UNIVERSITY OF NAIROBI

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND INFORMATION ENGINEERING

PROJECT TITLE:

INTERNET OF THINGS BUILDING MANAGEMENT SYSTEM

PROJECT INDEX: PRJ (61)

SUBMITTED BY:

OJAKAA PETER OKWARA

F17/36211/2013

SUPERVISOR: SAYYID AHMED ALI

EXAMINER: DR KAMUCHA

A project report submitted to the Department of Electrical and Information Engineering in partial fulfillment of the requirements of Bsc. Electrical and Electronic Engineering of the University of Nairobi.

DATE OF SUBMISSION: 29th April 2019

DECLARATION OF ORIGINALITY

NAME OF STUDENT:	OJAKAA PETER OKWARA
REGISTRATION NUMBER:	F17/36211/2013
COLLEGE:	ARCHITECTURE AND ENGINEERING
FACULTY:	ENGINEERING
DEPARTMENT:	ELECTRICAL AND INFORMATION ENGINEERING
COURSE NAME:	BACHELOR OF SCIENCE IN ELECTRICAL AND INFORMATION ENGINEERING
TITLE OF WORK:	INTERNET OF THINGS BUILDING MANAGEMENT SYSTEM

1. I understand what plagiarism is and I am aware of the university policy in this regard.
2. I declare that this final year project is my original work and has not been submitted elsewhere for examination, the award of a degree or publication. Where other people's work or my own has been used, this has properly been acknowledged and referenced in accordance with the University of Nairobi's requirement.
3. I have not sought or used the service of any professional agencies to produce this work.
4. I have not allowed, and shall not allow anyone to copy my work with the intention of passing it off as his/her own work.
5. I understand that any false claim in respect of this work shall result in disciplinary action, in accordance with the university anti-plagiarism policy.

Signature:

.....

Date:

.....

CERTIFICATION

This project report has been submitted to the Department of Electrical and Information Engineering at the University of Nairobi with my approval as a supervisor.

.....

SAYYID AHMED ALI

Date: .....

DEDICATION

I dedicate this work to my family, who have been my resource in many ways and forms. Julius Ojala, my brother, who has supported me and encouraged me to do my best during the tough times, my friends, Gilbert Kigen, Isaac Ndarwa, Brian Minyalwa, Alfie Wanjiru and Eunice Kimotho who have been a great source of inspiration throughout the long journey, Bob Afwata, and Arthur Ken Otieno who have given me advise on best ways to implement my project.

ACKNOWLEDGMENTS

I wish to acknowledge my supervisor Sayyid Ahmed who has been more generous with his expert knowledge and experience. I would like to thank him for the numerous motivational quotes, encouragement and life advice that he has given me throughout the project.

I would also like to acknowledge Ahlam, a software engineer who gave me advice on how to go about documenting the software aspects of my projects and how I could improve on my project.

Finally, I would like to acknowledge the University of Nairobi for giving me this opportunity to conduct this project and providing the resources that were needed to do the project.

Most of all, I thank **G**od for seeing me through completion.

ABSTRACT

This project seeks to demonstrate the use of MQTT protocol for an IoT Building Management System for commercial usage, where parameters such as building occupancy, energy demand, and energy consumption are measured and displayed on an online database and dashboard. MQTT brings its wireless technology, used for most IoT projects that are simple to implement, optimized for devices with low battery, limited processing power, and limited storage, can run across network constraints such as high link failure rates, supports low bandwidth and can run on any network that supports point to point data transfer.

In this project, we will be gathering data on the temperature, humidity, movement in and out of a room, current and power consumption of a bulb and fan, and control of the fan and a bulb within a room using relays.

We will be looking deeper into the technology that will be used in the project. The methodology and design of the project will be covered to show the process of a fully functional system.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ADC	Analog to Digital Converter
API	Application Interface
AWS	Amazon Web Service
CLI	Command Line Interface
DevOps	Development and Operations
DHT	Digital Humidity and Temperature
GPIO	General Purpose Input Output
IR	InfraRed
IC	Integrated Circuit
JSON	JavaScript Object Notation
PaaS	Platform As A Service
PCB	Printed Circuit Board
PIR	Passive InfraRed
SOC	System On Chip
UI	User Interface
USB	Universal Serial Bus

1 CHAPTER ONE: INTRODUCTION

1.1 Study Background

The national electrification level in Kenya is below 15%. Only about 5% of the rural households have access to electricity while biomass, mainly firewood accounts for 77% of the total energy consumed [1].

Kenya's energy sources consist of imported fossil fuels and renewable sources which include biomass, hydro, geothermal, solar and wind.

The current renewable energy source power generation is at about 5% of the total potential.

The output from Geothermal comes to 593 MW, from hydroelectricity comes to 827 MW, wind at 26 MW, fuel oil at 751 MW and biomass at 38 MW [2].

1.2 Problem Statement

In many countries, energy generation, distribution, and consumption are issues of vital importance and should receive top priority. Yet the challenges for the world's energy markets could not be more different. Energy demands in the growth markets are increasing rapidly, but in developed countries, the focus is on cost-effectiveness and climate protection.

Most energy in buildings is consumed by Heating, Ventilation and Air Conditioning (HVAC), water heating, telecommunication, and lighting.

Buildings account for 40 percent of total energy consumption. Examples include residential buildings as well as hotel shopping centers and industrial buildings.

The importance of energy management is brought up by three main factors:

i. The growing energy costs

ii. The ever-growing environmental regulations (also additional costs related to co2 emissions)

iii. 'Green customers' purchasing behaviors as regards to products and services.

Diesel and electricity costs have been on the high, with Kenyans paying 16.7% percent more for a liter of diesel in November compared to a year earlier. Household consuming about 200 units for electricity had to pay around 4,500 Ksh, representing a 12.63% increment as compared with before [3].

High oil prices have an effect on the pump prices because this determines the cost of transportation which in turn affects energy prices as some of our energy producing plants run on diesel.

Kenya heavily relies on hydro-electric power, at about 50%. Last year, consumers were faced with high power bills after a drop-in water level. This was caused by a prolonged period of drought.

It is apparent that in the future, electricity tariffs will continue to be on the rise due to climate change.

1.3 Study Objective

The objective is to implement a Smart Office System using IoT (Internet of Things) that is capable of controlling and automating most of the workplace devices effortlessly through a web application. Data gathered from the sensors will also be stored in the online database of the web application.

The devices and sensors will provide the employee with both qualitative and quantitative results to the employee with minimal effort.

1.4 Justification

The project seeks to solve the problem of energy usage through automation and energy monitoring to establish the best times to turn on equipment to efficiently conserve energy and monitor energy consumption patterns.

Wireless technologies are becoming more popular around the world and consumers appreciate this wireless existence which gives them reliance of the well-known tedious “cable chaos” smart office to provide an intelligent living environment for day to day easy and convenient life.

Automation enables holding routines enables responsibilities to a smart framework and decreases the cost of human blunder. The proposed system is intended to overcome the flaws of the previous system.

The proposed office automation using Esp8266, Raspberry Pi and Website makes the system quite interactive to provide ease in day to day life thereby improving the efficiency, flexibility, reliability and saves electricity and human efforts.

Lowering energy consumption using intelligent building automation system often requires little investments and help save a significant amount of energy, reduce CO_2 emissions and shorten payback periods.

In fact, measuring and controlling energy consumption are essential activities to accomplish the objectives of reducing energy wastes and raising energy efficiency awareness, and nowadays internet of things is nowadays fostering these activities, making a huge amount of data easily available for analyses.

These systems are in fact able to identify a reliable energy consumption baseline and to compare actual values of consumption with it, allowing energy managers to evaluate performances, identify malfunctions or energy efficiency opportunities and promptly undertake corrective actions. Moreover, after an efficiency project is concluded, a control system can be used to compare previous to current consumption, calculating the obtained energy saving.

1.5 Assumptions

The student has made some assumptions to increase the chances of the project to be relevant and objective centered. The assumptions made are

i. The building has a Wi-Fi access point where users can be able to connect to it and also connect to the internet.

ii. There are no pets that are moving within the building.

iii. The voltage from the power supply is constant and does not fluctuate.

1.6 Limitations

The limitations to the implementation of the project include;

i. The sensors will be placed in a small single room, which can be scaled to multiples room within a building.

ii. The room has one door, where one can enter or exit the room.

iii. Only current readings from the lamp and the fan are being recorded and saved.

iv. For the sake of practicality, control of appliances will only be limited to a single lamp and a single fan.

v. Dynamic IP addresses are being provided to the wireless devices, in this case, to the esp8266 Wi-Fi modules.

vi. Not many people are entering the room at a go.

vii. People are not standing at the door.

viii. No network analysis has been done regarding the communication between the devices, so an analysis of intelligent office is carried out to find what may be the flaws that can be seen in the system.

1.7 Scope

The scope of the project is to design a small model of an office building, where automation and energy consumption is measured.

2 CHAPTER TWO: LITERATURE REVIEW

2.1 Introduction

Here we will talk in depth about the various technologies used within the project

2.2 Hardware

2.2.1 ESP8266EX

ESP8266EX is an SoC that provides Wi-Fi capabilities and can be interfaced with external sensors and other devices through GPIO's. An ESP8266EX can perform either as a standalone application or as a slave to a host MCU [4].

The ESP8266EX comes with the Espressif Systems Smart Connectivity Platform (ESCP). This enables:

i. Energy efficiency by fast switching between sleep mode and wake up mode
ii. Adaptive radio biasing for low power operation
iii. Advanced signal processing

Some of the Wi-Fi key features include

i. 802.11 b/g/n support
ii. 802.11n support (2.4 GHz), up to 72.2 Mbps
iii. Defragmentation
iv. x virtual Wi-Fi interface
v. Automatic beacon monitoring (hardware TSF)
vi. Support Infrastructure BSS Station mode/Soft AP mode/Promiscuous mode
vii. Antenna diversity

The ESP8266EX can be used in a wide variety of applications including

i. Home automation
ii. Home appliances
iii. Baby Monitors
iv. IP cameras
v. Sensor Networks
vi. Security ID tags

vii. Wi-Fi position system beacons

viii. Industrial wireless control

ix. Wearable electronics

x. Wi-Fi location-aware devices

The ESP8266 comes in two variants, which will be discussed below.

2.2.1.1 ESP8266-01

The ESP8266-01 is a smaller Wi-Fi Module that allows microcontrollers to access a Wi-Fi network. It is a self-contained SoC meaning that it does not need a microcontroller to manipulate the inputs and outputs [5].

Figure 1 ESP8266-01 Wi-Fi module

It comes with a few GPIO pins which are programmed to do specific tasks as shown in Table 1 below

Table 1 Name and description of pins located in the ESP8266-01

NO	PIN NAME	FUNCTION
1	GND	GND
2	GPIO2	GPIO, Internal Pull-up
3	GPIO0	GPIO, Internal Pull up
4	RXD	UART0, data received pin RXD
5	VCC	3.3v power supply (VDD)
6	RST	1. External reset pin, active low

		2. Can loft or external MCU
7	CH_PD	Chip enable pin, Active High
8	TXD	UART0, data send pin RXD

Depending on the different states of the pin, the ESP8266-01 can boot into different modes as shown in Table 2 below

Table 2 Pinout and the different modes the ESP8266-01 can boot into

Mode	**GPIO15**	**GPIO0**	**GPIO2**
UART	Low	Low	High
Flash Boot	Low	Low	High

An image of the pins and their location on the chip is shown in Figure 2 below

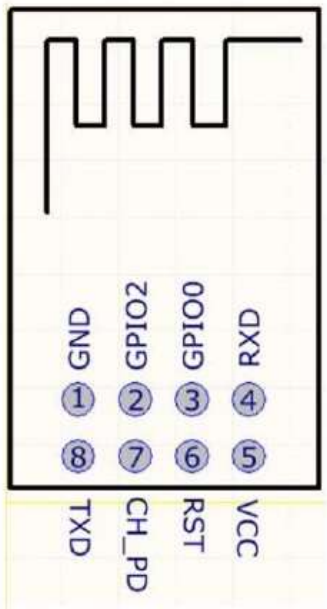


Figure 2 ESP8266-01 pinout adapted from [5]

2.2.1.2 NodeMCU

NodeMCU is an open source IoT platform, that includes firmware that uses the Lua scripting language and built based on hardware-based off the ESP-12 Wi-Fi module from Espressif Systems [6].

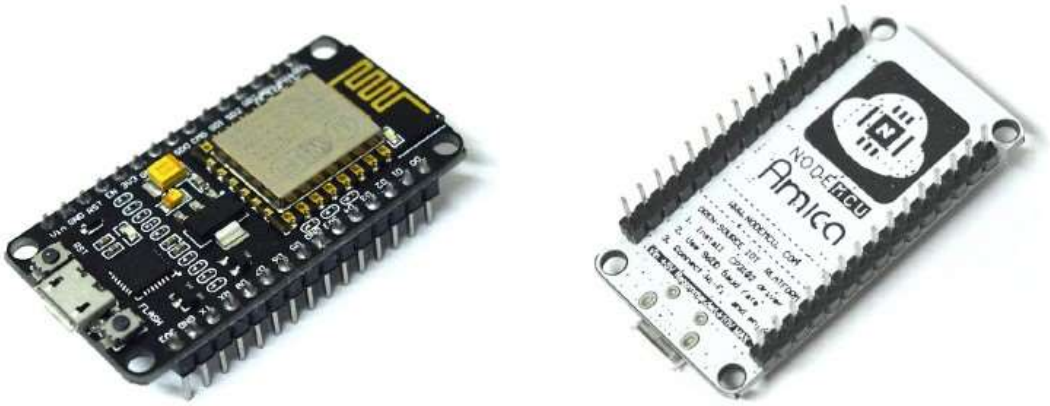


Figure 3 NodeMCU adapted from [6]

The NodeMCU comes with more pins as compared to the ESP8266-01. An image of the pins and their location on the chip are shown below.

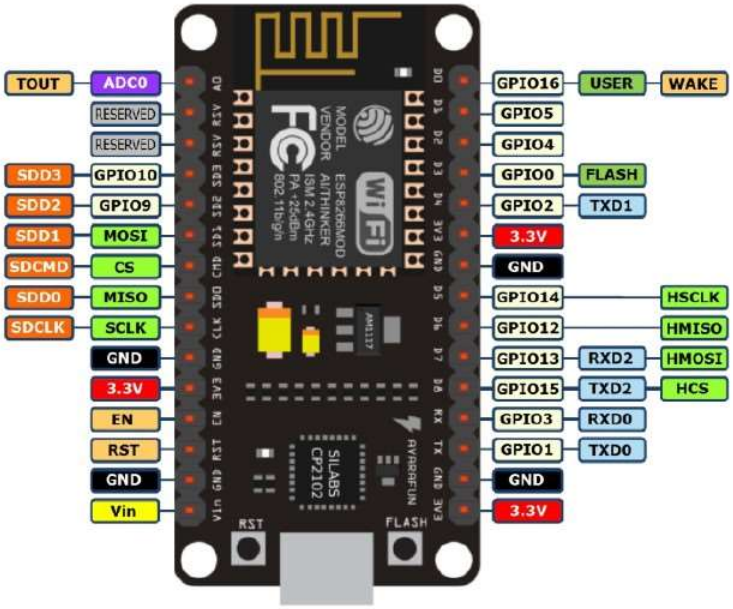


Figure 4 NodeMCU pinout adapted from [6]

2.2.2 Raspberry Pi

A RaspberryPi is a small computer developed by the Raspberry Pi foundation. Its purpose is to promote basic teaching of basic computer science, but the applications in which it is being used has been increasing [7].

Some of the features it has are:

i. A system on a chip, Broadcom BCM2836

ii. A GPU with Broadcom VideoCore

iii. CPU 900MHz Arm Cortex A7

iv. 1 GB memory

v. 4 USB ports

vi. MicroSD card for storage

Figure 5 Raspberry 2 Model B adapted from [7]

2.2.3 Sensors

Multiple sensors were used to gather data. This are:

2.2.3.1 PIR Motion Sensor

PIR motion sensors are sensors that allow one to detect movement within the sensor range. They consume less power, are small and inexpensive. They are made up of a pyroelectric sensor which detects levels of infrared radiation [8].

Figure 6 PIR motion sensor adapted from [8]

The PIR motion sensor normally has 3 pins

i. 3-5V DC Power: used to power the sensor, the sensor can work with 3.3V or 5V
ii. Ground: Connected to ground.

iii. Digital out: Givers the digital output of the sensor

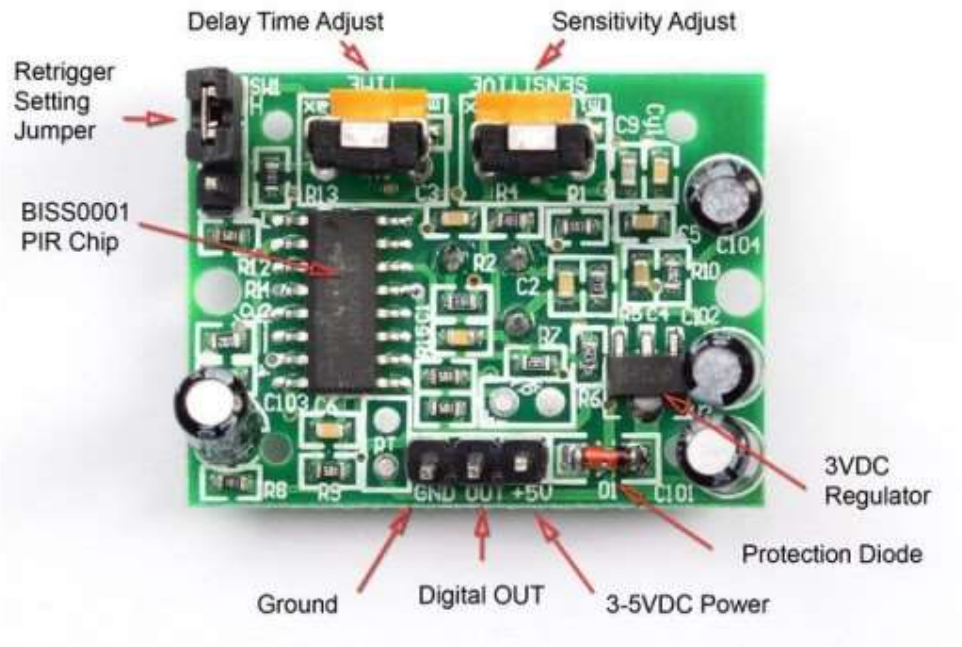


Figure 7 PIR motion sensor pinout adapted from [8]

A PIR motion sensor has two slots inside it. The materials are sensitive to IR. The sensors, in the presence of human IR radiation, the sensor detects the radiation and convert it to electrical pulses. The motion detection normally uses light to detect the presence of infrared light emitted by a warm object or absence of infrared light when an object interrupts a beam.

