CHAPTER ONE

INTRODUCTION

1.1 General Overview

The Internet of Things (IoT) is defined as a paradigm in which objects equipped with sensors, actuators, and processors communicate with each other to serve a meaningful purpose . Therefore, this research proposes an IOT integrated community service platform system architecture that can provide a single integrated and intelligent community services linked with the smart city and the smart home applied to all daily lives of users .by implementing prototype based on Internet of Things (IoT) technologies. These design typically combine freely available software source code and development tools with inexpensive hardware gadgets such as Arduino and Raspberry Pi microcontrollers , that manipulate the objects to be outfitted with sensing, identification and positioning devices and endowed with an IP address to become smart objects, and capable of communicating with other smart objects .To add new features to them or to alter them to serve a different purpose that they originally were designed for.

1.2 Problem Statement

The current control systems are still lacking in both potentials and the number of implementations , the idea of full control and connectivity needs a new technology to approach the ideal concept of modern society. to achieve the dream for higher quality agriculture, more safe and smart homes and industries .Not to mention the need for a more luxurious life style and ease of everyday routine , such issues have been concerning engineers and inventor for a long while , and now more than ever it needs the cooperation of all .

1.3 Objectives

The main objectives of this study are:

- To state a practical implementation of the Internet of Things .
- To demonstrate the ability of full integration in such systems ,and the ability to control it using a single platform .
- To improve smart homes by providing connectivity through the Internet
- To prevent theft in industries and improve automation .

1.4 Methodology

The methodology that has been followed in this project is:

- Review previous projects and models built on IoT .
- Research about IoT structure and Communication protocols.
- Design each part of the IoT platform .
- Implement and test the circuits .
- Arduino C , Python and HTML are used to program controllers.

1.5 Project Outlines

The first chapter stars with a brief introduction to demonstrate the idea of the Internet of Things and its implementations , after that the problem statement , objectives and the methodology taken throughout the project .The second chapter consist of the history of industrial revolutions that led to the birth of IoT, and a detailed overview about the internet of things platform and its fields . The third chapter provide detailed information about the hardware components and the software tools along with a block diagram of the system . The forth chapter focus on the implementation and testing of the platform along with the server , framework and communication protocols used . The fifth chapter contains the conclusion and recommendations for future Improvements .

CHAPTER TWO

LITERATURE REVIEW

2.1 Industrial Revolution

is the process of change from an agrarian and handicraft economy to one dominated by industry and machine manufacturing. It includes four types:

- **Industrial revolution 1.0:** Introducing mechanical production machines powered by water and steam End of the 18th century. 1760 to 1840 - Ushered in Mechanical production; railways and steam engine.
- **Industrial revolution 2.0 :** Introducing mass production lines powered by electric energy at the beginning of the 20th century. 1870 to 1940 - Mass production; electricity and assembly line .
- **Industrial revolution 3.0:** Through the use of electronics and IT further progression in autonomous production. 1960 to 2010 - Computers; semi-conductors, main frame computing, personal devices and the internet.
- **Industrial revolution 4.0:** is the ongoing automation of traditional manufacturing and industrial practices, using large-scale (M2M)and the internet of things (IOT)are integrated for increased automation, and production of smart machines without the need for human intervention.

• Industrial revolution 4.0 overview

A collective term for technologies and concepts of value chain organization. Based on the technological concepts of cyber-physical systems, the Internet of Things and the Internet of Services, it facilitates the vision of the Smart Factory. Within the modular structured Smart Factories of Industry 4.0, cyber-physical systems monitor physical processes, create a virtual copy of the physical world and make decentralized decisions.

Over the Internet of Things, Cyber-physical systems communicate & cooperate with each other & humans in real time. Via the Internet Services, both internal & cross-organizational services are offered & utilized by participants of the value chain. Some of the most important characteristics introduced by the industrial revolution 4.0 are :

- ➢ It's built on the Digital revolution.
- ➢ Smaller & powerful sensors.
- ➢ Machine Learning.
- ➢ Ubiquitous internet.
- ➢ Artificial Intelligence (AI).
- ➢ Labor & Energy Cost.

Industry 4.0 utilize many fields as shown in Figure 2.1each of them shoulde include Six Design Principles :

- **Interoperability**: the ability of cyber-physical systems (i.e. work piece carriers, assembly stations and products), humans and Smart Systems to connect and communicate with each other via the Internet of Things and the Internet of Services.
- **Virtualization:** a virtual copy of the Smart System which is created by linking sensor data (from monitoring physical processes) with virtual plant models and simulation models.
- **Decentralization:** the ability of cyber-physical systems within Smart Systems to make decisions on their own.
- **Real-Time Capability:** the capability to collect and analyze data and provide the insights immediately.
- **Service Orientation:** offering of services (of cyber-physical systems, humans and Smart Factories) via the Internet of Services.

- **Modularity:** flexible adaptation of Smart Factories for changing requirements of individual modules.

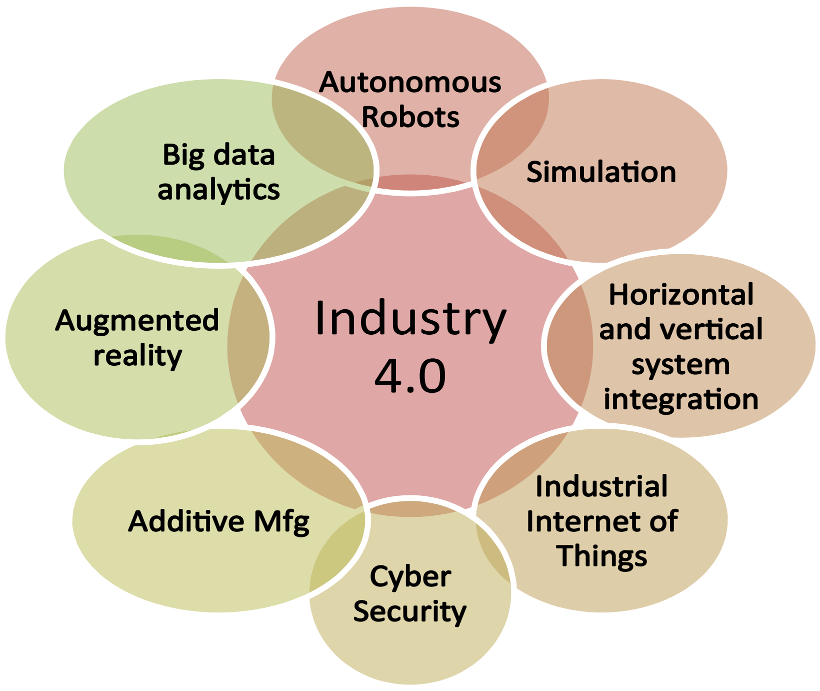


Figure 2.1:Building blocks of Industry 4.0

- **SIEMENS** :German manufacturing giant Siemens, an industrial user, is implementing an Industry 4.0 solution in medical engineering. **TRUMPF** :German toolmaker , an Industry 4.0 supplier and worldwide market leader of laser systems, has put the first "social machines" to work. Each component is "smart" and knows what work has already been carried out on it.
- **GEPredix**: the operating system for the Industrial Internet, is powering digital industrial businesses that drive the global economy. By connecting industrial equipment, analyzing data, and delivering real time insights. As shown in Figure 2.2 Examples of Product evolution: Connected and smart products.

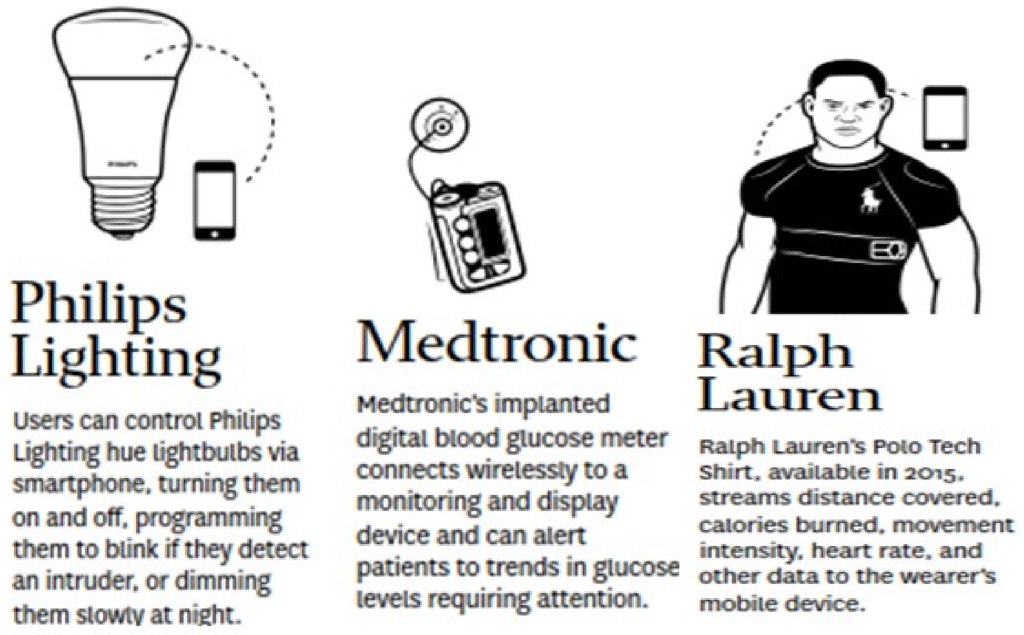

Philips Lighting
Users can control Philips Lighting hue lightbulbs via smartphone, turning them on and off, programming them to blink if they detect an intruder, or dimming them slowly at night.
Medtronic
Medtronic's implanted digital blood glucose meter connects wirelessly to a monitoring and display device and can alert patients to trends in glucose levels requiring attention.
Ralph Lauren
Ralph Lauren's Polo Tech Shirt, available in 2015, streams distance covered, calories burned, movement intensity, heart rate, and other data to the wearer's mobile device.

Figure 2.2 : Examples of smart products

2.2 The Internet of Things (IoT)

The internet of things is a communications network which is built on an "always-on" connectivity to the Internet. IoT can be well-thought-out as a grid of corporeal things which can be log on through the Internet .It is characterized by the combination of physical and digital components to create new products and enable novel models. Existence of increasingly efficient power management, broadband communication, reliable memory and advances in microprocessor technologies, it has become possible to digitalize functions and key capabilities of industrial-age products .

While the term Internet of Things is now more and more broadly used, there is no common definition or understanding today of what the IoT actually encompasses. The origins of the term date back more than 15 years and have been attributed to the work of the Auto-ID Labs, in 1999, at the Massachusetts Institute of Technology (MIT) on networked radio-frequency identification (RFID) infrastructures .Since then, visions for the Internet of Things have been further developed and extended beyond the scope of RFID technologies. The International Telecommunication Union(ITU) for instance now defines the Internet of Things as ‘a global infrastructure for the Information Society’, enabling advanced services by interconnecting (physical and virtual)things based on, existing and evolving, interoperable information and communication technologies.[16] At the same time, a multitude of alternative definitions has been proposed. Some of these definitions exhibit an emphasis on the things which become connected in the IoT. Other definitions focus on Internet-related aspects of the IoT, such as Internet protocols and network technology. And a third type centers on semantic challenges in the IoT relating to, e.g., the storage, search and organization of large volumes of information .[14]

2.2.1 Application for IoT technologies

The term “IoT” ,which was first proposed by Kevin Ashton, a British technologist, in 1999, has the potential to impact everything from new product opportunities to shop floor optimization to factory worker efficiency gains that will power top-line and bottom-line gains. It is believed that IoT will improve energy efficiency, remote monitoring, and control of physical assets and productivity through applications as diverse as home security to condition monitoring on the factory floor. Now IoT has been used in many application include, e.g., the smart industry, where the development of intelligent production systems and connected production sites is often discuss the heading of Industry 4.0. In the smart home or building area, intelligent thermostats and security systems are receiving a lot of attention, while smart energy applications focus on smart electricity, gas and water meters. Smart transport solutions include, e.g., vehicle fleet tracking and mobile ticketing, while in the smart health area, topics such as patients’ surveillance and chronic disease management are being addressed. And in the context of smart city projects, solutions like the real-time monitoring of parking space availability and intelligent lighting of streets are being explored .[14]

2.2.2 Architecture of IoT

There is no single consensus on architecture for IoT, which is agreed universally. Different architectures have been proposed by different researchers .Three- and Five-Layer Architectures. The most basic architecture is three-layer architecture. It has three layers, namely, the perception, network ,and application layers.

