IOT BASED HOME AUTOMATION AND SMART SECURITY SYSTEM

By

SATYAM ASOLKAR

SHUBHAM OTAVNEKAR

GAYATRI NAIK

PRATHAMESH JOSHI

NIKHIL BHOLNEKAR

M.SC. PART II ELECTRONICS

SCHOOL OF PHYSICAL AND APPLIED SCIENCE

GOA UNIVERSITY

2021 – 2022

IOT BASED HOME AUTOMATION AND SMART SECURITY SYSTEM

CERTIFICATE

This is to certify that the project entitled

"IOT BASED HOME AUTOMATION AND SMART SECURITY SYSTEM"

Is a record work done by

MR. SATYAM ASOLKAR

MR. SHUBHAM OTAVNEKAR

MISS. GAYATRI NAIK

MR. PRATHAMESH JOSHI

MR. NIKHIL BHOLNEKAR

M.Sc. Part II Electronics
For the year 2021 – 2022

The candidates themselves have worked on the project during the period of study under by guidance and to the best of my knowledge it has not previously formed the basis of award of any previous degree or diploma at Goa University or elsewhere.

H.O.D.	**Examiner**	**Project Guide**

IOT BASED HOME AUTOMATION AND SMART SECURITY SYSTEM

Master Thesis performed in Electronics

By

SATYAM ASOLKAR

SHUBHAM OTAVNEKAR

GAYATRI NAIK

PRATHAMESH JOSHI

NIKHIL BHOLNEKAR

GOA UNIVERSITY

2021 – 2022

DECLARATION

We, the students of Goa University's M.Sc. Electronics batch, hereby solemnly declare that this project report titled "IOT BASED HOME AUTOMATION AND SMART SECURITY SYSTEM" is a record of work that we have composed, and that this report has not been submitted anywhere else for the award of any diploma or degree to the best of our knowledge.

1. Satyam Asolkar _____
2. Shubham Otavnekar _____
3. Gayatri Naik _____
4. Prathamesh Joshi _____
5. Nikhil Bholnekar _____

ACKNOWLEDGEMENT

Expressing gratitude is a tough undertaking, and it is hard to do justice to everyone who contributed to the project. We would like to offer our heartfelt appreciation to everyone who gave us with direction, support, and inspiration during the course of this project; we are profoundly grateful to everyone.

We are grateful to everyone of the teaching faculty, particularly Prof. Rajendra S. Gad (Professor), Dr. Jivan S. Parab (Associate Professor), Dr. Marlon D. Sequeira (Assistant Professor), Dr. Narayan T. Vetrekar (Assistant Professor), and Dr. Aniketh A. Gaonkar (Assistant Professor), for their support, assistance, and direction on the project. The Registrar of Goa University for giving us with the necessary funding.

We also want to thank Mr. Vishant Malik (Lab. Technician) for his ongoing support and assistance in providing us with the necessary components for our project.

We would specially like to thank the non – teaching staff members William D'Souza (Lower Division Clerk), Pushpa Andrade (Multi – Tasking Staff) and Ashwini Velip (Multi – Tasking Staff) for their invaluable assistance and encouragement.

We are grateful to our family members for their spiritual support and encouragement during the process. Thank you to all of our colleagues and friends who offered their expertise, as well as those who volunteered to assist me with their skills.

And, of course, thanks to everyone we may have overlooked. All of the persons mentioned above are responsible for the success of our initiative.

ABSTRACT

This project creates a clever and cost-effective real-time home automation and smart security system. The primary goal of this project is to create a smart home device that can be used to control household appliances via the internet and Wi-Fi. It allows users to operate and monitor their devices from anywhere in the world using touch controls and voice commands in the Google Home or Amazon Alexa App. The gadget can be connected to an Android App via Wi-Fi, and the user will be able to operate the household appliances without the need for an internet connection. The user can also open his or her home door by scanning the fingerprint sensor on his or her mobile phone with the Android App. Intruder images with face detection notifications will be delivered in real time to the authenticated user's Telegram account and saved on the SD card inserted in the ESP32 Cam.

Table of Contents

INTRODUCTION

Chapter 1

1.1 IoT (Internet of Things)

The Internet of Things (IoT) is a network of physical items called "things" that are integrated with sensors, software, and other technologies that enable them to connect and share data with other devices and systems over the internet. These devices range from simple household items to big industrial machinery.

We are powerless without the Internet, which has become essential element of our social and educational lives. The Internet of Things (IoT) devices not only control but also monitor electronic, electrical, and mechanical systems used in many infrastructures. Because of advancements in automation technology, the demand for automated systems has skyrocketed. The tremendous growth in internet users has made the internet a way of life, and IoT is the most recent and rising internet technology.

IoT is the interconnection of embedded computer devices in everyday items via the internet, allowing them to send and receive data. It is a form of communication between multiple devices that requires no or little human intervention. It has the potential to reduce human effort. IoT analyses data retrieved from sensors and performs appropriate activities, saving human's time. People's confidence in their home activities and security led to the development of the Home Automation System.

Recent technological advancements that allow the use of wireless controlling environments such as Bluetooth and Wi-Fi have enabled various devices to connect with one another.

1.1.1 IoT Components

Here are four key components of an IoT system that explain how it works.

i) Sensors/Devices

To begin, sensors or devices help capture minute data from the surrounding environment. The complexity of the data collected might range from a simple temperature monitoring sensor to a comprehensive full video feed.

Multiple sensors on a device can collaborate to do more than just detect things. For example, our phone is a gadget with several sensors such as GPS, accelerometer, and camera, but it does more than just detect things.

The most fundamental stage will always be to choose and gather data from the surrounding environment, whether using a single sensor or several sensors.

Figure 1: IoT Components – Sensors and Devices [29]

ii) Connectivity

The collected data is subsequently sent to a cloud infrastructure, however it requires a medium for transit.

Sensors can be connected to the cloud by a number of communication and transport media, including as cellular networks, satellite networks, Wi-Fi, Bluetooth, wide-area networks (WAN), low power wide area networks, and others.

Each alternative has some requirements and trade-offs in terms of power consumption, range, and bandwidth. As a result, selecting the optimum connectivity choice in the IOT system is critical.

Figure 2: IoT Components – Connectivity [29]

iii) Data Processing

Once the data has been captured and sent to the cloud, the programme processes it.

This might be as simple as checking that the temperature reading on equipment such as air conditioners or heaters is within an acceptable

range. It may also be quite complex, such as using computer vision on video to recognize things (such as burglars in your home).

However, there may be times when a user input is necessary, such as when the temperature is too high or there is an intruder in your home. This is where the user enters the photo.

