, more attention to detail, faster reactions to events, and constant improvement while reducing the typical overhead of medical research and organizations.

1.3 Systems of Home Automation (Gibson, 2018)

There are 3 main network types of home automation systems that can be installed. These systems are categorized as power line systems, wired systems and wireless systems. Power Line Systems: This is called the most affordable system. Power line systems rely on the existing power lines to transfer things such as security camera feeds and lighting information to a common control interface. Usually, these are X10 technology-based systems.

Wired Systems: Wired systems use cables to communicate information. It is easier to install. Hardwiring all of the systems together through cables to a common junction point, the system is developed.

Wireless Systems: For this system no need to use traditional cables. Wireless systems integrate with Wi-Fi networks, meaning that they are easily compatible with any existing home networks, such as the ones formed by computers. However, some wireless systems use a different radio frequency for making them incompatible with open networks.

1.4 Worldwide Home Automation

Home automation technology is becoming more popular and globally it is highly used. For this, a large home automation market already is set up. According to the analysts, the market is expanding because of its extreme demand. Revenues earned from shipments of

home automation systems in Europe and North America will enhance at a compound annual growth rate of 43 percent from US\$ 2.2 billion in 2012 to closely US\$ 12.8 billion in 2017 (Elfriede Dustin, Thom Garrett, Bernie Gauf, 2009). The amount realizes us that automation provides ample amount of efficiency in different sector worldwide. American Smart Home market also is going to generate huge amount of revenues in coming 2020. From the statistical information, we are able to know that in 2013 the smart home market revenue was about 7.19 billion U.S.\$ but blessing of automation it will reach about 22.4 billion U.S.\$ in 2020 (Gibson, 2018).

Home Automation applications are mainly in lighting, safety and security, HVAC (Heating, Ventilation, and Air-Conditioning), entertainment (home, audio) and others (robotics, health care). In current years, automation systems were nourished in residences, shopping malls, sky scrapers, hotels in colourful ways. Many organizations such as 2GIG Technologies, Siemens AG, Johnson Controls, Honeywell International Inc., iControl Networks Inc., Vantage Controls are leading the market of home automation (Albany, Jan. 16, 2015)

1.5 Theory

1.5.1 What is a Database?

A [database](#) is an organized collection of data. The data is typically organized to model aspects of reality in a way that support process requiring information. Basically, database management systems are computer software applications that interact with user, other applications, and the database itself to capture and analyse data. A general aspect database system structured is to permit the explanation, creation, querying, update and administration of database. A well-organized database management system is including like, MySQL, PostgreSQL, Oracle, IBM DB2. Databases are used to support internal operations of organization and hold administrative information and more specialized data like any sort of data. (Wikipedia, 2020)

1.5.2 What is HTML?

[Hypertext Mark-up Language \(HTML\)](#) is the standard mark-up language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as `` and `<input />` directly introduce content into the page. Other tags such as `<p>` surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behaviour and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997. (Wikipedia, 2020)

1.5.3 What is SQLite?

[SQLite](#) is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.



SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures. These features make SQLite a popular choice as an Application File Format. SQLite database files are a recommended storage format by the US Library of Congress. Think of SQLite not as a replacement for Oracle but as a replacement for fopen()

SQLite is a compact library. With all features enabled, the library size can be less than 600KiB, depending on the target platform and compiler optimization settings. (64-bit code is larger. And some compiler optimizations such as aggressive function inlining and loop unrolling can cause the object code to be much larger.) There is a tradeoff between memory usage and speed. SQLite generally runs faster the more memory you give it. Nevertheless, performance is usually quite good even in low-memory environments. Depending on how it is used, SQLite can be faster than direct filesystem I/O (Hipp, D. Richard, SQLite Development Team, 2020).

1.5.4 What is JSON?

[JSON \(JavaScript Object Notation\)](#) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON is built on two structures:

-  A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
-  An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures. (json.org, 2020)

1.5.5 What is Python?

[Python](#) is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program

maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. (Python.org, 2020)

1.5.6 What is Django?

Django is a free and open source web application framework written in Python. A framework is nothing more than a collection of modules that make development easier. They are grouped together, and allow you to create applications or websites from an existing source, instead of from scratch.

This is how websites - even simple ones designed by a single person - can still include advanced functionality like authentication support, management and admin panels, contact forms, comment boxes, file upload support, and more. In other words, if you were creating a website from scratch you would need to develop these components yourself. By using a framework instead, these components are already built, you just need to configure them properly to match your site.

The official project site describes Django as "a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source."

Django offers a big collection of modules which you can use in your own projects. Primarily, frameworks exist to save developers a lot of wasted time and headaches and Django is no different.

You might also be interested in learning that Django was created with front-end developers in mind. "Django's template language is designed to feel comfortable and easy-to-learn to those used to working with HTML, like designers and front-end developers. But it is also flexible and highly extensible, allowing developers to augment the template language as needed."

Chapter 2. Ethical Considerations

For this project the use of IoT (Internet of Things) is required. This however, leads to some ethical issues that are being considered. These ethical issues, appear since the Internet of Things has become more and more popular. Some examples of technologies that cooperate with the Internet of Things are, RFID (Radio frequency Identification), NFC (Near Field Communication), LTE and 5G communications. Each one of the above technologies uses people's data in order to make their life easier. However, most of the data is sensitive and in case of a remote attack on the web-server all of the user's data is exposed. The use of IoT might cause ethical problems such as:

- ✚ *Ubiquity, pervasiveness* – the user is attracted to IoT, absorbed by it and there is no clear way to give up the IoT (since all the producers equip devices with Internet connectivity) (Daniela Popescul, Mircea Georgescu, 2013)
- ✚ *Miniaturization, invisibility* – computers, the way they are in our days, will disappear – the devices will become smaller and smaller, transparent, as a result there will be no inspections or quality control on them (Daniela Popescul, Mircea Georgescu, 2013)
- ✚ *Ambiguity, vagueness* – the distinction between natural objects and artificial created objects will be more and more difficult to be made as everything is absorbed in a network of artifacts. This will lead to serious problems of identity and system boundaries (Daniela Popescul, Mircea Georgescu, 2013)
- ✚ *Difficult identification* – for all the objects that are connected to IoT, an identity is required. The access to these objects or the management of their identity might cause serious problems on security and control in a globalized world (Daniela Popescul, Mircea Georgescu, 2013)
- ✚ *Ultra-connectivity* - The connections between objects and people will increase in number and reach unpredicted scales. In result, the amount of transferred data and products will increase in a large scale (Big Data) and this data could be maliciously used (Daniela Popescul, Mircea Georgescu, 2013)
- ✚ *Autonomous and unpredictable behaviour* – The interconnected devices and objects might interfere in peoples' lives in an unpredicted way for the users or designers. The people will be part of the IoT ecosystem together with artificial created objects and devices, creating hybrid systems with unpredicted behaviour. The progressive

development of IoT will lead to emerging behaviours without they users being aware the environment they are exposed to (Daniela Popescul, Mircea Georgescu, 2013)

🌈 *Incorporated intelligence*, as the IoT progresses, objects and devices are being seen as a substitute for the social life – the objects are getting intelligent and dynamic with an emerging behaviour. Being devoid of these devices will cause significant problems – see the teenagers who consider their self socially handicapped without a Smartphone or social media (Daniela Popescul, Mircea Georgescu, 2013)

🌈 *Difficult control* - As the number of hubs, switches and data increases, it is really difficult to control the IoT and accumulate it. Although the information flow will be easier in the future and the transfers will be quicker and cheaper, the management of them will be tougher. There will appear emerging behaviours and phenomena which will require monitoring and management (Daniela Popescul, Mircea Georgescu, 2013)

Chapter 3. System Analysis and Design







3.1 Development Process and Method

The development process and method followed to implement this project was the Waterfall model. This model has 6 sequential phases starting from the gathering of requirements up to the maintenance of the system. On the design phase, a mock-up (sketch) of the system was done. This mock-up includes the interaction of the relays and the NodeMCU as well as their connectivity. To design the connectivity of the relays and a simple device as a lamp, a 12V power source was used for safety.

3.1.1 Requirements

Starting from the requirements, the hardware had to be implemented first. To implement the hardware a final hardware specification design is required. The research in the beginning of this project included Arduino Mega controller, but in the final hardware specification it was not used since NodeMCU can control the devices state and help us to achieve IoT.

After the hardware collection and assembling is finished, the project can proceed to implementation phase. The requirements for this phase of the project were:

-  To assemble a Home Automation Arduino-based board
-  To make the microcontrollers communicate together
-  To be able to power a device
-  Jumper wires to connect relays with the NodeMCU
-  Power source for NodeMCU and relays
-  Power source for a device

3.1.2 Use cases

The use case diagram below represents the main logic of the system, and the ability for a user to be able to power on/off a device connected to a relay. However, this is a mock-up and not final design. The final design includes more complicated functions and more automated processes.

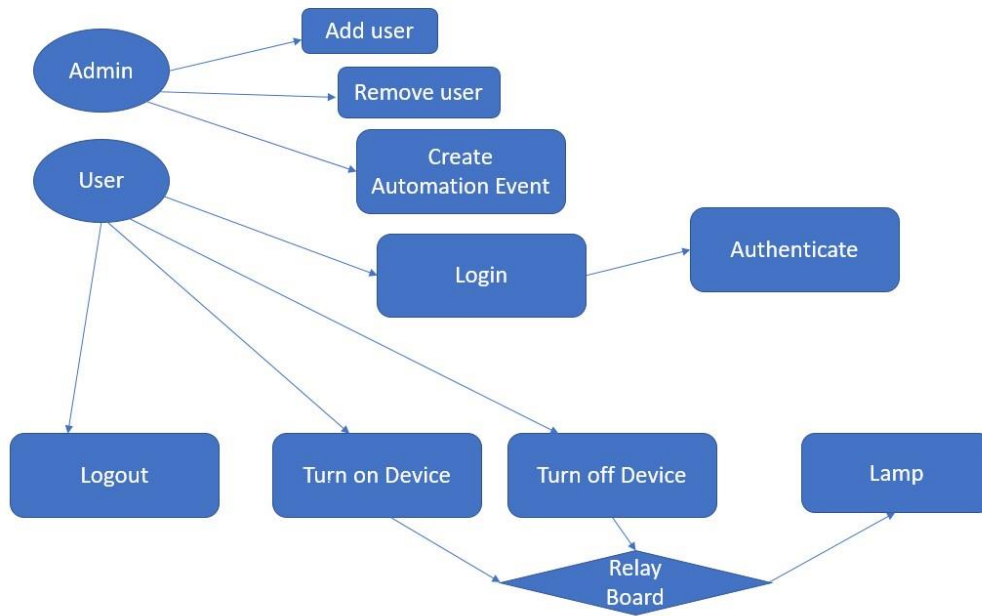


Figure 1: Use cases Diagram

3.2 Design and development tools

To design this project a good research was prior-made. This research included software and hardware components as well as better and more effective approaches for IoT. The software used for this IoT project was Arduino IDE. This tool is written in C++ language and it includes libraries and examples related to Arduino. On this tool, the code for the ESP8266 was written.

Other tools used on this project include: Python, Django framework, HTML, CSS, and JSON.