- **The perception layer**

is the physical layer, which has sensors for sensing and gathering information about the environment. It senses some physical parameters or identifies other smart objects in the environment.

- **The network layer**

is responsible for connecting to other smart things, network devices, and servers. Its features are also used for transmitting and processing sensor data.

- **The application layer**

is responsible for delivering application specific services to the user. It defines various applications in which the Internet of Things can be deployed, for example, smart homes, smart cities, and smart health.

The three-layer architecture defines the main idea of the Internet of Things, but it is not sufficient for research on IoT because research often focuses on finer aspects of the Internet of Things. That is why, we have many more layered architectures proposed in the literature. One is the five layer architecture, which additionally includes the processing and business layers.
The five layers are perception, transport, processing, application, and business layers. The role of the perception and application layers is the same as the architecture with three layers.

- **The transport layer**

transfers the sensor data from the perception layer to the processing layer and vice versa through networks such as wireless, 3G, LAN, Bluetooth, RFID.

- **The processing layer**

is also known as the middleware layer. It stores, analyzes, and processes huge amounts of data that comes from the transport layer. It can Manage and provide a diverse set of services to the lower layers.

- **The business layer**

Manages the whole IoT system ,including applications, business and profit models ,and users' privacy.

Another architecture proposed by (Ning and Wang)is inspired by the layers of processing in the human brain. It is inspired by the intelligence and ability of human beings to think, feel, remember, make decisions, and react to the physical environment. It is constituted of three parts. First is human brain, which is analogous to the processing and data management center. Second is the spinal cord, which is analogous to the distributed network of data processing nodes and smart gateways. Third is the network of nerves, which corresponds to the networking component and sensors.[12]

2.3 The Internet of Things Platform

At a high level, an Internet of Things (IoT) platform is the support software that connects edge hardware, access points, and data networks to other parts of the value chain (which are generally the end-user applications). An IoT platform facilitates communication, data flow, device management, and the functionality of applications. A platform is not the application itself, although many applications can be built entirely within an IoT platform framework. It links machines, devices, applications, and people to data and control centers. It employs better, quicker search engines and data storage systems with the capacity and sophistication to handle volume far beyond what has brought industry to the present moment. Most of its elements are cloud-based and running on wireless connectivity, which may be established via third-party providers, application programming interfaces (APIs), or most likely a combination of these technologies. Through dashboards, APIs, data engines, and algorithms, a platform enables elements and sectors of a business network to connect, monitor, and communicate with each other with far greater speed and flexibility than we have yet seen .In the context of discussions about IoT technologies, a frequently-used concept is that of IoT platforms. In computing, the term ‘‘platform’’ itself is a relatively broad concept, which has been defined as a group of technologies that are used as a base upon which other applications, processes or technologies are developed. Within the Internet of Things, IoT platforms are essentially software products, which offer comprehensive sets of application-independent functionalities that can be utilized to build IoT applications. There is no standard configuration of an IoT platform, but a multitude of IoT platforms exists, which address specific needs and areas of application. While some IoT platforms, e.g., Eclipse, are rather thing-focused and offer mainly functionality to develop and operate embedded applications on things. There are three main types of platforms from most sophisticated to least sophisticated:

- **Application enablement and development:** This includes platforms that offer templates, modules, or widget-based frameworks in order to create actual end-user applications. These platforms can quickly turn data into either intelligence or action.
- **Network, data, and subscriber management**: In the wireless carrier these platforms are trying to simplify connecting cellular M2M data, so you don’t have to build much of the data infrastructure behind it.

- **Device management:** These platforms are more about monitoring, troubleshooting, and administrating the provisioning and health of the endpoints. This is more typical of a lot of hardware in the IoT space.

2.3.1 Smart home and smart building

the rise of Wi-Fi’s role in home automation has primarily come about due to the networked nature of deployed electronics where electronic devices (TVs and AV receivers, mobile devices, etc.) have started becoming part of the home IP network and due the increasing rate of adoption of mobile computing devices (smartphones, tablets, etc.).Several organizations are working to equip homes with technology that enables the occupants to use a single device to control all electronic devices and appliances. The solutions focus primarily on environmental monitoring, energy management, assisted living, comfort, and convenience. The solutions are based on open platforms that employ a network of intelligent sensors to provide information about the state of the home. These sensors monitor systems such as energy generation and metering.

2.3.2 smart factory and smart manufacturing

The role of the Internet of Things is becoming more prominent in enabling access to devices and machines, which in manufacturing systems. The IoT will connect the factory to a whole new range of applications, which run around the production. This could range from connecting the factory to the smart grid, sharing the production facility as a service or allowing more agility and flexibility within the production systems themselves. In this sense, the production system could be considered one of the many Internets of Things (IoT), where a new ecosystem for smarter and more efficient production could be defined. The first evolutionary step towards a shared smart factory could be demonstrated by enabling access to today’s external stakeholders in order to interact with an IoT-enabled manufacturing system. These stakeholders could include the suppliers of the productions tools (e.g. machines, robots), as well as the production logistics (e.g. material flow), and maintenance and re-tooling actors. An IoT-based architecture that challenges the closed factory automation pyramid, by allowing the above-mentioned stakeholders to run their services in multiple tier flat production system. The room for innovation in the application space could be increased in the same degree of magnitude as this has been the case for embedded applications or Apps, which have exploded since the arrival of smart phones (i.e. the provision of a clear and well standardized interface to the embedded hardware of a mobile phone to be accessed by all types of Apps).

2.3.3 Smart agriculture (Farming)

Smart agriculture is considered as the basis of life for the human species as it is the main source of food grains and other raw materials. It plays vital role in the growth of country's economy. It also provides large ample employment opportunities to the people. Growth in agricultural sector is necessary for the development of economic condition of the country. Unfortunately, many farmers still use the traditional methods of farming which results in low yielding of crops and fruits. But wherever automation had been implemented and human beings had been replaced by automatic machineries, the yield has been improved. Hence there is need to implement modern science and technology in the agriculture sector for increasing the yield.

Most of the papers signifies the use of wireless sensor network which collects the data from different types of sensors and then send it to main server using wireless protocol. The collected data provides the information about different environmental factors which in turns helps to monitor the system. Monitoring environmental factors is not enough and complete solution to improve the yield of the crops. There are number of other factors that affect the productivity to great extent. These factors include attack of insects and pests which can be controlled by spraying the crop with proper insecticide and pesticides. Secondly, attack of wild animals and birds when the crop grows up. There is also possibility of thefts when crop is at the stage of harvesting. Even after harvesting, farmers also face problems in storage of harvested crop. So, in order to provide solutions to all such problems, it is necessary to develop integrated system which will take care of all factors affecting the productivity in every stages like; cultivation, harvesting and post harvesting storage .

2.4 Microcontroller Overview

A microcontroller is a single-chip computer .Micro suggests that the device is small, and controller suggests that it is used in control applications. Another term for microcontroller is embedded controller, since most of the microcontrollers are built into or embedded in the devices that controlling. Microcontrollers have traditionally been programmed using the assembly language of the target device. Microcontrollers have different assembly languages, so the user must learn a new language with every new microcontroller he or she uses. Microcontrollers can also be programmed using a high-level language, such as BASIC, PASCAL, or C. High-level languages are much easier to learn than assembly languages and also facilitate the development of large and Complex programs . microcontroller is designed for a very specific task to control a particular system. As a result, the parts can be simplified and reduced, which cuts down on production cost.

2.4.1 Microcontroller components

A microcontroller contains one or more following components:

- **Central processing unit**

Central Processing Unit is the brain of a microcontroller. CPU is responsible for fetching the instruction, decodes it, and then finally executed. CPU connects every part of a microcontroller into a single system.

- **Memory**

Memory in a microcontroller is same as microprocessor. It is used to store data and program.

- **Parallel input/output ports**

Parallel input/output ports are mainly used to drive/interface various devices such as LCD'S, LED'S, printers, memories, etc. to a microcontroller.

- **Serial interfacing ports**

Serial ports provide various serial interfaces between microcontroller and other peripherals like parallel ports.

- **Timers and counters**

A microcontroller may have more than one timer and counters. The timers and counters provide all timing and counting functions inside the microcontroller. The major operations of this section are performing clock functions, modulations, pulse generations, frequency measuring, making oscillations, etc.

2.4.2 Microcontroller applications

Microcontrollers are widely used in modern electronic equipment. They used in biomedical instruments, Communication systems, as peripheral controller in Personal Computer (PC), robotics and in automobile fields. Microcontroller applications found in many lives filed, for example indoor lock, temperature and pressure monitoring, etc...[13]

CHAPTER THREE

HARDWARE AND SOFTWARE COMPONENTS

3.1 Introduction

The platform is constructed of two parts which is hardware and software . the hardware part resembles the controller circuits installed in the facilities ,each has its own set of sensors ,actuators and program depending on its function . the software part consists of the programs within controllers , the communication protocols , servers and firmware which are described in details later .

3.2 Hardware Components

The basic hardware components of the platform are as follows :

- The raspberrypi 3.
- The ESP8266 Wi-Fi module .
- The Node MCU .
- Smoke detector .
- Motion detector.
- Wi-Fi router.
- The L293D motor driver.
- Servo motor.

3.2.1 The raspberrypi

The Raspberry Pi is a single-board computer created by the Raspberry Pi Foundation, a charity formed with the primary purpose of reintroducing low-level computer skills to children in the UK. The aim was to rekindle the microcomputer revolution from the 1980s, which produced a whole generation of skilled programmers. Even before the computer was released at the end of February 2012, it was clear that the Raspberry Pi had gained a huge following worldwide and has now sold over 2 million units . The name, Raspberry Pi, was the combination of the desire to create an alternative fruit-based computer (such as Apple, BlackBerry, and Apricot) and a nod to the original concept of a simple computer that can be programmed using Python , so PI is the short to Python .[2] figure 3.1 below show the raspberrypi 3.

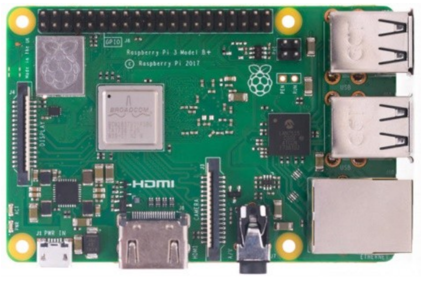


Figure 3.1 : The Raspberrypi 3 model B .

Several generations of Raspberry Pi have been released. All models feature a Broadcom system on a chip (SoC) with an integrated ARM-compatible central processing unit (CPU) and on-chip graphics processing unit (GPU). Processor speed ranges from 700 MHz to 1.4 GHz for the Pi 3 Model B+, on-board memory ranges from 256 MB to 1 GB random-access memory (RAM). Secure Digital (SD) cards in MicroSDHC form factor (SDHC on early models) are used to store the operating system and program memory. The boards have one to five USB ports. For video output, HDMI and composite video are supported, with a standard 3.5 mm tip-ring-sleeve jack for audio output. Lower-level output is provided by a number of GPIO pins, which support common protocols like I^{2}C.

The Raspberrypi acts as a MQTT server that manages the communications between devices use a publish and subscribe criteria , it's also contains the Python code that holds the necessary libraries and arguments for the operations to run smoothly . embedded in the python code is Flask Firmware that manages the graphical user interface to make orders .