2.2.3.2 DHT11

A DHT11 is a Digital Humidity and Temperature sensor. They are made up of two components, a capacitive humidity sensor, and a thermistor. The data is then converted from an analog signal into a digital signal, digital signals which are very easy for a microcontroller to read [9].

Some of the features of the DHT11 sensor are

i. Low cost

ii. Can work with voltages of 3V up to 5V

iii. Has a maximum current of about 2.5mA

iv. It is good for measuring humidity of around 20% to 80% with a reading of around 5% accuracy.

v. It is good for measuring temperature of around 0- 50°C

vi. It has no more than 1 Hz as it's the sampling rate

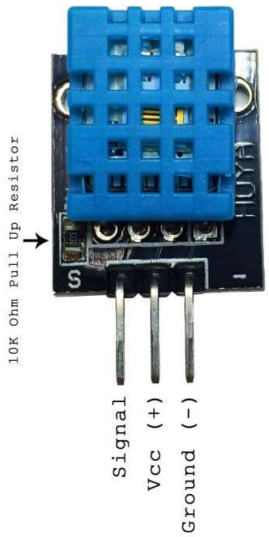


Figure 8 DHT11 pinout adapted from [10]

The DHT11 has 3 pins

i. Signal: This pin is connected to the digital pin of the microcontroller to get the values for temperature and humidity.

ii. Vcc: The sensor is powered by 3V up to 5V.

iii. Ground: The pin is connected to ground

It detects water vapor by measuring the resistance between two electrodes, the humidity sensing component being a moisture holding substrate with electrodes on the surface. The substrate absorbs water vapor, releasing ions and increasing the conductivity between the two electrodes. High humidity reduces the resistance between the two electrodes. Temperature is measured by a surface mounted NC temperature sensor (thermistor) built into the unit [10].

2.2.3.3 ACS712 Current Sensor

The ACS712 is a current sensor that provides an economical and industrial solution for AC or DC current sensing for both industrial and commercial use. It consists of a Hall effect Linear current sensor with a copper conduction path located along the surface of the die. When applied current flows through the copper path, it generates a magnetic field which is detected by the Hall effect IC itself and converted to a proportional voltage. The output voltage is then connected to an ADC pin of a microcontroller which converts the output voltage to a digital value which can be easily understood [11].

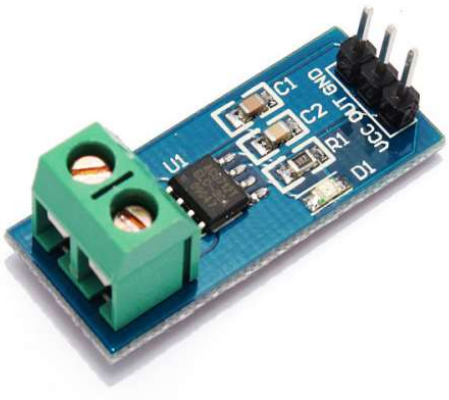


Figure 9 ACS712 5A Current Sensor

The ACS712 has 3 pins

i. Vcc: This pin is used to power the sensor, the sensor is powered by 5V

ii. Out: This is the output of the sensor. The sensor outputs a voltage between 0v and 5v which represents the current that is being measured by the sensor

iii. Gnd: The ground pin is connected to the ground pin of the power supply of the sensor.

2.2.4 Relay

The relay used is the songle relay. Some of the main features of the relay are [12]:

i. Switching capability for 10A

ii. Simple magnetic circuit to meet the low cost of mass production

Some of its applications include

i. Domestic appliances

ii. Office machines

iii. Audio equipment

iv. Automobiles etc.

Figure 10 Songle relay

2.2.5 Voltage Regulator

The LM1117 is a linear voltage regulator. It provides fixed output voltages of 1.8, 2.5, 3.3 and 5 volts. A 10 µF capacitor is added to the input and the output of the voltage regulator to reduce the noise [13].

Figure 11 LM1117 Voltage Regulator

The LM1117 has 3 pins as shown in Figure 12.

i. ADJ/GND: Adjust pin for adjusting the output voltage. Ground pin for a fixed output voltage.
ii. Vin: Input pin for the Voltage Regulator.
iii. Vout: Output pin for the Voltage Regulator.

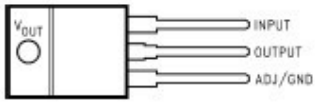


Figure 12 LM1117 Voltage Regulator pinout adapted from [13]

2.3 Software

2.3.1 MQTT Protocol

Wireless networks have been in the increase as more and more devices are being connected to the internet. Data is being transferred from these devices and sensors to other devices, machines or sensors within the network. The network is also very dynamic, with devices changing their network address at any time, network links failing and also the devices and sensors themselves failing. The conventional method of using network addresses can be problematic because of their dynamic and temporary nature [14].

This problem is solved by using a data-centric approach. In this approach, data is delivered to the devices within the network not based on their network addresses, but based their interests. A system that uses this is the publish/subscribe messaging system that MQTT uses. It decouples the communication devices from each other, so as to make it easier to add devices acting as data sources or devices consuming the data.

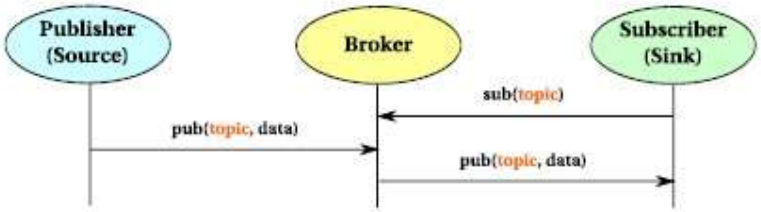


Figure 13 MQTT publish subscribe model, adapted from [14]

In a publish/subscribe model, devices which are interested in consuming certain data register their interest, a process is known as a subscription. Devices which want to get certain information register their interest, they are known as subscribers and the process is known as a subscription. Components which want to produce certain information are known as publishers and do so by publishing their information. A broker is a device that acts as an intermediary between the publisher and subscriber and is responsible for ensuring that data gets from the publishers to the subscribers.

MQTT is such a protocol that uses a topic-based publish/subscribe. MQTT was designed with the following in mind

i. Optimization for tiny devices with low battery, limited processing power, and storage.

ii. To be reliable across network constraints such as high link failure rates, low bandwidth, and devices submitting data with a short message payload.

iii. To run on any network so long as the network supports point to point data transfer.

2.3.2 Mosquitto

Mosquitto is an open source software that provides standards compliant to server and client implementation of the MQTT messaging protocol. The model used in Mosquitto is the publish and subscribe model, which has the advantages of having low network overhead and can be implemented on low power devices such as microcontrollers that might be used in remote Internet of Things sensors [15].

The software consists of 2 parts

i. The main mosquitto server

ii. The mosquitto_pub and mosquitto_sub client utilities that are a method of communication with an MQTT server

2.3.3 Heroku

Heroku is described as a PaaS, a cloud computing environment that helps developers to write, run and manage the lifecycles of the application. They not only responsible for provisioning and managing the lower infrastructure resources, but also for providing a fully managed application development and deployment platform [16].

It acts as a middleman between the application developer and AWS. It tries to abstract the complexity of the DevOps from the developer. It does this by offering a PaaS software that gives the user a smooth web interface and CLI to easily deploy their applications.

2.3.4 Micropython

It is a lean and efficient implementation of the Python3 programming language that includes a small subset of the python standard library. It is optimized to run on microcontrollers and in constrained environments [17].

Micropython has advanced features such as

i. Interactive prompts

ii. Precision integers which are arbitrary

iii. Closures

iv. List comprehension generators

2.3.5 MERN Stack

Database servers are computers that are dedicated to a large data store or a computer that has a database software installed so that it can store data while running websites or applications alongside it. An application server is a computer that runs developer applications. A web server stores and delivers content for a website [16] [18] .

MERN Stack involves 4 technologies which are listed below

i. MongoDB: MongoDB is document-based database. It is free and open source. It uses JSON and schemas. MongoDB stores data in a document-oriented, file structure. The database consists of multiple collections. Data in the collections are stored in the form of JSON objects.

ii. Express Js: This is a Node.js web application that is minimal and flexible. It provides a wide range of web application features. It makes it easy to create APIs that websites and databases can interact with.

iii. React: React.js is a front-end JavaScript library for creating user interfaces. It is mostly used for client-side operations. It is an open source JavaScript Library maintained by Facebook.

iv. Node Js: It is asynchronous and event-driven, designed to make scalable web applications and APIs.

Some of the advantages of MERN stack include

i. Use of Javascript everywhere

ii. Use of JSON everywhere

iii. Node Js is a fast and resilient web server

3 CHAPTER THREE: SYSTEM DESIGN AND IMPLEMENTATION

3.1 Introduction

The Building Management System was set up in a room. The room setup includes a fan and a bulb. It has one door where people can come in and out of the room. Sensors for gathering data on current, power, temperature, humidity and motion sensors are also set up within the room. The Building Management System also includes components that reside in the cloud, such as the web server and the online cloud database.

The building management system consists of two components, the hardware design, and the software design, both of which will be discussed extensively below.

3.2 Hardware

The hardware setup for the room is shown below.

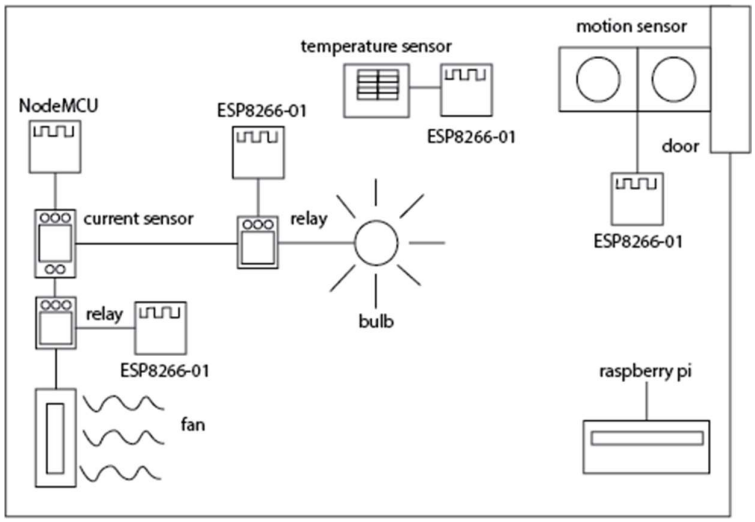


Figure 14 A room with the hardware setup

Two motion sensors are set up close to the door, as shown in the figure above. Depending on the order in which the PIR motion sensors are activated, one can know whether someone is moving into a room or out of a room. The motion sensors are connected to an ESP8266-01 to capture and transmit the data over a Wi-Fi connection.

The fan and the bulb are connected to relays, that enable one to turn on and off the fans and the relays. Each of the relays is connected to an ESP8266-01, that receive commands to turn on or off the relays and the fans and relay back to the Raspberry Pi.

The fan and the bulb are wired back to a current sensor that detects the current being consumed. The current sensor is connected to a NodeMCU which gathers the current sensor data and relays it the Raspberry Pi. The set up is shown in Figure 15 below.

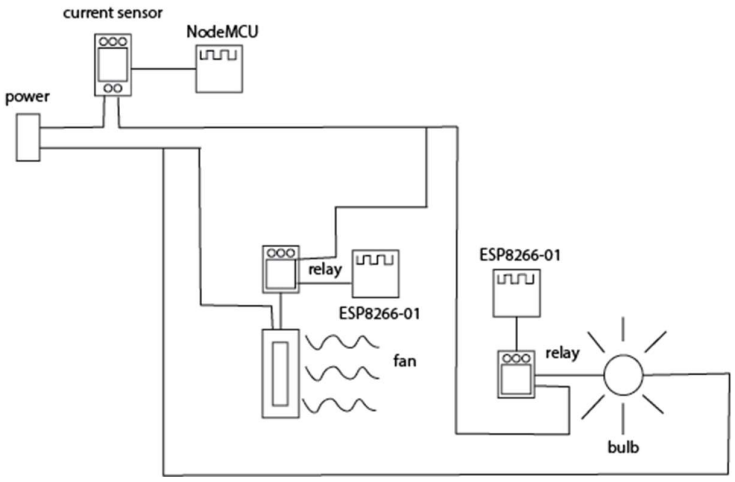


Figure 15 Wiring set up for the current sensor to measure current from both the fan and the bulb

The Raspberry Pi which is connected to the Wi-Fi gathers all the data from the Soc.

In the next pages, we will dive deeply into the construction of the circuits.

3.2.1 Flashing circuit

One of the difficulties of using the ESP8266-01 is flashing firmware into the chip itself. Improperly flashing code into the ESP8266-01 may damage the chip itself and making any data that is stored in it to become corrupt. Voltages which are higher than 3.3v may damage the chip [19].

To address this issue, a flashing circuit with the following design is used

i. Capacitors being used to provide noise reduction on the power bus.

ii. Pull up resistors added to ensure that the pins GPIO2 and GPIO0 pins on the ESP8266-01 are not allowed to float.

The flashing circuit was set up as shown in Figure 16 below

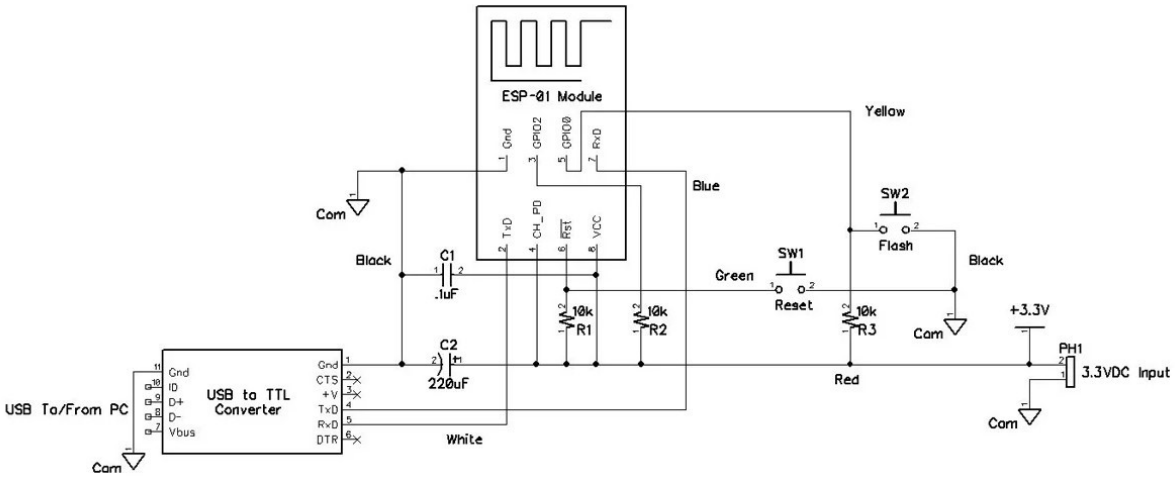


Figure 16 ESP8266-01 Flashing Circuit, adapted from [19]

The reset button resets the ESP8266-01 by setting GPIO0 to ground. The flash button enables the ESP8266-01 to boot into flash mode, enabling the user to enter flash board where he or she can flash firmware into the SoC. The circuit components were then soldered on to a stripboard as shown in *Figure 57*

3.2.2 Power supply

The ESP8266-01 operates with a voltage of 3.3V. Components such as the DHT11 and the PIR Motion Sensor also operate within this voltage. To get a power supply of 3.3V, a linear voltage regulator, the LM1117 was used [13].

Using an LM1117 requires a 10 μF capacitor at the input of the voltage regulator and at the output of the voltage regulator. The input to the voltage regulator is 5V which is then stepped down to a voltage of 3.3V. The schematic diagram of the power supply is shown in the figure below.

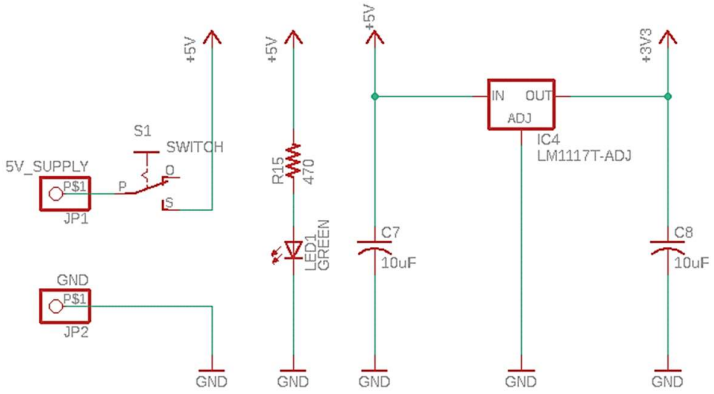


Figure 17 Schematic diagram of the power supply connection

Initially, the input to the voltage regulator was connected to a 9V battery, but using a 9V battery resulted in the voltage regulator heating up most of the time, and would require a heat sink to be added to the voltage regulator. A better alternative which was to use a USB cable, since most USB cables are capable of supplying 5V. The top part of the USB cable was stripped away to leave out the two wires, 5V, and ground, to be used to power circuits.
A switch and a led were also added to the circuit to enable the user to switch on or off, the circuit, and to indicate whether the circuit is on or off.