Libraries used for this project:

- 🚦 ArduinoJson is a C++ JSON library for Arduino and IoT
- 🚦 ESP8266 library to connect the ESP with the WiFi network
- 🚦 ESP8266HTTPClient library to send or receive requests to web-server

3.2.1 Webserver and flow of Data

The flow of data on the web-server is as shown in the Figure 2 below. The WiFi module is connecting to the Internet through our Modem and to our web-application. This application is on a cloud server called *Heroku*. The clouding platform is responsible to keep data for our relays in its' database. The cloud-server can be accessed from any device or platform that it has a web browser.

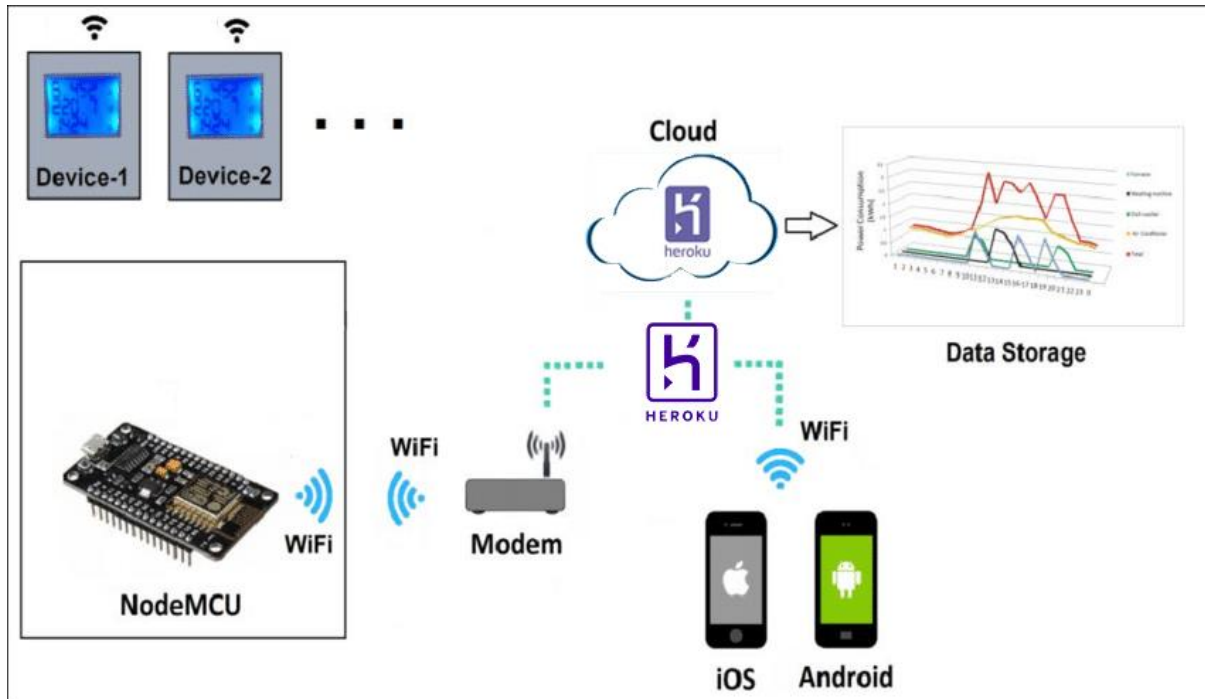


Figure 2: Flow of Data on Webserver

3.2.2 Web-application analysis

The web-application created for this project is based in *Python* and *Django* framework. Key feature of this web-application is that its **responsive** on every device, platform or operating system. This exploits a huge advantage for Home automation since the user is able to control every device without having to install anything in order to access this web-application. The logic (function) between this interaction of user – device through the web-application is clearly stated on the Figure 3 below.

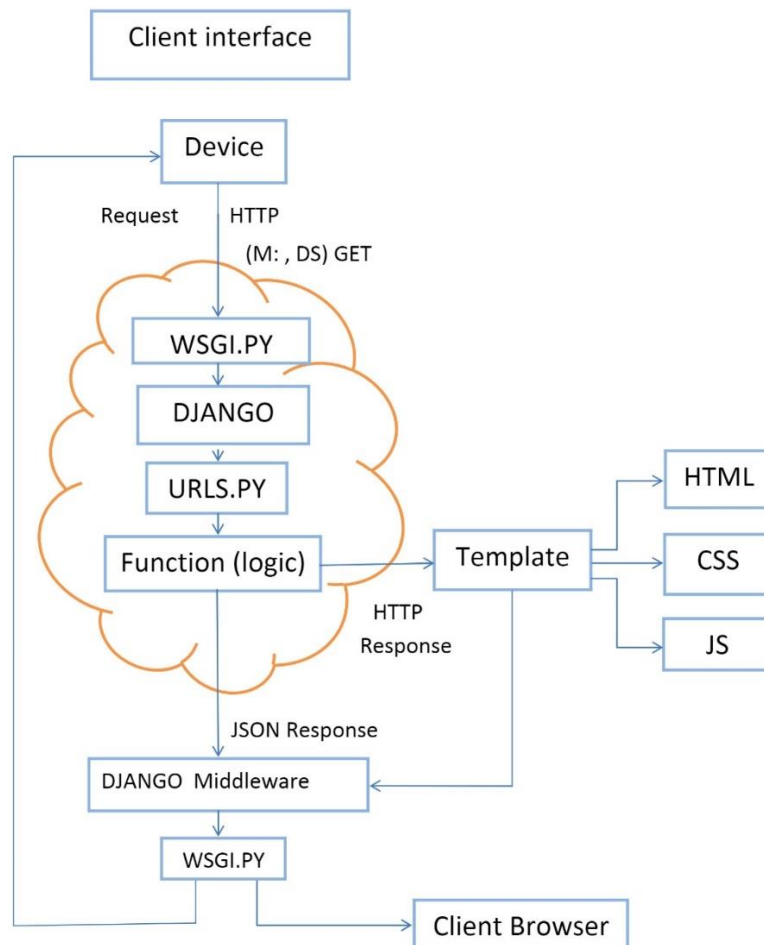


Figure 3: Flow of Data on Webserver and Logic

Other features of this web-application, is the Admin panel page which can be accessed only by the developer. This Admin dashboard offers management to three different instances.

- + Add or Remove Users
- + Add or Remove Groups
- + Add or Remove Relays



Figure 4: Django administration dashboard

3.2.3 Adding relays to web-application

Visiting the admin dashboard, we add our 8 relays on the system. We can name the relays however we want; For demonstrating purposes we named the relays ("Relay 1", "Relay 2" etc.)

In the future this could change on final connection of devices, so "Relay 1" could be "Garage Door", "Relay 2" could be "Main Door" etc.

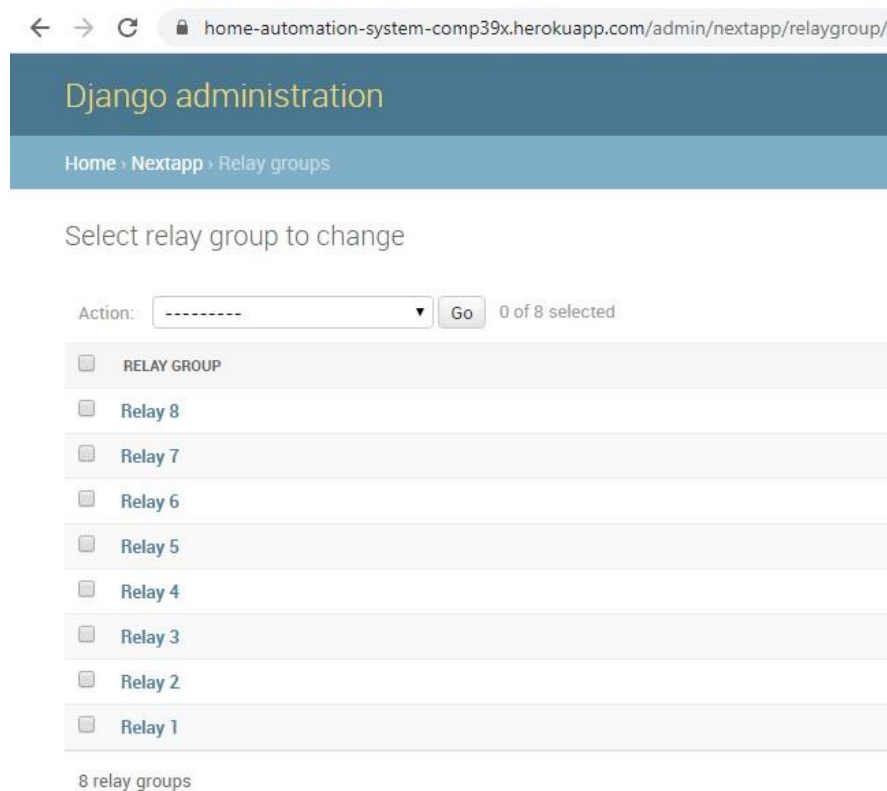


Figure 5: Adding Relays through Admin Dashboard

3.2.4 Arduino code Analysis

The logic behind the interaction of NodeMCU and the web-server is an http request that is sent from the ESP8266 to the web-server on the specific link that it stores the relay information. This information contains the id of the relay, the name of the relay, the status at the moment and the last update date and time.

```
[{"id": 1, "r_name": "Relay 1", "r_status": false, "r_lastupdateinfo": "2020-04-17T23:28:48.540Z"},
```

Figure 6: Relay information in JSON

The ESP8266 while it runs, it loops every 2 seconds and checks the status of the relays. If a relay is turned on through the webserver it forces that relay to turn on by sending `signal(relay_number, LOW)`. To achieve the management of the relays a JSON array was required. This array contains a Boolean for every relay.

The logic to turn On/Off a specific relay is as shown:

```
if (root_0_r_status == true ) {  
    Serial.println("Relay 1 is on ");  
    digitalWrite(relay1, LOW);  
} else if (root_0_r_status == false ) {  
    Serial.println("Relay 1 is off ");  
    digitalWrite(relay1, HIGH);  
}
```

Figure 7: Function to determine relay status

3.3 Interacting with Amazon Echo Dot

To interact with Amazon Dot or commonly known as “Alexa” the Sinric Repository was used. The Sinric is a project free to use, published on Github including python scripts and Arduino IDE functions. The logic behind the interaction with Amazon Dot and the web-server created is simple, yet effective. The NodeMCU connects to the Sinric server with a unique API and a device id. When the connection is successful, it can manage the state of the device. On the same time, the NodeMCU connects to our web-server using the same API and device id and checks the status of a relay. The python script by Sinric updates the status on our web-server too. On each loop, our WiFi module, keeps on checking the status of the devices and relays. A device can either be turned on using voice commands or through the web-server. They interact and if a device is turned on (either way) the other server refreshes accordingly.

```
if (root_0_r_status == true || sinric1 == true) {  
    Serial.println("Relay 1 is on ");  
    digitalWrite(relay1, LOW);  
    setPowerStateOnServer(relay1_id, "ON");  
    EEPROM.write(100, 1);  
    EEPROM.commit();  
} else {  
    Serial.println("Relay 1 is off ");  
    digitalWrite(relay1, HIGH);  
    setPowerStateOnServer(relay1_id, "OFF");  
    EEPROM.write(100, 0);  
    EEPROM.commit();  
}
```

Figure 8: Function to check if voice command is given

The logic behind this function is that the NodeMCU is checking the web-server (if a device is turned on via the online dashboard) and the Sinric server (if a device is turned on using voice commands) and it decides whether the relay will turn On or Off. This function enables us to use a device through the web-application or with voice commands at the same time.