- **Operating Systems :**The Raspberry Pi Foundation provides Raspbian, a Debian-based Linux distribution for download. It promotes Python and Scratch as the main programming languages, with support for many other languages. The default firmware is closed source, while an unofficial open source is available. Many other operating systems can also run on the Raspberry Pi, including the formally verified microkernel .

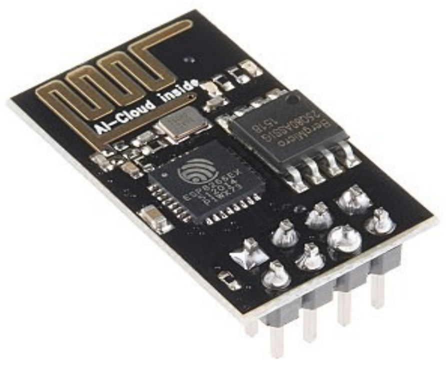
- **Use in home automation :** There are a number of developers and applications that are leveraging the Raspberry Pi for home automation. These programmers are making an effort to modify the Raspberry Pi into a cost-affordable solution in energy monitoring and power consumption. Because of the relatively low cost of the Raspberry Pi, this has become a popular and economical alternative to the more expensive commercial solutions.

- **Use in Industrial Automation :** In June 2014, Polish industrial automation manufacturer TECHBASE released ModBerry, an industrial computer based on the Raspberry Pi Compute Module. The device has a number of interfaces, most notably RS-485/232 serial ports, digital and analogue inputs/outputs , all of which are widely used in the automation industry. The design allows the use of the Compute Module in harsh industrial environments, leading to the conclusion that the Raspberry Pi is no longer limited to home and science projects, but can be widely used as an Industrial IoT solution and achieve goals of Industry 4.0. [5]

3.2.2 Wi-Fi module (ESP8266)

The ESP8266 is a low-cost Wi-Fi microchip with full TCP/IP stack and microcontroller capability produced by manufacturer Espressif Systems in Shanghai , China . The chip first came to the attention of Western makers in August 2014 with the ESP-01 module, made by a third-party manufacturer Ai-Thinker. This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands.

It has a Processor L106 32-bit RISC microprocessor core based on the Tensilica Xtensa Diamond Standard 106Micro running at 80 MHz , a memory of 32 KB , instruction RAM 32 KB , instruction cache RAM 80 KiB , user-data RAM 16 KB , flash up to 16 MB , IEEE 802.11 Wi-Fi , Integrated TR switch , power amplifier and matching network WEP or WPA/WPA2 authentication . the figures below shows the shape and pinout of the ESP.



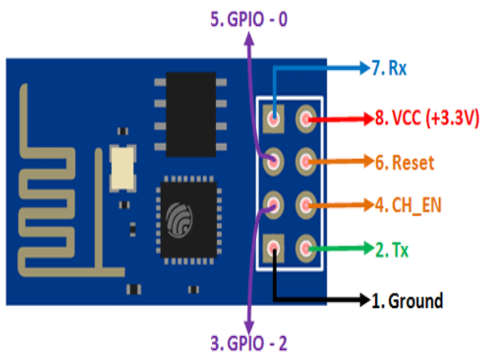


Figure 3.2 : The ESP8266–01 Figure 3.3 : ESP8266-01 Pin out .

3.2.3 The NodeMCU

is a low-cost open source IoT platform. It initially included firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which was based on the ESP-12 module, figure 3.4 shows the node MCU. The name "NodeMCU" combines "node" and "MCU" (micro-controller unit). The term "NodeMCU" strictly speaking refers to the firmware rather than the associated development kits . The prototyping hardware typically used is a circuit board functioning as a dual in-line package (DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. widely used in IoT applications .

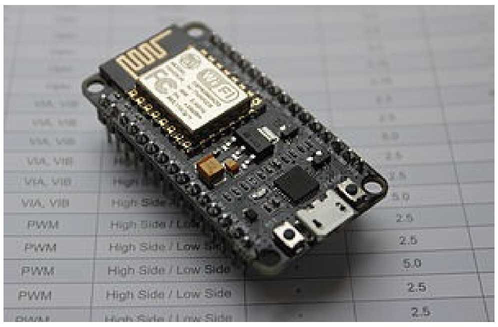


Figure 3.4 : The NodeMCU – ESP-12

As "Arduino.cc" began developing new MCU boards based on non-AVR processors like the ARM/SAM MCU and used in the Arduino Due, they needed to modify the Arduino IDE so that it would be relatively easy to change the IDE to support alternate toolchains to allow Arduino C/C++ to be compiled for these new processors. [5]

3.2.4 Motion Detector

A motion detector is a device that detects moving objects, particularly people. Such a device is often integrated as a component of a system that automatically performs a task or alerts a user of motion in an area. They form a vital component of security, automated lighting control, home control, energy efficiency and other useful systems. An electronic motion detector contains an optical, microwave, or acoustic sensor, and in many cases a transmitter for illumination.. Most low-cost motion detectors can detect up to distances of at least 15 feet (4.6 m). Specialized systems cost more, but have much longer ranges. Tomographic motion detection systems can cover much larger areas because the radio waves are at frequencies which penetrate most walls and obstructions, and are detected in multiple locations, not only at the location of the transmitter. Such a detector may also trigger a security camera to record the possible intrusion .

- **Infrared Sensor** : is an electronic device, that emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detects the motion. These types of sensors measure only infrared radiation, rather than emitting it that is called a passive IR sensor. Usually, in the infrared spectrum, all the objects radiate some form of thermal radiation. These types of radiations are invisible to our eyes, that can be detected by an infrared sensor. The emitter is simply an IR LED (Light Emitting Diode) and the detector is simply an IR photodiode that is sensitive to IR light of the same wavelength as that emitted by the IR LED. When IR light falls on the photodiode, the resistances and the output voltages will change in proportion to the magnitude of the IR light received. [9]

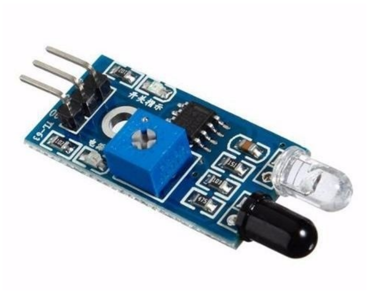


Figure 3.5 : Infrared sensor chip

3.2.5 Smoke Detector

A smoke detector is a device that senses smoke, typically as an indicator of fire. Commercial security devices issue a signal to a fire alarm control panel as part of a fire alarm system, while household smoke detectors, also known as smoke alarms, generally issue a local audible or visual alarm from the detector itself or from a number of detectors if there are multiple smoke detectors interlinked . Smoke detectors are housed in plastic enclosures, typically shaped like a disk about 150 millimeters (6 in) in diameter and 25 millimeters (1 in) thick, but shape and size vary. Smoke can be detected either optically (photoelectric) or by physical process (ionization); detectors may use either, or both, methods. Sensitive alarms can be used to detect, and thus deter, smoking in areas where it is banned. Smoke detectors in large commercial, industrial, and residential buildings are usually powered by a central fire alarm system, which is powered by the building power with a battery backup. Domestic smoke detectors range from individual battery-powered units, to several interlinked mains-powered units with battery backup; with these interlinked units, if any unit detects smoke, all trigger even if household power has gone out.[5]

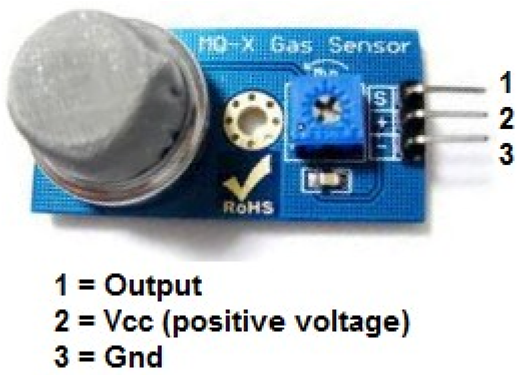


Figure 3.6 : Smock sensor chip

3.2.6 Wireless Router

A wireless router is a device that performs the functions of a router and also includes the functions of a wireless access point. It is used to provide access to the Internet or a private computer network. Depending on the manufacturer and model, it can function in a wired , wireless or mix of them in local area network LAN . The most common operating system on such embedded devices is Linux. More seldomly,. It is possible for a computer running a desktop operating system with appropriate software to act as a wireless router. This is commonly referred to as a SoftAP.[5]

3.2.7 IC L293D motor driver

The L293D are quadruple high-current half-H drivers. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

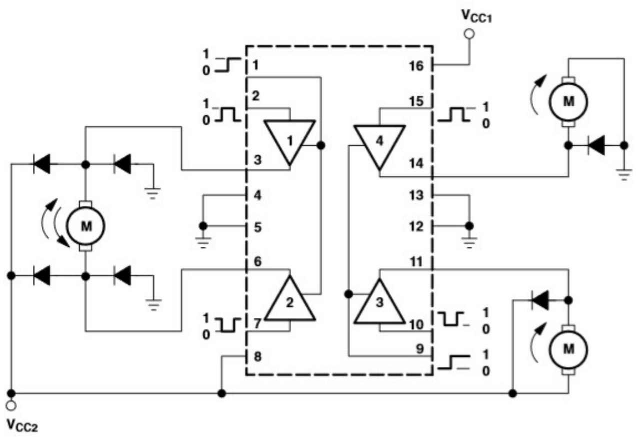


Figure 3.7 : L293D pinout and interior connection

3.2.8 Servo Motor

A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors. Servomotors are not a specific class of motor, although the term servomotor is often used to refer to a motor suitable for use in a closed-loop control system.

Figure 3.8 : Servo Motor

Servomotors are used in applications such as robotics, CNC machinery or automated manufacturing . A servomotor is a closed-loop servomechanism that uses position feedback to control its motion and final position. The input to its control is a signal (either analogue or digital) representing the position commanded for the output shaft. The motor is paired with some type of position encoder to provide position and speed feedback

3.2.9 Other Components

Other components used in this project are : wires , resistors , LEDs , buzzer , mini breadboard , DC motor and DC 5v fan .

3.3 Programming Languages and remote communication protocols

The software component of the platform represented by the programming languages and communication protocols are as follows .

3.3.1 Python programming language

Python is a general-purpose, high-level language that can be extended and embedded (included in applications as a tool for writing macros). That makes Python a smart choice for many programming problems, both small and large , Python is ideal for projects that require quick development. It supports multiple programming philosophies, so it's good for programs that require flexibility. The many packages and modules already written for Python provide versatility and save you time. Python also has many features and some of them is:

- Scripting language: A script is a program that controls other programs. Scripting languages are good for quick development and prototyping because they're good at passing messages.
- Indentation for statement grouping: Python specifies that several statements are part of a single group by indenting them. The indented group is called a code block High-level data types: Computers store everything in 1s and 0s, but humans need to work with data in more complex forms, such as text. A language that supports such complex data is said to have high-level data types. A high-level data type is easy to manipulate. For example, Python strings can be searched, sliced, joined, split, set to upper- or lowercase, or have white space removed. High-level data types in Python, such as lists and dictionaries (which

can store other data types), encompass much more functionality than in other languages.