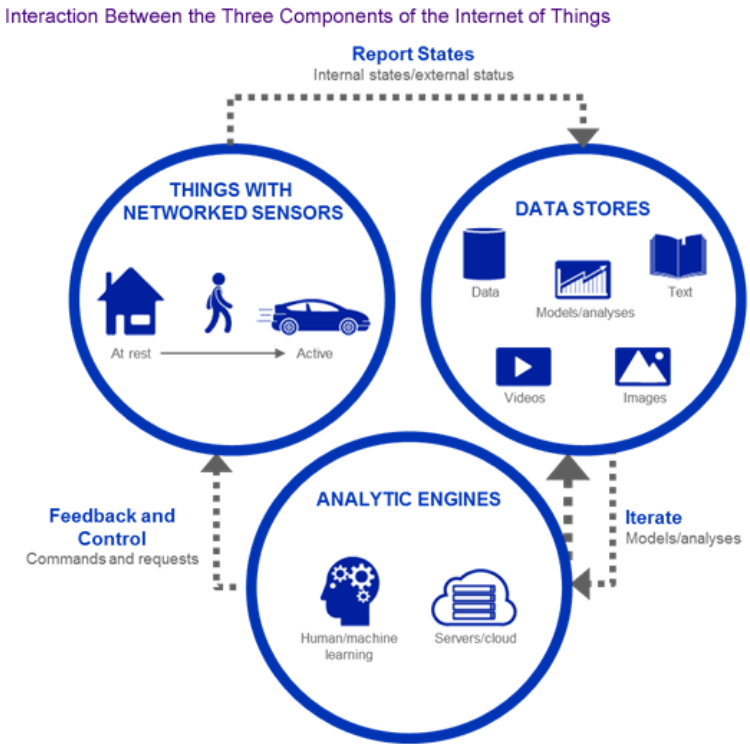


Figure 3: IoT Components – Data Processing [29]

iv) User Interface

The information is then made available to the end user in some manner. This may be accomplished by setting alarms on their phones or contacting them via SMS or emails.

A user may also have an interface via which they may actively monitor their IOT system. For example, if a user has a camera placed in his

home, he may wish to access the video recordings and all feeds via a web server.

However, it is not always this simple and one-way. Depending on the IoT application and system complexity, the user may potentially be able to conduct an action that might backfire and disrupt the system.

For example, if a user notices certain changes in the refrigerator, the user can alter the temperature remotely using their phone.

Specific activities are also performed automatically in some instances. The complete IOT system may alter the settings automatically by developing and applying some specified rules, and no person is required to be physically present.

In addition, if any intruders are detected, the system may send an alarm not only to the home's owner, but also to the appropriate authorities.

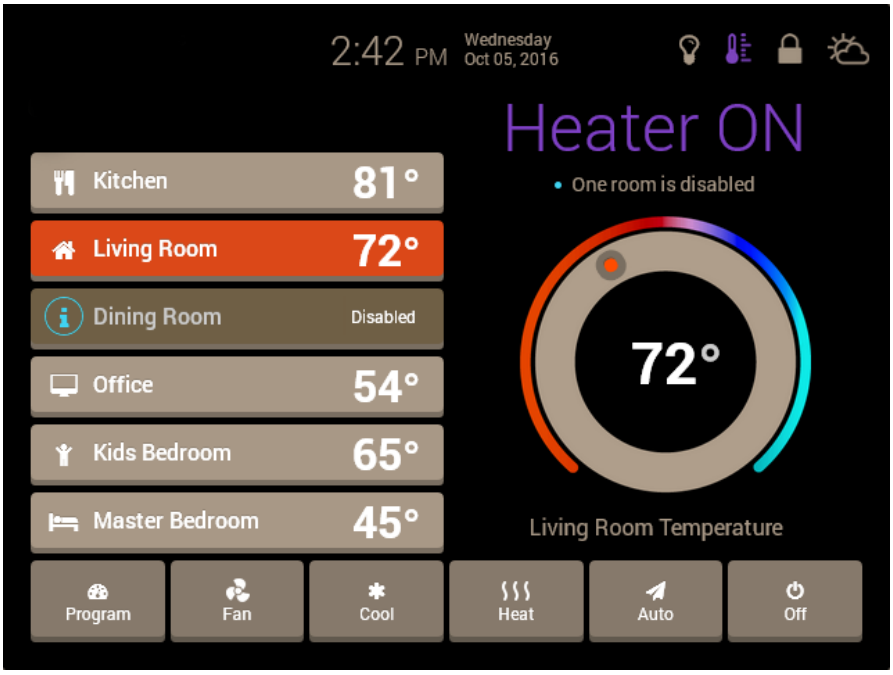


Figure 4: IoT Components – User Interface [29]

1.1.2 IoT Architecture Layers

Essentially, there are three IoT Architecture layers:

- IoT Device Layer (mainly the client-side)
- IoT Gateway Layer (server-side operations)
- IoT Platform Layer (acts as a bridge for connecting clients and operators).

The following are the major IoT layers that provide the solution for IoT Architecture:

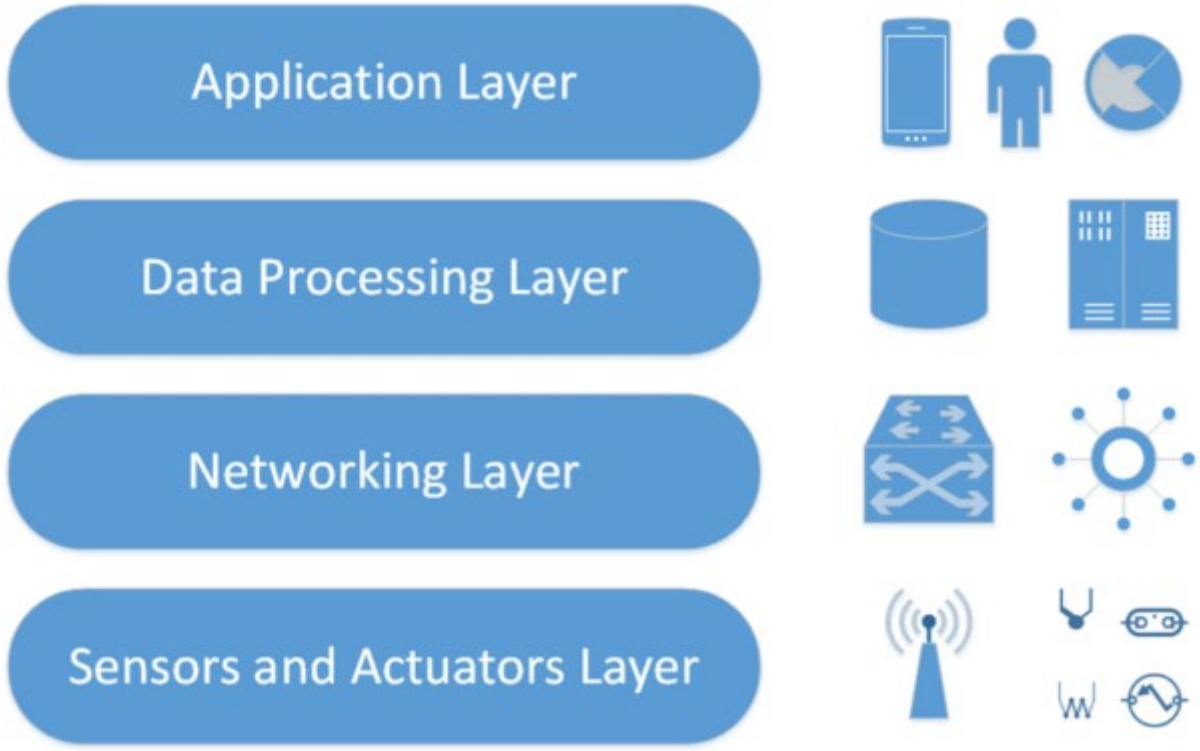


Figure 5: Architecture Layers of IoT [30]