3.4 Hardware Specification

The table below shows the hardware components used to implement this project. This table was finalised after reviewing similar projects, research papers and data from web pages referring to IoT Home Automation projects.

No.	Component	Part number	Quantity
1	WiFi Module	CH340 NodeMcu V3 - ESP8266	1
2	Relay Module	5V 8-Channel (250V)	1
3	Bread Board	400 Tie-points & 830 Tie Points	1
4	Jumper Wires	1. 40Pcs M/M Dupont 2. 40Pcs M/F Dupont 3. 40Pcs F/F Dupont	1

Table 1: Hardware Components

3.4.1 Hardware Components in Details

1. *What is NodeMCU?*

The NodeMCU (Node Microcontroller Unit) is an open source software and hardware development environment that is built around a very inexpensive System-on-a-Chip (SoC) called also, ESP8266.

This module is an Arduino-like device with programmable pins and built-in WiFi.

The advantages of this module are its' low-cost price, the low power consumption that it uses and also the simplicity of its design.

NodeMCU Pins

This Microcontroller has many pins which are used either to connect other sensors or to send/receive signals. General-Purpose Input/Output (GPIO) is a pin on an IC (Integrated Circuit) which can be used as input or output pin and its' behaviour can be controlled at run time. Each GPIO pin is assigned to a different number on the board. Example: D0 pin is assigned with GPIO16.

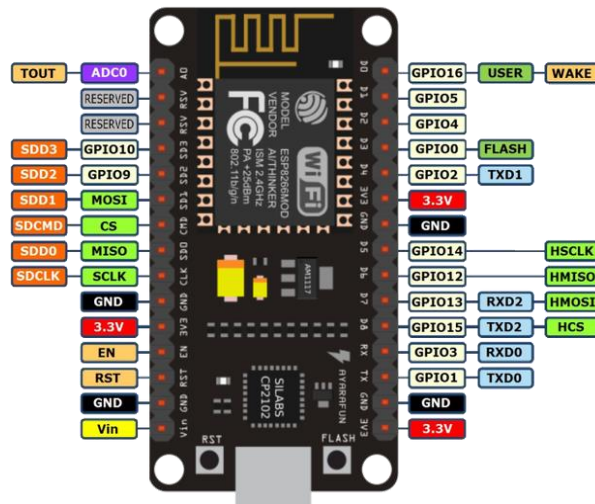


Figure 9: NodeMCU

2. 8-channel Relay Module



Figure 10: 8-channel Relay Module

Features:

- ✚ Support control of 10A 30V DC and 10A 250V AC signals
- ✚ 5V 8-Channel Relay interface board
- ✚ LOW level trigger, equipped with indicator easy to recognize the working status
- ✚ Each relay has NO and NC ports, easier to connect and control the connected devices
- ✚ Selection of plastic material for high temperature and better chemical solution performance.
- ✚ PCB size: 5.7 x 13.8 cm

Ports of 8-channel relay module

Input:

- ✚ VCC: Connected to positive supply voltage (supply power according to relay voltage)
- ✚ GND: Connected to negative supply voltage
- ✚ IN1: Signal (LOW Level) triggering terminal 1 of relay module
- ✚ IN2: Signal(LOW Level) triggering terminal 2 of relay module
- ✚ IN3: Signal(LOW Level) triggering terminal 3 of relay module
- ✚ IN4: Signal(LOW Level) triggering terminal 4 of relay module
- ✚ IN5: Signal(LOW Level) triggering terminal 5 of relay module
- ✚ IN6: Signal(LOW Level) triggering terminal 6 of relay module
- ✚ IN7: Signal(LOW Level) triggering terminal 7 of relay module
- ✚ IN8: Signal(LOW Level) triggering terminal 8 of relay module

3. Breadboard

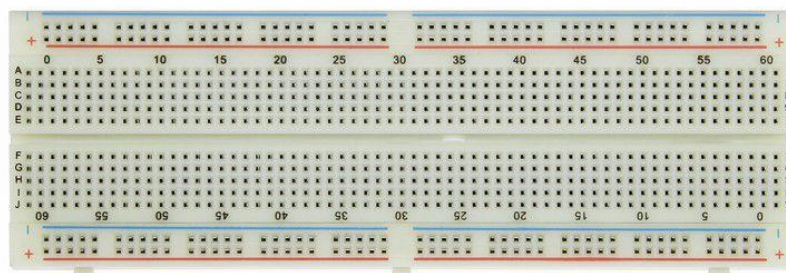


Figure 11: Breadboard

Used to build and test the circuit before finalising the connectivity between the ESP and other sensors.

4. Jumper wires



Figure 12: Jumper wires

They are responsible for transmitting signals to pins or powering on/off sensors

Chapter 4. Implementation

4.1 Connecting hardware components

NodeMCU with 8 Relay Module

To connect the NodeMCU microcontroller with the 8-channel relay module jumper wires are required. The jumper wires needed for this connection are female to female end since both hardware components have pins. The pins on the NodeMCU ranging from D1 to D8 have to be connected to the appropriate number on the 8-channel relay module. Each of these pins on the NodeMCU is responsible to turn on/off the relay matched to it.

Along with the input/output pins another connection should be made, which is VCC pin from the ESP8266 to VCC pin on the 8-channel relay and GND pin to GND.

The VCC pin is responsible for powering on the relays but not for powering on the device connected to the relay. The GND pin is also required to be connected in order to make a complete electronic circuit between ESP and 8-channel relay module.

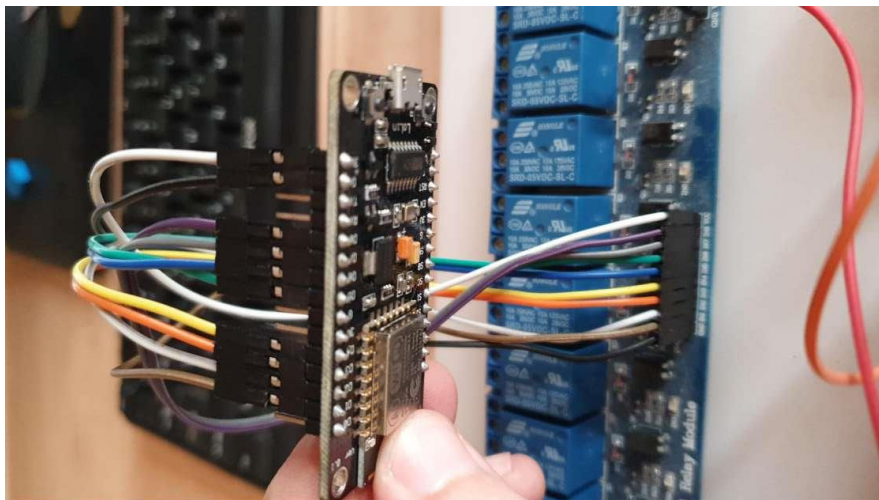


Figure 13: Connecting NodeMCU with Relays

4.2 Matching NodeMCU Pins with GPIO Pins

Each of the pins on the NodeMCU is assigned to an internal GPIO pin number. This match should be made in order to program our ESP.

NodeMCU Pins	GPIO Pins
D1	GPIO05
D2	GPIO04
D3	GPIO00
D4	GPIO02
D5	GPIO14
D6	GPIO12
D7	GPIO13
D8	GPIO15

Table 2: NodeMCU ↔ GPIO mapping

4.3 Connecting device to relay

To connect a device to a relay a power source is needed. This can be a battery or even a 220v cable connected to a lamp. For the relay to work, there should be 2 wires connecting to the lamp. One is connecting from the lamp to the positive pole of the source, and the other cable is split into 2 pieces connecting to a relay. On the relay, we connect one wire to COM (power from source) and the other to NO.

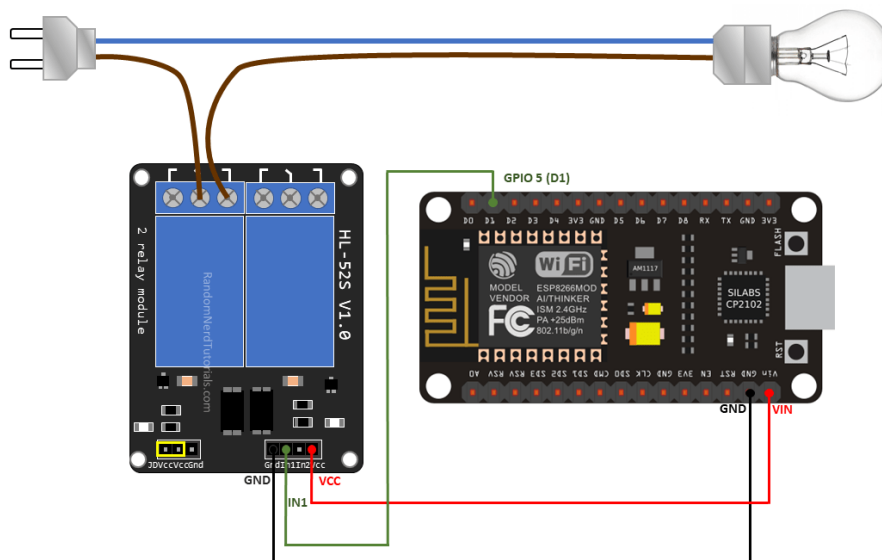


Figure 14: Connecting a device to a relay

4.4 Declaring libraries and constants

The libraries that are required to be declared at the top of our Arduino IDE code are:

- ✚ ArduinoJSON
- ✚ ESP8266
- ✚ ESP8266HTTPCLIENT

```
#include <ArduinoJson.h>
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
```

Figure 15: Including libraries in Arduino IDE

The constants that are required to be declared are:

- ✚ WiFi SSID
- ✚ WiFi Password
- ✚ Relay 1-8 each one matching to an internal GPIO pin number (see table on previous page)
- ✚ The webserver URL page where the JSON data is stored

```
const char* ssid = "Wifi_name";
const char* password = "Wifi_password";

const int relay1 = 05; // GPIO05
const int relay2 = 04; // GPIO04
const int relay3 = 00; // GPIO00
const int relay4 = 02; // GPIO02
const int relay5 = 14; // GPIO14
const int relay6 = 12; // GPIO12
const int relay7 = 13; // GPIO13
const int relay8 = 15; // GPIO15
```

Figure 16: Declaring constants

```
http.begin("http://home-automation-system-comp39x.herokuapp.com/home/get_relay_group_status/");
int httpCode = http.GET(); //Make the request
```

Figure 17: Declaring web-server address

4.5 Compiling the program | Uploading to ESP8266

We click verify to ensure that there are no syntax errors then we upload the program to NodeMCU by choosing “Generic ESP8266 Module” under Tools -> Board.