- Extensibility: An extensible programming language can be added to.
- Interpreted: Interpreted languages run directly from source code that humans generate. [4]

3.3.2 Arduino C and Arduino IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them. Programs written using Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and are saved with the file extension .ino. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right-hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor. The Arduino code is actually just plain old c without all the header part (the includes and all). when you press the 'compile' button, the IDE saves the current file as arduino.c in the 'lib/build' directory then it calls a makefile contained in the 'lib' directory. This makefile copies arduino.c as prog.c into 'lib/tmp' adding 'wiringlite.inc' as the beginning of it. this operation makes the Arduino/wiring code into a proper c file (called prog.c.The core files are supported by pascal Stangs Procyon avr-lib that is contained in the 'lib/avrlib' directory .At this point the code contained in lib/tmp is ready to be compiled with the c compiler contained in 'tools'. If the make operation is successful then you'll have prog.hex ready to be downloaded into the processor. [7]

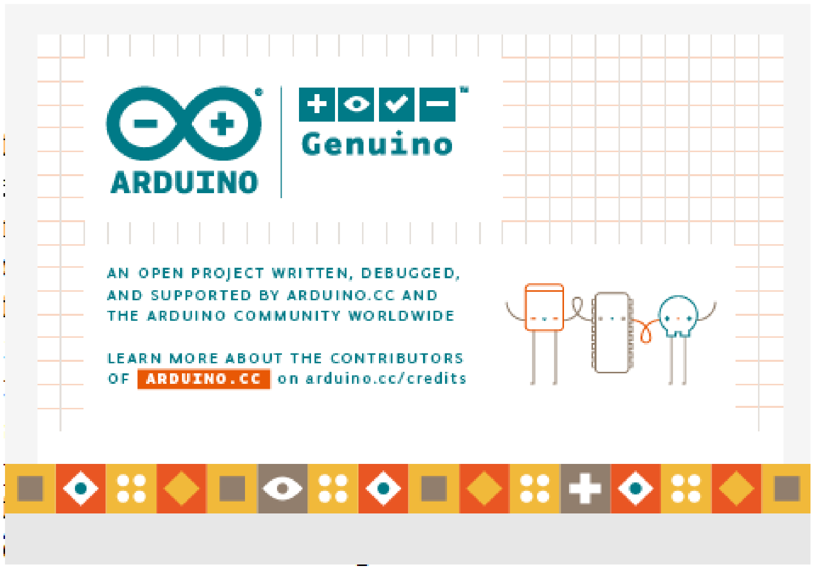

ARDUINO
Genuino
AN OPEN PROJECT WRITTEN, DEBUGGED, AND SUPPORTED BY ARDUINO.CC AND THE ARDUINO COMMUNITY WORLDWIDE
LEARN MORE ABOUT THE CONTRIBUTORS OF ARDUINO.CC on arduino.cc/credits

Figure 3.9 : Arduino Genuino IDE

3.3.3 Hypertext Markup Language (HTML)

HTML is the most basic building block of the Web. It defines the meaning and structure of web content. Other technologies besides HTML are generally used to describe a web page's appearance/presentation (CSS) or functionality/ behavior (JavaScript) . "Hypertext" refers to links that connect web pages to one another, either within a single website or between websites. Links are a fundamental aspect of the Web. By uploading content to the Internet and linking it to pages created by other people, you become an active participant in the World Wide Web. HTML uses "markup" to annotate text, images, and other content for display in a Web browser. HTML markup includes special "elements" such as <head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, <div>, <span>, <img>, <aside>, <details>, <nav>, <ul>, <ol>, <li> and many others.[11]

3.3.4 Cascading Style Sheets (CSS)

CSS is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript . CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file which reduces complexity and repetition in the structural content as well as enabling the .css file to be cached to improve the page load speed between the pages that share the file and its formatting. Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.[5]

3.3.5 The HTTP Protocol

HTTP stands for Hyper Text Transfer Protocol is a simple request-response protocol that normally runs over TCP. It specifies what messages clients may send to servers and what responses they get back in return. The request and response headers are given in ASCII. The contents are given in a MIME-like format. This simple model was partly responsible for the early success of the Web because it made development and deployment straightforward.

HTTP is an application layer protocol because it runs on top of TCP and is closely associated with the Web. However, in another sense HTTP is becoming more like a transport protocol that provides a way for processes to communicate content across the boundaries of different networks. These processes do not have to be a Web browser and Web server. Developers could use HTTP to fetch project files. Consumer electronics products like digital photo frames often use an embedded HTTP server as an interface to the outside world. Machine-to-machine communication increasingly runs over HTTP.

The usual way for a browser to contact a server is to establish a TCP connection to port 80 on the server's machine, although this procedure is not formally required. The value of using TCP is that neither browsers nor servers have to worry about how to handle long messages, reliability, or congestion control. All of these matters are handled by the TCP implementation. Although HTTP was designed for use in the Web, it was intentionally made more general than necessary with an eye to future object-oriented uses. For this reason, operations, called methods, other than just requesting a Web page are supported. [3]

3.3.6 The Message Queuing Telemetry Transfer (MQTT)

MQTT stands for MQ Telemetry Transport. It is a publish / subscribe , extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks. The design principles are to minimize network bandwidth and device resource requirements whilst also attempting to ensure reliability and some degree of assurance of delivery. These principles also turn out to make the protocol ideal of the emerging "machine-to-machine" (M2M) or "Internet of Things" world of connected devices, and for mobile applications where bandwidth and battery power are at a premium. MQTT was invented by Dr.Andy Stanford-Clark of IBM, and Arlen Nipper of Arcom (now Eurotech), in 1999 and it has been widely implemented across a variety of industries since 1999.

You can pass a user name and password with an MQTT packet in V3.1 of the protocol. Encryption across the network can be handled with SSL, independently of the MQTT protocol itself (it is worth noting that SSL is not the lightest of protocols, and does add significant network overhead).

Additional security can be added by an application encrypting data that it sends and receives, but this is not something built-in to the protocol, in order to keep it simple and lightweight. [6]

3.4 Block Diagram Of The System

The figure below shows the block diagram of the platform including its software part .

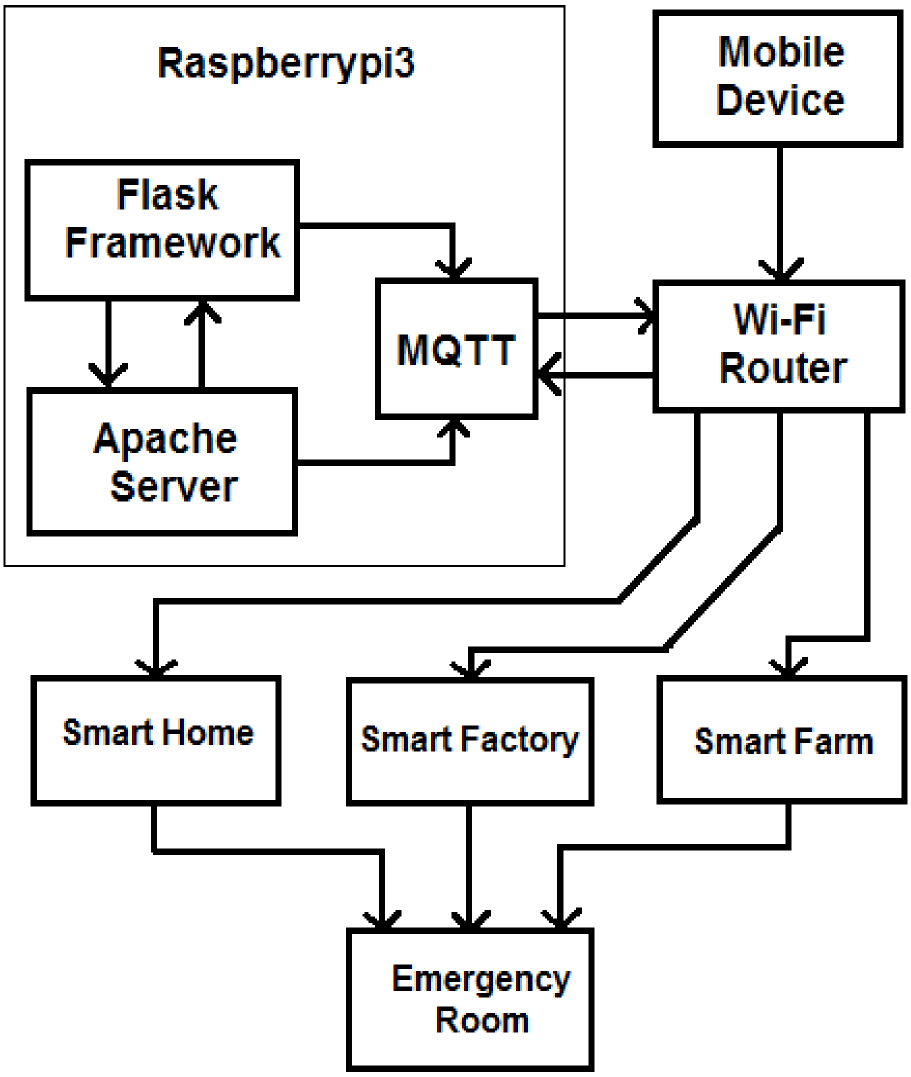


Figure 3.10 : Block Diagram of The Platform

CHAPTER FOUR

CIRCUIT DESIGN, IMPLEMENTATION AND TESTING

4.1 Introduction

In this chapter we describe the procedure of installing and implementing the software and the hardware part of the system ,The structure of the platform consists of four facilities each is represented by circuit ,a network server (Router) and a MQTT server represented by the raspberrypi , The system heavily depends on the software part as the hardware part responds and actuate the orders going back and forth .

4.2.Building the circuits

Each facility in the platform has a circuit representing a certain functionality , The design of the circuits was made using Fritzin emulator for circuit designing , each design and implementation is described below .

4.2.1 Smart Home

For the smart home implementation we made a simple representation using an esp8266 as a controller and a Wi-Fi module , In the house a 5V fan resembles the Electric devices to be controlled in the house , and to operate the fan we used an L293D dc motor driver so the 3.3v esp8266 can control it with the help of a 5v adapter .

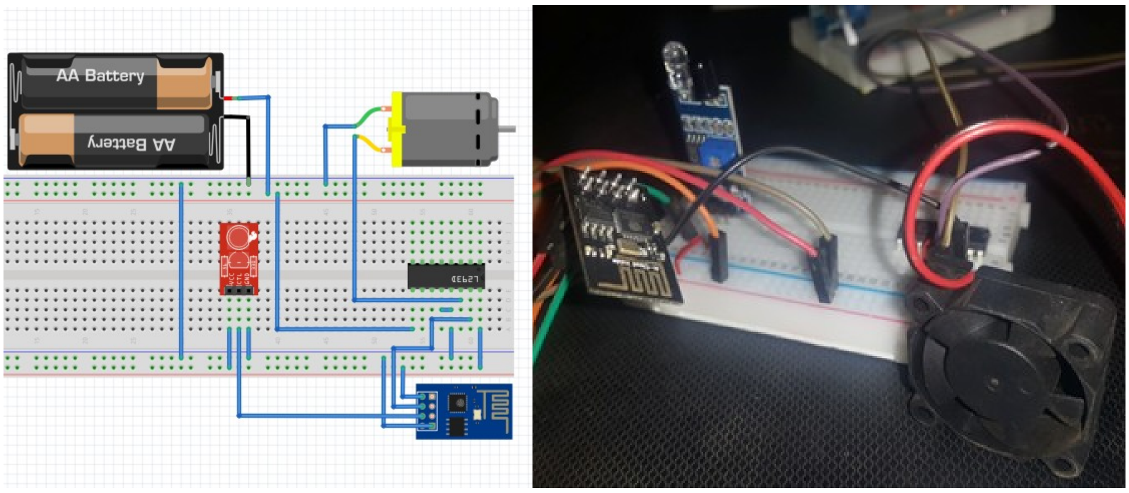


Figure 4.1 : The smart home circuit

An Infrared sensor chip is also connected to the esp8266 as an input to represent home security sensors against theft and burglary , so when a physical object cross in front of the sensor the esp will send an alert message directly to the emergency room through the MQTT server installed in the raspberrypi which will fire an alarm automatically .

4.2.2 Smart Manufacturing

In this case a dc motor represent factory machines which can be turned on and off remotely without risking the operators life ,The motor operate on 5v and the signal is sent by the esp8266 through the l293d .