1. **Application Layer**

   All IoT-enabled apps are included in the application layer. It acts as a link between end-user IoT devices and the network. Smart homes, smart health, smart cities, and other IoT applications. It is authorized to deliver services

to apps. Because of services based on data collected by sensors, services may change for each application. It is implemented at the device level via a specific application.

For a computer, the application layer is applied by the browser. The browser is responsible for executing application layer protocols such as HTTP, HTTPS, SMTP, and FTP. There are several problems on the application layer, the most important of which is security.

2. **Data Processing Layer**

Data were delivered directly to the network layer in a three-layer design. The risk of harm increases when data is sent directly. In a four-layer design, data from a perception layer is transferred to this layer. The Data Processing Layer is responsible for two tasks: ensuring that data is given by authorized users and protecting against threats.

The most common approach for confirming individuals and data is authentication. It is applied to the affected person through the use of pre-shared keys and passwords. The second purpose of the layer is to deliver data to the network layer. The data transmission medium from the Data Processing Layer to the network layer might be wireless or wire-based.

3. **Network Layer**

This layer is also known as the transmission layer. It serves as a bridge, transporting and sending data obtained from physical objects via sensors. The medium might be wireless or wired. It also connects network devices

and networks. As a result, it is particularly vulnerable to attacker assaults. It has significant security concerns regarding the integrity and authenticity of data delivered to the network.

4. **Perception Layer/Sensor Layer**

   The sensor layer is in charge of recognizing objects and collecting data from them. RFID, sensors, and 2-D barcodes are just a few of the sensors that may be attached to items to acquire information. The sensors are chosen to meet the needs of the applications. These sensors may collect information on their location, changes in the air, their surroundings, and so on. However, they are the prime target of attackers who want to utilize them to replace the sensor with their own.

1.1.3 Why should you use IoT?

Because the Internet of Things (IoT) allows items to be handled remotely over the internet, almost every sector might benefit from IoT devices. IoT devices have opened up new avenues for directly linking and integrating the physical world with machine platforms via sensors and the internet. IoT-enabled devices communicate with one another and aid in the creation of automation for applications.

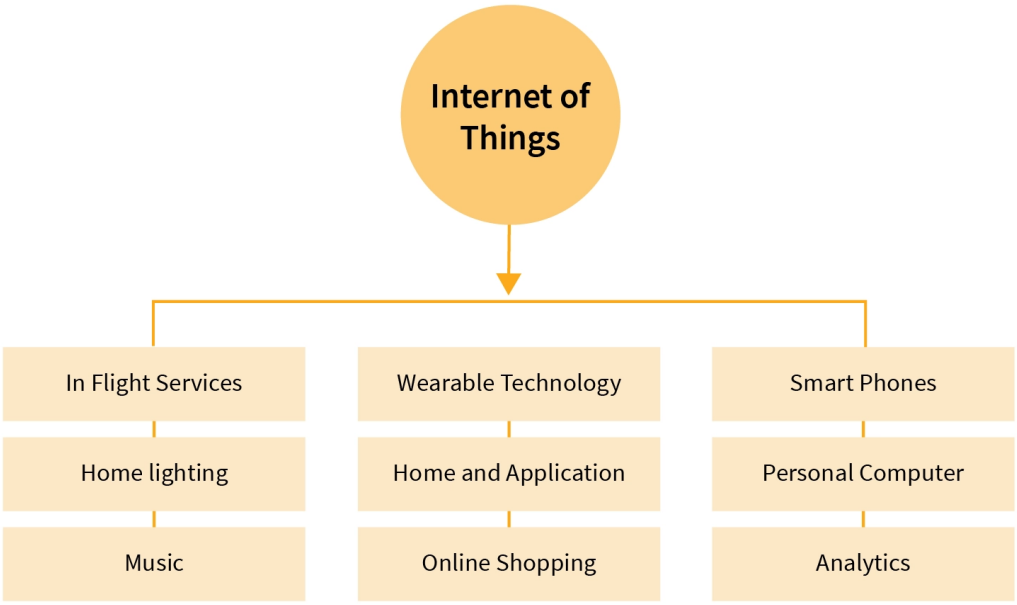


Figure 6: Internet of Things Applications [31]

We may automate the process by using IoT, which allows sensors to communicate data straight to a cloud server, where a number of operations can be run for significantly more efficient output after examination. Based on the outputs or outcomes, the cloud then performs predetermined activities. With less human involvement, accuracy, productivity, and financial value improve even more.

The following are some advantages of adopting IoT:

i) More data leads to improved decision making. Assume sensors are installed, and they collect a vast quantity of data from various situations, allowing it to make smart judgments in any new scenarios and circumstances.

ii) You have the ability to track and control all of your things and commodities.

iii) Found in healthcare management systems. Such is monitoring and managing the patient's heartbeats and blood pressure.

iv) Used in the construction of smart houses.

v) Found in wearables. As in wristwatch, GPS, heart rate monitoring, and other technologies.

1.1.4 The Future of IoT

The Internet of Things has evolved into a world-class technology. It has gained popularity in a relatively short period of time. Furthermore, advancements in Artificial Intelligence and Machine Learning have eased IoT device automation. To enable optimal automation, AI and ML algorithms are fundamentally connected with IoT devices. As a result, IoT applications have expanded across a wide range of sectors. This section will go through the uses and future reach of IoT in the healthcare, automotive, and agriculture industries.