3.2.3 Pull up resistors

A pull up resistor is a resistor used to ensure a known state for a signal. They are very common when using microcontrollers or any other digital logic device. When a system with a SoC is configured as an input and the program reads the state of the pin, the state of the pin alternates between high and low. This state is known as a floating state. This can affect the normal operation of the SoC, in this case the ESP8266-01 [20].

If the GPIO pins of the ESP8266-01 are in the low state, the SoC goes into the flashing mode, which allows the user to erase the firmware on the SoC or update firmware in the SoC. In this state, the SoC cannot perform the normal operations such as connecting to Wi-Fi, subscribing to messages, publishing messages among other things.

To prevent this, a pull up resistor is connected to the GPIO pins. An example of this connection is shown below in Figure 18.

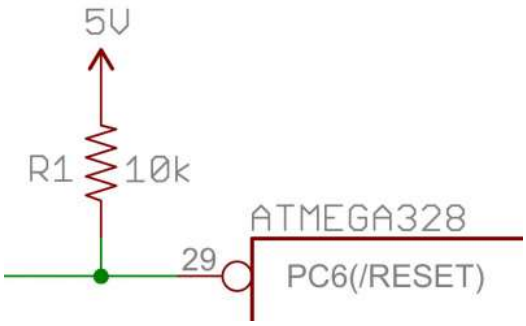


Figure 18 An example of a pull up resistor connected to an atmega328, adapted from [20]

In our case, the GPIO pins of the ESP8266-01 are connected to a resistor of 10kΩ and the end connected to 3.3V, to pull the state of the pins of the GPIO pins to high.
A resistor in the order of 10kΩ is used. This high resistor value is known as a weak pull up resistor, and allows for less current to flow through it. The amount of current flowing is calculated below

$$I = \frac{v}{R} = \frac{3.3v}{10{,}000\Omega} = 0.00033A$$

3.2.4 ESP8266-01 with a DHT11 sensor

The ESP8266-01 was connected to a DHT11 sensor. The output pin of the DHT11 pin is connected to the GPIO2 pin of the SoC. This allows the SoC to be able to gather temperature and humidity data. Both the DHT11 and the SoC are powered by 3.3V, which is supplied from the output of the voltage regulator.

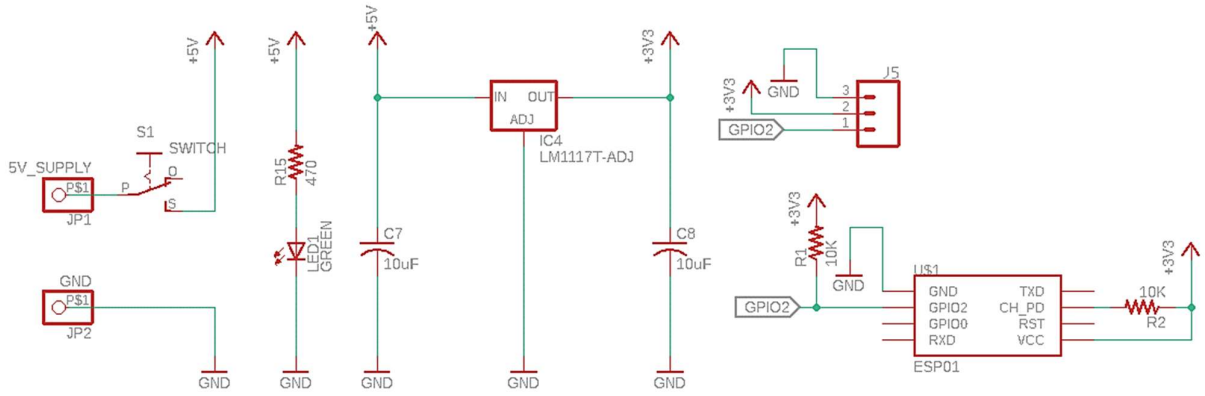


Figure 19 Schematic diagram of the ESP8266-01 connected to a DHT sensor and a power supply

The circuit was first set up on a breadboard and tested to see whether it works as expected before moving to a PCB design. The breadboard design is as shown in Figure 45.

A pull up resistor is connected to the GPIO2 pin and the CH_PD pin of the SoC. This is to avoid floating pins, a condition where the SoC registers the pin as a high then as a low and alternates between the two states. This behavior may affect the proper functioning of the SoC. The resistor value used in this case is a 10kΩ resistor, one end connected to 3.3V and the other end connected to the respective pin of the SoC.

Once the circuit was working, the PCB design was constructed as shown in Figure 50 and was etched on a presensitized PCB. Components were then added and soldered on to the PCB, with the final design as shown in Figure 55.

3.2.5 ESP8266-01 with a pair of PIR motion sensors

The PIR motion sensors were connected to the GPIO pins of the ESP8266-01 SoC. The output of the PIR motion sensors is connected to GPIO2 pin and the GPIO0 pin of the SoC. Pull up resistors of 10kΩ are added to GPIO_2, CH_PD and GPIO_0 pin of the SoC to avoid floating pins.

The circuit was first set up on a breadboard and tested to see whether it is working, as shown in Figure 46 before moving to a PCB design as shown in Figure 51. The components were then soldered on to a presensitized PCB as shown in Figure 54

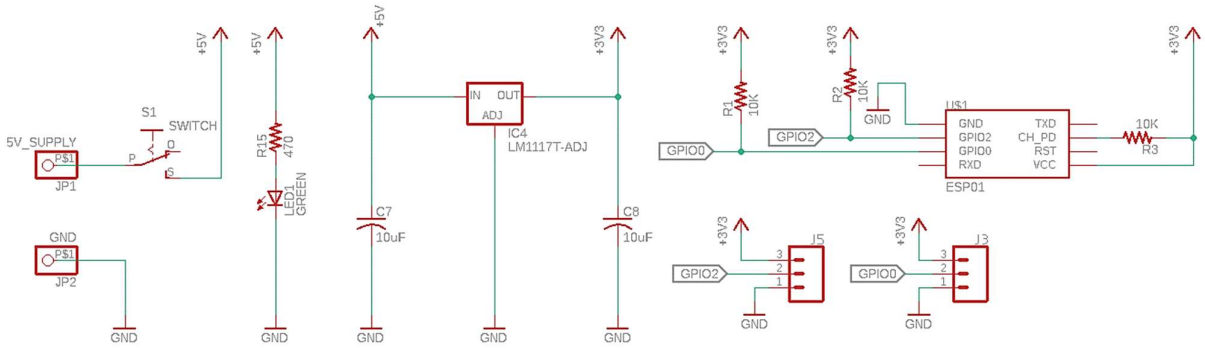


Figure 20 Schematic diagram of an ESP8266-01 connected to a pair of PIR motion sensors

3.2.6 ESP8266-01 with a relay

The ESP8266-01 was connected to a relay. The output of the relay is connected to the collector of the transistor, while the emitter is connected to the ground. The base of the transistor is connected to a 4.7kΩ resistor which is then connected to GPIO2 pin of the SoC. This allows the SoC to be able to turn on an off appliances, in this case, the fan and the bulb. The calculations on how 4.7kΩ was obtained.

$$3.3v - 0.7v = 2.6v$$

$$v = IR \quad R = \frac{v}{I}$$

$$R = \frac{2.6V}{0.0006A} = 4{,}333.33\Omega$$

$$\cong 47k\Omega$$

The SoC is powered by 3.3V, which is supplied from the output of the voltage regulator. The relay is powered by 5V.

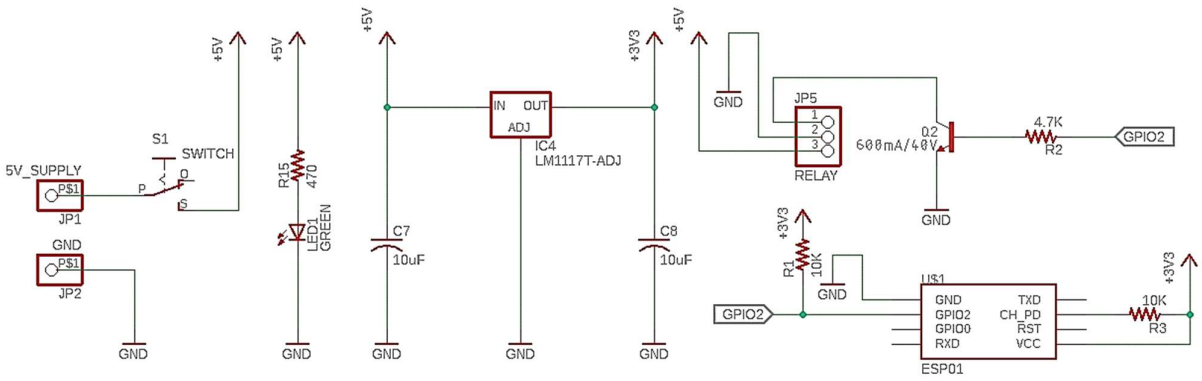


Figure 21 Schematic diagram of an ESP8266-01 connected to a relay

Pull up resistors of 10kΩ are added to GPIO_2 and CH_PD of the SoC to avoid floating pins.

The circuit was first set up on a breadboard and tested to see whether it is working, as shown in Figure 47 before moving to a PCB design as shown in Figure 52. The components were then soldered on to a presensitized PCB as shown in Figure 56.

3.2.7 NodeMCU with ACS712 current sensor

The NodeMCU was connected to an ACS712 current sensor. The output pin of the current sensor pin is connected to the GPIO2 pin of the SoC. This allows the SoC to be able to gather data on current consumption. Both the NodeMCU and the current sensor are powered by 5V.

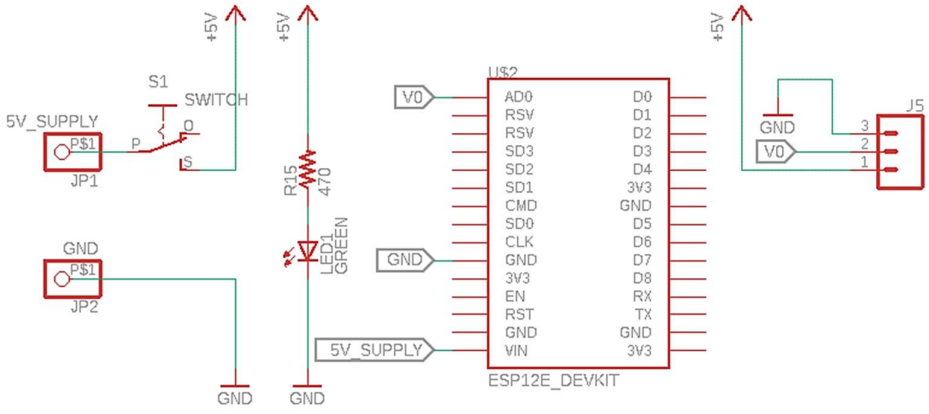


Figure 22 Schematic design of a NodeMCU connected to a current sensor

The circuit was first set up on a breadboard and tested to see whether it is working, as shown in Figure 48 before moving to a PCB design as shown in Figure 49. The components were then soldered on to a presensitized PCB as shown in Figure 53.

3.3 Software

3.3.1 System Requirements

System Requirements Specification is a document that describes the features and behavior of a system or software application. The System Requirements for the Building Management System are shown below.

Table 3 System requirement specification

REQUIREMENTS
1. A user should be able to see and analyze the energy demand and energy consumption patterns in an online dashboard. 2. Energy demand and energy consumption data to be stored in an online database for analysis to get insight into energy usage for the purpose of energy efficiency and cost reduction 3. Raspberry Pi with associated sensors for gathering data.

3.3.2 Class Diagram

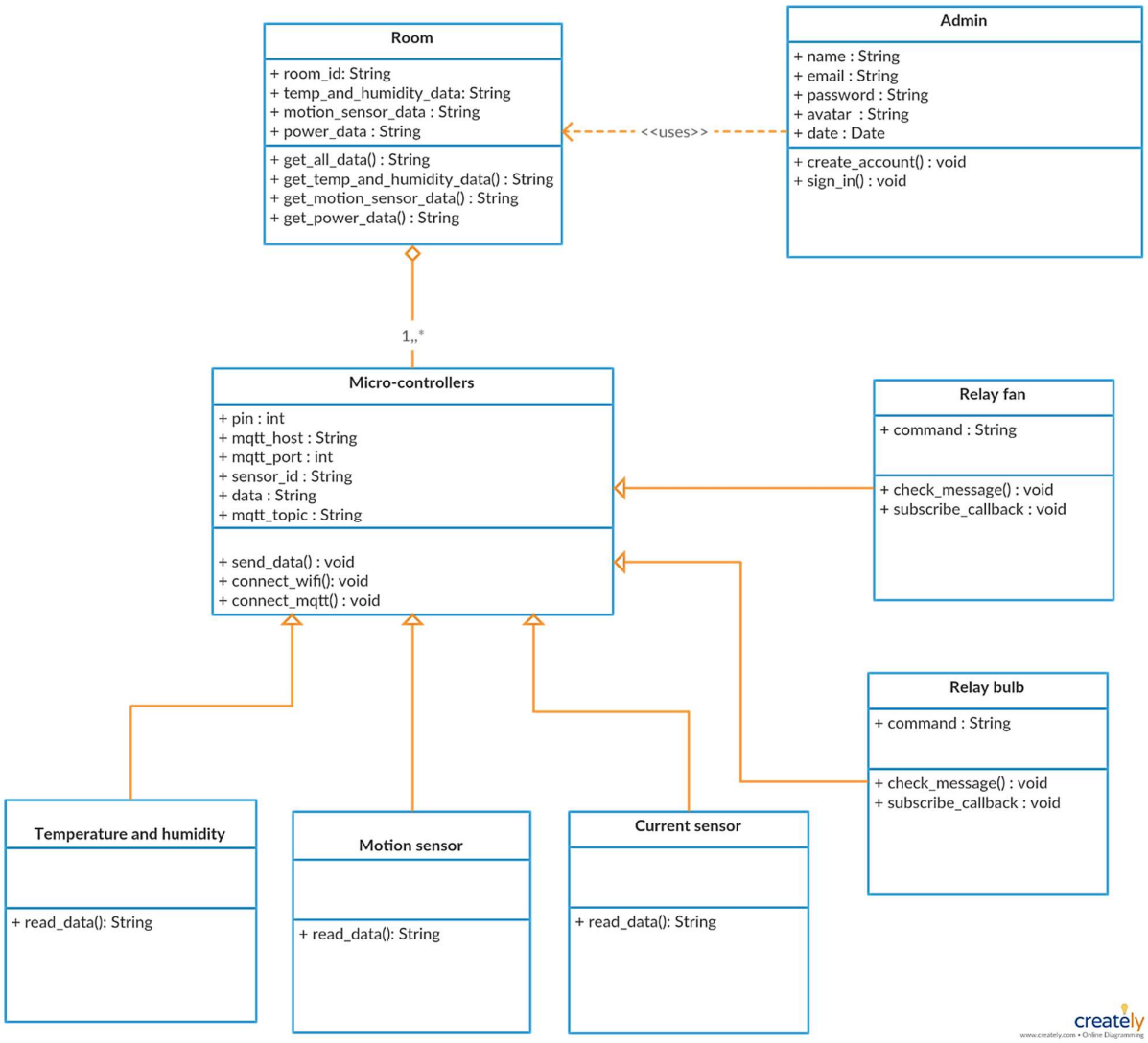


Figure 23 Class diagram of the Building Management System

A class diagram describes the structure of a system by showing the system's classes, their attributes, operations, and relationships.

We have 8 classes

i. Room

ii. Admin

iii. Microcontrollers

iv. Temperature and humidity

v. Motion sensor

vi. Current sensor

vii. Relay-fan

viii. Relay-bulb

Most of the hardware components share most of the class functions with the microcontroller class only with slight differences with the functions that run and also the variables used.

A single room can have multiple microcontrollers within it. The building can also have multiple rooms which can be accessed by the admin.

3.3.3 Use Case Diagram

Figure 24 Use case diagram of the Building Management System

Use case diagrams attempt to model the functionality of a system using actors and use cases. Use cases are described as functions that need to be performed by the system. In this case, the Building

Owner can be able to sign in, check room occupancy, check the temperature and humidity of the room, check which devices are active, check power consumption within the room, create a new user and create a new room. Microcontrollers send all the data to a database and data is fetched from the database to be displayed to the website for the building owner to see. Other background tasks are done such as checking sign-in credentials, display login errors, save room details to the database and save the sign-in credentials to the database.