We used a smock detector as an input to the esp ,So when the smock detector senses a gas leak or a smock the esp8266 will immediately send an alert message to the emergency room and the alarm will turn on ,All through the MQTT protocol .

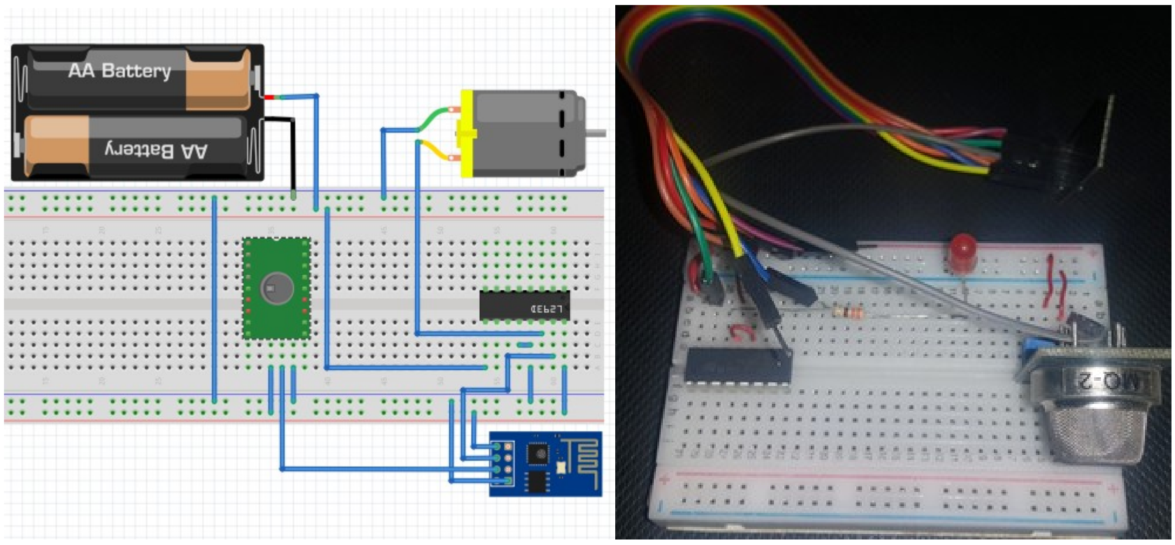


Figure 4.2 : The Smart Factory Circuit

4.2.3 Smart Farming (Irrigation)

In this model we used a servo motor to manifest and emulate the behavior of pivotal irrigation , the servo will start operating when a remote signal is sent from the mobile device through the internet using MQTT

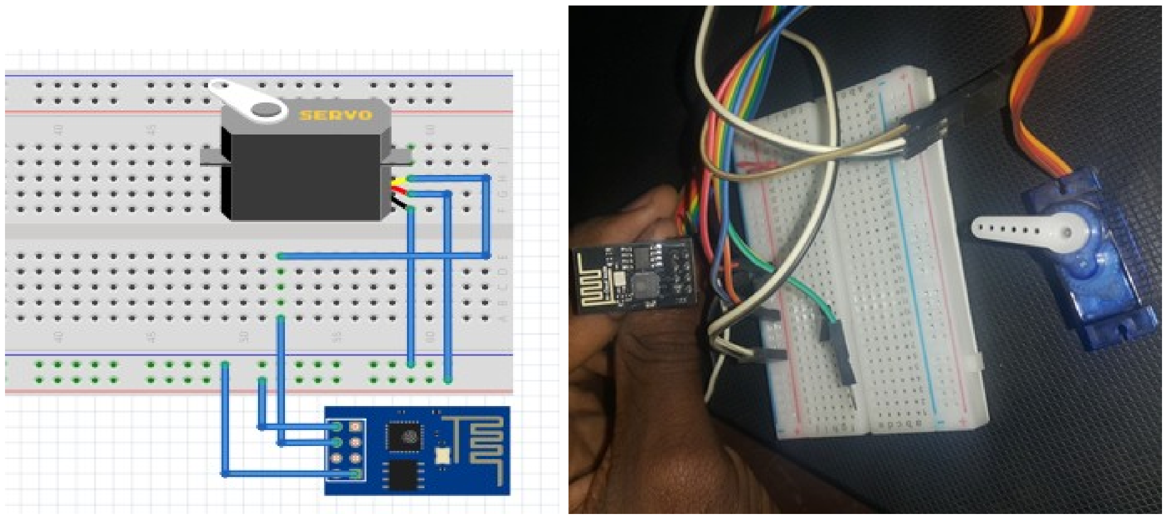


Figure 4.3 : The Smart Irrigation circuit

4.2.4 Emergency Room

This room is wirelessly connected to the three previous circuit through the MQTT and will automatically fire an alarm whenever a sensor in one of them is activated , the emergency room doesn't need human interference and can perform a machine to machine interaction .

To emulate the alarm action we used the NodeMCU which is a module based on the esp8266 and connected two LEDs and a buzzer , when the NodeMCU receive alert message the LEDs will turn on simultaneously and the buzzer will make alarm sound for a brief period of time.

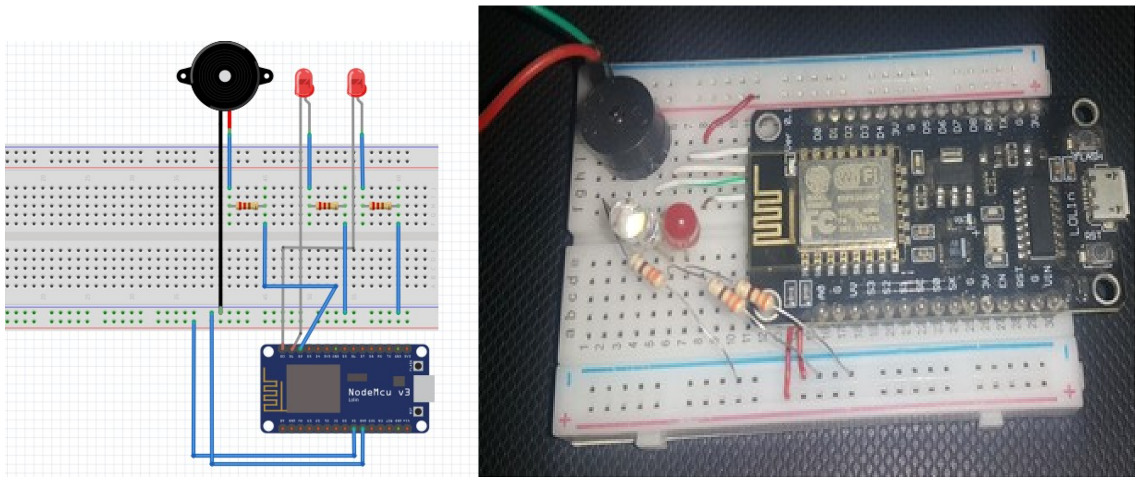


Figure 4.4 : The Emergency room circuit

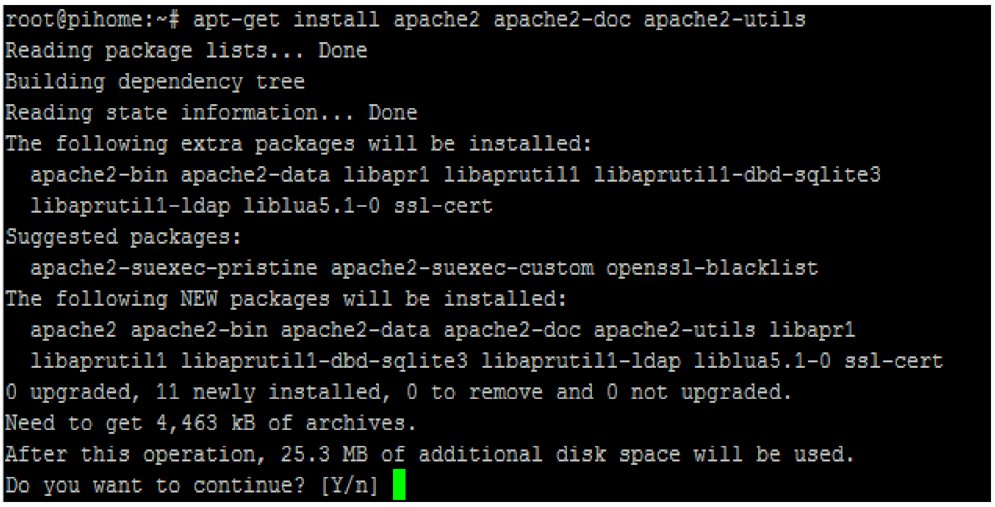
4.3 Installation of servers and framework

After designing and building the circuits comes the second most important part which is installing the servers and framework that manages the frontend and backend of the platform .

4.3.1 Server (APACHE2)

The Apache HTTP Server Project is a collaborative software development effort aimed at creating a robust, commercial-grade, featureful, and freely-available source code implementation of an HTTP (Web) server. The project is jointly managed by a group of volunteers located around the world, using the Internet and the Web to communicate, plan, and develop the server and its related documentation. This project is part of the Apache Software Foundation. In addition, hundreds of users have contributed ideas, code, and documentation to the project. [10]