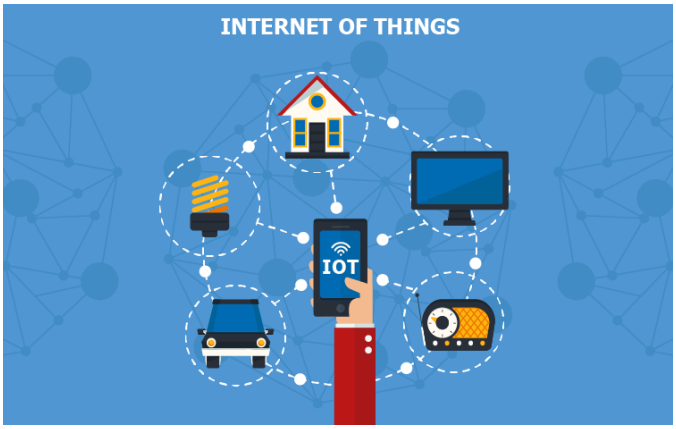


Figure 7: Internet of Things [32]

1.2 Smart Home Automation

IoT-powered smart home automation solutions are intended to monitor and control the variables you wish to manage. A variety of Wi-Fi gadgets, for example, acquire and share data via Internet protocols. Each gadget contains sensors or detectors that communicate with a central home automation hub. Sensors send and receive orders to one or more hubs, which subsequently send the results to the cloud network.

Home Automation Systems (HASs) include centralized control and remote status monitoring of a home's lighting, security system, and other appliances and systems. HASs improves energy efficiency, security, and most importantly for user comfort and ease. Home automation is important for security and to assist handicapped and elderly people by controlling home appliances and alerting them in critical situations.

The server, hardware interface module, and software package are the three basic components. Switches, sensors, lights, locks, and other sensory organs of the home are examples of end devices. Their primary role is to offer the user with control flexibility. The most appropriate gadgets respond automatically under regular household situations. Their primary responsibility is to meet established criteria and send notifications to a user's mobile phone or internet app.

The data is then routed through an IoT gateway after the various end devices broadcast or receive orders from their central controllers — hubs — through

Connection Devices. The hubs are in charge of ensuring that data is transmitted smoothly between a single and several end devices.

Figure 8: Smart Home [33]

Because IoT devices create data packages that must be kept, a Data Storage element is used to accomplish this purpose. Furthermore, any platform should be dependable in order to assure the high performance of home IoT security.

Z-wave wireless systems are becoming increasingly popular because their protocols allow for easy tracking of smart devices such as light bulbs, plugins, and thermostats. This widespread popularity is frequently coupled with a dependable security grade. Several internal communication possibilities are accessible via Network Protocols – MQTT, HTTP, and CoAP. When analyzing protocol performance requirements, the decision will always be based on the level of efficiency. The many forms of IoT system needs will aid in selecting the most dependable alternative.

1.3 Components used in our project

1.3.1 ESP32 Development Board

ESP32 is a low-cost, low-power system-on-a-chip microcontroller family featuring built-in Wi-Fi and dual-mode Bluetooth. The ESP32 Development board is designed around the ESP WROOM32 WIFI + BLE Module. This is the most recent version of the ESP32 IoT development module. This development board features a 3.3 Volt power regulator, a Reset and programming button, and an inbuilt CP2102 USB to TTL converter for programming directly through USB port. The ESP32 offers many more functionalities than the ESP8266.

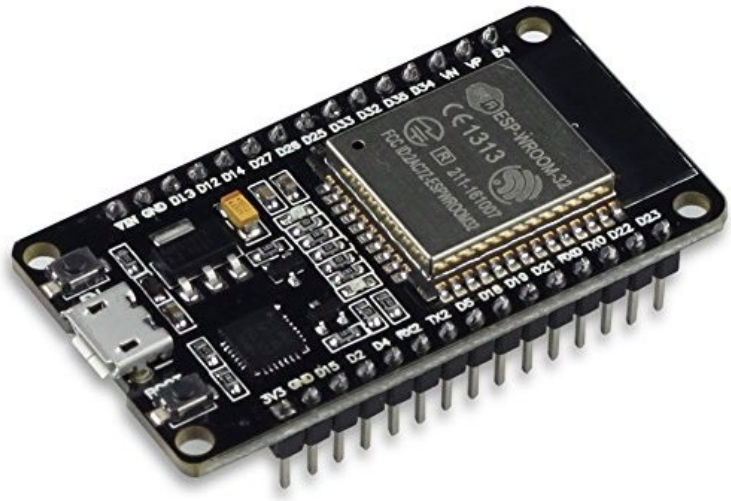


Figure 9: ESP32 Board [40]

1.3.2 ESP32 CAM Board

The ESP32-CAM is a full-featured microcontroller with an inbuilt video camera and a micro SD card slot. It's cheap and simple to use, making it ideal for IoT devices that require a camera with complex functions like image tracking and identification.

With a footprint of only 27 x 40.5 x 4.5mm and a deep sleep current of up to 6mA, the ESP32 – CAM features a highly competitive tiny size camera module that can run independently as a minimal system.

The ESP32 – CAM may be found in a wide range of IoT applications. It is appropriate for smart home gadgets, industrial wireless control, wireless monitoring, QR wireless identification, wireless positioning system signals, and other Internet of Things applications. It is an excellent choice for IoT applications.

ESP32 – CAM uses a DIP package and may be immediately inserted into the backplane to enable speedy product manufacturing while offering customers with a high – reliability connection mode that is suitable for use in a variety of IoT hardware terminals.

Figure 10: ESP32 CAM [24]

1.3.3 Four Channel 5V Optically Isolated Relay Module

This is a LOW Level 5V 4-channel relay interface board with a 15-20mA driver current need for each channel. It may be utilized to regulate a variety of high-current appliances and equipment. It has high-current relays that operate at AC250V 10A or DC30V 10A. It features a standard interface that may be directly controlled by a microcontroller. This module is optically separated from the high voltage side for safety and to prevent ground loops when connected to a microcontroller.

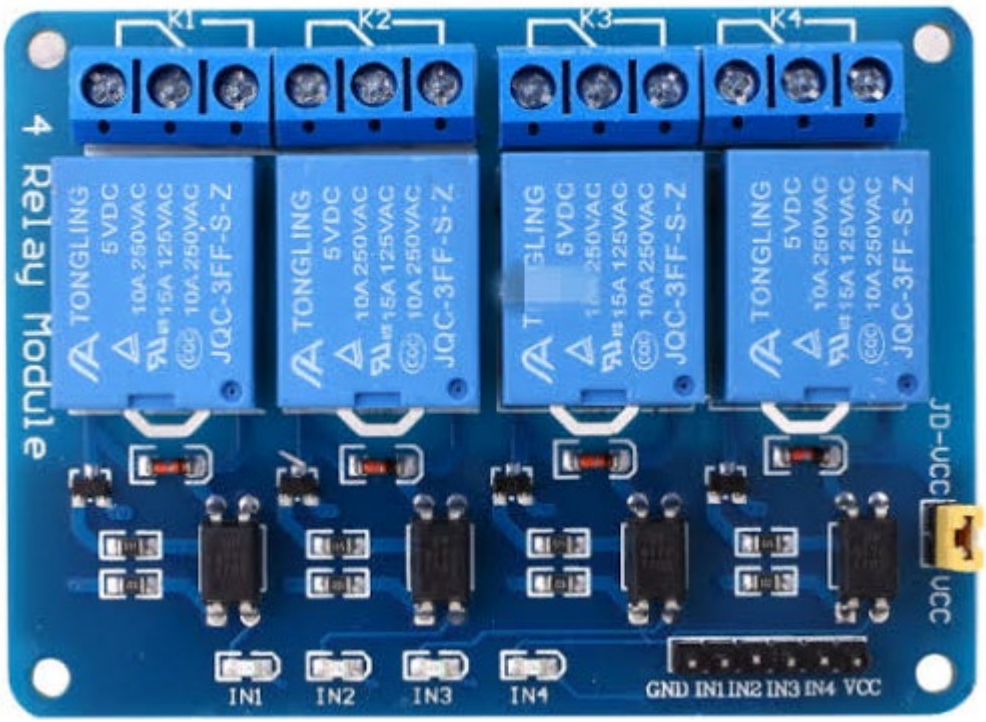


Figure 11: 4 Channel Relay Module [41]