3.3.4 Database Diagram

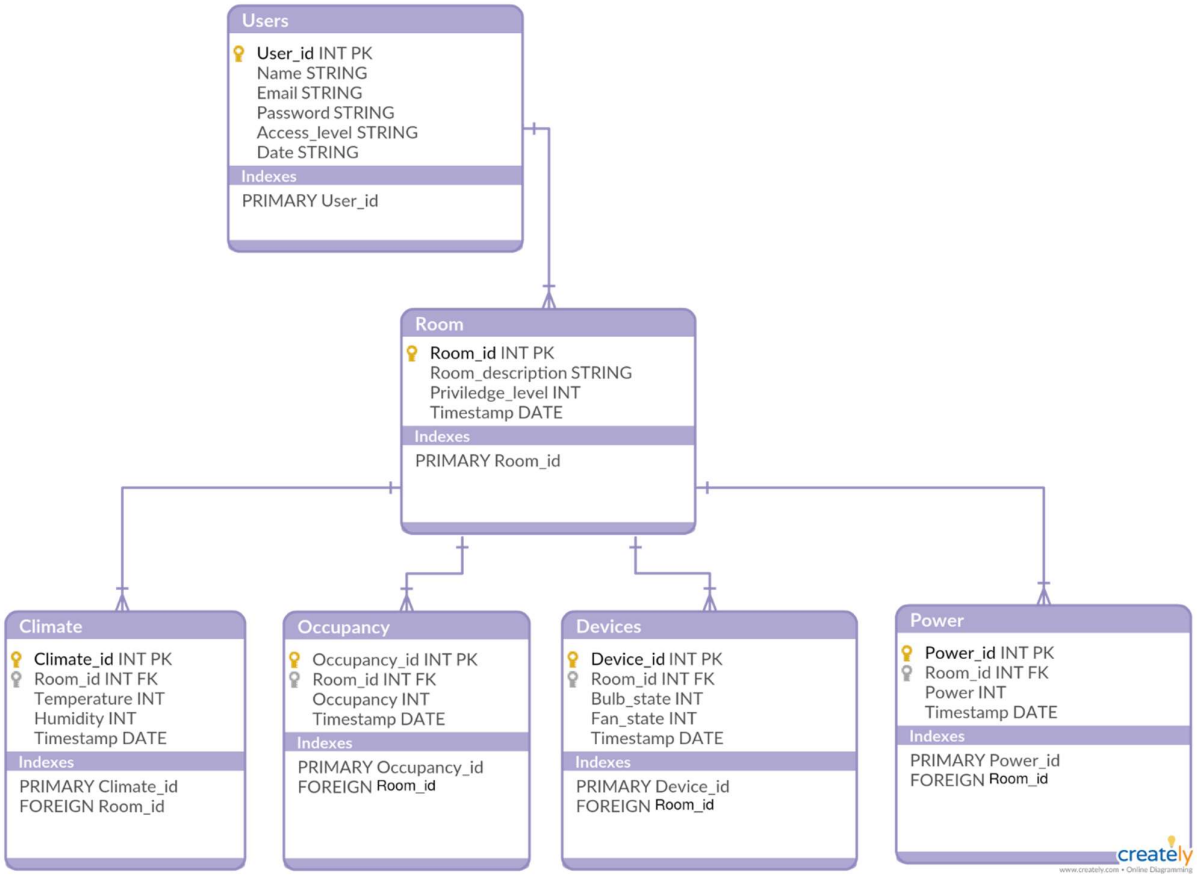


Figure 25 Database diagram of the Building Management System

The figure above shows the database structure for the Building Management System. The database has 6 records, the users, room, climate, occupancy, devices, and power. A room can have multiple collections for data on climate, occupancy, devices, and power. Users can also access multiple rooms. The foreign key for data that has been sent to the database from the sensors is the room id, meaning that data sent can be tracked to a specific room.

3.3.5 Sequence Diagram

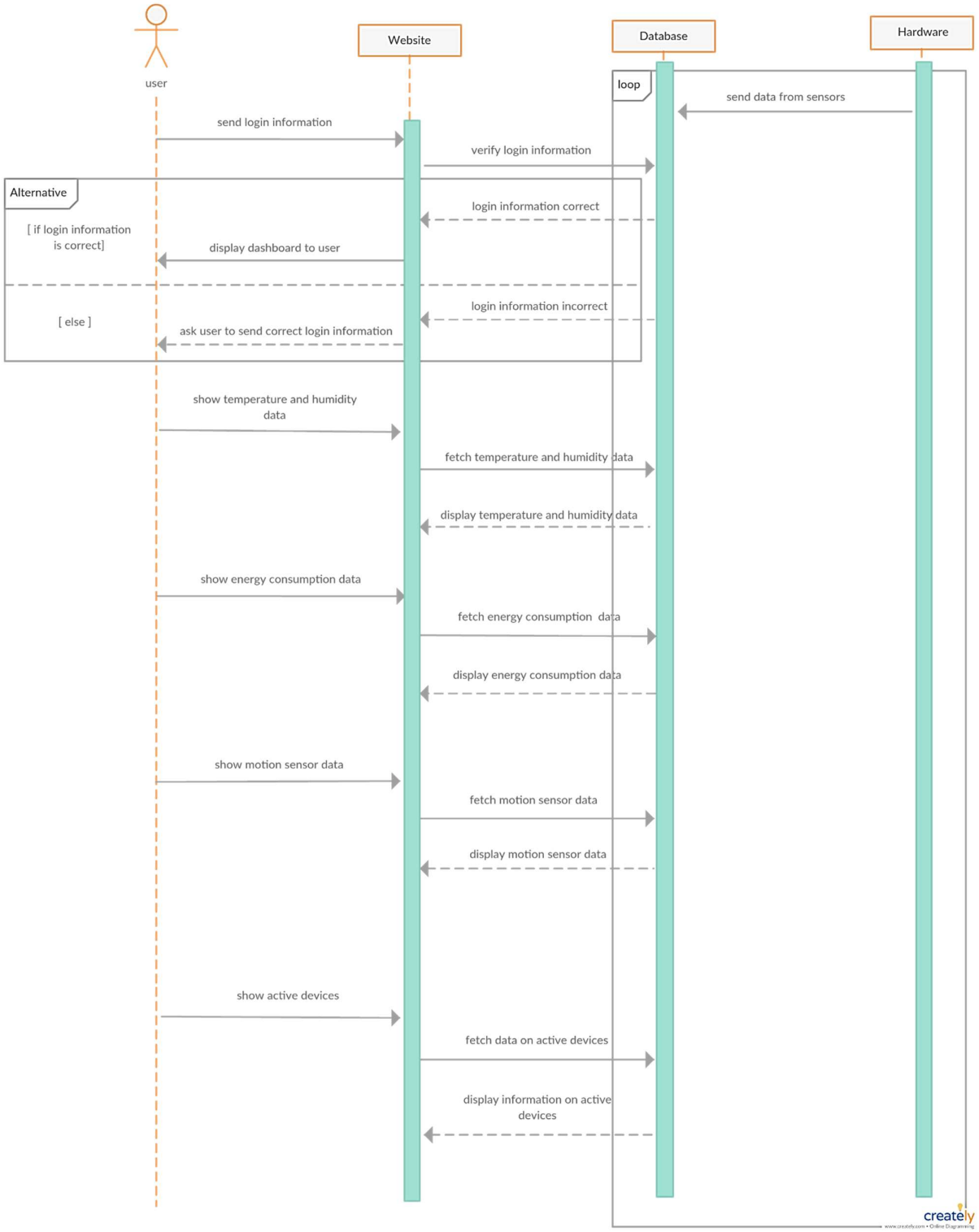


Figure 26 Sequence diagram of the Building Management System

A sequence diagram depicts the interaction between objects within the system. The system begins with the hardware sending the data on temperature, humidity, building occupancy, the current

consumed, the power consumed and the state of the devices. The user then logs in and the credentials are sent to the website then to the database. If the login credentials are good, the user is presented with the dashboard where the user can see all the data about the system in a summarized form. After that, the user can now request for specific data such as climate data, power data, room occupancy and the state of the devices.

3.3.6 Wireframes

A wireframe is a low fidelity way of showing a design. Some of the things a wireframe does are:

i. Showing the main chunks of content

ii. Showing the most basic UI

iii. Showing the outline and the layout structure

3.3.6.1 Sign In

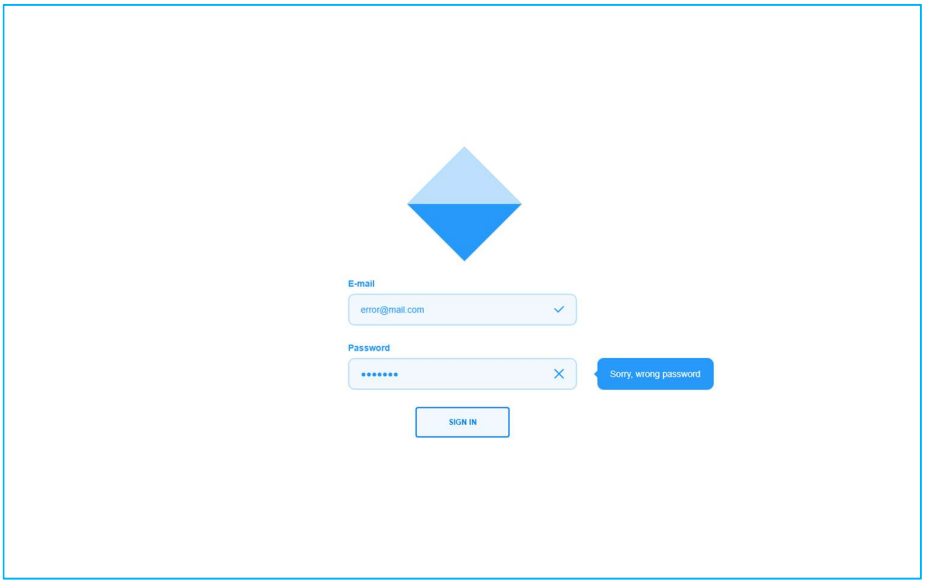


Figure 27 Sign in wireframe for the Building Management System Dashboard

The figure above shows the wireframe for the sign in page for the Building Management System website. The user needs to enter the email address and the password and then will be redirected to the dashboard.

3.3.6.2 Sign Up

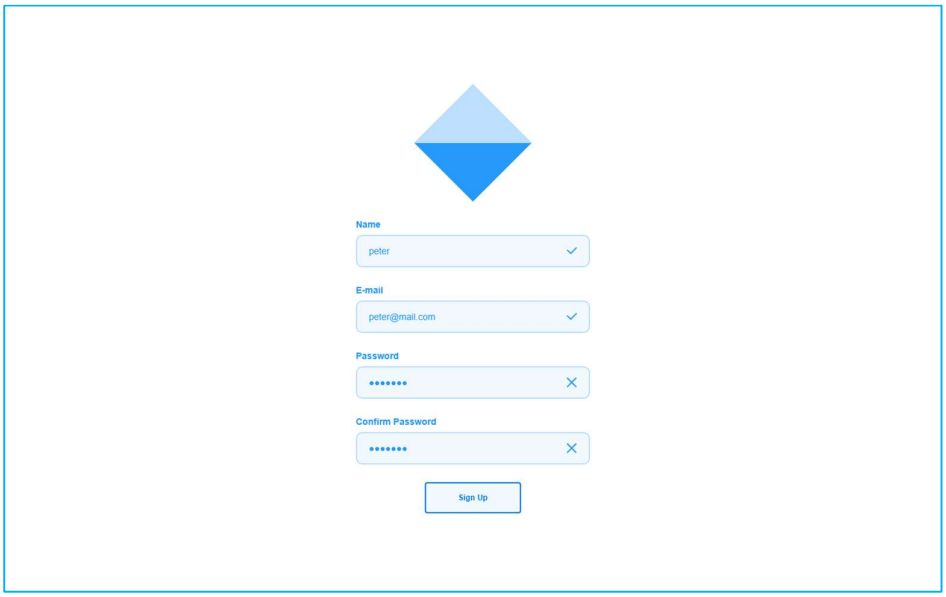


Figure 28 Sign up wireframe for the Building Management System Dashboard

For the user to access the dashboard, the user has to sign up to the system. This involves entering the name, email address, and a suitable password. The password is entered twice to ensure that the user has entered the correct password.

3.3.6.3 Dashboard

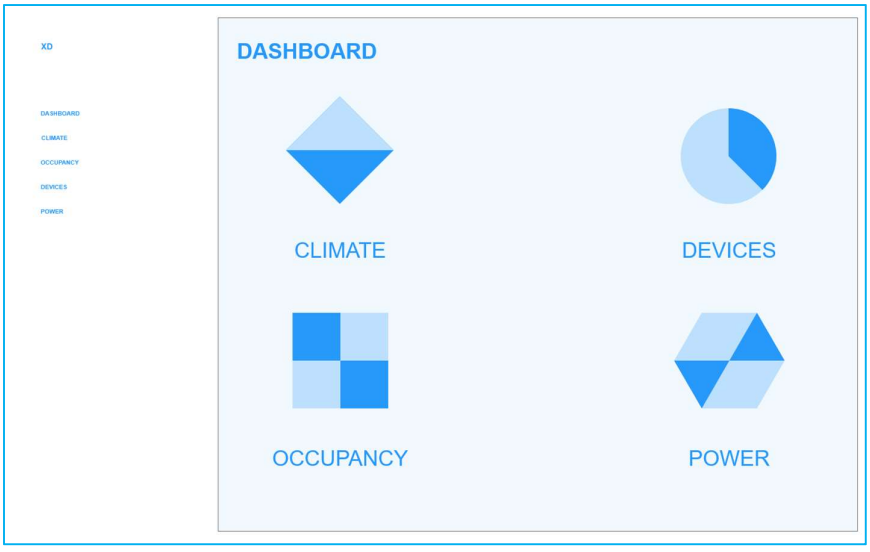


Figure 29 Dashboard wireframe for the Building Management System Dashboard

Once the user has signed up, he or she is redirected to the dashboard. Here the user can see the latest data on climate, devices, room occupancy, and power.

3.3.6.4 Climate

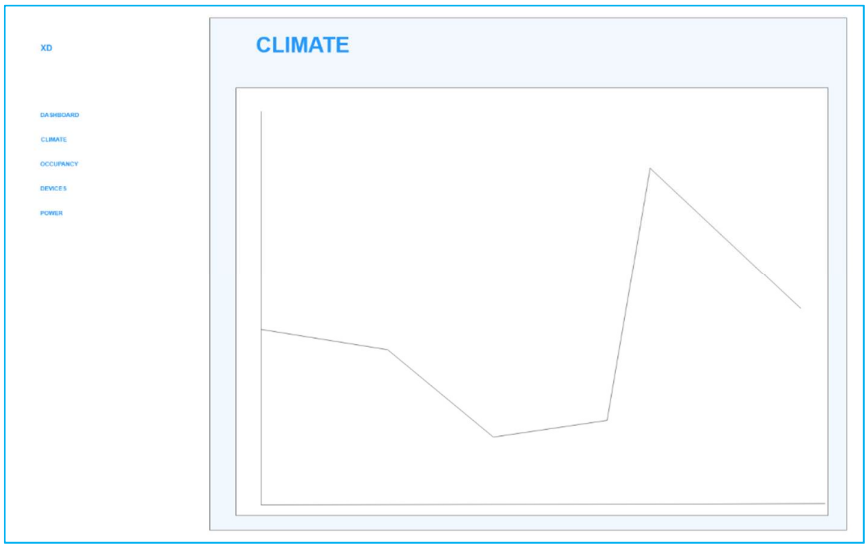


Figure 30 Climate wireframe for the Building Management System Dashboard

When the user clicks on the climate tab, there are redirected to the climate page. Here a plot of temperature and humidity are plotted against time. The user can then see the temperature and humidity of the room at a given time by just hovering on top of the graph.

3.3.6.5 Occupancy

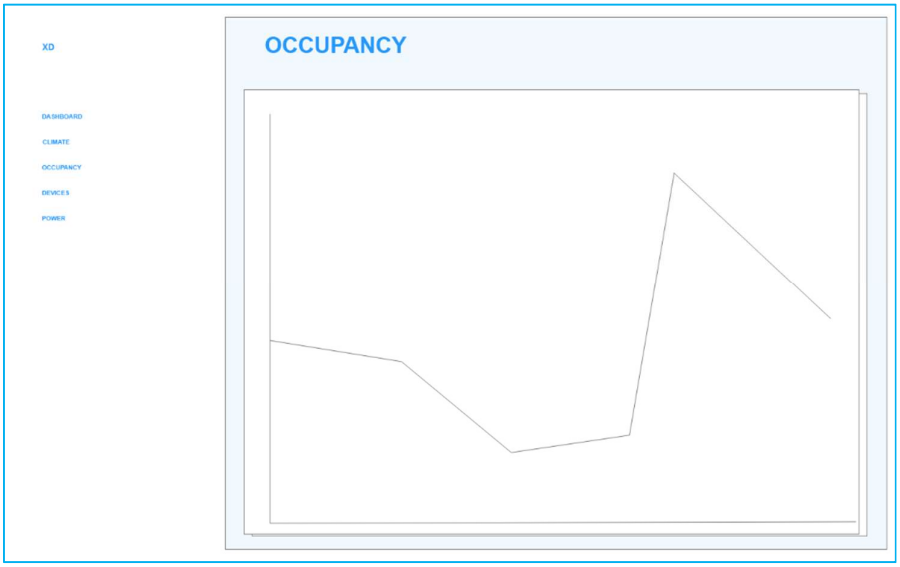


Figure 31 Occupancy wireframe for the Building Management System Dashboard

Clicking on occupancy takes the user to the occupancy page, where the user can see the number of users who have entered a room over a period of time, in the form of a line plot.

3.3.6.6 Power

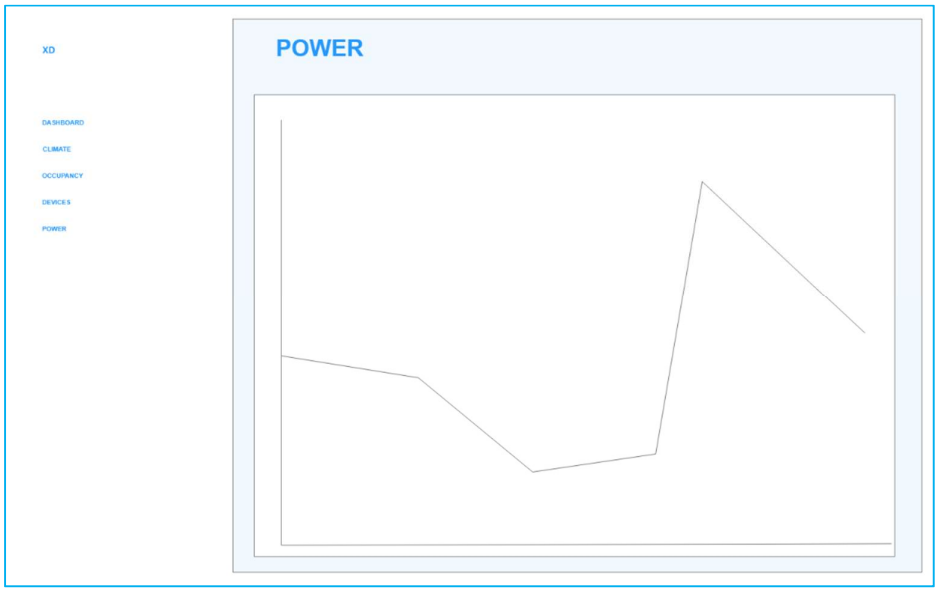


Figure 32 Power wireframe for the Building Management System Dashboard

When the user clicks on power, they are taken to the power page where a plot of current and power measured in watts is plotted against time.