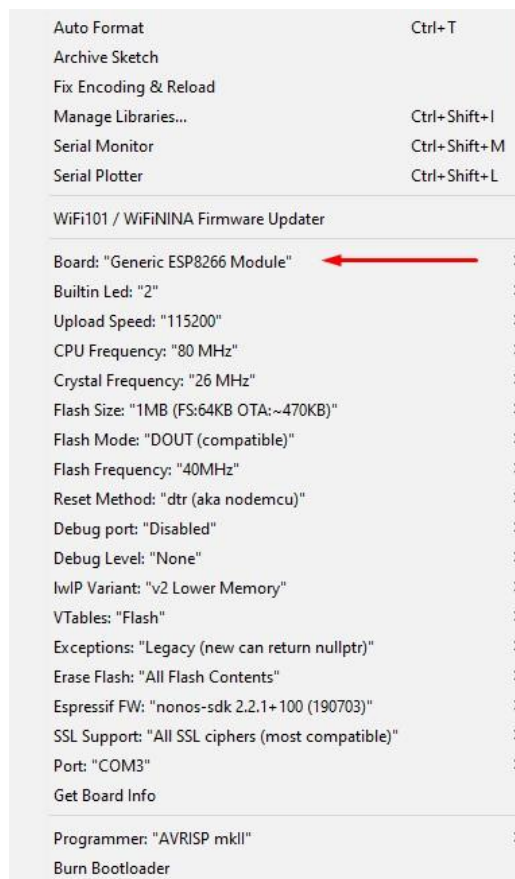


Figure 18: Uploading the program to ESP8266

We also have to verify and make sure the Port we are uploading the program to. Clicking Get Board Info can help us to make sure we are uploading to ESP8266.

While the uploading process is finished, we receive this message.

```

Done uploading.
Writing at 0x00014000... (31 %)
Writing at 0x00018000... (36 %)
Writing at 0x0001c000... (42 %)
Writing at 0x00020000... (47 %)
Writing at 0x00024000... (52 %)
Writing at 0x00028000... (57 %)
Writing at 0x0002c000... (63 %)
Writing at 0x00030000... (68 %)
Writing at 0x00034000... (73 %)
Writing at 0x00038000... (78 %)
Writing at 0x0003c000... (84 %)
Writing at 0x00040000... (89 %)
Writing at 0x00044000... (94 %)
Writing at 0x00048000... (100 %)
Wrote 416656 bytes (302843 compressed) at 0x00000000 in 26.7 seconds (effective 124.8 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...

```

Figure 19: Writing program to ESP memory

This message means that our code is okay and without syntax errors and the program is successfully uploaded on the NodeMCU memory.

While the program has been uploaded it stays on the memory of our microcontroller and we do not have to upload it again.

4.6 Implementing the webserver locally | Running python commands

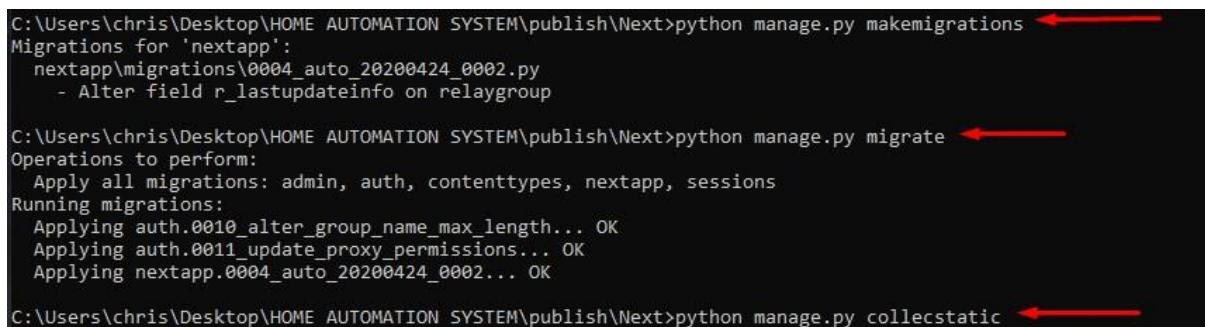
To implement the webserver and run it locally, 3 python commands are required.

These are:

Python manage.py makemigrations

Python manage.py migrate

Python manage.py collectstatic



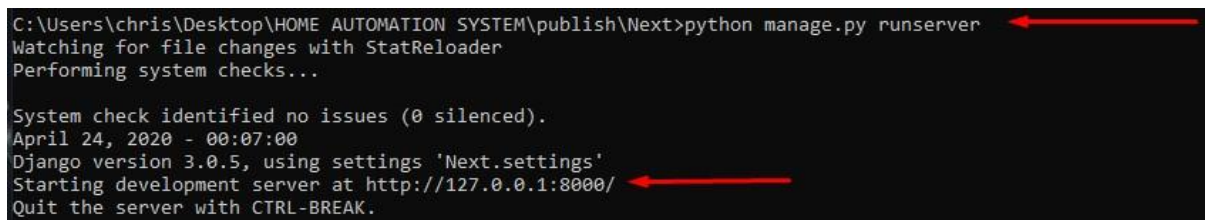
```
C:\Users\chris\Desktop\HOME AUTOMATION SYSTEM\publish\Next>python manage.py makemigrations
Migrations for 'nextapp':
  nextapp\migrations\0004_auto_20200424_0002.py
    - Alter field r_lastupdateinfo on relaygroup

C:\Users\chris\Desktop\HOME AUTOMATION SYSTEM\publish\Next>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, nextapp, sessions
Running migrations:
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying nextapp.0004_auto_20200424_0002... OK

C:\Users\chris\Desktop\HOME AUTOMATION SYSTEM\publish\Next>python manage.py collecstatic
```

Figure 20: Python commands

To run the server the command **python manage.py runserver** is required.



```
C:\Users\chris\Desktop\HOME AUTOMATION SYSTEM\publish\Next>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 24, 2020 - 00:07:00
Django version 3.0.5, using settings 'Next.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Figure 21: Command to run server locally

The server starts locally at **127.0.0.1:8000**

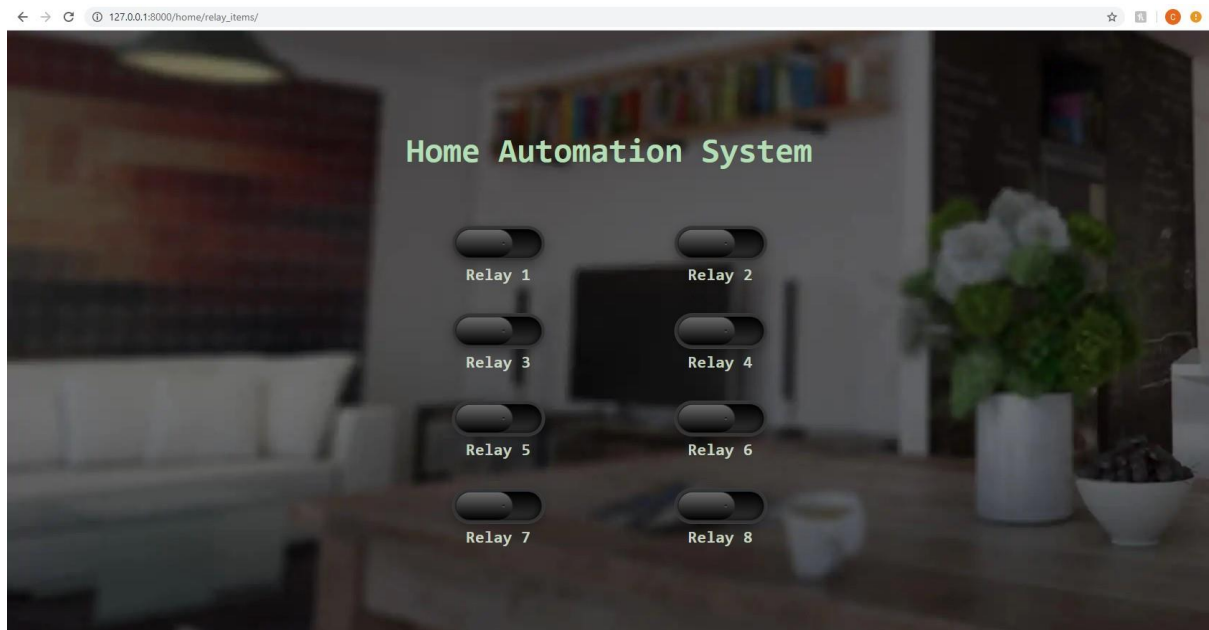


Figure 22: Server running locally

4.7 Publishing web-server to cloud platform

Why this is important?

This step is crucial for this project because it implements the idea of IoT. To publish the project to the cloud platform called *Heroku* there are some commands that are required to be executed.

First, we have to navigate to the path our project is stored. Then we have to add this project to a GitHub repository. This is done by command:

 **git add --all**

```
C:\Users\chris\Desktop\HOME AUTOMATION SYSTEM\publish - backup\Next>git add --all
```

Figure 23: Git command_1

Then we have to commit a change so they files are updated. This is done by command:

 **git commit -m "COMP39X Commit"**

```
C:\Users\chris\Desktop\HOME AUTOMATION SYSTEM\publish - backup\Next>git commit -m "COMP39X Commit"
```

Figure 24: Git command_2

Then, we have to create a Python application on the *Heroku* dashboard.

Our application is called **home-automation-system-comp39x**



Figure 25: Python application on Heroku Server

Next, we have to assign the project path with the Python application we created online.

This is done by:

 **Heroku git:remote -a home-automation-system-comp39x**

```
C:\Users\chris\Desktop\HOME AUTOMATION SYSTEM\publish - backup\Next>heroku git:remote -a home-automation-system-comp39x
set git remote heroku to https://git.heroku.com/home-automation-system-comp39x.git
```

Figure 26: Git command_3

Finally, we have to push the project files from our path to the cloud platform.

This is done by command:

 **Git push Heroku master --force**