1.3.4 Solenoid Lock

Solenoids are essentially electromagnets, consisting of a large coil of copper wire with an armature (a slug of metal) in the center. When the coil is powered, the slug is drawn into the coil's core. This allows the solenoid to pull from only one end. This solenoid in particular is sturdy, with a slanted cut slug and a well-designed mounting bracket. It's essentially an electronic lock intended for a simple cabinet, safe, or door. When the lock is activated, you cannot open the door because the solenoid slug is in the way. In this condition, it does not utilize any electricity. When 9-12VDC is applied, the slug draws in so that it no longer sticks out and the door can be opened.

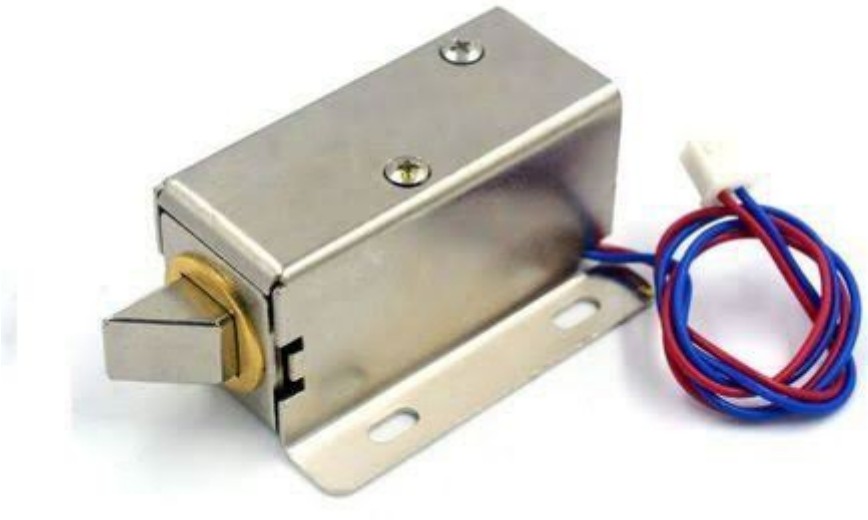


Figure 12: Solenoid Lock [42]

1.4 Cloud Service Platform

1.4.1 Sinric Pro

With Sinric Pro, you can use a dependable skill to link your IOT development board to Alexa or Google Assistant - no more glitch emulation. Sinric Pro supports all Amazon and Google IoT device types and provides Python, C++, and NodeJS libraries with examples to help you get started quickly. Advanced users may build their own custom devices, configure rooms and routines, and monitor all of their devices via the REST API.

Users may use this to control their development boards such as Arduino, WEMOS D1 Mini and ESP32 using Amazon Alexa and Google Home for Free. Users may use Sinric.com to design switches, light bulbs, and other devices and link them to their development board to Amazon Eco system or Google's Smart Home ecosystem without a hassle.

Figure 13: Sinric Pro App Icon

1.4.2 Google Home

Google Home is at the core of the company's smart home ecosystem. The Google Home system began as a basic wireless speaker that could respond

to voice commands and has evolved into a robust system for managing your entire home. It is controlled via the Google Home app and allows you to ask queries, activate applications, and set up routines to manage home gadgets.

Figure 14: Google Home App Icon

1.4.3 Amazon Alexa

Amazon's Alexa is an all-knowing, interactive voice assistant. Alexa can do fast arithmetic for you, launch your favorite songs, check news and weather, and operate many of your home's smart gadgets. Alexa is available on Amazon's Echo speakers, smart thermostats, sound bars, lamps and lights, and even on your phone through the Alexa app.

Figure 15: Amazon Alexa App Icon

1.5 Android App Development Platform

Kodular is a free online suite for mobile apps development. It primarily offers an online drag-and-drop Android app generator, allowing anybody to create any type of app without writing a single line of code.

Kodular blocks-based editor makes it simple to design Android apps. No coding knowledge is necessary. Your apps will stand out with the Material Design UI.

Figure 16: Kodular App Icon

1.6 Motivation and Objectives

As technology is growing so houses are also becoming smarter. Modern homes are rapidly migrating away from traditional switches and toward centralized control systems with remote-controlled switches. Currently, standard wall switches situated around the home make it difficult for the user to get close to them in order to activate them. It becomes even more difficult for the elderly or physically disabled folks to do so. With smart phones, a remote controlled home automation system gives the most current solution. To do this, a Wi-Fi module is interfaced on the ESP32 board at the receiving

end, while a mobile application on the cell phone transmits ON/OFF or LIVESTREAM commands to the receiver where loads are attached at the transmitter end. The instruction to toggle ON/OFF/ LIVESTREAM remotely is transmitted by clicking the button on the app.

This project provides a smart and cost efficient real time Home Automation and Smart Security System for a Home. The main objective of this project is to build a smart home device which can be used to control the home appliances via internet and Wi-Fi. It gives users the ability to control and monitor their devices through touch controls and Voice Commands in the Google Home or Amazon Alexa App from any part of the world. The device can be connected to an Android App using Wi-Fi, using this app the user will be able to control the home appliances without internet with ease. The user can also unlock the door of his/her home by scanning fingerprint sensor of mobile phone using the Android App. Real Time Intruder alert images with face detection will be sent to the authenticated user on his Telegram account and also will be saved in the SD Card mounted on ESP32 CAM.

LITERATURE REVIEW

Chapter 2

2.1 Manual Systems:

2.1.1 IoT Based Remote Lock System using ESP-32 Microcontroller

Methodology:

The design employs the modular technique. The design is divided into two parts: hardware and software. The hardware component is intended to power a solenoid lock. A solenoid lock was chosen because it is dependable and simple to use. The software is meant to enhance the functionality of the physical device. The software's complicated and elaborate running routine is achieved by writing the programme in modules. The software was created in the C++ programming language and will be written in portions to facilitate debugging and troubleshooting.