3.3.6.7 Devices

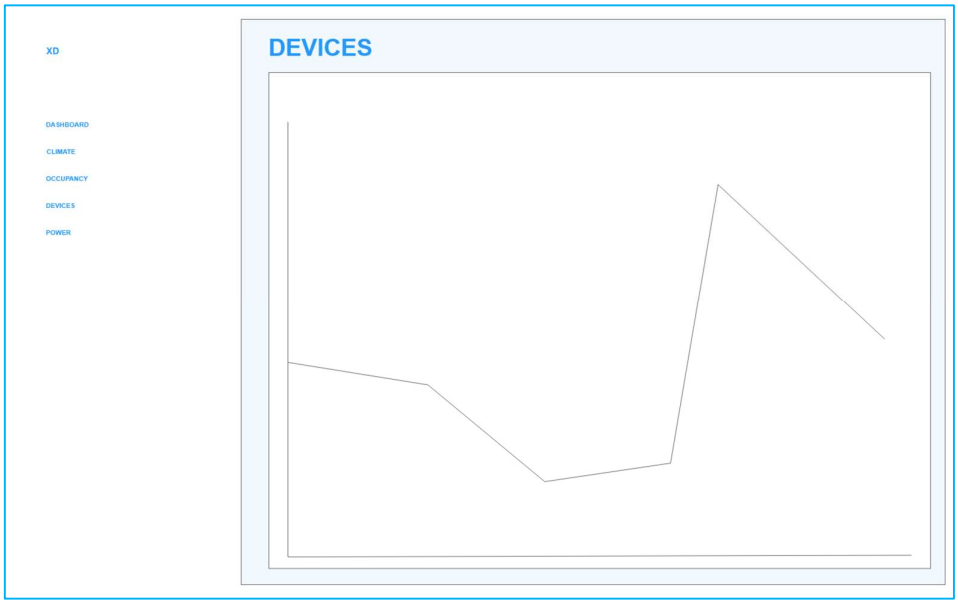


Figure 33 Devices wireframe for the Building Management System Dashboard

Clicking on devices takes the user to the devices page where the user can see whether the devices are on or off, represented as 1 for on and 0 for on, in the form of a line graph.

3.3.7 Flowchart

A flowchart is a graphical representation of a system or computer algorithm. The section below shows the flowcharts for the hardware components.

3.3.7.1 ESP8266-01 with a DHT11 sensor

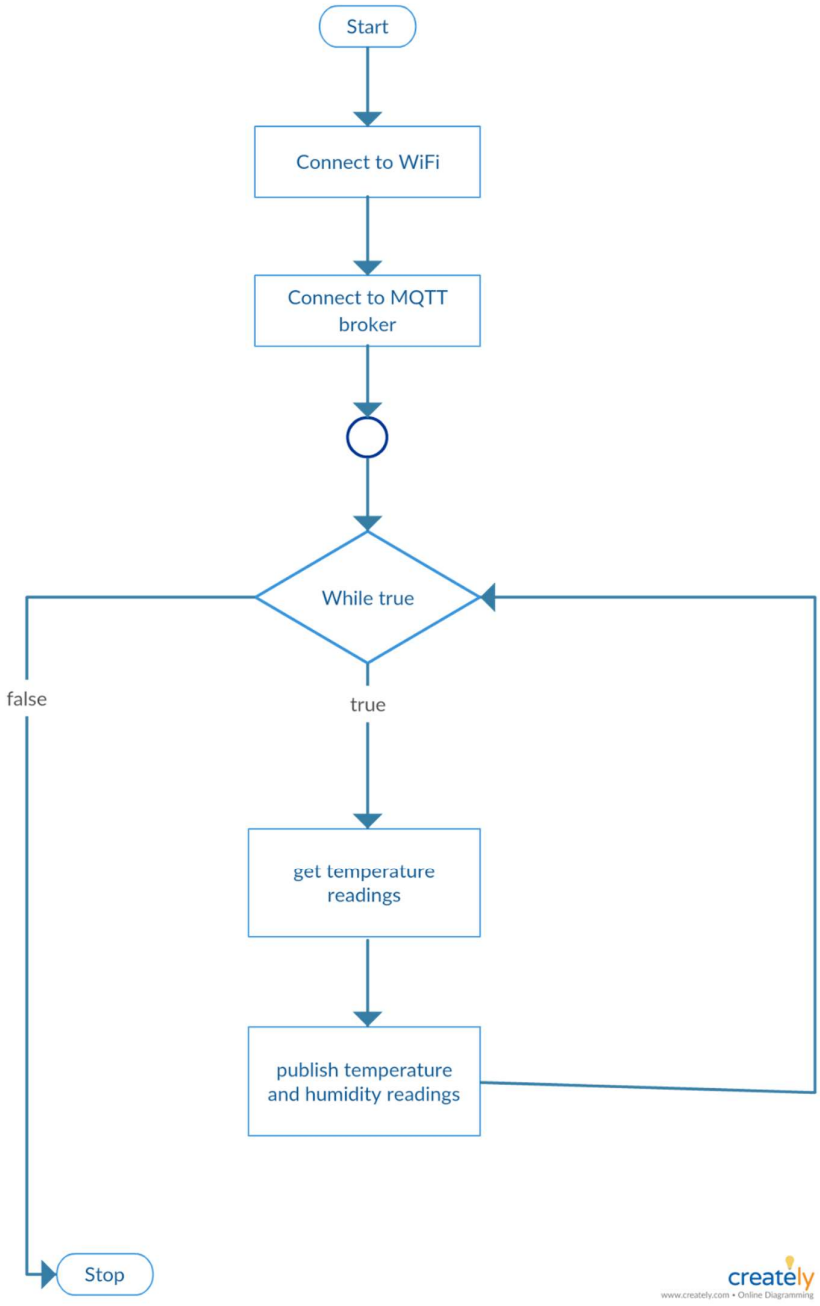


Figure 34 Flowchart for ESP8266-10 connected to a DHT11 sensor

3.3.7.2 ESP8266-01 with a pair of PIR motion sensors

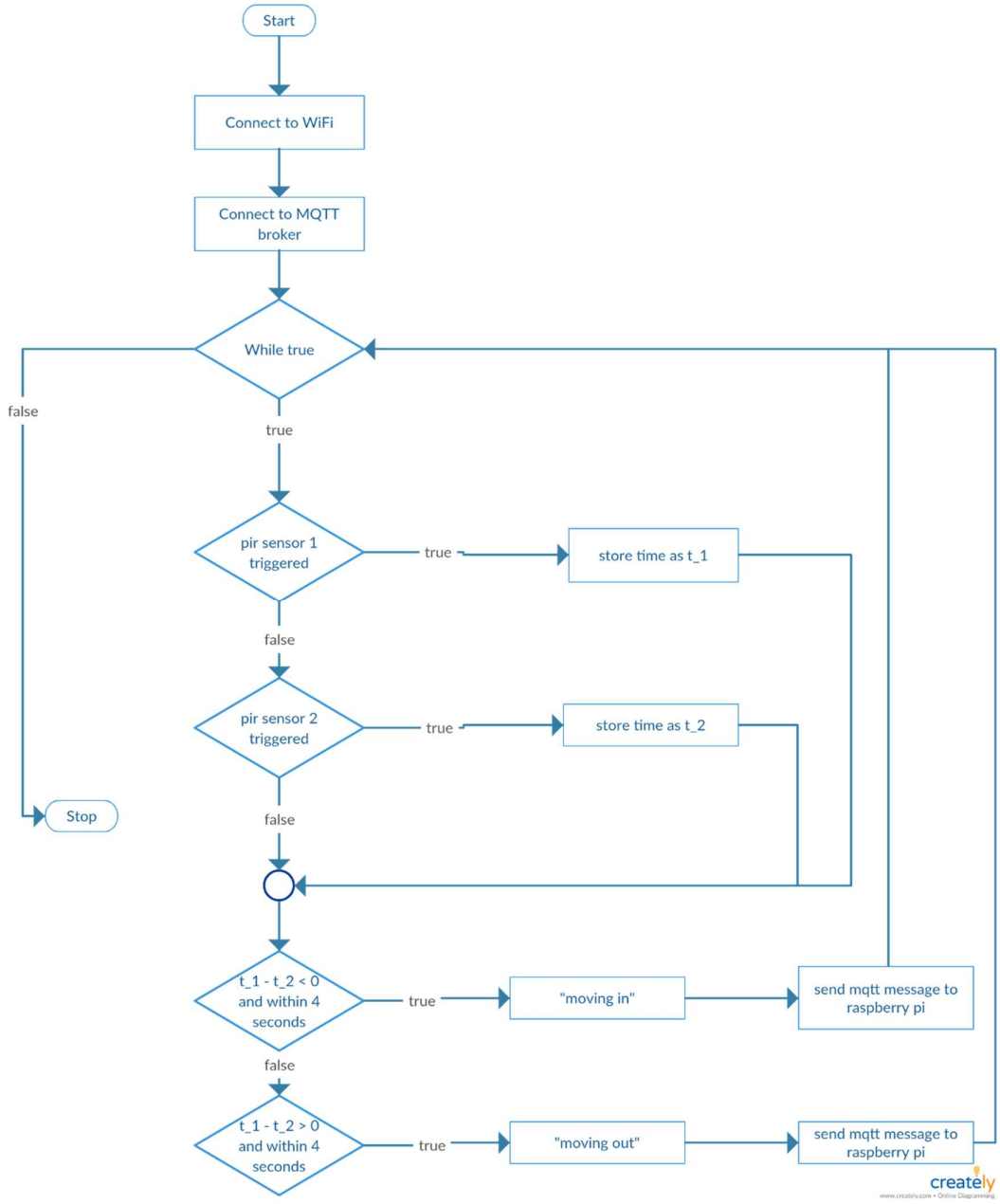


Figure 35 Flowchart for an ESP8266-01 for capturing motion using a motion sensor and publishing it online

3.3.7.3 ESP8266-01 with relay

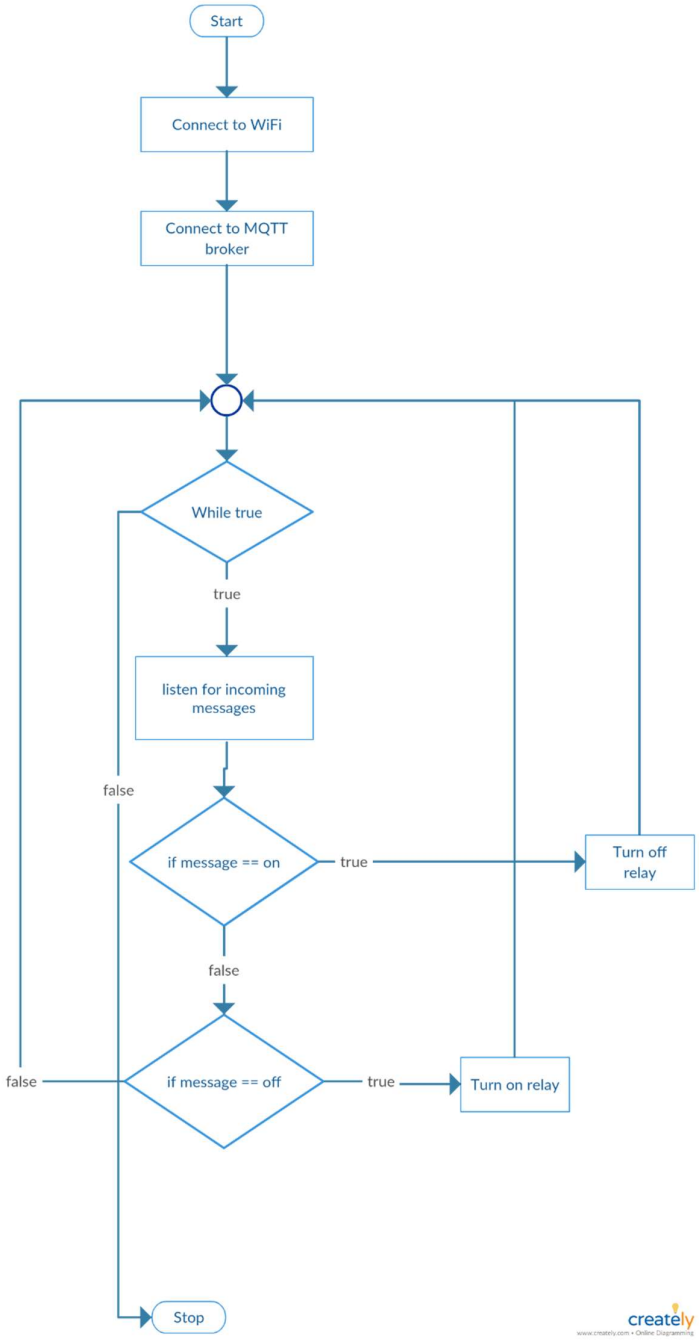


Figure 36 Flowchart for ESP8266-01 connected to a relay

3.3.7.4 NodeMCU with Current sensor

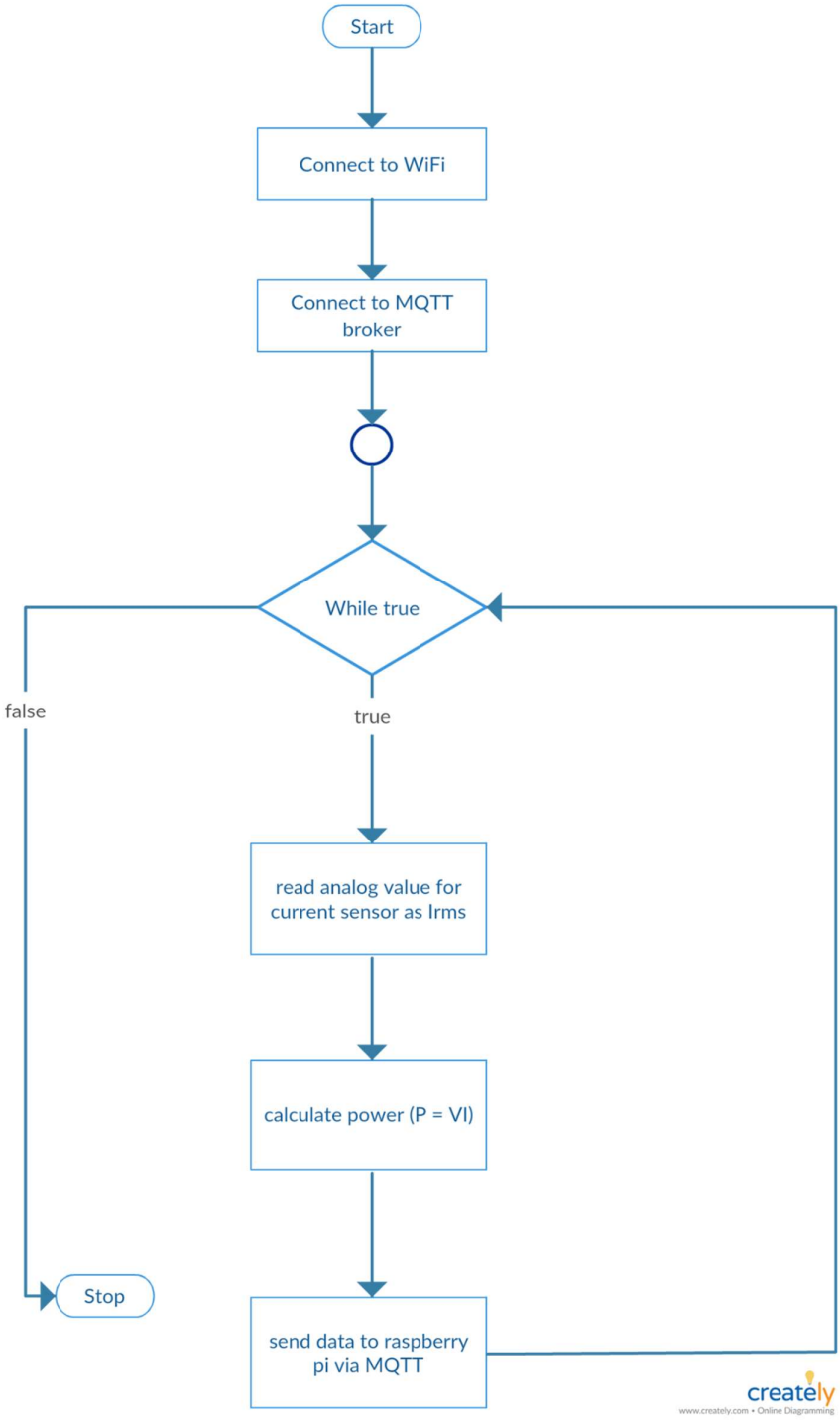


Figure 37 Flowchart for the NodeMCU to fetch and display power and current consumption