```
C:\Users\chris\Desktop\HOME AUTOMATION SYSTEM\publish - backup\Next>git push heroku master --force_
```

Figure 27: Git command_4

Now that our project is published on the World Wide Web we can test its' functionality not only locally but also globally.

4.8 Summary of Implementation (Waterfall model)

Implemented successfully software and hardware and made them communicate together with IoT. To implement the hardware some electro-logical resources were required. While implementing the software knowledge from many fields was required. For the web-application, HTML knowledge and CSS was required, as well as knowledge in Python. For storing the data of the relays, an SQLite database was used. This data includes the status of a relay as well as the name of it, and the last update it has (date and time). Secure SSL protocol provided by the cloud platform also needs to be mentioned, making our web-application more secure and less vulnerable. This means that with the use of an SSL protocol, all the data transmitting from the user to the online dashboard is encrypted. Moreover, another big advantage of using SSL is to ensure that all data requested or submitted is actually delivered. For the coding part of this project, C++ knowledge was used as well as JSON environment knowledge. The milestones that were successfully implemented are stated below.

-  Created an HTML website to store our Home Automation dashboard.
-  HTML website stored under my personal link from the university
-  Used CSS to control the HTML elements and the layout of the website
-  Created a user-friendly GUI with the feature to turn on/off devices
-  Created SQLite database to store all information related to relays
-  Use SSL protocol in order to ensure that all requests sent to the server are received
-  Used C++ coding language for the Arduino IDE and main functions on the ESP8266 board
-  Created JSON array for the relay status

Chapter 5. Testing & Evaluation

5.1 Testing phase in Waterfall Model

To classify this project as functional there were many tests that needed to be carried out. On the testing phase of the Waterfall model followed, we had to test both software and hardware components.

1. The tests that occurred are stated below:
 - a. Tested the hardware components individually
 - a. Checked and ensured that they are in working conditions
 - b. Tested the C++ code on NodeMCU individually
 - a. Checked that NodeMCU is connecting to the WiFi
 - b. Checked that NodeMCU is sending the request to web-server
 - c. Tested the python scripts on the web-application individually
 - a. Checked that the status of relays changes on manual activation of a relay
 - d. Tested the overall communication of the above
 - a. Checked turning on a device through web-server
 - i. Checked if Serial Monitor is capturing the new status of a relay.
 - b. Checked turning on a device with voice commands
 - i. Checked if Serial Monitor is capturing the new status of a relay.

5.2 Testing in run-time

After the process of uploading to the NodeMCU is finished we can open the Serial Monitor of the Arduino IDE and check for the status of the relays.

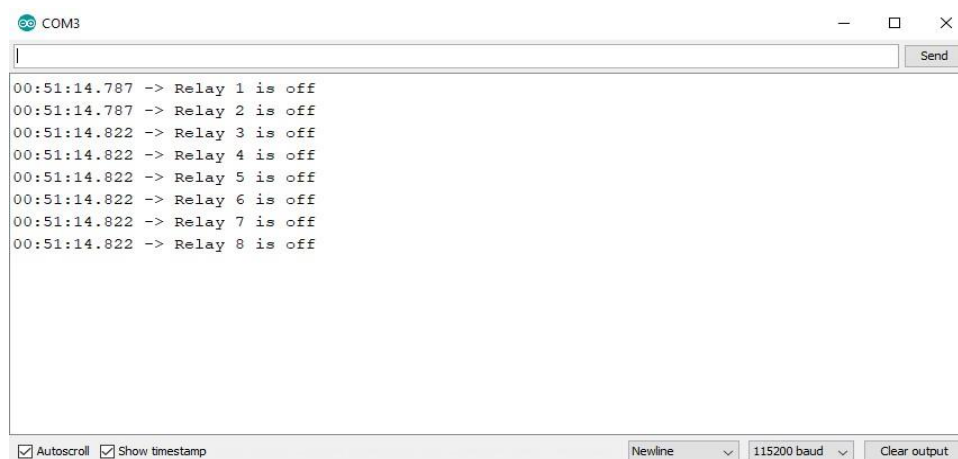


Figure 28: Serial Monitor showing Relay Status

To test the functionality of our project, we simply have to turn on the relays through the web-server.



Figure 29: Turning Relays on through Webserver

We visit the web-server and turn on all relays. This will result to a new relay status on the Serial Monitor of our Arduino IDE.

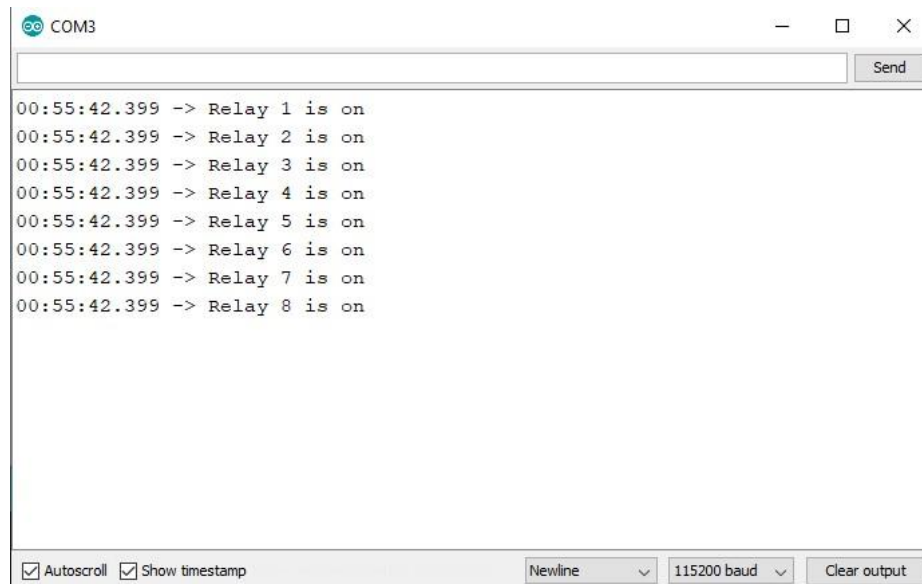


Figure 30: Serial Monitor showing new Relay Status

As we can see this test was successfully made and the Serial Monitor updates the relay status and also triggers them to turn on.

5.2.1 Powering on a Lamp

A 12V lamp is connected to Relay 1 with a 12V battery source.

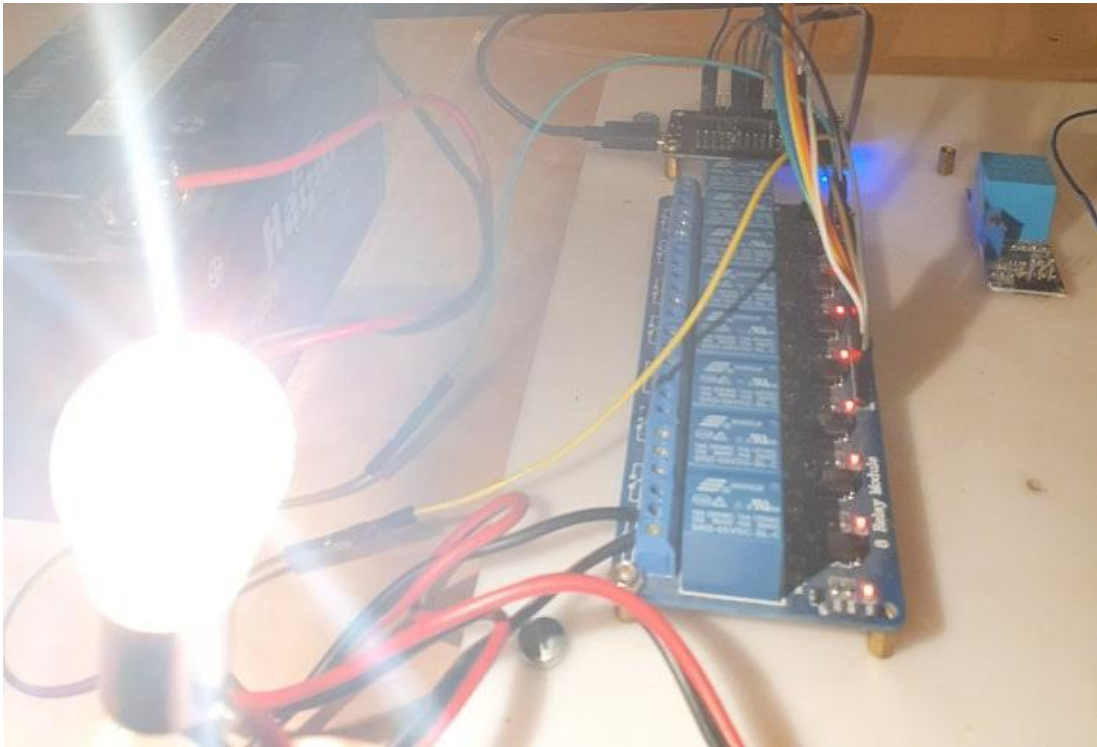


Figure 31: Hardware Components in run-time

The previous test we conducted not only switched on relays 1-8 on, but also triggered electricity to flow on the Lamp connected to Relay 1.

5.2.2 Testing voice commands

To test voice commands, we simply have to say “Alexa turn on <device name>”.