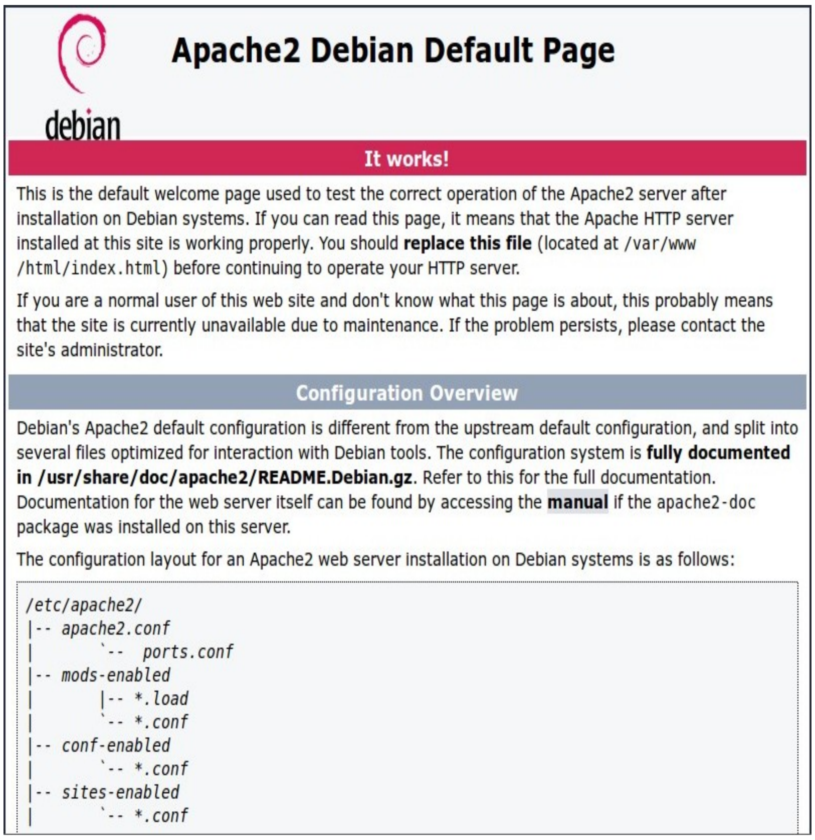
We installed Apache2 from the raspberrypi cloud repository using the command apt-get .



```
root@pihome:~# apt-get install apache2 apache2-doc apache2-utils
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  apache2-bin apache2-data libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap liblua5.1-0 ssl-cert
Suggested packages:
  apache2-suexec-pristine apache2-suexec-custom openssl-blacklist
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-doc apache2-utils libapr1
  libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.1-0 ssl-cert
0 upgraded, 11 newly installed, 0 to remove and 0 not upgraded.
Need to get 4,463 kB of archives.
After this operation, 25.3 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Figure 4.5 : Installation of Apache2 web server

And after clicking yes the installation is complete and apache2 web server is ready to be used , all it takes is entering the IP address of the raspberrypi in the web browser and a default webpage will appear showing that the installation is successfully completed .



Apache2 Debian Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at /var/www/html/index.html) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

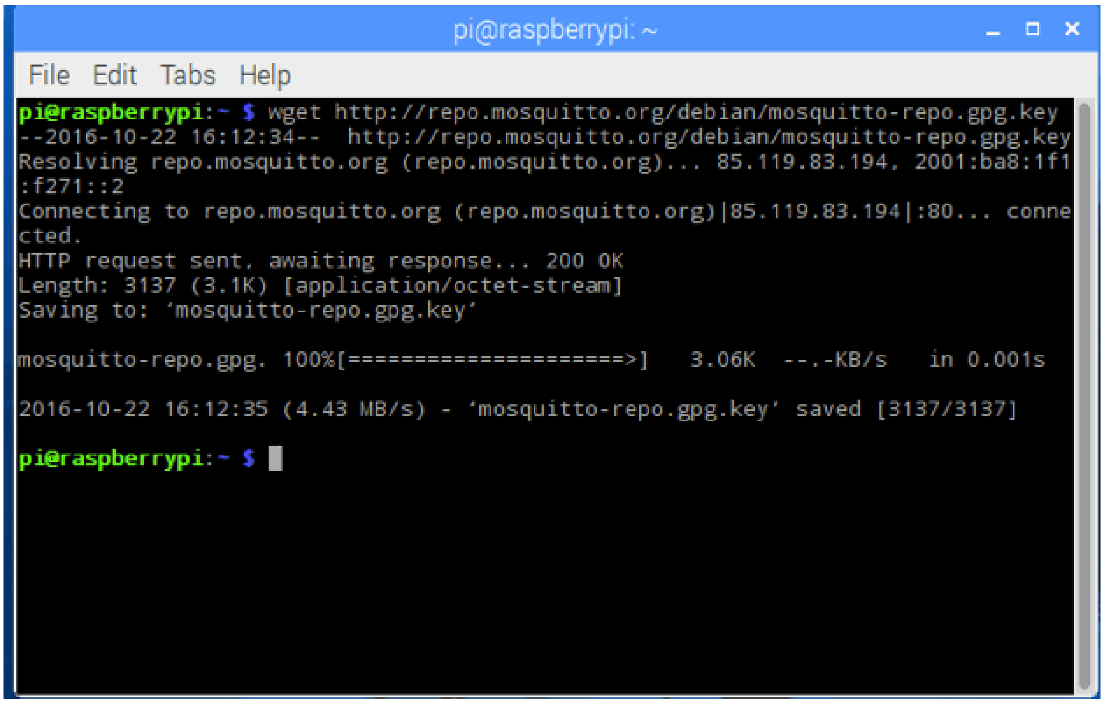
The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

Figure 4.6 : Test page showing completion of installation

4.3.2 MQTT Server Installation

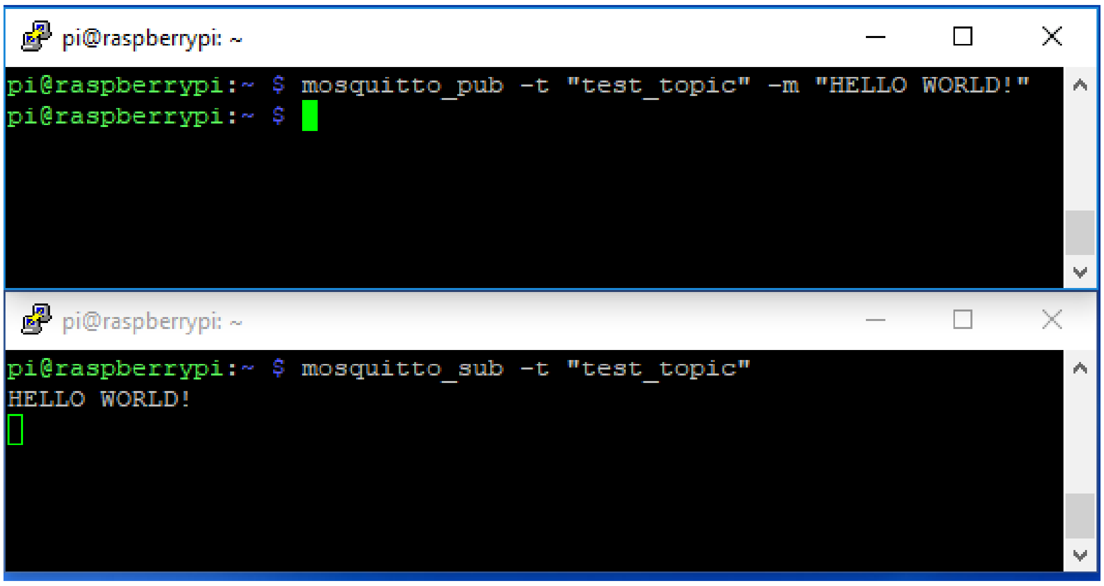
As mentioned earlier ,The MQTT server is responsible of the communication between all parts of the platform using the MQTT protocol of publishers and subscribers . There is a compatible version of the server for Debian which is the operating system of the raspberrypi , it can be downloaded from the official website mosquitto.org and easily installed.



```
pi@raspberrypi:~ $ wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
--2016-10-22 16:12:34--  http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
Resolving repo.mosquitto.org (repo.mosquitto.org)... 85.119.83.194, 2001:ba8:1f1
:f271::2
Connecting to repo.mosquitto.org (repo.mosquitto.org)|85.119.83.194|:80... conne
cted.
HTTP request sent, awaiting response... 200 OK
Length: 3137 (3.1K) [application/octet-stream]
Saving to: 'mosquitto-repo.gpg.key'

mosquitto-repo.gpg. 100%[===================>]   3.06K  --.-KB/s   in 0.001s

2016-10-22 16:12:35 (4.43 MB/s) - 'mosquitto-repo.gpg.key' saved [3137/3137]

pi@raspberrypi:~ $
```

Figure 4.7 : Installation of MQTT server

The figure below shows a test code in the raspberrypi shell to see if the server is working perfectly , and that's by sending a subscribe and publish message .

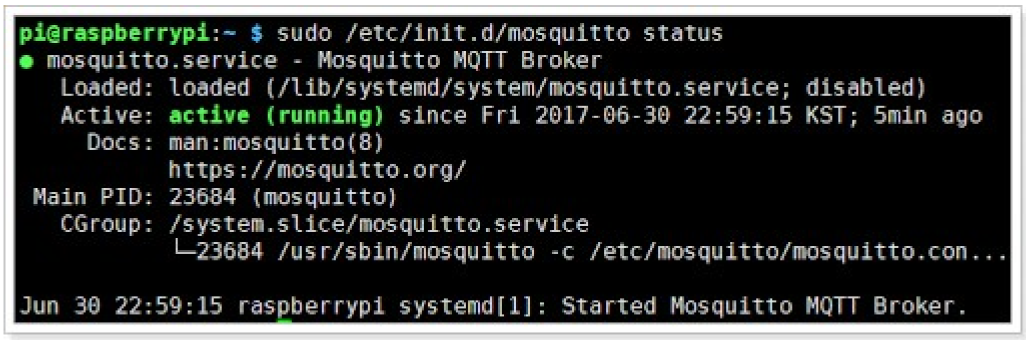


```
pi@raspberrypi:~ $ mosquitto_pub -t "test_topic" -m "HELLO WORLD!"
pi@raspberrypi:~ $
```

```
pi@raspberrypi:~ $ mosquitto_sub -t "test_topic"
HELLO WORLD!
```

Figure 4.8 : testing the MQTT server

Next figure shows a test code for MQTT status in the raspberrypi shell



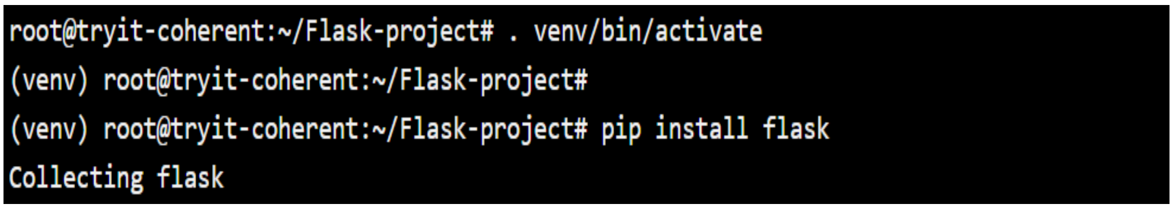
```
pi@raspberrypi:~ $ sudo /etc/init.d/mosquitto status
● mosquitto.service - Mosquitto MQTT Broker
   Loaded: loaded (/lib/systemd/system/mosquitto.service; disabled)
   Active: active (running) since Fri 2017-06-30 22:59:15 KST; 5min ago
     Docs: man:mosquitto(8)
           https://mosquitto.org/
 Main PID: 23684 (mosquitto)
   CGroup: /system.slice/mosquitto.service
           └─23684 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.con...

Jun 30 22:59:15 raspberrypi systemd[1]: Started Mosquitto MQTT Broker.
```

Figure 4.9 : Status test for the MQTT server

4.3.3 Framework (Flask)

Flask (source code) is a Python web framework built with a small core and easy-to-extend philosophy. Flask is considered more Pythonic than the Django web framework because in common situations the equivalent Flask web application is more explicit. Flask is also easy to get started with as a beginner because there is little boilerplate code for getting a simple app up and running. [8]

In our project Flask manages the graphical user interface that appears in the smart phone when requesting the webpage of the platform, And that's by integrating a HTML and CSS code into the main python code . The installation of Flask can easily be completed using pip which is a command in the raspberrypi shell .