3.3.7.5 Raspberry Pi

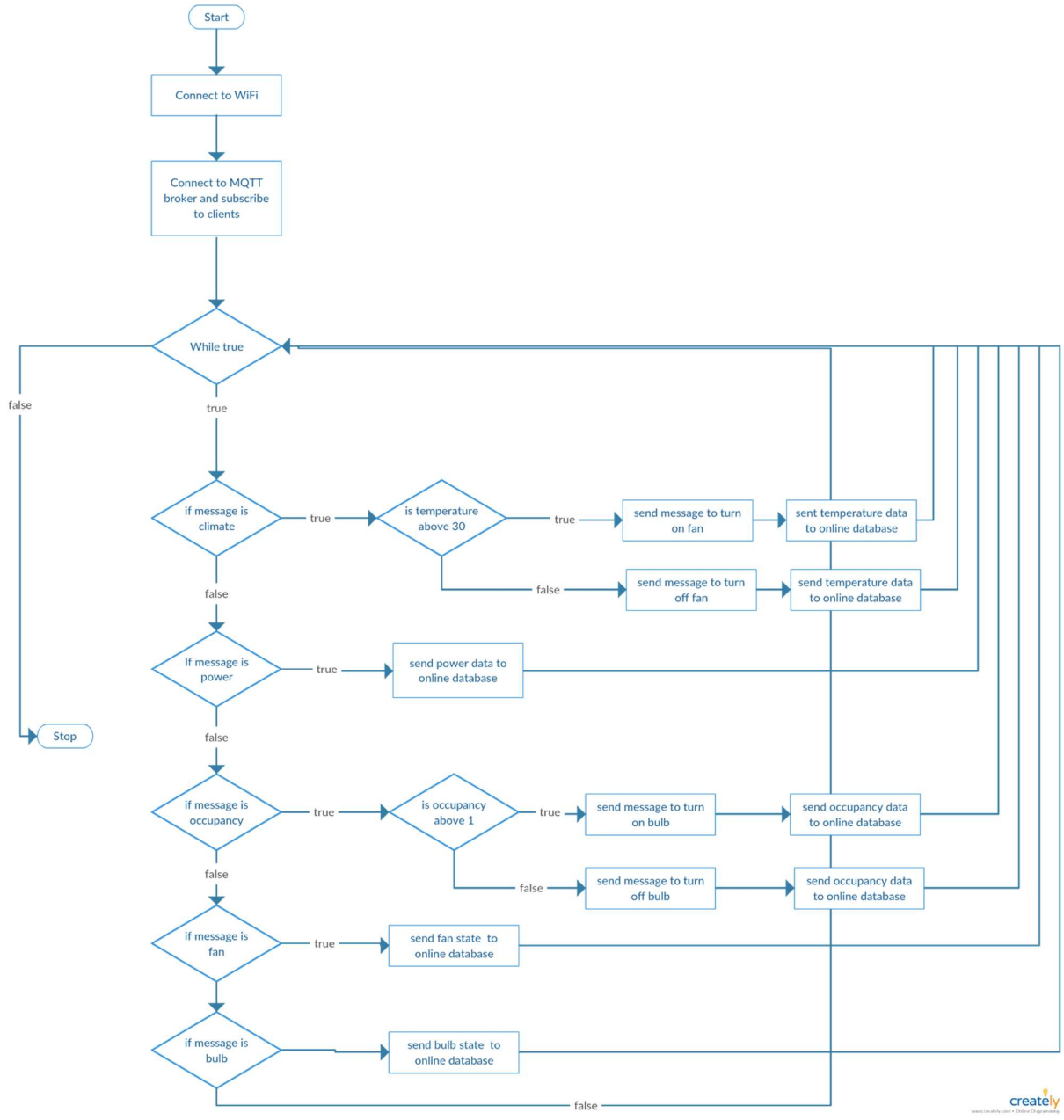


Figure 38 Flowchart for Raspberry Pi for saving data to the database and automating the fan and bulb

4 CHAPTER FOUR: RESULTS ANALYSIS AND DISCUSSION

4.1 Sign in and Sign up

For the user to sign up as shown in Figure 39 below, the user needs to enter the name, email address, password confirmation password. Once the user is able to sign up, he/she is taken to the sign in page as shown in Figure 40 where they enter the credentials they signed up with. From there the user is taken to the dashboard page, as shown in Figure 41.

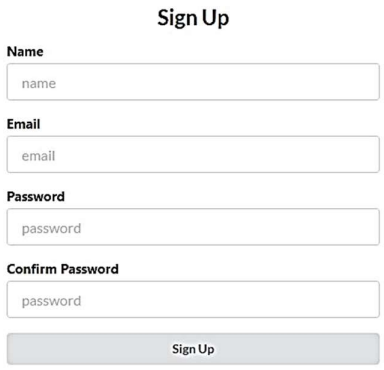


Figure 39 Website sign up page

Figure 40 Website sign in page

4.2 Dashboard

From the dashboard, as shown in Figure 41 the user has access to current data, such as the current temperature (temperature in degrees Celsius) and humidity(measured in terms of relative humidity), devices state (1 indicates the device is on and 0 indicates that the device is off), room occupancy, power (measured in watts) and current (measured in amperes).

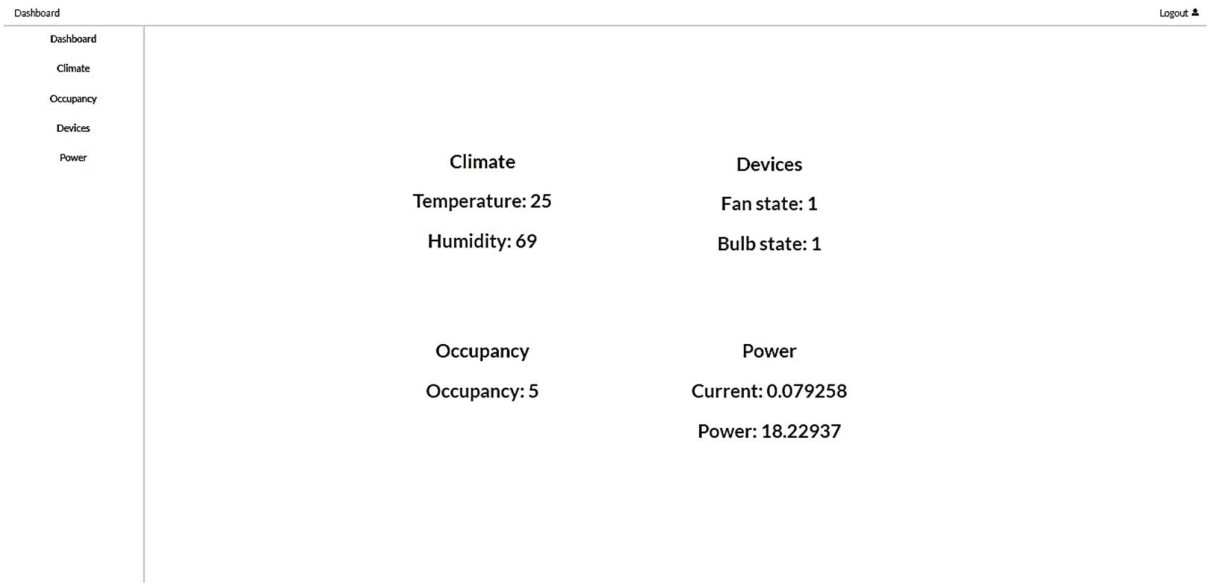


Figure 41 Website dashboard showing current data on climate, devices, occupancy, and power consumption

4.3 Time series results for climate data

Figure 42 shows a plot of humidity and temperature over a period of 6 hours. Here we can see some correlation between temperature and humidity. When a flame was put close to the DHT11 sensor, we see a rise in temperature levels and a decrease in humidity level before it went back to the normal average temperature.

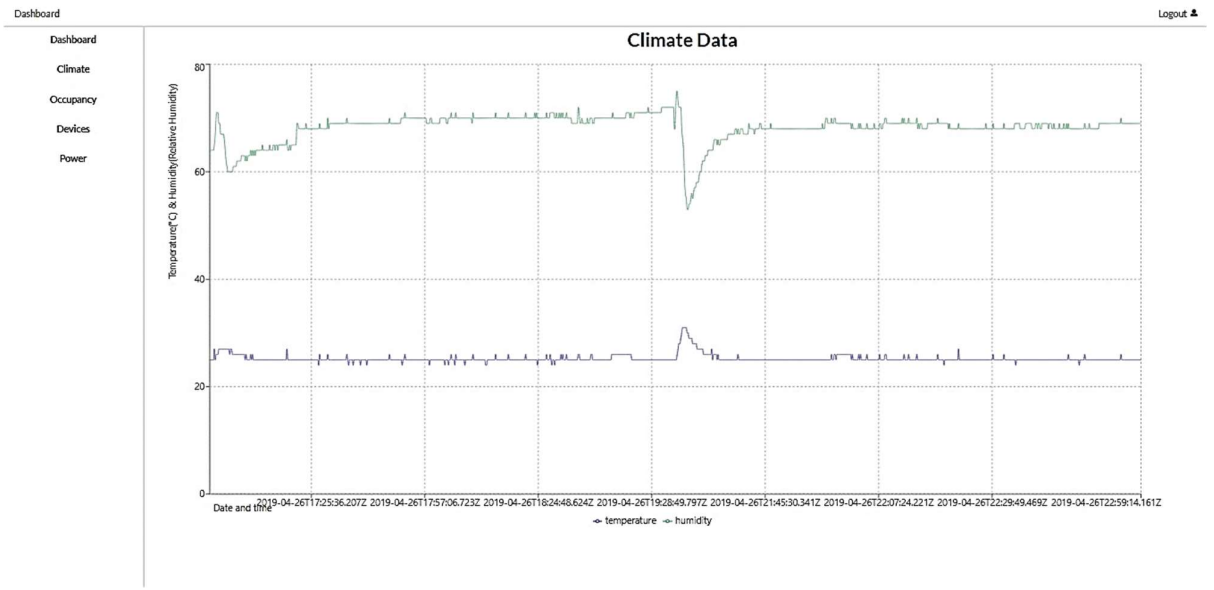


Figure 42 Website showing a line plot of climate data

4.4 Time series data for the devices

Figure 43 shows a plot of the state of the devices. 1 indicates that the device is on and 0 indicates that the device is off. The state of the bulb and the fan was observed over a period of time and plotted on a line graph.

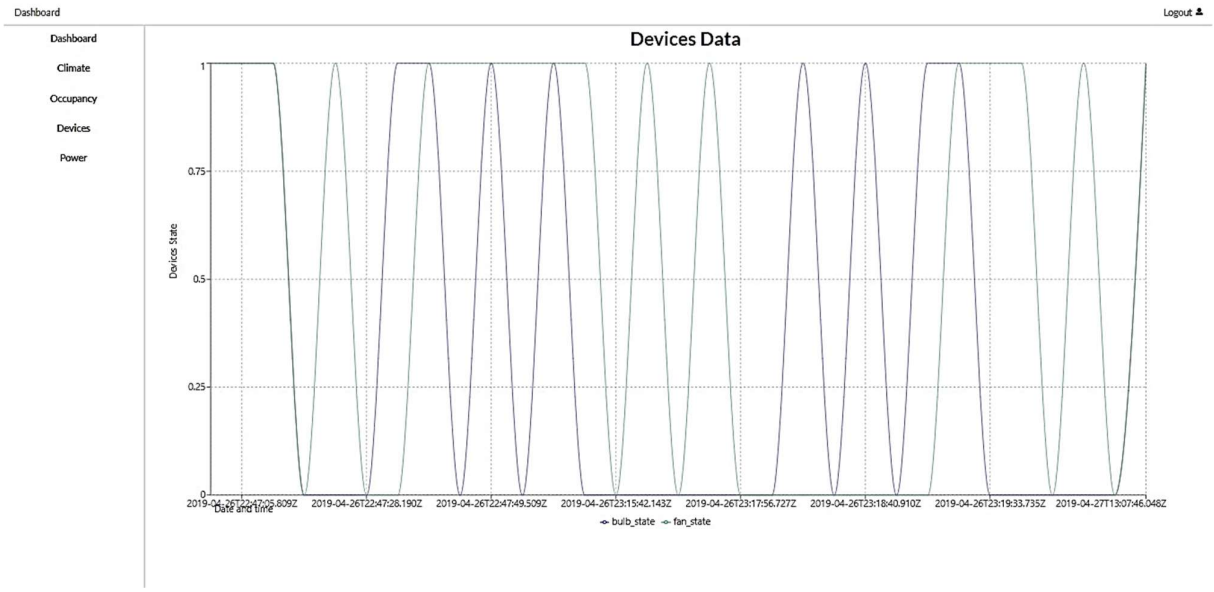


Figure 43 Website showing a line plot of the devices state

4.5 Time series data for power consumption

Power and current consumption were captured and displayed on the website as shown in Figure 44. The values for current and power fluctuate rapidly since the current consumption that is being

measured is sinusoidal in nature, and there are points that the current is captured at the low point of the sine wave and there are points where the current value is captured at the top point of the sinusoidal wave. When the fan is turned on, the current and power consumption rises and maintains a constant position. When the fan is turned off, the current and power consumption drops down and retains that position.

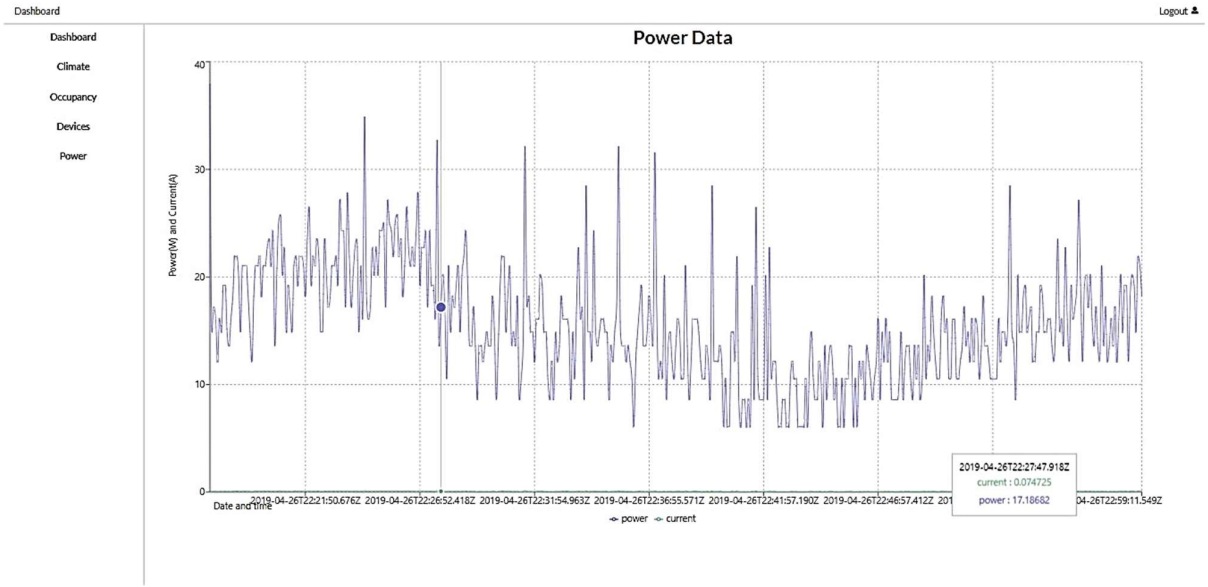
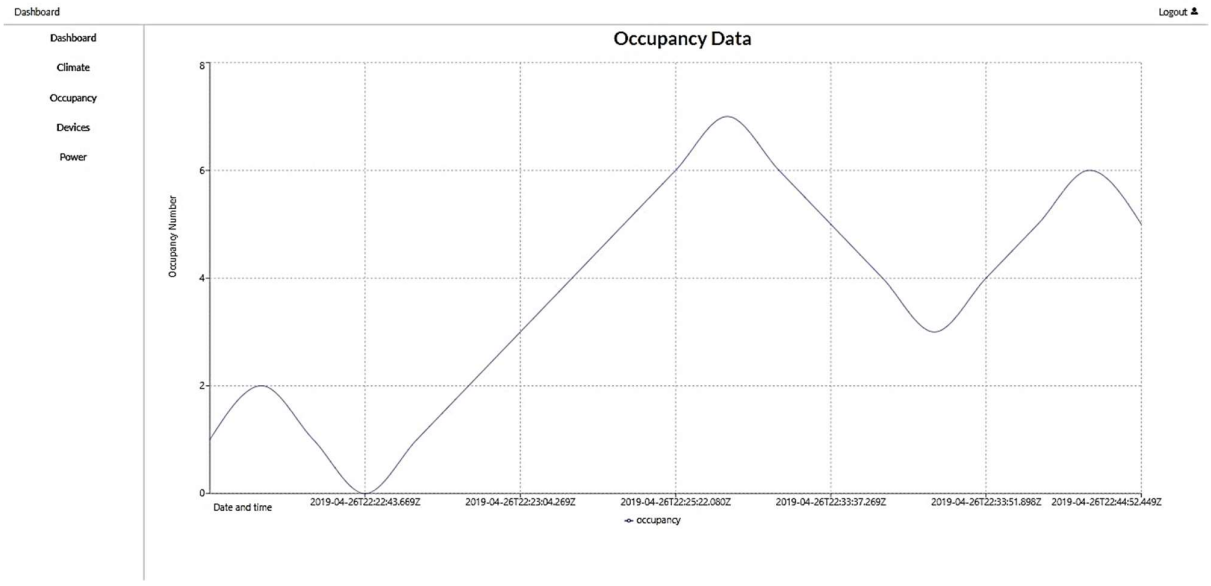


Figure 44 Website showing a line plot of power consumption data

4.6 Time series data for building occupancy

Occupancy data was captured and displayed on a line graph. When room occupancy rises, the line graph rises and when the occupancy reduces, the line graph goes on a downward trend.



5 CHAPTER FIVE: CONCLUSION AND RECOMMENDATION

5.1 Recommendation for further improvement

Although a simple and effective prototype was implemented, some improvements can be made to improve the prototype.

i. Having access control were depending on the user access level they can only access certain pieces of data.

ii. Having a setup software or webpage where the user can register the devices sending data and also set up to what room id, the data is being stored instead of hard coding to what room id the data is being stored.

iii. Being able to delete, add or update user information and also information about rooms.

iv. To provide EMF shielding in case you are placing the sensors very close to the ESP8266-01 and the NodeMCU. This is to avoid the EMF fields affecting the microcontrollers and making them nonfunctional.

v. The NodeMCU is only using 3 pins (Analog pin, Vin and Ground pin) out of 30 pins. All the other pins, as a result, are not being used. Using a microcontroller with fewer pins as opposed to using the NodeMCU could save on cost.

vi. Setting up security on MQTT and on the server. At the moment, if anyone knows the MQTT host address and the server address, he/she can spoof the data that goes into the server.

vii. PIR sensors were found to be very unstable in terms of detecting motion. The sensors would sense a lot of false negatives. Using a different sensor for detecting motion and for populating room occupancy is advised.

viii. Having separate power supply for all the electronics components instead of one.

5.2 Conclusion

A Building Management System was successfully designed and implemented based on the ESP8266-01, NodeMCU, ACS712 current sensor, PIR motion sensor, DHT11 sensor, relays, raspberry pi, a server application hosted online and an online cloud database. The system was successfully able to capture building occupancy, humidity and temperature, current consumption and power consumption and state in which appliances are on or off. This information was able to be published successfully to

an online dashboard where the user can log in and access all the data. The data was also stored in an online database.