```
17:09:46.481 -> Relay 1 is on by Alexa
17:09:46.514 -> Relay 2 is off
17:09:46.514 -> Relay 3 is off
17:09:46.514 -> Relay 4 is off
17:09:46.514 -> Relay 5 is off
17:09:46.549 -> Relay 6 is off
17:09:46.549 -> Relay 7 is off
17:09:46.549 -> Relay 8 is off
```

Figure 32: Serial Monitor showing Relay activated with Alexa

Relay 1 and device attached to it, is successfully turned on by Alexa.

5.3 Evaluation

The tests conducted for this project were successful and they were used to evaluate the work done. The project meets the aims and objectives set in the beginning and it proposes a Home Automation System as described. Another evaluation of this project was to send the web-server link to friends and tell them to turn on specific relays and to test if the appropriate relay turned on by my side. This evaluated the idea of a Home Automation System that can be controlled globally over the World Wide Web. Final evaluation for this project was to visit the web-server with every possible combination. This was done by visiting the link through mobile phone, laptop and tablet, as well as, visiting the web-server using LTE network. All of the above tests were successful and the outcome was as expected.

Chapter 6. Conclusion

On this project we have used NodeMCU module which has built-in WiFi to control 8 relays locally or globally on every place on earth via visiting the web-application dashboard. This module is one of the easiest to code and most pocket-size friendly for Home Automation System based on IoT. This project proposes an efficient implementation for IoT (Internet of Things) used for monitoring and controlling home appliances via World Wide Web. Home Automation System is an easy to use and friendly online dashboard, which communicates with the NodeMCU through an Internet gateway using low power communication protocol like WiFi. Through the web-application the user can manage home appliances like lights, speakers, fans etc. The server interacts with these devices which each of them is connected to a relay. This project also proposes and implementation for IoT with voice commands with the help of Sinric repository and user can manage every device by saying “Alexa turn on <device-name>”.

6.1 Results

The results of the status of the relays were as expected either controlling them with the web-application dashboard or verbally with the Amazon Dot. All relays respond to commands and either turn on or off. The NodeMCU is responsible to send the signal which signal is LOW or HIGH. When a relay receives LOW signal from NodeMCU it lets electricity flow, while on HIGH signal electricity stops flowing. The Waterfall model followed for this project helped on to design and implement the project in a sufficient way.

6.2 Future Scope

Future scope of this project includes ideas on:

1. Deployment of the system and possible future research on this project
 - a. Further research on this project can include:
 - i. A newer version of this project based on this version
 - ii. Use of new technologies to communicate with the relays (LTE or 5G).

2. Deployment of the system and possible release on the market
 - a. We can look for things to improve and make it more appealing to the market (more automated so the user will do less things, more time saver for the user.)
 - b. Things that could be improved
 - i. More Efficient System
 - ii. Better Arduino Modules (less power consumption, more accuracy)
 - iii. Improving safety features
 1. More secure website (we can hide the login form, only the user will know the login form)
 2. More secure database by changing the tables prefix
3. Maintenance: Maintaining the system in the future
 - a. Supporting the system as a developer
 - i. We can add feature so the user can do changes on the system
 - b. Supporting the system as a user (user will be able to modify the system)
 - i. User can add or remove relays.
 - c. Looking for newer modules (to replace the ones used)
 - i. The use of new modules is highly recommended because
 1. Newer modules will use modern technologies
 2. Newer modules will use less power
 3. Newer modules can be more secure
 - d. Looking for different uses of the current project
 - i. This project can be used in many fields except homes, these are:
 1. Schools
 2. Offices
 3. Organizations
 4. Companies

Chapter 7. BCS Criteria & Self-Reflection

7.1 BCS Criteria

This thesis on Home Automation helped me gather and use skills gained during my degree programme. To make this project work, I had to use knowledge from many fields including modules that I have been taught in the previous years. Starting this project, I had to analyse the requirements and create an idea logically enough to be implemented. For all this to work, I used information from 4 modules that helped me finalise this project. These modules are:

- COMP201 – Software Engineering I
- COMP207 – Database Development
- COMP284 – Scripting Languages
- COMP329 – Robotics and Autonomous Systems

COMP201 module was really helpful on this project in order to understand what I have to build, what are the software resources needed and how to design and test my project. The waterfall model taught in the module was followed on this project.

COMP207 module helped me on to create and manage a Database which stores information regarding the relays of my project. The information stored in the database is:

1. Relay ID
2. Relay Name
3. Relay Status
4. Last update date and time

COMP284 module was extremely helpful regarding this project since I learned how to write and execute commands in python environment. Resources from this module were studied again regarding user permissions in python.

COMP329 module that was taught this year, included information on how to work with JSON environment and store data in this environment.

To be able to call this project innovative I had to use all this knowledge from 4 different modules into one project and also make this knowledge cooperate and work. This was successfully done in this thesis since the webserver which is based in Python has a database and it stores the relay information in JSON environment.

Synthesising all this information and making the project work was successfully made in 3 steps which are interconnected.

- ✚ Created a Python project which is the web-application
- ✚ Created a JSON array for storing the relays' information
- ✚ Created a Database to store the JSON information from the relays

Information from similar projects was also used, on how to update the status from the NodeMCU to the webserver.

This project not only achieves a Home Automation and takes us to a whole new era but it can also be released to the market and be sold. This of course requires many tests and evaluations before the release.

This project can also be used in industries, companies or even schools.

7.2 Self-Reflection

While doing this project on Home Automation I personally learned many things. The Research conducted for this project helped me to understand the idea of IoT, its' advantages and why it is becoming more and more powerful on our days. From the research to the implementation of this project, it was a personal process which was self-managed. The writing of this thesis also helped me to manage my time according to where it should spend more time on. In the beginning I had the idea of Home Automation clearly stated on my mind but in fact I was not sure how I will implement it. Research helped me a lot to understand what is needed for this project and the reason why it is needed.

The good things I learned during this project is to read publications related to my topic and to understand the scientific aspect of this project. This is a huge advantage for me that can be used in the future in order to do more research on similar projects. Another thing that needs mentioning is that I learned to code in C++ in a better and more effective way. The level of my C++ coding in the past was significantly lower than it is right now. The knowledge that I gained on C++ will be really helpful for me in the future while I have a better understanding of the coding language and I can code now certain functions without needing to look on guidance for them.

Of course, during this project not all things went well. In the beginning I had problems with the NodeMCU and the web-server. These problems were significant since I was not able to turn on and off relays. However, while researching on electro logical websites I found out that I was missing a jumper on my relay board and that was the issue the relays did not turn on. Without this jumper, the circuit was not complete and the electricity flowing from the source to the relays was not delivered. After managing to solve this problem I had another problem with Voice commands and Alexa. This problem was happening because the NodeMCU was connecting to the Sinric server and it was updating the status of a device there and not on my server. I managed to solve this problem too, with the use of a python script on my server where it checks on the Sinric website and it updates the status on my server based on the status of Sinric server.

In conclusion, on every project there will be errors and things that do not work as they should, but that is challenging. To make all these things work, you gain experience and you become better on problem-solving. This project helped me to fully understand the nature of every problem that I faced and how to solve it effectively.

References

- Albany, Jan. 16, 2015. *Home Automation Market - Global Industry*, NY: GLOBE NEWSWIRE.
- Daniela Popescul, Mircea Georgescu, 2013. *Internet Of Things–Some Ethical Issues*. [Online] Available at: https://www.researchgate.net/publication/260290933_Internet_Of_Things-Some_Ethical_Issues [Accessed 2020].
- Daniela Popescul, Mircea Georgescu, 2013. *Internet Of Things–Some Ethical Issues*. [Online] Available at: https://www.researchgate.net/publication/260290933_Internet_Of_Things-Some_Ethical_Issues [Accessed 2020].
- Elfriede Dustin, Thom Garrett, Bernie Gauf, 2009. *Implementing Automated Software Testing - Continuously Track Progress and Adjust Accordingly*. [Online] Available at: <http://www.methodsandtools.com/archive/archive.php?id=94> [Accessed 24 April 2020].
- Fritz, R., 2020. *Lifewire*. [Online] Available at: <https://www.lifewire.com/reasons-to-automate-your-home-817695>
- Gibson, C., 2018. *Types of Home Automation*. [Online] Available at: https://hipages.com.au/article/types_home_automation [Accessed 2020].
- Hipp, D. Richard, SQLite Development Team, 2020. *About SQLite*. [Online] Available at: <https://www.sqlite.org/about.html> [Accessed 2020].
- json.org, 2020. *Introducing JSON*. [Online] Available at: <https://www.json.org/json-en.html> [Accessed 2020].
- Python.org, 2020. *What is Python? Executive Summary*. [Online] Available at: <https://www.python.org/doc/essays/blurb/> [Accessed 2020].
- Rouse, M., 2020. *Internet of Things IoT*. [Online] Available at: <https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT> [Accessed April 2020].
- Wikipedia, 2020. *Database*. [Online] Available at: <https://en.wikipedia.org/wiki/Database> [Accessed 2020].
- Wikipedia, 2020. *HTML*. [Online] Available at: <https://en.wikipedia.org/wiki/HTML> [Accessed 2020].

Appendices

Arduino IDE code

```
#include <ESP8266HTTPClient.h>
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <WebSocketsClient.h> // https://github.com/kakopappa/sinric/wiki/How-to-add-dependency-libraries
#include <ArduinoJson.h> // https://github.com/kakopappa/sinric/wiki/How-to-add-dependency-libraries
#include <StreamString.h>
#include <EEPROM.h>

ESP8266WiFiMulti WiFiMulti;
WebSocketsClient webSocket;
WiFiClient client;

#define MyApiKey "API_KEY" // API From Sinric.com
#define MySSID "WIFI_SSID" // Wifi network SSID
#define MyWifiPassword "WIFI_PASSWORD" // Wifi network password

#define HEARTBEAT_INTERVAL 300000 // 5 Minutes

#define relay1_id "*****" // switch 1 id from sinric.com
#define relay2_id "*****" // switch 2 id from sinric.com
#define relay3_id "*****" // switch 3 id from sinric.com
#define relay4_id "*****" // switch 4 id from sinric.com
#define relay5_id "*****" // switch 5 id from sinric.com
#define relay6_id "*****" // switch 6 id from sinric.com
#define relay7_id "*****" // switch 7 id from sinric.com
#define relay8_id "*****" // switch 8 id from sinric.com

const int relay1 = 05; // GPIO05
const int relay2 = 04; // GPIO04
const int relay3 = 00; // GPIO00
const int relay4 = 02; // GPIO02
const int relay5 = 14; // GPIO14
const int relay6 = 12; // GPIO12
const int relay7 = 13; // GPIO13
const int relay8 = 15; // GPIO15

bool sinric1 = false; // sinric status
bool sinric2 = false; // sinric status
bool sinric3 = false; // sinric status
bool sinric4 = false; // sinric status
bool sinric5 = false; // sinric status
bool sinric6 = false; // sinric status
bool sinric7 = false; // sinric status
bool sinric8 = false; // sinric status

uint64_t heartbeatTimestamp = 0;
bool isConnected = false;
```

```

void setPowerStateOnServer(String deviceId, String value);
void setTargetTemperatureOnServer(String deviceId, String value, String scale);

// deviceId is the ID of the device from Sinric.com

void turnOn(String deviceId)
{
  if (deviceId == relay1_id) // Device ID of 1 device
  {
    Serial.print("Turn on device id: ");
    Serial.println(deviceId);
    digitalWrite (relay1,LOW);
    sinric1 = true;
    EEPROM.write(100, 1);
    EEPROM.commit();
  }
  else if (deviceId == relay2_id) // Device ID of 2 device
  {
    Serial.print("Turn on device id: ");
    Serial.println(deviceId);
    digitalWrite (relay2,LOW);
    sinric2 = true;
    EEPROM.write(101, 1);
    EEPROM.commit();
  }
  else if (deviceId == relay3_id) // Device ID of 3 device
  {
    Serial.print("Turn on device id: ");
    Serial.println(deviceId);
    digitalWrite (relay3,LOW);
    sinric3 = true;
    EEPROM.write(102, 1);
    EEPROM.commit();
  }
  else if (deviceId == relay4_id) // // Device ID of 4 device

  {
    Serial.print("Turn on device id: ");
    Serial.println(deviceId);
    digitalWrite (relay4,LOW);
    sinric4 = true;
    EEPROM.write(103, 1);
    EEPROM.commit();
  }
  else if (deviceId == relay5_id) // // Device ID of 5 device

  {
    Serial.print("Turn on device id: ");
    Serial.println(deviceId);
    digitalWrite (relay5,LOW);
    sinric5 = true;
    EEPROM.write(104, 1);
    EEPROM.commit();
  }
  else if (deviceId == relay6_id) //// Device ID of 6 device
  {
    Serial.print("Turn on device id: ");

```

```

Serial.println(deviceId);
digitalWrite (relay6,LOW);
sinric6 = true;
EEPROM.write(105, 1);
EEPROM.commit();
}
else if (deviceId == relay7_id) // Device ID of 7 device
{
  Serial.print("Turn on device id: ");
  Serial.println(deviceId);
  digitalWrite (relay7,LOW);
  sinric7 = true;
  EEPROM.write(106, 1);
  EEPROM.commit();
}
else if (deviceId == relay8_id) // Device ID of 8 device
{
  Serial.print("Turn on device id: ");
  Serial.println(deviceId);
  digitalWrite (relay8,LOW);
  sinric8 = true;
  EEPROM.write(107, 1);
  EEPROM.commit();
}
else {
  Serial.print("Turn on for unknown device id: ");
  Serial.println(deviceId);
}
}

void turnOff(String deviceId)
{
  if (deviceId == relay1_id) // Device ID of 1 device
  {
    Serial.print("Turn off Device ID: ");
    Serial.println(deviceId);
    digitalWrite (relay1,HIGH);
    sinric1 = false;
    EEPROM.write(100, 0);
    EEPROM.commit();
  }
  else if (deviceId == relay2_id) // Device ID of 2 device
  {
    Serial.print("Turn off Device ID: ");
    Serial.println(deviceId);
    digitalWrite (relay2,HIGH);
    sinric2 = false;
    EEPROM.write(101, 0);
    EEPROM.commit();
  }
  else if (deviceId == relay3_id) // Device ID of 3 device
  {
    Serial.print("Turn off Device ID: ");
    Serial.println(deviceId);
    digitalWrite (relay3,HIGH);
    sinric3 = false;

```

```

EEPROM.write(102, 0);
EEPROM.commit();
}
else if (deviceId == relay4_id) // Device ID of 4 device
{
    Serial.print("Turn off Device ID: ");
    Serial.println(deviceId);
    digitalWrite (relay4,HIGH);
    sinric4 = false;
    EEPROM.write(103, 0);
    EEPROM.commit();
}
else if (deviceId == relay5_id) // Device ID of 5 device
{
    Serial.print("Turn off Device ID: ");
    Serial.println(deviceId);
    digitalWrite (relay5,HIGH);
    sinric5 = false;
    EEPROM.write(104, 0);
    EEPROM.commit();
}
else if (deviceId == relay6_id) // Device ID of 6 device
{
    Serial.print("Turn off Device ID: ");
    Serial.println(deviceId);
    digitalWrite (relay6,HIGH);
    sinric6 = false;
    EEPROM.write(105, 0);
    EEPROM.commit();
}
else if (deviceId == relay7_id) // Device ID of 7 device
{
    Serial.print("Turn off Device ID: ");
    Serial.println(deviceId);
    digitalWrite (relay7,HIGH);
    sinric7 = false;
    EEPROM.write(106, 0);
    EEPROM.commit();
}
else if (deviceId == relay8_id) // Device ID of 1 device
{
    Serial.print("Turn off Device ID: ");
    Serial.println(deviceId);
    digitalWrite (relay8,HIGH);
    sinric8 = false;
    EEPROM.write(107, 0);
    EEPROM.commit();
}
else {
    Serial.print("Turn off for unknown device id: ");
    Serial.println(deviceId);
}
}

void websocketEvent(WStype_t type, uint8_t * payload, size_t length)
{
    switch(type) {

```