```
root@tryit-coherent:~/Flask-project# . venv/bin/activate
(venv) root@tryit-coherent:~/Flask-project#
(venv) root@tryit-coherent:~/Flask-project# pip install flask
Collecting flask
```

Figure 4.10 : Installation of Flask framework

4.4 Testing the system

On any browser on a smart phone , by typing the IP address of the raspberrypi which has Apache2 server installed ,The default webpage under the name index will load , That HTML webpage is connected through flask with the main python program to activate MQTT orders according to the buttons pressed in the webpage .

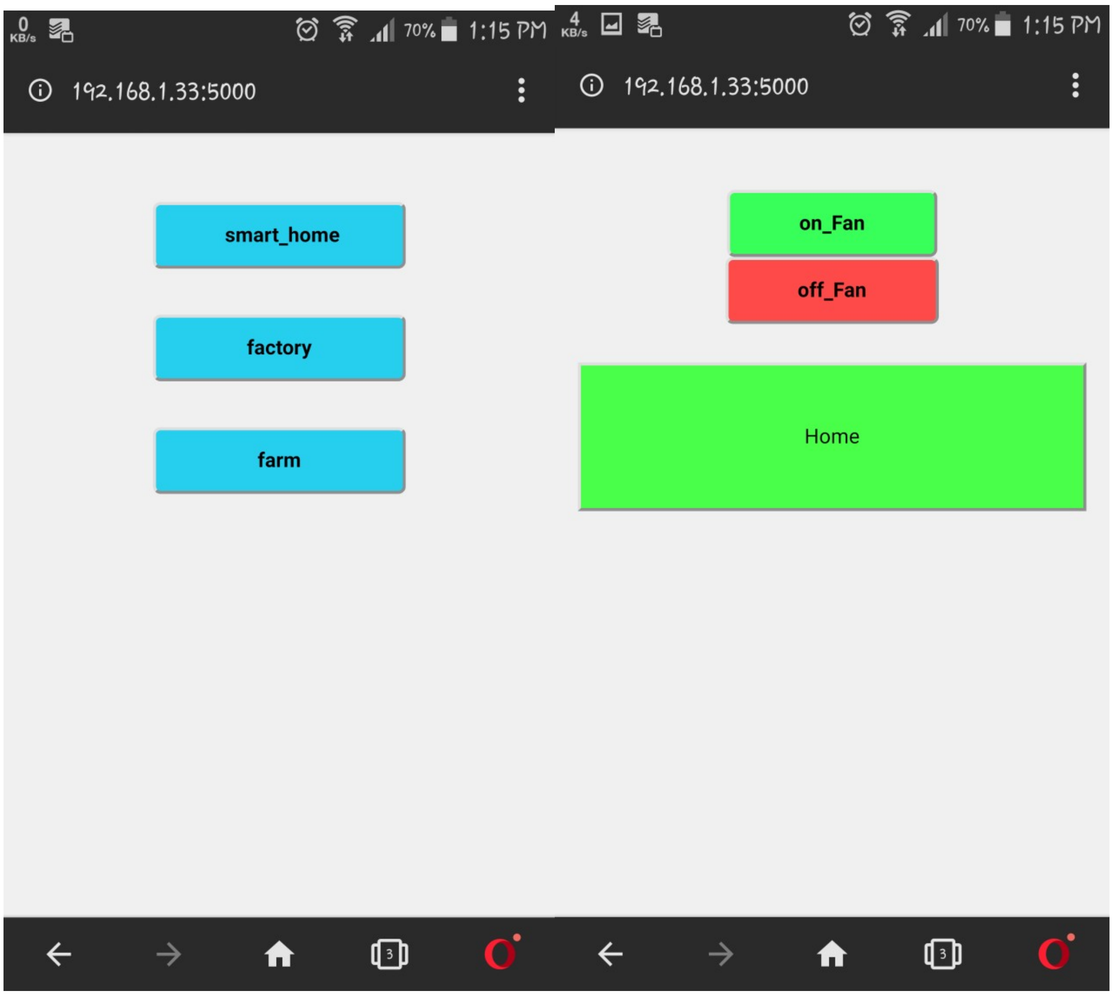
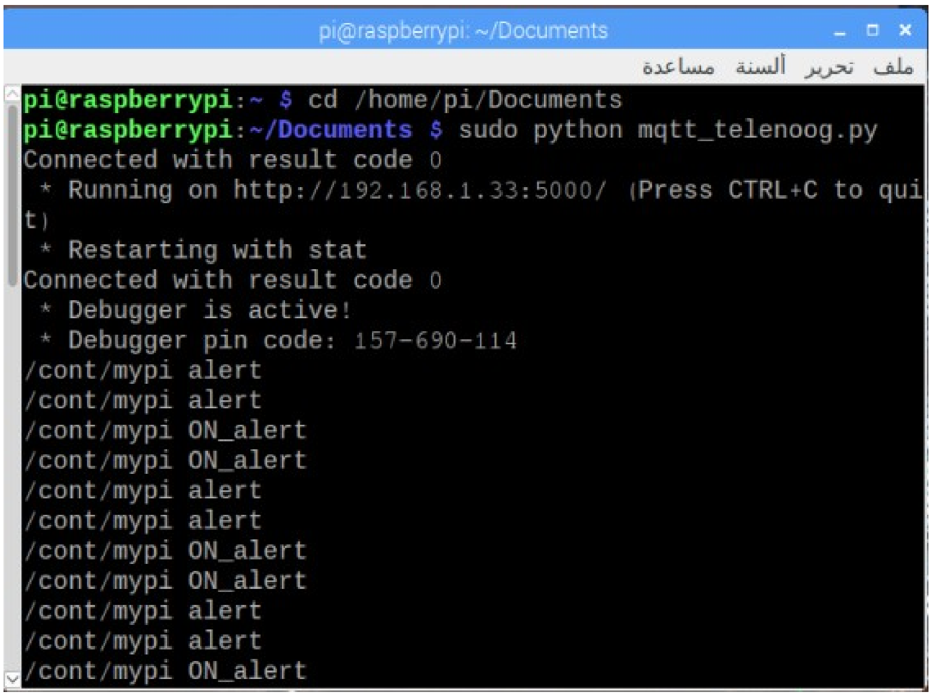


Figure 4.11 : The graphical user interface working perfectly

But first of all we need to run the python program and activate the MQTT server . the best way to do so is through the shell window of the raspberrypi . the nest figure shows the activation and testing of the program .



```
pi@raspberrypi:~ $ cd /home/pi/Documents
pi@raspberrypi:~/Documents $ sudo python mqtt_telenoog.py
Connected with result code 0
 * Running on http://192.168.1.33:5000/ (Press CTRL+C to qui
t)
 * Restarting with stat
Connected with result code 0
 * Debugger is active!
 * Debugger pin code: 157-690-114
/cont/mypi alert
/cont/mypi alert
/cont/mypi ON_alert
/cont/mypi ON_alert
/cont/mypi alert
/cont/mypi alert
/cont/mypi ON_alert
/cont/mypi ON_alert
/cont/mypi alert
/cont/mypi alert
/cont/mypi ON_alert
```

Figure 4.12 : MQTT server at work

Next step is making sure that the circuits is responding to the commands written in the code . figure 4.13 below shows the activation of the alarm after receiving alert signal through the MQTT from one of the esp controllers after a sensor detected something .

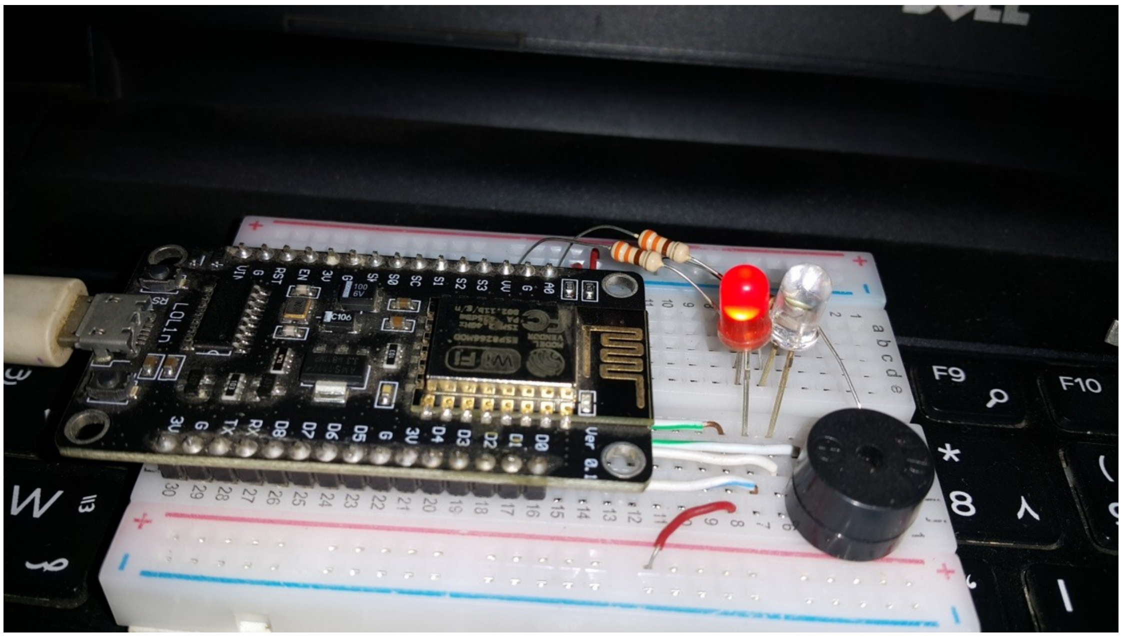


Figure 4.13 : Emergency room receiving alert

The previous test shows that the machine to machine communication is working fine through MQTT ,and now comes testing a direct command from the smart phone to a device connected to one of the esp's , the figure below shows operating a dc motor representing a factory machine from the platform through IoT .

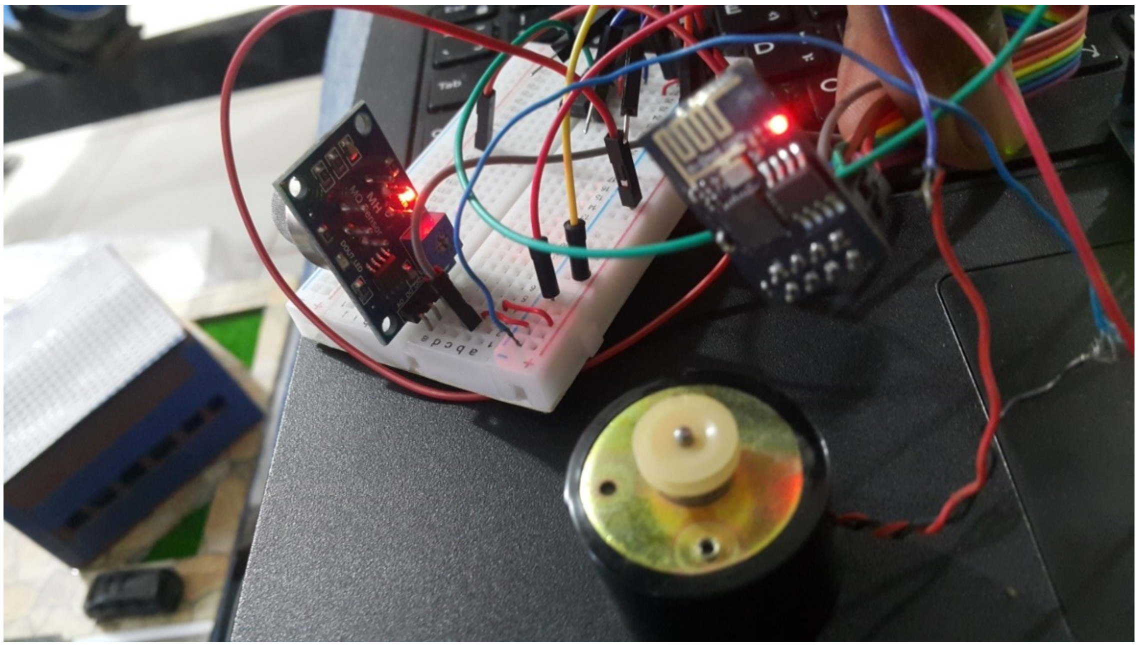


Figure 4.14 : Motor working after receiving order

The next figure shows the servo motor representing pivotal irrigation , the motor made a cycle after receiving orders from the smart phone . and with that the testing is complete .

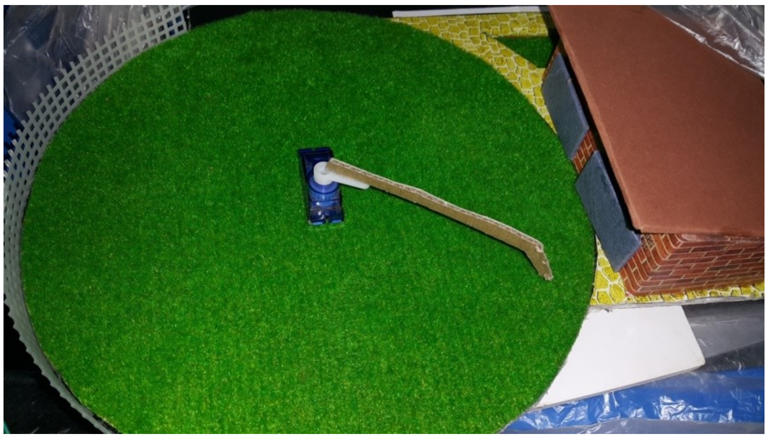


Figure 4.15 : Servo motor acting as pivot irrigation

Figure 4.16 shows the facilities in which the circuits are installed to represent the platform , we used an arduino uno as a power supply to provide 5v and 3.3v to each circuit .

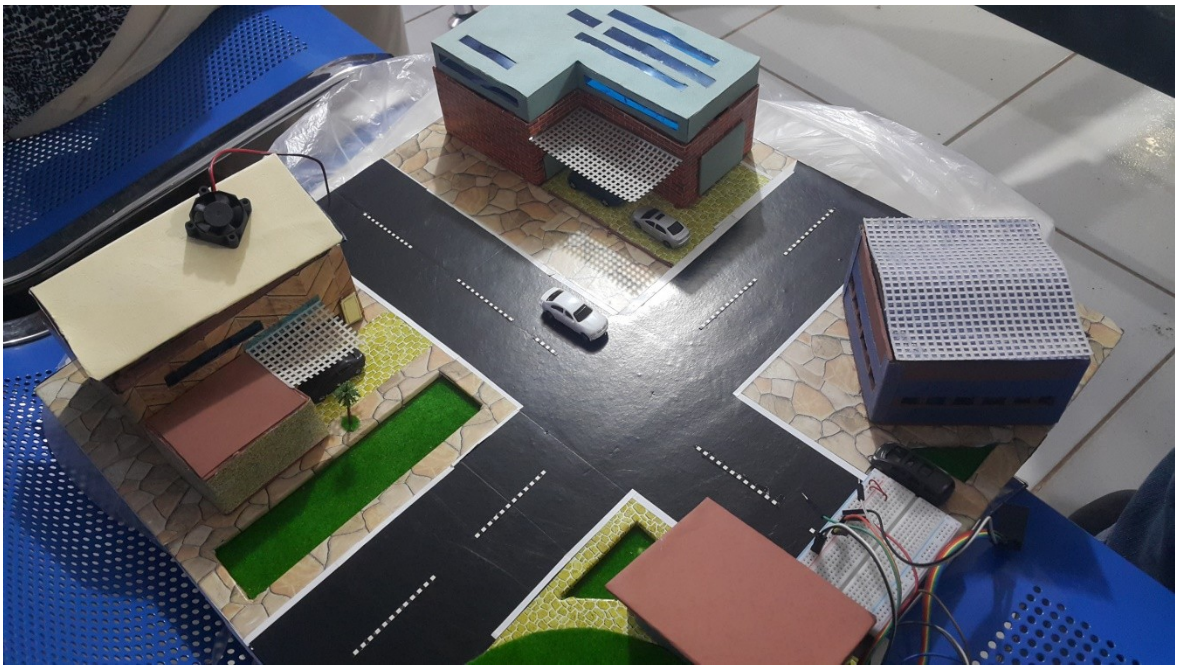


Figure 4.16 : The platform embedded within the facilities

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The Internet of Things have shown the mesmerizing potentials of embedding microcontrollers in everyday devices and providing them with wireless connectivity enabling them to perform various applications that would make human life not only easier but also more save and productive .

After a thorough research of available and easily obtainable techniques we could design and build a platform based on IoT showing that not only the implementation of IoT can easily be done ,but that the integration between those parts can provide efficiency through connectivity .and with such cheap parts like those used in this project as shown we could improve smart homes to not only work locally but also through the internet if the proper Equipments are present . In agriculture on the other hand ; irrigation is made easy ,less consumptive and more efficient after adding connectivity.

In this project we also proved the efficiency of adding sensors to various types of facilities ,that can provide security measures and safety and that's also by using wireless connectivity ,Thus the IoT platform can provide not only remote control, but more efficiency ,reduction of power consumption ,safety and security .

5.2 Recommendations

In this project we implemented the most basic pieces of IoT and there is room for improvements like :

- Increasing information security by using https and password secured MQTT or other methods of encryption .
- Use a server with larger memory to be able to create databases for multiple clients ,and Get permission from the international telecommunication company for a static IP address to enable using the actual internet instead of a LAN .
- The esp8266 has only two GPIO so a microcontroller with more of them without increasing the size would be better, besides a PCB would save much space and increase the efficiency .
- Another useful applications can be implemented like smart healthcare , and using IoT to reduce electrical consumption .

References

[1] B.K Tripathy & J.Anuradha ,"Internet of Things (IoT) ", CRC press 2018.

[2] Tim Cox ,"Raspberry Pi Cookbook for Python Programmers ", PACKT Publishing , 2014 .

[3] ANDREW S. TANENBAUM ,"Computer Networks 5th edition" , University of Washington Seattle , 2011 .

[4] Stef Maruch and Aahz Maruch ,"Python For Dummies" , John Wiley & Sons 2006 .

[5] Wikipedia.com

[6] www.MQTT.org .

[7] www.arduino.cc/en/main/documentation.

[8] www.FullStackPython.com/Flask.

[9] www.elprocus.com/infrared-ir-sensor-circuit-and-working/

[10] www.apache.org/about.

[11] https://developer.mozilla.org/en-US/docs/Web/HTML

[12] Journal of Electrical and computer Engineering, Jan 2017.

[13] D. Cuff ,M. Hansen and J.Kang, "Urban sensing: out of the woods", communications of the ACM , 2014.

[14] Gunther Griding and Bettina Weiss , "Introduction to microcontrollers " , Vienna university 2007.

Appendix A

Python Code