The Raspberry pi 2 Model B was connected to a USB Wi-Fi dongle. Mosquitto was set up on the Raspberry Pi, to enable MQTT to run on the Raspberry Pi. The ESP8266-01 and the NodeMCU were then set up to connect to the same Wi-Fi network and collect data on current consumption, building occupancy, temperature, and humidity. Some of the ESP8266-01 were set up to accept instructions to turn on and off the relays. All this data was relayed to the Raspberry Pi.

A web server was set up to accept data that is being sent by the Raspberry Pi and store it on an online database. A website was then set up that could query the data that was stored in the database and display it on the dashboard of the website.

Line plots showing time series data on current, power, temperature, humidity, device state and building occupancy were plotted and displayed on the dashboard. There was a correlation in the data, for example, when the fan is turned on, the current and power consumption rose and the device state for the fan changed from an off state (0) to an on the state (1). Turning on the bulb did not affect the current and power consumption, due to the fact that the bulb that was being used as an energy saving bulb.

There was difficulty in plotting occupancy data, as the motion sensors were behaving erratically and one could not capture building occupancy as well.

The objectives of the Building Management System were met. More iterations are needed to ensure that the system works perfectly. The next step would be incorporating the improvements listed in the recommendations and incorporating future technologies such as Artificial Intelligence to make the system intelligent. Use of Data Science can also be helpful in making sense of all the data that is collected by the Building Management System.

REFERENCES

[1] Hivos, "Kenya: Energy Profi le," [Online]. Available: https://www.hivos.org/sites/default/files/kenya_profile.pdf. [Accessed 29 April 2019].

[2] Power Africa, "Development of Kenya's power sector 2015 - 2020," May 2016. [Online]. Available: https://www.usaid.gov/sites/default/files/documents/1860/Kenya_Power_Sector_report.pdf. [Accessed 29 April 2019].

[3] C. MUNDA, "Diesel and power cost rise highest in December," Business Daily, 27 DECEMBER 2018. [Online]. Available: https://www.businessdailyafrica.com/news/Diesel-and--power-cost--rise-highest--in-December/539546-4912008-byrcy6z/index.html.

[4] Espressif Systems, "https://www.espressif.com/," 2018. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf.

[5] sherry@aithinker.com, "ESP-01 WiFi Module," 2015. [Online]. Available: http://www.microchip.ua/wireless/esp01.pdf.

[6] Einstronic, "Introduction to NodeMCU ESP8266," Julu 2017. [Online]. Available: https://einstronic.com/wp-content/uploads/2017/06/NodeMCU-ESP8266-ESP-12E-Catalogue.pdf.

[7] L. Ada, "Introducing the Raspberry Pi 2 - Model B," 22 August 2018. [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/introducing-the-raspberry-pi-2-model-b.pdf. [Accessed 29 April 2019].

[8] L. Ada, "PIR Motion Sensor," 22 August 2018. [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/pir-passive-infrared-proximity-motion-sensor.pdf. [Accessed 22 April 2019].

[9] L. Ada, "DHT11, DHT22 and AM2302 Sensors," 12 January 2019. [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/dht.pdf. [Accessed 22 April 2019].

[10] Circuit Basics, "How to set up the DHT11 humidity sensor on an arduino," Circuit Basics, [Online]. Available: http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/. [Accessed 22 April 2019].

[11] I. Allegro MicroSystems, "ACS712, Fully Integrated, Hall Effect-Based Linear Current Sensor," 2007,. [Online]. Available: https://www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf. [Accessed 22 April 2019].

[12] "Songle Relay," [Online]. Available: http://www.circuitbasics.com/wp-content/uploads/2015/11/SRD-05VDC-SL-C-Datasheet.pdf. [Accessed 22 April 2019].

[13] Texas Instruments, "LM1117 800-mA Low-Dropout Linear Regulator," JANUARY 2016. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm1117.pdf. [Accessed 25 April 2019].

[14] U. T. H. L. S.-c. A. Hunkeler, "MQTT-S – A Publish / Subscribe Protocol For Wireless Sensor Networks," *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08),* p. 8, 2008.

[15] R. A Light, "Mosquitto: server and client implementation of the MQTT protocol," *The Journal of Open Source Software,* vol. 2, no. 13, pp. 1-2, 2017.

[16] C. Stapper, "An analysis of Heroku and AWS for growing startups," Salesforce, [Online]. Available: https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1134&context=cscsp. [Accessed 23 April 2019].

[17] P. S. a. c. Damien P. George, "MicroPython Documentation," 22 December 2018. [Online]. Available: https://buildmedia.readthedocs.org/media/pdf/micropython-pfalcon/latest/micropython-pfalcon.pdf. [Accessed 23 April 2019].

[18] P. B. Dr Dinesh Kumar, "A Review Paper on MERN Stack for Web Development," April 2018. [Online].

[19] C. R. Hampton, "How to Flash ESP-01 Firmware to the Improved SDK v2.0.0," All About Circuits, 17 March 2017. [Online]. Available: https://www.allaboutcircuits.com/projects/flashing-the-ESP-01-firmware-to-SDK-v2.0.0-is-easier-now/. [Accessed 22 April 2019].

[20] SparkFun, "Pull-up Resistors," [Online]. Available: https://learn.sparkfun.com/tutorials/pull-up-resistors/all. [Accessed 07 May 2019].

APPENDIX

Breadboard connections

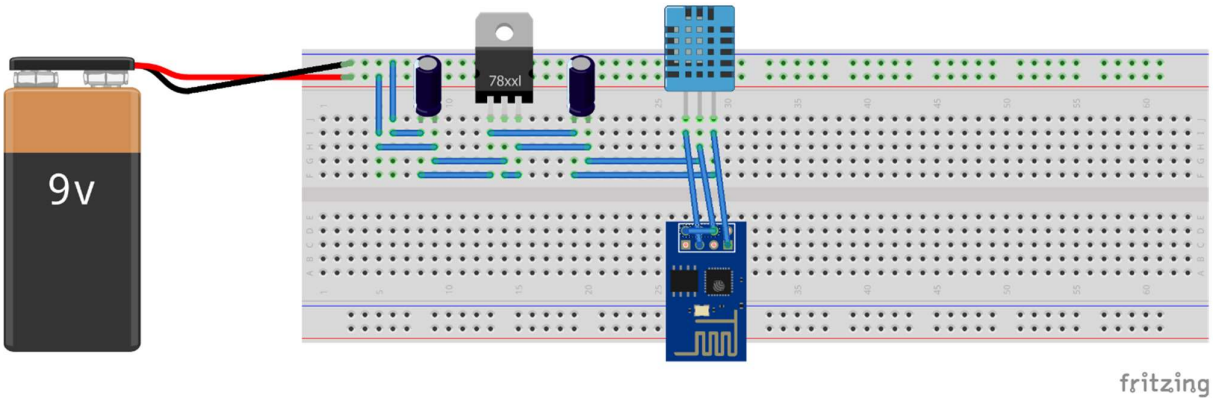


Figure 45 Breadboard design of a DHT11 sensor connected to an ESP8266-01

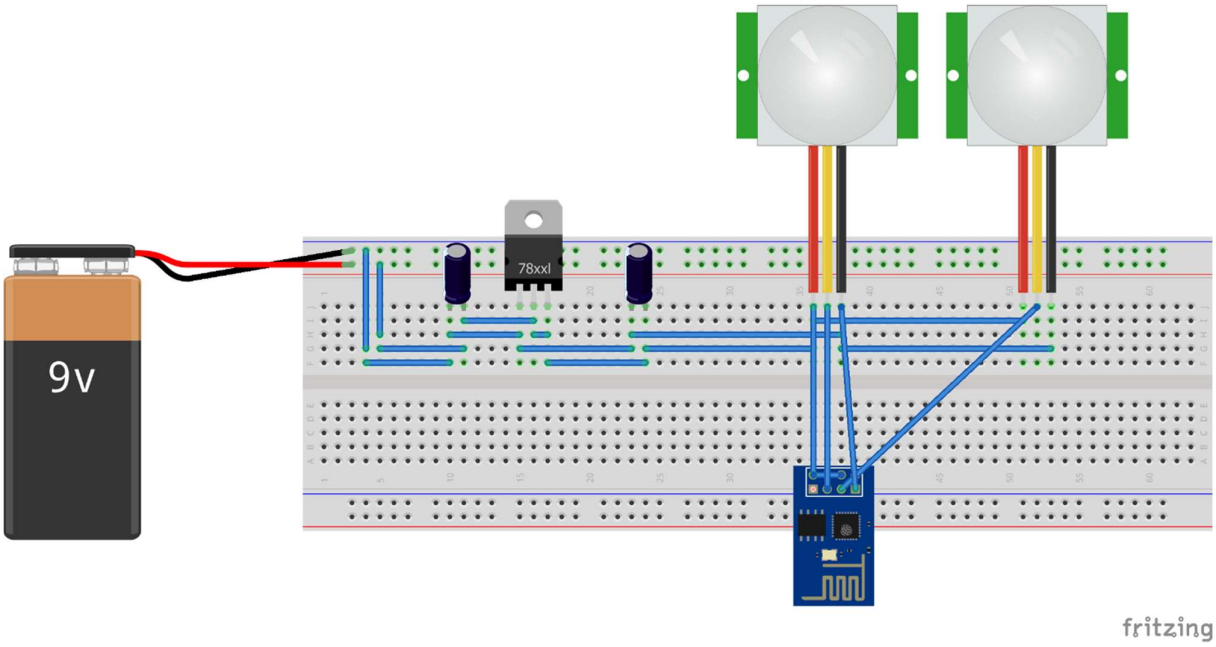


Figure 46 Breadboard design of a pair of PIR motion sensor connected to an ESP8266-01