```

case WType_DISCONNECTED:
    isConnected = false;
    Serial.printf("[WSc] Webservice disconnected from sinric.com!\n");
    break;
case WType_CONNECTED: {
    isConnected = true;
    Serial.printf("[WSc] Service connected to sinric.com at url: %s\n", payload);
    Serial.printf("Waiting for commands from sinric.com ...\n");
}
    break;
case WType_TEXT: {
    Serial.printf("[WSc] get text: %s\n", payload);

    #if ARDUINOJSON_VERSION_MAJOR == 5
        DynamicJsonBuffer jsonBuffer;
        JsonObject& json = jsonBuffer.parseObject((char*)payload);
    #endif
    #if ARDUINOJSON_VERSION_MAJOR == 6
        DynamicJsonDocument json(1024);
        deserializeJson(json, (char*) payload);
    #endif

    String deviceId = json ["deviceId"];
    String action = json ["action"];

    if(action == "setPowerState") { // Switch or Light
        String value = json ["value"];
        if(value == "ON") {
            turnOn(deviceId);
        } else {
            turnOff(deviceId);
        }
    }
    else if(action == "SetColor") {
        // Alexa, set the device name to red
        // get text: {"deviceId":"xxx","action":"SetColor","value":{"hue":0,"saturation":1,"brightness":1}}
        String hue = json ["value"]["hue"];
        String saturation = json ["value"]["saturation"];
        String brightness = json ["value"]["brightness"];

        Serial.println("[WSc] hue: " + hue);
        Serial.println("[WSc] saturation: " + saturation);
        Serial.println("[WSc] brightness: " + brightness);
    }
    else if(action == "SetBrightness") {

    }
    else if(action == "AdjustBrightness") {

    }
    else if (action == "test") {
        Serial.println("[WSc] received test command from sinric.com");
    }
}
    break;
case WType_BIN:
    Serial.printf("[WSc] get binary length: %u\n", length);

```

```

    break;
    default: break;
}
}

void setup() {
  Serial.begin(115200);
  EEPROM.begin(1024);
  pinMode (relay1,OUTPUT); // Initialising the relays
  pinMode (relay2,OUTPUT); // Initialising the relays
  pinMode (relay3,OUTPUT); // Initialising the relays
  pinMode (relay4,OUTPUT); // Initialising the relays
  pinMode (relay5,OUTPUT); // Initialising the relays
  pinMode (relay6,OUTPUT); // Initialising the relays
  pinMode (relay7,OUTPUT); // Initialising the relays
  pinMode (relay8,OUTPUT); // Initialising the relays

  // HIGH = Closed relay, No electricity flows
  // LOW = Open relay, Electricity flows
  // Setting relays to HIGH state on boot For safety reasons

  digitalWrite(relay1,HIGH); // Setting relay to HIGH for safety
  digitalWrite(relay2,HIGH); // Setting relay to HIGH for safety
  digitalWrite(relay3,HIGH); // Setting relay to HIGH for safety
  digitalWrite(relay4,HIGH); // Setting relay to HIGH for safety
  digitalWrite(relay5,HIGH); // Setting relay to HIGH for safety
  digitalWrite(relay6,HIGH); // Setting relay to HIGH for safety
  digitalWrite(relay7,HIGH); // Setting relay to HIGH for safety
  digitalWrite(relay8,HIGH); // Setting relay to HIGH for safety

  WiFiMulti.addAP(MySSID, MyWifiPassword);
  Serial.println();
  Serial.print("Connecting to Wifi: ");
  Serial.println(MySSID);

  // Waiting for Wifi connect
  while(WiFiMulti.run() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  if(WiFiMulti.run() == WL_CONNECTED) {
    Serial.println("");
    Serial.print("WiFi connected. ");
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
  }

  // server address, port and URL
  websocket.begin("iot.sinric.com", 80, "/");

  // event handler
  websocket.onEvent(webSocketEvent);
  websocket.setAuthorization("apikey", MyApiKey);

  // try again every 5000ms if connection has failed
  websocket.setReconnectInterval(5000); // If you see 'class WebSocketsClient' has no member named
  'setReconnectInterval' error update arduinoWebSockets

```

```

if (EEPROM.read(100) == 1){digitalWrite (relay1,LOW);}
else if (EEPROM.read(100) == 0){digitalWrite (relay1,HIGH);}
if (EEPROM.read(101) == 1){digitalWrite (relay2,LOW);}
else if (EEPROM.read(101) == 0){digitalWrite (relay2,HIGH);}
if (EEPROM.read(102) == 1){digitalWrite (relay3,LOW);}
else if (EEPROM.read(102) == 0){digitalWrite (relay3,HIGH);}
if (EEPROM.read(103) == 1){digitalWrite (relay4,LOW);}
else if (EEPROM.read(103) == 0){digitalWrite (relay4,HIGH);}
if (EEPROM.read(104) == 1){digitalWrite (relay5,LOW);}
else if (EEPROM.read(104) == 0){digitalWrite (relay5,HIGH);}
if (EEPROM.read(105) == 1){digitalWrite (relay6,LOW);}
else if (EEPROM.read(105) == 0){digitalWrite (relay6,HIGH);}
if (EEPROM.read(106) == 1){digitalWrite (relay7,LOW);}
else if (EEPROM.read(106) == 0){digitalWrite (relay7,HIGH);}
if (EEPROM.read(107) == 1){digitalWrite (relay8,LOW);}
else if (EEPROM.read(107) == 0){digitalWrite (relay8,HIGH);}

}

void loop()
{
  websocket.loop();
  delay(1000);
  HTTPClient http;

  http.begin("http://home-automation-system-comp39x.herokuapp.com/home/get_relay_group_status/");
  //The URL of the web-server
  int httpCode = http.GET(); //Make the request

  if (httpCode > 0) { //Check for the returning code

    String payload = http.getString();

    const size_t capacity = JSON_ARRAY_SIZE(8) + 8 * JSON_OBJECT_SIZE(4) + 620;
    DynamicJsonDocument doc(capacity);

    char charBuf[capacity];

    payload.toCharArray(charBuf, capacity);

    deserializeJson(doc, charBuf);

    JsonObject root_0 = doc[0];
    bool root_0_r_status = root_0["r_status"];

    JsonObject root_1 = doc[1];
    bool root_1_r_status = root_1["r_status"];

    JsonObject root_2 = doc[2];
    bool root_2_r_status = root_2["r_status"];

    JsonObject root_3 = doc[3];
    bool root_3_r_status = root_3["r_status"];

    JsonObject root_4 = doc[4];

```



```

bool root_4_r_status = root_4["r_status"];

JsonObject root_5 = doc[5];
bool root_5_r_status = root_5["r_status"];

JsonObject root_6 = doc[6];
bool root_6_r_status = root_6["r_status"];

JsonObject root_7 = doc[7];
bool root_7_r_status = root_7["r_status"];

    if (root_0_r_status == true || sinric1 == true) { // checking status on web-server && sinric server
        Serial.println("Relay 1 is on ");
        digitalWrite(relay1, LOW); // turn on relay
        setPowerStateOnServer(relay1_id, "ON"); // turn on relay
        EEPROM.write(100, 1);
        EEPROM.commit();
    } else {
        Serial.println("Relay 1 is off ");
        digitalWrite(relay1, HIGH);
        setPowerStateOnServer(relay1_id, "OFF");
        EEPROM.write(100, 0);
        EEPROM.commit();
    }
    if (root_1_r_status || sinric2 == true) {
        Serial.println("Relay 2 is on ");
        digitalWrite(relay2, LOW);
        setPowerStateOnServer(relay2_id, "ON");
        EEPROM.write(101, 1);
        EEPROM.commit();
    } else {
        Serial.println("Relay 2 is off ");
        digitalWrite(relay2, HIGH);
        setPowerStateOnServer(relay2_id, "OFF");
        EEPROM.write(101, 0);
        EEPROM.commit();
    }
    if (root_2_r_status || sinric3 == true) {
        Serial.println("Relay 3 is on ");
        digitalWrite(relay3, LOW);
        setPowerStateOnServer(relay3_id, "ON");
        EEPROM.write(102, 1);
        EEPROM.commit();
    } else {
        Serial.println("Relay 3 is off ");
        digitalWrite(relay3, HIGH);
        setPowerStateOnServer(relay3_id, "OFF");
        EEPROM.write(102, 0);
        EEPROM.commit();
    }
    if (root_3_r_status || sinric4 == true) {
        Serial.println("Relay 4 is on ");
        digitalWrite(relay4, LOW);
        setPowerStateOnServer(relay4_id, "ON");
    }

```

```

EEPROM.write(103, 1);
EEPROM.commit();
} else {
  Serial.println("Relay 4 is off ");
  digitalWrite(relay4, HIGH);
  setPowerStateOnServer(relay4_id, "OFF");
  EEPROM.write(103, 0);
  EEPROM.commit();
}

if (root_4_r_status || sinric5 == true) {
  Serial.println("Relay 5 is on ");
  digitalWrite(relay5, LOW);
  setPowerStateOnServer(relay5_id, "ON");
  EEPROM.write(104, 1);
  EEPROM.commit();
} else {
  Serial.println("Relay 5 is off ");
  digitalWrite(relay5, HIGH);
  setPowerStateOnServer(relay5_id, "OFF");
  EEPROM.write(104, 0);
  EEPROM.commit();
}

if (root_5_r_status || sinric6 == true) {
  Serial.println("Relay 6 is on ");
  digitalWrite(relay6, LOW);
  setPowerStateOnServer(relay6_id, "ON");
  EEPROM.write(105, 1);
  EEPROM.commit();
} else {
  Serial.println("Relay 6 is off ");
  digitalWrite(relay6, HIGH);
  setPowerStateOnServer(relay6_id, "OFF");
  EEPROM.write(105, 0);
  EEPROM.commit();
}

if (root_6_r_status || sinric7 == true) {
  Serial.println("Relay 7 is on ");
  digitalWrite(relay7, LOW);
  setPowerStateOnServer(relay7_id, "ON");
  EEPROM.write(106, 1);
  EEPROM.commit();
} else {
  Serial.println("Relay 7 is off ");
  digitalWrite(relay7, HIGH);
  setPowerStateOnServer(relay7_id, "OFF");
  EEPROM.write(106, 0);
  EEPROM.commit();
}

if (root_7_r_status || sinric7 == true) {
  Serial.println("Relay 8 is on ");
  digitalWrite(relay8, LOW);
  setPowerStateOnServer(relay8_id, "ON");
  EEPROM.write(107, 1);

```

```

    EEPROM.commit();
} else {
    Serial.println("Relay 8 is off ");
    digitalWrite(relay8, HIGH);
    setPowerStateOnServer(relay8_id, "OFF");
    EEPROM.write(107, 0);
    EEPROM.commit();
}
}

if(isConnected)
{
    uint64_t now = millis();
    // Send heartbeat in order to avoid disconnections during ISP resetting IPs over night. Thanks @MacSass
    if((now - heartbeatTimestamp) > HEARTBEAT_INTERVAL)
    {
        heartbeatTimestamp = now;
        websocket.sendTXT("H");
    }
}
http.end(); //Free the resources
}

```

```

void setPowerStateOnServer(String deviceId, String value) {
    #if ARDUINOJSON_VERSION_MAJOR == 5
        DynamicJsonBuffer jsonBuffer;
        JsonObject& root = jsonBuffer.createObject();
    #endif
    #if ARDUINOJSON_VERSION_MAJOR == 6
        DynamicJsonDocument root(1024);
    #endif
    root["deviceId"] = deviceId;
    root["action"] = "setPowerState";
    root["value"] = value;
    StreamString databuf;
    #if ARDUINOJSON_VERSION_MAJOR == 5
        root.printTo(databuf);
    #endif
    #if ARDUINOJSON_VERSION_MAJOR == 6
        serializeJson(root, databuf);
    #endif

    websocket.sendTXT(databuf);
}

```

//eg: setPowerStateOnServer("deviceid", "CELSIUS", "25.0")

// Call ONLY If status changed. DO NOT CALL THIS IN loop() and overload the server.