Conclusion:

The lock was created to enhance user convenience by allowing him to examine the picture of a genuine visitor and remotely open or close the door lock. Another useful system function is that when a valid user approaches, he can turn on the camera. The owner no longer needs to be concerned about forgetting the key, being locked out, or having their hands full with groceries since the Smart Lock system has them covered. When compared to existing digital door lock systems, the suggested system may be marketed into a useful product, such as a secure security system with greater convenience [1].

2.2 Semi – Automatic Systems:

2.2.1 Arduino Based Home Automation using Internet of Things

Methodology:

This article describes a low-cost, adaptable, and dependable home automation system with added security using an Arduino microcontroller and IP connection through local Wi-Fi for accessing and controlling devices remotely using a smart phone application. The suggested system is server-agnostic and employs the Internet of Things to operate human-desired items ranging from industrial machines to consumer products. The user can also utilize various devices to operate the system, such as a web browser or a smart phone. To show the efficacy and feasibility of this system, they provided a home automation system based on an Arduino UNO microcontroller and an esp8266 connection module in this article. It enables the user to remotely operate numerous appliances such as lights, fans, and televisions, as well as make decisions based on sensor data. They put their technology to the test by doing experiments in diverse environments.

Conclusion:

The IOT system they created has been tested under a variety of load circumstances for a few residences. Following the installation of the experimental setup, the user must install the programme on his or her laptop or Android phone. The IP address will be shown on the 16X2 LCD display after correct installation of the given software. After obtaining the IP address and port address, the user may log in through the Android application. When the setup is finished, a home

page will emerge where the user may keep track of all the electronic and electrical devices that are linked to the server [2].

2.2.2 Door Locking and Unlocking with ESP32 CAM and Blynk App using IoT

Methodology:

In this project, we are photographing individuals with an ESP-32 camera. The ESP-32 has a camera with a flash that consumes less power. It features an embedded Wi-Fi and Bluetooth connection with two high-performance components: a power supply, a PIR sensor, a relay, a solenoid lock, a buzzer, a push button, red and green led lights, and an ESP32cam. After powering up the components, the ESP-32 cam will capture images anytime an object is identified. The PIR sensor detects the thing and provides data to the ESP-32 camera. The photographs are provided to the owner after they have been taken. The owner obtains photographs by installing and logging into the Blynk programme. A program will be used to link them both. In this case, we're writing software in C++. This is loaded into the ESP-32 cam module. The owner has the ability to remotely open and close the door. When the owner grants access to the door, the relay changes the solenoid lock to open or close it. There is also a push button inside the house, so that if a person wants to come out, by pressing the push button, the door will automatically open, allowing the person to go outdoors.

Conclusion:

When a person approaches the door, the IR sensor detects it and sends information to the ESP32 microcontroller, which processes the information and updates the server, which in this case is the Blynk app. The Blynk will refresh and notify you that someone has arrived. In addition, we can photograph a person in a certain situation. The Blynk will display the identity of the individual who has come. We offer the command to open or close the door based on that instruction. This system may be upgraded in the future to include dual verification techniques such as a retina scanner, fingerprint scanner, OTP, PIN code, and so on. This system will begin by recognizing the face. If a face is located in the database, the second verification procedure must be one of the ones listed above. Only the door will open if the individual passes the verification exam. The photograph will be sent to the website if the face is not located in the database. The system will be extremely secure. Face recognition can be used in conjunction with any other technique [3].

2.2.3 Face Detection Door Lock System using ESP32

Methodology:

This study used an experimental setup-based research technique to show hardware connection with a cloud server via a Wi-Fi local hotspot to remotely transfer data for monitoring and control-based applications. The hardware employed here was a high-performance low-cost Wi-Fi enabled computational

platform with an integrated camera, namely the ESP32CAM board. It contains a number of GPIO's for connecting to external input/output devices. A solenoid lock, a buzzer, a two-channel relay module, an infrared proximity sensor module, a tactile push button, +5V and +12V DC power sources, a USB TTL UART Convertor to programme the ESP32-CAM board, and finally a smart-phone with a "Blynk" account were also utilized to execute the suggested system. This hardware experimental setup's experimental findings should confirm the work. The research includes data collection from datasheets on various components utilized in hardware development. The information gleaned from these research initiatives was utilized to construct the overall approach for designing, developing, and implementing the system. The final report includes appropriate tables, flowcharts, and figures, as well as pictures of the functioning prototype.

Conclusion:

After successfully completing this work, it was concluded that through IoT, we can connect multiple input/output devices, multiple sensors and actuators in a network so that they can communicate with each other, data acquired from these things can be used to keep a log, monitor, or control the other things without human intervention, and much more. Thus, IoT is analogous to global networks that enable communication between things, humans, and things. IoT is the evolution of existing internet infrastructure to handle anything that exists in the world or will exist in the future. Surveillance, according to this work, is the process of careful conscious perception or monitoring kept up over an individual,

gathering, and so on, particularly one under care or under suspicion. For the before mentioned goals, he created a system that included a sensor, camera, CPU, relays, buzzers, LED indicators, and actuators based on the application requirements. The system performed effectively in the local context and behaved as expected [4].

2.3 Automatic Systems:

2.3.1 IoT Based Door Access Control System using Face Recognition

Methodology:

If you utilize a traditional key-based system, you'll either have to go to a key-cutting business and get genuine keys for the authorized people, or you'll have to hide your key outdoors where anybody may find it. If electronic lock code breaking occurs, some of it is performed by the most knowledgeable. If your password is significant, it is advised that you pick a random password to boost your security because numbers at important dates are simpler to guess. If you're not sure what the advantages and disadvantages of acquiring a biometric entry door lock are, keep in mind that fingerprint scan door locks require energy to work, and a power outage might render them utterly useless.

Conclusion:

This proves that it is trustworthy as well as fun. It can only be reached from a distance. Locker rooms, bank keys, automobile doors, bank vaults, and home offices are all simple to use. It's a global look at what's committed to the Internet.

We have successfully developed a robust face-to-face export system that might be utilized as a low-cost replacement for fingerprint / facial recognition systems [5].

2.3.2 Home Automation using Internet of Things

Methodology:

The suggested system is a distributed home automation system that includes a server as well as sensors. The server manages and monitors the numerous sensors, and it is easily expandable to accept new hardware interface modules (sensors). The Intel Galileo development board serves as a web server since it has a built-in Wi-Fi card connector into which the card is put. The Automation System may be accessed via the web browser of any local PC in the same LAN using the server IP, or remotely using the proper web browser on any PC or mobile handheld device connected to the internet via the server actual IP (Internet Protocol). Wi-Fi technology is used as the network architecture that connects the server and sensors. Wi-Fi is chosen to increase system security (through a secure Wi-Fi connection) and to save time.