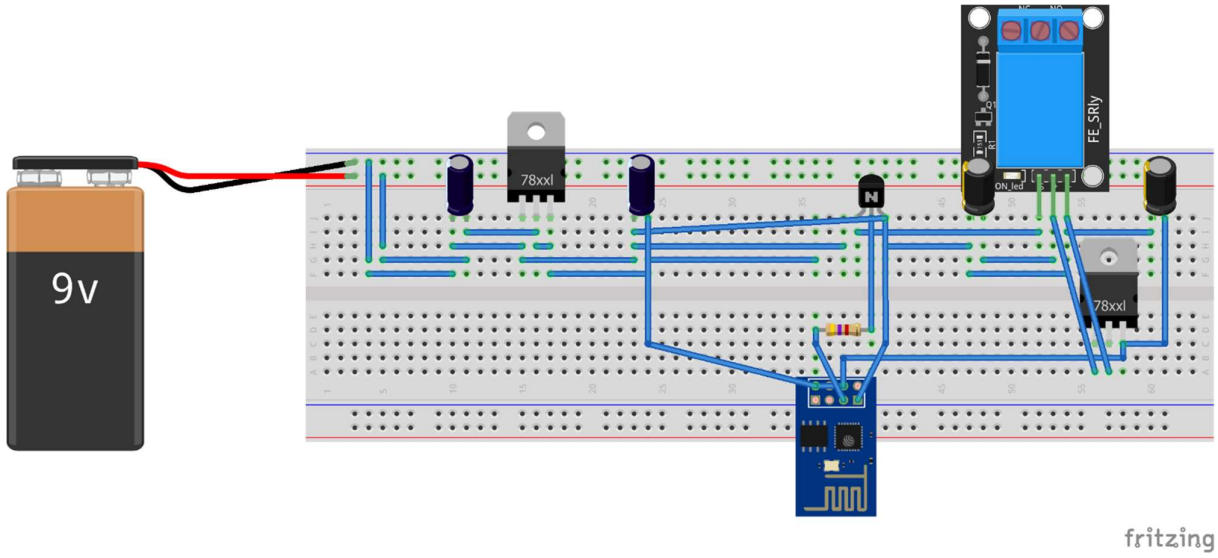


Figure 47 Breadboard design of a relay connected to an ESP8266-01

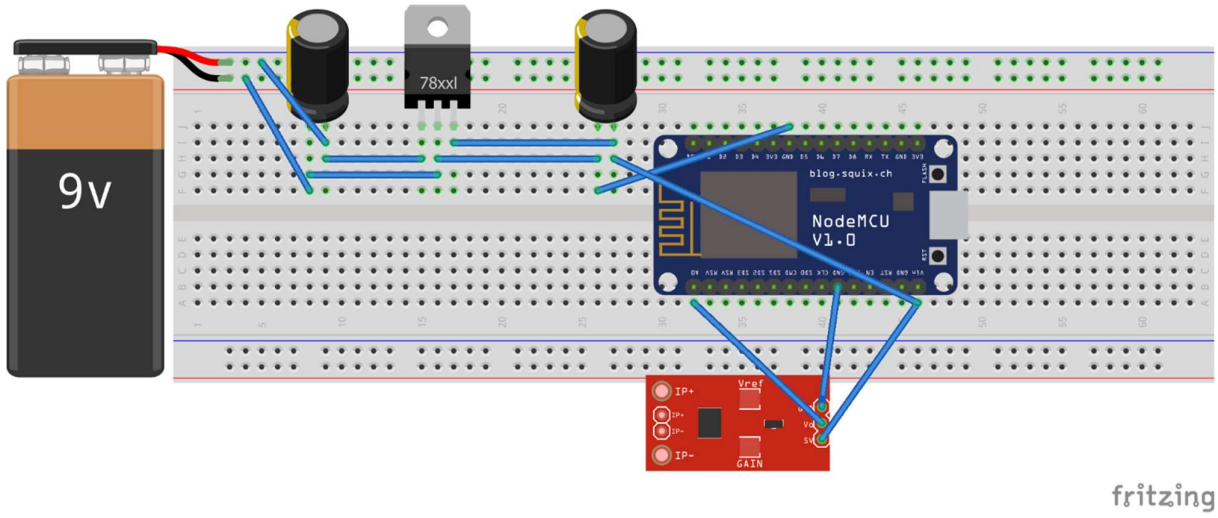


Figure 48 Breadboard design of a current sensor connected to a NodeMCU

PCB schematics

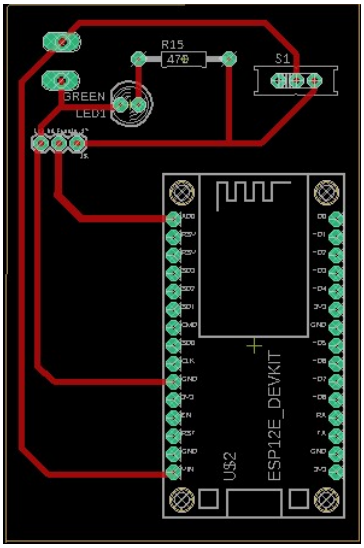


Figure 49 PCB design for the current sensor

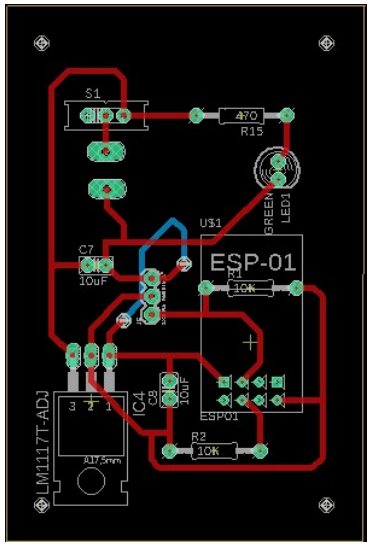


Figure 50 PCB design for the DHT11 sensor

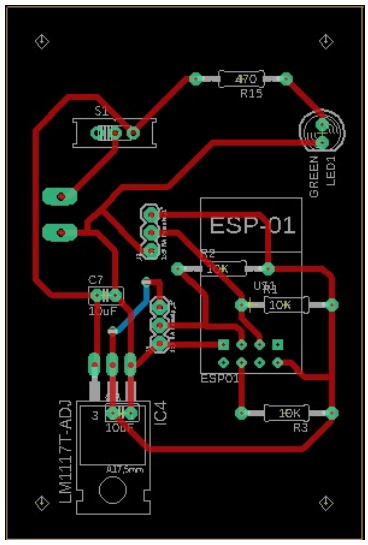


Figure 51 PCB design for PIR motion sensors

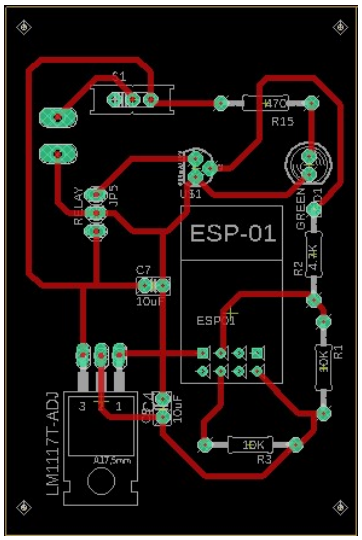


Figure 52 PCB design for the relays

Board images

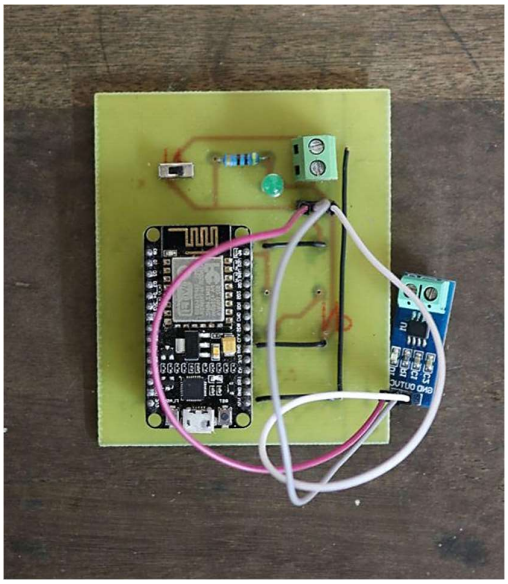


Figure 53 Current sensor connected to a NodeMCU on a PCB

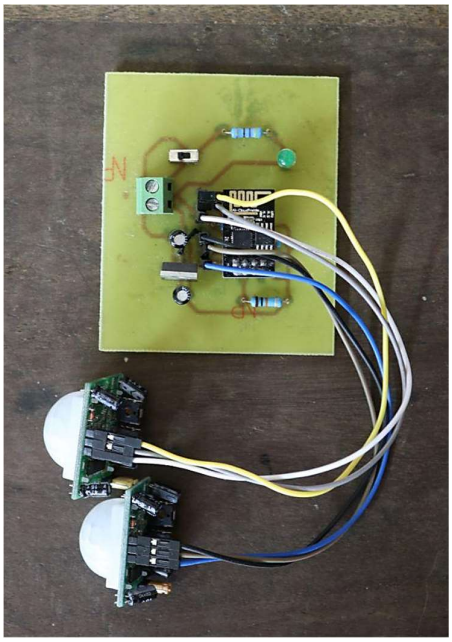


Figure 54 ESP8266-01 connected to a pair of PIR motion sensors

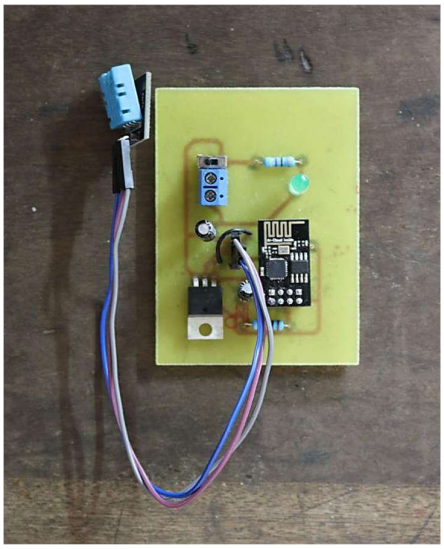


Figure 55 ESP8266-01 connected to a DHT11 sensor

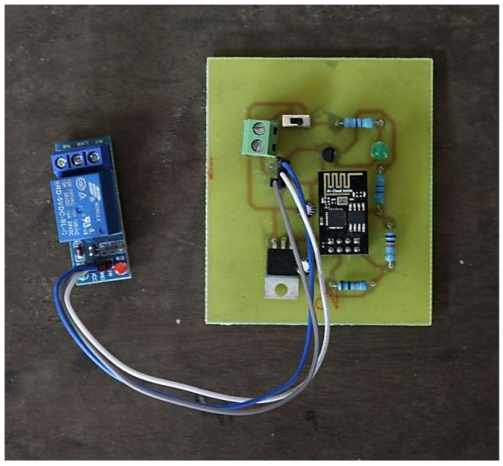


Figure 56 ESP8266-01 connected to a relay

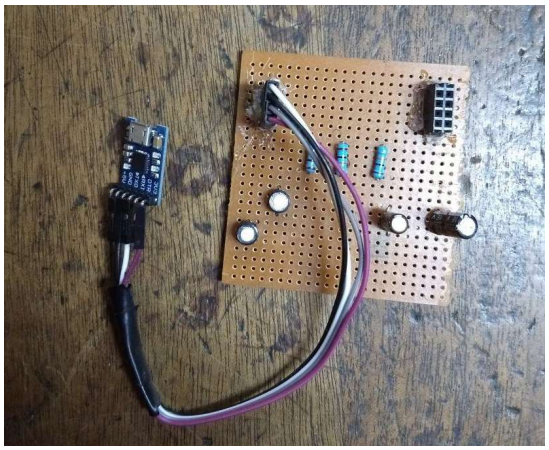


Figure 57 ESP8266-01 flashing circuit

Figure 58 Board setup with current sensor, DHT11 sensor, motion sensor and relay

SoC code

The SoC code is split into 6 parts

- Current sensor
- DHT11 sensor
- Motion sensor

- Relay (both for the fan and the bulb)
- Wi-Fi Setup

Current sensor

```
# Peter Okwara
# F17/36211/2013

# This piece of code collects analog values for current from the ACS712 5A current sensor
# and provides the current and power consumed by devices connected to the current sensor

import connectWifi
import time
import config
import json
import time
import utime
import machine
import math
from umqtt.simple import MQTTClient

connectWifi.connect()
ADC_SCALE = 1023.0
VREF = 5.0
sensitivity = 0.185
U = 230
adc = machine.ADC(0)

SERVER = config.MQTT_CONFIG['MQTT_HOST']
PORT = config.MQTT_CONFIG['PORT']
SENSOR_ID = config.MQTT_CONFIG['SENSOR_ID']
PUB_TOPIC = config.MQTT_CONFIG['PUB_TOPIC']
```

```
# function for callibrating the current sensor
def calibrate():
    global zero
    acc = 0
    x = 0
    while x < 10:
        acc += adc.read()
        x = x + 1
    zero = acc / 10
    return zero

# function for getting the current and power value from sensor
def read_sensor():
    frequency = 50
    period = 1000000 / frequency
    Isum = 0
    measurements_count = 0
    t_start = utime.ticks_ms()

    while (utime.ticks_ms() - t_start < period):
        Inow = adc.read() - zero
        Isum += Inow*Inow
        measurements_count = measurements_count + 1

    Irms = math.sqrt(Isum/measurements_count) / ADC_SCALE * VREF / sensitivity

    P = U * Irms
    return {
        "current": Irms,
        "power": P
    }

# function for sending data to the MQTT broker
def send(data):
```

```
    c = MQTTClient(SENSOR_ID, SERVER, 1883)
    c.connect()
    c.publish(PUB_TOPIC, data)
    c.disconnect()


def main():
    # start by callibrating the sensor
    calibrate()
    while True:
        # read the analog value from the sensor
        data = read_sensor()
        # transmit the analog value via MQTT
        send(data)


if __name__ == "__main__":
    main()
```

DHT11 sensor

```
# Peter Okwara
# F17/36211/2013

# This piece of code collects temperature and humidity values from the DHT11 tempera-
ture
# and humidity sensor and transmits them to the mqtt broker

import connectWifi
import time
import config
import json
from dht import DHT11
from machine import Pin
from umqtt.simple import MQTTClient
```

```
# connect to wifi first
connectWifi.connect()

# set the pin for reading the humidity ant temperature values
d = DHT11(Pin(2))

SERVER = config.MQTT_CONFIG['MQTT_HOST']
PORT = config.MQTT_CONFIG['PORT']
SENSOR_ID = config.MQTT_CONFIG['SENSOR_ID']
PUB_TOPIC = config.MQTT_CONFIG['PUB_TOPIC']

# function for reading the temperature and humidity values
def read_sensor():
    d.measure()
    return {
        "temperature": d.temperature(),
        "humidity": d.humidity()
    }

# function for sending the temperature and humidity values
def send(data):
    c = MQTTClient(SENSOR_ID, SERVER, 1883)
    c.connect()
    c.publish(PUB_TOPIC, json.dumps(data))
    c.disconnect()


def main():
    while True:
        # get the values for temperature and humidity
        data = read_sensor()
        print("Sending data", data)
        # transmit the values for temperature and humidity
```

```
        send(data)
        # delay for 8 seconds before reading the values again
        time.sleep(8)


if __name__ == "__main__":
    main()
```

Motion sensor

```
# Peter Okwara
# F17/36211/2013

# This piece of code detects whether motion has been detected on the respective pir mo-
tion
# sensor and determines whether someone is entering a room or leaving a room, relays
that
# information as occupancy level

import esp
import machine
import connectWifi
import time
import config
import utime
import json

from machine import Pin
from umqtt.simple import MQTTClient

# connect to wifi first
connectWifi.connect()

# define pins for the pir motion sensor
pin_1 = machine.Pin(0, machine.Pin.IN)
```

```
pin_2 = machine.Pin(2, machine.Pin.IN)

state = "on"
motiondetected_1 = 0
motiondetected_2 = 0
time_s1 = 0
time_s2 = 0
time_comp = 0
count = 0


SERVER = config.MQTT_CONFIG['MQTT_HOST']
PORT = config.MQTT_CONFIG['PORT']
SENSOR_ID = config.MQTT_CONFIG['SENSOR_ID']
PUB_TOPIC = config.MQTT_CONFIG['PUB_TOPIC']

# set up interrupt functions for in case pir motion sensor 1 is triggered
def handle_interrupt_1(p_0):
    print('pin change', p_0)
    global motiondetected_1, time_s1
    motiondetected_1 = 1
    time_s1 = time.time()

# set up interrupt functions for in case pir motion sensor 2 is triggered
def handle_interrupt_2(p_1):
    print('pin change', p_1)
    global motiondetected_2, time_s2
    motiondetected_2 = 1
    time_s2 = time.time()

# set up interrupts for the two pins
pin_1.irq(trigger=Pin.IRQ_RISING, handler=handle_interrupt_1)
pin_2.irq(trigger=Pin.IRQ_RISING, handler=handle_interrupt_2)
```

```
def main(server=SERVER, port=PORT):
    # set up and connect to the MQTT client
    c = MQTTClient(SENSOR_ID, SERVER, 1883)
    c.connect()
    while True:
        global state, motiondetected_1, motiondetected_2, time_s1, time_s2, time_comp,
count
        while True:
            time_comp = time_s1 - time_s2
            # determine if someone is moving in to a room and transmit the occupancy
            if time_comp < 0 and abs(time_comp) < 4:
                print("moving in")
                print(time_comp)
                time.sleep(2)
                count = count + 1
                time_s1 = 0
                time_s2 = 0
                data = {"occupancy": count}
                c.publish(PUB_TOPIC, json.dumps(data))
            # determine if someone is movint out of a room and transmit the occupancy
            elif time_comp > 0 and abs(time_comp) < 4:
                print("moving out")
                print(time_comp)
                time.sleep(2)
                count = count - 1
                if count < 0:
                    count = 0
                time_s1 = 0
                time_s2 = 0
                data = {"occupancy": count}
                c.publish(PUB_TOPIC, json.dumps(data))
            time_comp = 0
if __name__ == "__main__":
```

```
main()
```

Wifi setup

```
# Set up connection to wifi access point
def connect():
    import network
    import config

    ssid = config.WIFI_CONFIG['WIFI_ESSID']

    password = config.WIFI_CONFIG['WIFI_PASSWORD']

    station = network.WLAN(network.STA_IF)

    if station.isconnected() == True:
        print("Already connected")

    station.active(True)
    station.connect(ssid, password)

    while station.isconnected() == False:
        pass

    print("Connection successful")
    print(station.ifconfig())
```

Relay

```
#Peter Okwara
# F17/36211/2013

# This piece of code awaits for instructions to turn on or off a GPIO pin to control a relay

import connectWifi
```

```
import time
import config
import json
from dht import DHT11
from machine import Pin
from umqtt.simple import MQTTClient

# connect to wifi
connectWifi.connect()

# set GPIO_2 pin as an output pin
RPin = Pin(2, Pin.OUT, value=0)

SERVER = config.MQTT_CONFIG['MQTT_HOST']
PORT = config.MQTT_CONFIG['PORT']
SENSOR_ID = config.MQTT_CONFIG['SENSOR_ID']
PUB_TOPIC = config.MQTT_CONFIG['PUB_TOPIC']

c = MQTTClient(SENSOR_ID, SERVER)

# function to set relay on
def relay_on():
    send("Relay on")
    RPin.on()

# function to set relay off
def relay_off():
    send("Relay off")
    RPin.off()

# function to send confirmation that relay is on or off
def send(data):
    c.publish(PUB_TOPIC, json.dumps(data))
```

```
# function to subscribe for incomimg commands
def sub_cb(topic, msg):
    print((topic, msg))
    command = msg.decode('ASCII')
    if command == "on":
        relay_on()
    elif command == "off":
        relay_off()


def main(server=SERVER):
    c.set_callback(sub_cb)
    c.connect()
    c.subscribe(PUB_TOPIC)
    # check for incoming messages/commands
    while True:
        if True:
            # Blocking wait for message
            c.wait_msg()
        else:
            # Non-blocking wait for message
            c.check_msg()
            # Then need to sleep to avoid 100% CPU usage (in a real
            # app other useful actions would be performed instead)
            time.sleep(1)

    c.disconnect()


if __name__ == "__main__":
    main()
```

Raspberry Pi code

```
// Peter Okwara

// This piece of code performs automation (if temperature is above 30 degrees turn on fan
// else turn off fan) and (if occupancy is below 0 turn off bulb, if it is above 0 turn on
// bulb) and also relays all incoming information to a database

const mqtt = require("mqtt");
const axios = require("axios");

// client configuration
const broker = require("./config/broker").brokerUrl;

// connect to the broker
const client = mqtt.connect(broker);

// server configuration
const server = require("./config/server").serverUrl;

let fan = "0";
let bulb = "0";

// subscribe to climate
client.on("connect", function() {
  client.subscribe("building/room/climate");
  console.log("client has subscribed successfully");
});

// subscribe to power info
client.on("connect", function() {
  client.subscribe("building/room/power");
  console.log("client has subscribed successfully");
});
// subscribe to relay for the fan
client.on("connect", function() {
```

```
  client.subscribe("building/room/relay/fan");
  console.log("client has subscribed successfully");
});
// subscribe to relay for the bulb
client.on("connect", function() {
  client.subscribe("building/room/relay/bulb");
  console.log("client has subscribed successfully");
});

// subscribe to room occupancy
client.on("connect", function() {
  client.subscribe("building/room/occupancy");
  console.log("client has subscribed successfully");
});

client.on("message", function(topic, message) {
  console.log("message is " + message.toString());
  console.log("topic is " + topic.toString());
  switch (topic.toString()) {

    // when we receive information on climate in the room
    case "building/room/climate":

      // check if temperature is above 30, if it is turn on fan, if it is not turn off fan
      if (JSON.parse(message).temperature > 30) {
        client.publish("building/room/relay/fan", "on");
      } else {
        client.publish("building/room/relay/fan", "off");
      }

      // format the climate data into a json format and post it to a database
      var data = {
        temperature: JSON.parse(message).temperature,
        humidity: JSON.parse(message).humidity,
```

```
    date: Date.now(),
    room_id: "1"
   };
   axios
    .post(server + "/api/climate", data)
    .then(res => console.log(res))
    .catch(err => console.log(err));
   return;

// when we receive information about power in the room
case "building/room/power":

   // format the data into a json format and then post it to an online database
   var data = {
    power: JSON.parse(message).power,
    current: JSON.parse(message).current,
    date: Date.now(),
    room_id: "1"
   };
   axios
    .post(server + "/api/power", data)
    .then(res => console.log(res))
    .catch(err => console.log(err));
   return;

// when we receive information on room occupancy
case "building/room/occupancy":

   // if room occupancy is less than zero, turn off the bulb. if it is not turn on
   // the bulb
   if (JSON.parse(message).occupancy <= 0) {
    client.publish("building/room/relay/bulb", "off");
   } else {
    client.publish("building/room/relay/bulb", "on");
```

```
  }

  // put the occupancy data into a json format and post it to an online database
  var data = {
    occupancy: JSON.parse(message).occupancy,
    room_id: "1"
  };
  axios
    .post(server + "/api/occupancy", data)
    .then(res => console.log(res))
    .catch(err => console.log(err));
  return;
// when we receive information about the fan
case "building/room/relay/fan":
  let fan_state = message;

  // if fan state is on, publish on the database that the fan state is on
  // if fan state is off, publish on the database that the fan state is off
  if (fan_state == "on") {
    fan = "1";
    var data = {
      fan_state: fan,
      bulb_state: bulb,
      room_id: "1"
    };
    axios
      .post(server + "/api/devices", data)
      .then(res => console.log(res))
      .catch(err => console.log(err));
  } else if (fan_state == "off") {
    fan = "0";
    var data = {
      fan_state: fan,
      bulb_state: bulb,
```

```
      room_id: "1"
    };
    axios
      .post(server + "/api/devices", data)
      .then(res => console.log(res))
      .catch(err => console.log(err));
  }
  return;
// when we receive information about the bulb
case "building/room/relay/bulb":
  let bulb_state = message;

  // if bulb state is on, publish on the database that the fan state is on
  // if bulb state is off, publish on the database that the fan state is off
  if (bulb_state == "on") {
    bulb = "1";
    var data = {
      fan_state: fan,
      bulb_state: bulb,
      room_id: "1"
    };
    axios
      .post(server + "/api/devices", data)
      .then(res => console.log(res))
      .catch(err => console.log(err));
  } else if (bulb_state == "off") {
    bulb = "0";
    var data = {
      fan_state: fan,
      bulb_state: bulb,
      room_id: "1"
    };
    axios
      .post(server + "/api/devices", data)
```

```
            .then(res => console.log(res))
            .catch(err => console.log(err));
        }
        return;
      default:
        break;
    }
});
```

Website code

Due to the large size of the code for the website, the code has been made available at https://github.com/peterokwara/BuildingManagement/tree/master/site

Bill of quantities

Table 4 Bill of quantities

Item no	Component	Quantity	Cost per piece	Total cost	Supplier
1	ESP8266-01	4	400	1600	Nerokas
2	NodeMCU	1	1400	1400	Nerokas
3	RaspberryPi 2 Model B	1	5000	5000	Nerokas
4	ACS712 current sensor	1	450	450	Nerokas
5	PIR motion sensor	2	200	400	Nerokas
6	DHT11 sensor	1	400	400	Ktechnics
7	Relay	2	100	200	Ktechnics
8	10 kΩ Resistor	7	2	14	Ktechnics
9	470 Ω Resistor	5	2	10	Ktechnics
10	Connectors	5	20	100	Massimo
11	Wires	10	10	100	Nerokas
12	Capacitor 10 µF	8	5	40	Nerokas
13	Capacitor 220 µF	1	5	5	Nerokas
14	Capacitor 100 nF	1	5	5	Nerokas
15	Female header 48 pin	3	20	60	Nerokas
16	Switch	5	20	100	Ktechnics
17	Led-green	5	3	15	Ktechics
18	Presensitized pcb	3	550	1650	Nerokas
19	Wifi Dongle	1	600	600	Nerokas
			TOTAL	12,149	