```

void setTargetTemperatureOnServer(String deviceId, String value, String scale) {
    #if ARDUINOJSON_VERSION_MAJOR == 5
        DynamicJsonBuffer jsonBuffer;
        JsonObject& root = jsonBuffer.createObject();
    #endif
    #if ARDUINOJSON_VERSION_MAJOR == 6
        DynamicJsonDocument root(1024);
    #endif

```

```

#endif
root["action"] = "SetTargetTemperature";
root["deviceId"] = deviceId;

#if ARDUINOJSON_VERSION_MAJOR == 5
JsonObject& valueObj = root.createNestedObject("value");
JsonObject& targetSetpoint = valueObj.createNestedObject("targetSetpoint");
#endif
#if ARDUINOJSON_VERSION_MAJOR == 6
JsonObject valueObj = root.createNestedObject("value");
JsonObject targetSetpoint = valueObj.createNestedObject("targetSetpoint");
#endif
targetSetpoint["value"] = value;
targetSetpoint["scale"] = scale;

StreamString databuf;
#if ARDUINOJSON_VERSION_MAJOR == 5
root.printTo(databuf);
#endif
#if ARDUINOJSON_VERSION_MAJOR == 6
serializeJson(root, databuf);
#endif

websocket.sendTXT(databuf);
}

```

Web-application code

Settings.py file

```

import django_heroku
import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'h(q+cgnga8ijqas+b*e-i_bdear&um6+hm*q5cx06j_!$5icy2'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['*']

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',

```

```

'django.contrib.messages',
'django.contrib.staticfiles',
'nextapp.apps.NextappConfig',

]

MIDDLEWARE = [
'django.middleware.security.SecurityMiddleware',
'django.contrib.sessions.middleware.SessionMiddleware',
'django.middleware.common.CommonMiddleware',
'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'Next.urls'

TEMPLATES = [
{
'BACKEND': 'django.template.backends.django.DjangoTemplates',
'DIRS': [os.path.join(BASE_DIR, 'templates')],
'APP_DIRS': True,
'OPTIONS': {
'context_processors': [
'django.template.context_processors.debug',
'django.template.context_processors.request',
'django.contrib.auth.context_processors.auth',
'django.contrib.messages.context_processors.messages',
],
},
},
]

WSGI_APPLICATION = 'Next.wsgi.application'

# Database
# https://docs.djangoproject.com/en/2.0/ref/settings/#databases

DATABASES = {
'default': {
'ENGINE': 'django.db.backends.sqlite3',
'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
}
}

# Password validation
# https://docs.djangoproject.com/en/2.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
{
'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
},
{

```

```

        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

```

Internationalization

<https://docs.djangoproject.com/en/2.0/topics/i18n/>

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

DATE_INPUT_FORMATS = ['%d-%m-%Y']

Static files (CSS, JavaScript, Images)

<https://docs.djangoproject.com/en/2.0/howto/static-files/>

STATIC_URL = '/static/'

STATIC_ROOT = os.path.join(BASE_DIR, 'static')

MEDIA_URL = '/media/'

MEDIA_ROOT = os.path.join(BASE_DIR, "media")

Activate Django-Heroku.

django_heroku.settings(locals())

Urls.py

```

from django.conf.urls import url, include
from . import views
from django.views.generic import TemplateView
from django.contrib.auth import views as auth_views
from django.urls import path

```

#from .views import FileView

app_name = 'nextapp'

urlpatterns = [

```

    path('ajax_update_relay_status/', views.ajax_update_relay_status, name='ajax_update_relay_status'),
    path('get_relay_group_status/', views.get_relay_group_status, name='get_relay_group_status'),
    path('relay_items/', views.relay_items, name='relay_items'),

```

]

Urls.py in app

```
# -*- coding: utf-8 -*-
```

```
from django.http import JsonResponse
# from .serializers import PlayerSerializer
from django.views.decorators.csrf import csrf_exempt
from .models import *
from django.shortcuts import render
from django.core import serializers
import datetime
from django.core.files.storage import default_storage
```

```
##### 11-04-2020
#####
```

```
def relay_items(request):
    all_relay_items = RelayGroup.objects.all().order_by('pk')
    context={"all_relay_items":all_relay_items}
    return render(request, "templates/relay_items.html",context)
```

```
def ajax_update_relay_status (request):
    status = request.GET.get('status')
    relay_number = request.GET.get('relay_number')
    #import pdb;pdb.set_trace()
    try :
        relay = RelayGroup.objects.get(pk=relay_number)
        relay.r_status = True if status=="1" else False
        relay.r_lastupdateinfo = datetime.datetime.now()
        relay.save()
        data = {"Status": "Ok"}
    except :
        data = {"Status": "Error"}
    #print(data)
    return JsonResponse(data, safe=False)
```

```
def get_relay_group_status (request):
    #import pdb;pdb.set_trace()
    try :
        relay_group = list(RelayGroup.objects.all().values())
    except :
        relay_group = None
    #print(data)
    return JsonResponse(relay_group, safe=False)
```

Models.py

```
from django.db import models
from django.contrib.auth.models import User
import datetime,os
class RelayGroup(models.Model):
```

```
    r_name = models.CharField(max_length=100,blank=True,null=True,unique=True)
    r_status = models.BooleanField(default=False)
    r_lastupdateinfo = models.DateTimeField(default= datetime.datetime.now())
    #ow_phone_store = models.CharField(max_length=30, blank=True, null=True)
    #ow_phone_store_sec = models.CharField(max_length=30, blank=True, null=True)
```

```
def __str__(self):
    return self.r_name
class Meta:
    db_table = 'RelayGroup'
```

admin.py

```
from django.contrib import admin
from .models import *
# Register your models here.

admin.site.register(RelayGroup)
```

Relay_items.html

```
{% load static %}
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="refresh" content="300">
    <title>COMP39X</title>
    <link rel="stylesheet" type="text/css" href="{% static 'style.css' %}">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
</head>
<body style="background-image: url({% static 'images/cover.jpg' %});">

    <div class="center">
        <table style="border-style:none;border: 1">
            <thead>
                <td colspan="2" style="border-style: none;border: 1;padding: 25px;color: #b0deb5;font-family:
Monospace; font-weight: bold;font-size: 180%">
                    <h1> Home Automation System
                </h1>
            </td>

        </thead>
        <tbody>

            {% for relay in all_relay_items %}
            {% if forloop.counter0|divisibleby:2 %}
                <tr><th style="padding: 20px;"><label style="color: #dce8d7e3;;font-family: Monospace;font-weight:
bold;font-size: 180%"><input type="checkbox" name="{{ relay.id }}" {% if relay.r_status %} checked{% endif
%} /></br>{{ relay.r_name}}</label>
                </th>

            {% else %}
                <th style="padding: 20px;"><label style="color: #dce8d7e3;;font-family: Monospace;font-weight:
bold;font-size: 180%"><input type="checkbox" name="{{ relay.id }}" {% if relay.r_status %} checked{% endif
%} /></br>{{ relay.r_name }}</label>
                </th></br>
            </tr>
            {% endif %}
            {% endfor %}

        </tbody>
```



```

</table>

</div>
<script>

$(document).ready(function()
{
    $('input[type=checkbox]').change(function(e)
    {
        console.log(''+e.target.name+" Status : "+ e.target.checked);

        status = 0;
        var relay_number = e.target.name;

        if (e.target.checked){status=1}
        else {status=0}

        $.ajax({
            url: "{% url 'nextapp:ajax_update_relay_status' %}",
            data: {
                status:status,
                relay_number :relay_number
            },
            dataType: 'json',
            type: "GET",
            success: function (data){
                if (data.Status == "Ok"){

                    console.log (''+relay_number+" : "+status +"updated successfully");

                }
                else {
                    console.log (''+relay_number+" : "+status +" Not updated ");
                }

                //window.location.reload()

            },
            error: function( error )
            {
                alert("error:" + error.responseText);
            }

        });

    });

});

</script>

```

```
</body>
</html>
```

Style.css

```
body
{
    margin: 0;
    padding: 0;
    /*background: #212d33d1;*/
    /* background: #292929;
    */

}

.center
{
    position: absolute;
    top: 40%;
    left: 50%;
    transform: translate(-50%,-50%);
}

input[type="checkbox"]
{
    position: relative;
    width: 120px;
    height: 40px;
    -webkit-appearance: none;
    background: linear-gradient(0deg,#333,#000);
    outline: none;
    border-radius: 20px;
    box-shadow: 0 0 4px #353535, 0 0 5px #3e3e3e , inset 0 0 10px rgba(0,0,0, 1), 0 5px 20px rgba(0,0,0,.5),
inset 0 0 15px rgba(0,0,0,.2);

}

input:checked[type="checkbox"]
{
    background: linear-gradient(0deg,#50ff00,#04fd00);
    box-shadow: 0 0 2px #4dff00, 0 0 4px #353535, 0 0 5px #3e3e3e, inset 0 0 10px rgba(0,0,0, 1), 0 5px
20px rgba(0,0,0,.5), inset 0 0 15px rgba(0,0,0,.2);
}

input[type="checkbox"]:before{
    content: "";
    position: absolute;
    top: 0;
    left: 0;
    width: 80px;
    height: 40px;
    background:linear-gradient(0deg,#000, #6b6b6b);
    border-radius: 20px;
    box-shadow: 0 0 1px #232323;
    transform:scale(.98,.96);
```

```

    transition: 0.5s;

}
input:checked[type="checkbox"]:before{
    left: 40px;
}

input[type="checkbox"]:after{
    content: "";
    position: absolute;
    top: calc(50% - 2px);
    left: 65px;
    width: 4px;
    height: 4px;
    background: linear-gradient(0deg,#6b6b6b,#000) ;
    border-radius: 50%;
    transition: 0.5s;
}
input:checked[type="checkbox"]:after{
    background: #7aff00;
    left: 105px;
    box-shadow: 0 0 5px #ffd100, 0 0 15px #fff000;
}

```