Conclusion:

Home automation utilizing the Internet of Things has been experimentally proved to operate satisfactorily by connecting basic appliances to it and effectively controlling the equipment remotely over the internet. The developed system not only monitors sensor data such as temperature, gas, light, and motion sensors, but

it also actuates a process based on the need, such as turning on the light when it gets dark. It also stores sensor parameters in the cloud (Gmail) on a regular basis. This will allow the user to assess the state of numerous parameters in the house at any time and from any location [6].

2.3.3 IOT Based Electrical Equipment Control for Automation

Methodology:

This paper discusses the use of the Internet of Things for home automation. This study focuses mostly on IoT attention, which connects all changes of items like as smart phones, tablets, digital cameras, and sensors to the internet and so provides various services and a great quantity of data and content. They also focused on cloud computing, which allows us to connect the objects that surround us so that we can easily access everything at any time and from any location.

Conclusion:

Today, Automaton is the world's most influential mobile platform open source working system, allowing us to effortlessly apt any functionality we had in mind. This article is about wireless home computerization by Android smartphone, which enables you to create such a magnificent system in our house at a very reasonable price by employing low-cost gadgets. As a result, it disables various difficulties like as costs, intransigence, safety, and so on. In addition, it will provide additional benefits such as lowering our energy expenditures and

improving house safety. Furthermore, it is perfectly fit for use and will restore the comfort of our home [7].

2.3.4 Voice Based Smart Home Automation using Internet of Things

Methodology:

In this paper Alexa receives a voice command from the user, such as Turn on Fan 1, Turn off Bulb 2, and so on. Alexa then looks for the command in its accessible cloud (The cloud available here is Sinric). If the command is located in the cloud, it is sent to the NodeMCU. The important point to note here is that the operation is only carried out if there is an updated operation on the device that is not the same as the prior one. NodeMCU is linked to Embedded C code that displays the status of the devices depending on the command on the LCD and also makes the same request to Atmega8, which processes the request and transmits it to the relay. The relay reacts to the request and performs the necessary operations on the devices. Based on the request, the gadget turns on or off.

Conclusion:

They completed the job and used Amazon Echo to operate the house appliances. Six gadgets were under our control. Echo hears the voice rapidly and reacts to orders promptly. For communication with Echo, we used SINRIC. One critical issue that we attempted to solve in this study is non-smartness. We combined one of the smart gadgets on the market today, the Amazon Echo, with the Arduino NodeMCU. The module we used to do home automation activities performed

admirably. Furthermore, real-time testing of the programme on Amazon Echo yielded good results. We feel that this is an important step toward achieving a low-cost smart home [8].

2.3.5 Internet of Things-Based Intelligent Smart Home Control System

Methodology:

They created an Android application in this study for remote control of basic electrical appliances, as well as monitoring and displaying environmental elements in the home and its surrounds. To put in place a sensor-based automation system that detects movement and trespass in the house. To avoid false alerts from the security system, a machine learning algorithm will be used to recognize photos in the home.

Conclusion:

This paper described an intelligent home automation system built with IoT technology, cloud computing, and a machine learning algorithm. The Positive Home automation system provides for both remote and local control of the home via an Android-based mobile application. Positive system regulates electrical household appliances, monitors environmental conditions via temperature, humidity, and light sensors, and provides home security via a motion sensor and an IoT camera.

Positive system makes intelligent selections to automatically turn on or off lights and allows the user to view and download a photo of the person taken by the

camera. Positive system is expandable, and the application allows for more appliances with an additional configured point, simply referred to as an outlet in our work. In addition, the mobile application records data acquired using a real-time cloud database. It also visually displays the trend of the readings on the mobile app's screen [9].

2.3.6 Internet Of Things (IoT) Based Home Automation System

Methodology:

The ESP module is an integrated chip that includes built-in robust and long-range Wi-Fi connection. Data is wirelessly sent to a database in order to operate the devices on the cloud or over the internet. As a result, this module delivers the most efficient service in terms of quality. It's a cheap Wi-Fi chip. Wi-Fi is far more convenient and user-friendly than a microcontroller. A Wi-Fi connected gadget may send and receive data from the host wirelessly and easily. Wi-Fi is a simple and quick method of data transport. Data is sent to the cloud in this project using a Wi-Fi module controlled by the server.

The Wi-Fi module has a very long range and can pierce through objects, allowing it to be positioned anywhere in the house. This module will be used in the project to transport data acquired by the hardware. It delivers data to the server and stores it in the buffer memory, which is subsequently sent to the cloud through Wi-Fi and operations may be conducted on the data.

Conclusion:

The user will receive all of the information via the cloud. The system will notify the user if any suspicious behavior is detected. Because of its compact size and great usability, the Raspberry Pi is chosen as the primary processing unit. A web page or an Android app with a user-configurable front end can be used to regulate and monitor the load. The user can transmit commands to the Wi-Fi module using the assigned IP address. The Wi-Fi module is set up to connect to the internet through any local wireless modem. A programme within a Wi-Fi module executes the commands received by the module. Commands are used to switch on and off the Wi-Fi module interfaced through the payload. The load status (ON or OFF) will be displayed on the web page and app [10].

2.3.7 Arduino Based Security System – An Application of IOT

Methodology:

In this paper the suggested system would be a security system automation. This system should sound an alert and communicate data to the owner as well as to the cloud, where it may be analyzed afterwards. The security system's automation is powered by an Arduino Uno, which is compatible with a wide range of physical devices.

Conclusion:

By automating the device using IOT, security can be supplied without the need for human interaction. Any infiltration may be detected here, and the device's functionality can be validated using the information provided to the cloud [11].