```
from flask import Flask , render_template ,request
import paho.mqtt.client as mqtt


#functions will be called when MQTT events happen like
# connecting to the server or receiving data from a subscribed feed.
def on_connect(client, userdata, flags, rc):
        # Connection renewed when lost.
        print("Connected with result code " + str(rc))
        client.subscribe("/cont/mypi")
# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
        print(msg.topic+" "+str( msg.payload))
        # Check if this is a message for the Pi.
        if msg.topic == '/cont/mypi':
# Create MQTT client and connect to localhost "Raspberry Pi"
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect('localhost', 1883, 60)
# Connect to the MQTT server and process messages in a background thread.
client.loop_start()


app = Flask(__name__)
@app.route('/',methods=['GET','POST'])
def home():
        if request.method =="POST":
                if 'smart_home' in request.form:
                        return render_template("smart_home.html")
                elif 'farm' in request.form:
                        return render_template("farm.html")
                elif 'factory' in request.form:
                        return render_template("factory.html")
        elif request.method =="GET":
                return render_template("home1.html")


@app.route('/smart_home',methods=['GET','POST'])
def smh():
        if request.method =="POST":
```

```
        if 'on_Fan' in request.form:
            client.publish('/cont/esp_SH', 'ON_fan')
            return render_template("smart_home.html")
        elif 'off_Fan' in request.form:
            client.publish('/cont/esp_SH', 'OFF_fan')
            return render_template("smart_home.html")
        elif 'Home' in request.form:
            return render_template("home1.html")
    elif request.method =="GET":
        return render_template("smart_home.html")
@app.route('/farm',methods=['GET','POST'])
def farm():
    if request.method =="POST":
        if 'on_Irr' in request.form:
            client.publish('/cont/esp_farm', 'ON_Irr')
            return render_template("farm.html")
        elif 'off_Irr' in request.form:
            client.publish('/cont/esp_farm', 'OFF_Irr')
            return render_template("farm.html")
        elif 'Home' in request.form:
            return render_template("home1.html")
        elif request.method =="GET":
                return render_template("farm.html")
@app.route('/factory',methods=['GET','POST'])
def factory():
        if request.method =="POST":
                if 'on_machine' in request.form:
                        client.publish('/cont/esp_factory', 'ON_machine')
                        return render_template("factory.html")
                elif 'off_machine' in request.form:
                        client.publish('/cont/esp_factory', 'OFF_machine')
                        return render_template("factory.html")
                elif 'Home' in request.form:
                        return render_template("home1.html")
        elif request.method =="GET":
                return render_template("factory.html")
if __name__=='__main__':
        app.run(debug=True,host="192.168.1.33")
# Main loop to listen for button presses.
print('Script is running, press Ctrl-C to quit...')
```

Appendix B

Arduino C Code

```
#include <Servo.h>

#include <ESP8266WiFi.h>

#include <Adafruit_MQTT.h>

#include <Adafruit_MQTT_Client.h>

const char* ssid = "Name of the local network" ;

const char* password = "password" ;

#define MQTT_USERNAME   ""

#define MQTT_PASSWORD   ""

#define MQTT_SERVER   "192.168.1.33" // static ip address

#define MQTT_PORT     1883

WiFiClient client;

Adafruit_MQTT_Client mqtt(&client, MQTT_SERVER, MQTT_PORT,
MQTT_USERNAME, MQTT_PASSWORD);

Adafruit_MQTT_Publish Alarm = Adafruit_MQTT_Publish(&mqtt,
MQTT_USERNAME "/cont/Alarm");

Adafruit_MQTT_Publish RPi = Adafruit_MQTT_Publish(&mqtt,
MQTT_USERNAME "/cont/mypi");

Adafruit_MQTT_Subscribe esp = Adafruit_MQTT_Subscribe(&mqtt,
MQTT_USERNAME "/cont/esp_farm");
```

```
void MQTT_connect();
void setup() {
 myservo.attach(2);  // attaches the servo on GIO2 to the servo object
 Serial.begin(9600);
 delay(10);
 WiFi.mode(WIFI_STA);
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) { delay(500); }
 // Setup MQTT subscription for esp feed.
 mqtt.subscribe(&esp);}
void loop() {
 MQTT_connect();
 // this is our 'wait for incoming subscription packets' busy subloop
 Adafruit_MQTT_Subscribe *subscription;
 while ((subscription = mqtt.readSubscription())) {
   if (subscription == &esp) {
     char *message = (char *)esp.lastread;
     // Check if the message was ON, OFF, or TOGGLE.
     if (strncmp(message, "ON", Length of message) == 0) {
      // Order to turn on device
 }
```

```
      if (strncmp(message, "OFF", Length of message) == 0) {
       // Order to turn off device
} }}}
void MQTT_connect() {   // Stop if already connected.
  if (mqtt.connected()){ return; }
  Serial.print("Connecting to MQTT... ");
  uint8_t retries = 3;
  while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
    Serial.println(mqtt.connectErrorString(ret));
    Serial.println("Retrying MQTT connection in 5 seconds...");
    mqtt.disconnect();
    delay(5000);  // wait 5 seconds
    retries--;
    if (retries == 0) {
      // basically die and wait for WDT to reset me
      while (1); } }
  Serial.print("MQTT connected");
  RPi.publish("connected");
}
```

Appendix C

HTML and CSS Code

```
<!DOCTYPE html lang="en">
<html>
        <head>
                <title></title>
                <meta name="viewport" content="width=device-width,initial-scale=1,minimum-scale=1"><link rel="stylesheet" href="{{url_for('static',filename='style.css')}}">
        </head>
        <body>
                <form action='/factory' method='POST' class='section'>
                        <input type="submit" name="on_machine" value="on_machine" class="button_on"><br>
                        <input type="submit" name="off_machine" value="off_machine" class="button_off">
                </form>
                <footer>
                        <form action='/factory' method='POST'>
                                <input type="submit" name="Home" value="Home" class="home_button"></form></footer></body></html>
```

CSS Code :

```
body{
        background-color : #efefef ;
        text-align:center ;
        padding : 15px ;
        margin : auto ;}
.home_but{
        text-align : center ;
        background-color : #22ccee ;
        padding : 12px 45px ;
        border-radius : 5px ;
        font-weight : bold ;
        margin-top : 30px ;
        height : 15% ;
        width : 50% ;}
.home_button{
        text-align: bottom ;
        background-color: #44ff44 ;
        height : 20% ;
        width: 100% ;
        margin: auto ;}
```

```
.button_on {
        text-align : center ;
        background-color : #33ff55 ;
        padding : 12px 45px ;
        border-radius : 5px ;
        font-weight : bold ;}
.button_on:hover {
        background-color : #22ee44 ;}
.button_off {
        text-align : center ;
        background-color : #ff4444 ;
        padding : 12px 45px ;
        border-radius : 5px ;
        font-weight : bold ;}
.button_off:hover {
        background-color : #ee3333 ;}
.section{
        background-color: #efefef ;
        border-radius: 5px ;
        padding: 15px ;
        margin: 10px ;}
```