IMPLEMENTATION

Chapter 3

3.1 Block Diagram

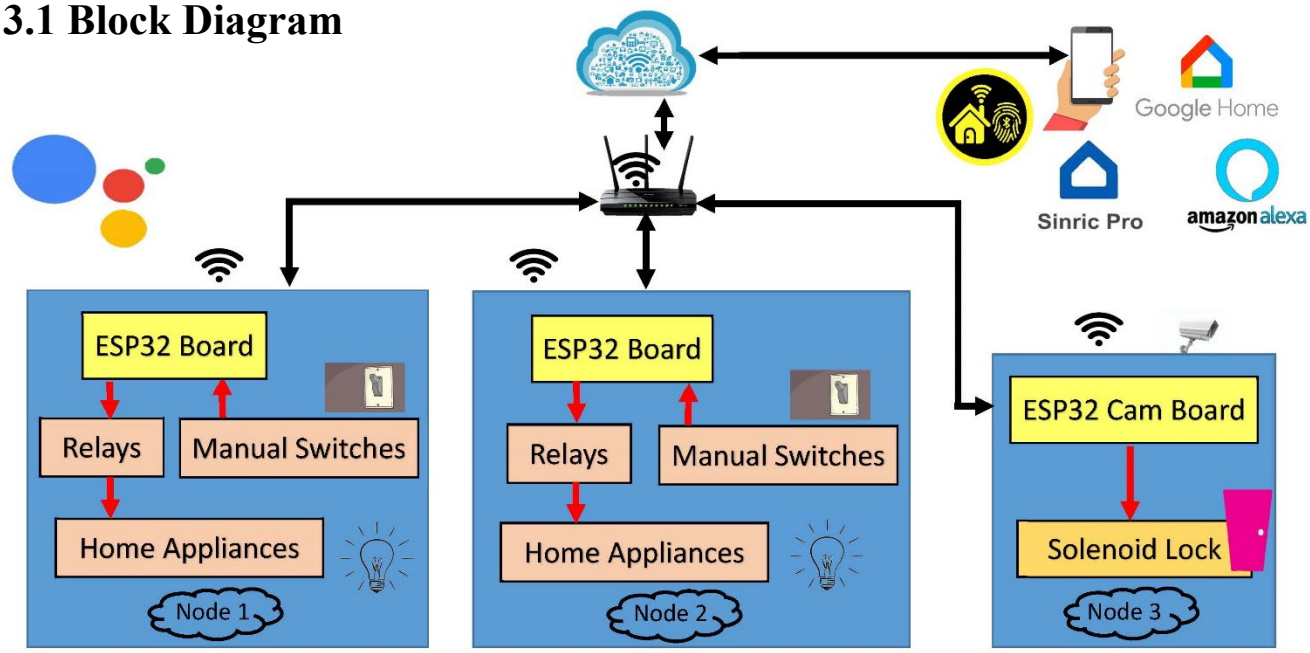


Figure 17: System Block Diagram

The above figure 17 describes the working block of our implementation. In our project we have used two ESP32 modules and one ESP32 CAM module. Node 1 and Node 2 represents two different rooms in a home and Node 3 is placed at the main door of home. All the Nodes are connected to Wi-Fi router for sending and receiving data. Relays and Flip switches are interfaced with the ESP32 modules in Node 1 and Node 2 for controlling the Home Appliances. The Solenoid lock is interfaced with ESP32 CAM in Node 3 to unlock the main door. ESP32 CAM in Node 3 do the live streaming with face detection. The Wi-Fi router is connected to the internet for the user to control and monitor home appliances, view the live streaming from ESP32 CAM, send intruder images to user's telegram account and unlock the main door of home by scanning fingerprint on phone. The user can connect to Wi-Fi router and control gadgets through an Android App and Voice Commands using Google Home, Amazon Alexa or Sinric Pro App.

3.2 Flowcharts

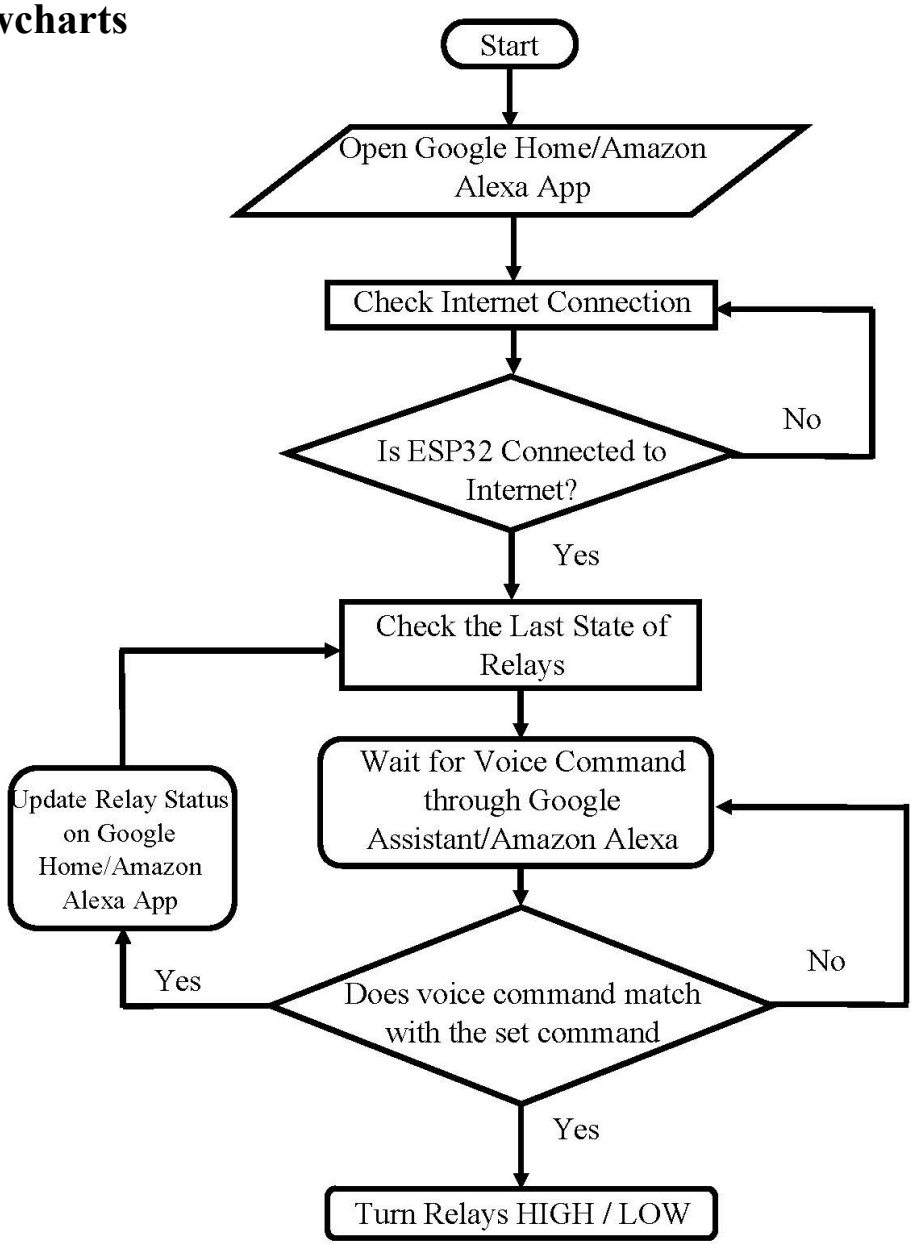


Figure 18: Flowchart for Internet and Voice Controlled Home Automation

The above flowchart in figure 18 describes about controlling home appliances over internet and voice commands using Google Home or Amazon Alexa App. First we open the Google Home or the Amazon Alexa App on our mobile phone device. The ESP32 module will check if it is connected to Wi-Fi router and has an active internet connection. Once connected it will check for the last state of relays and it will turn relays HIGH / LOW. Then it will wait for the user's voice

commands, and if the command matches it will trigger the relays HIGH / LOW, if not then it will wait for proper voice command from user. Also it will update the state of relays on the cloud server.

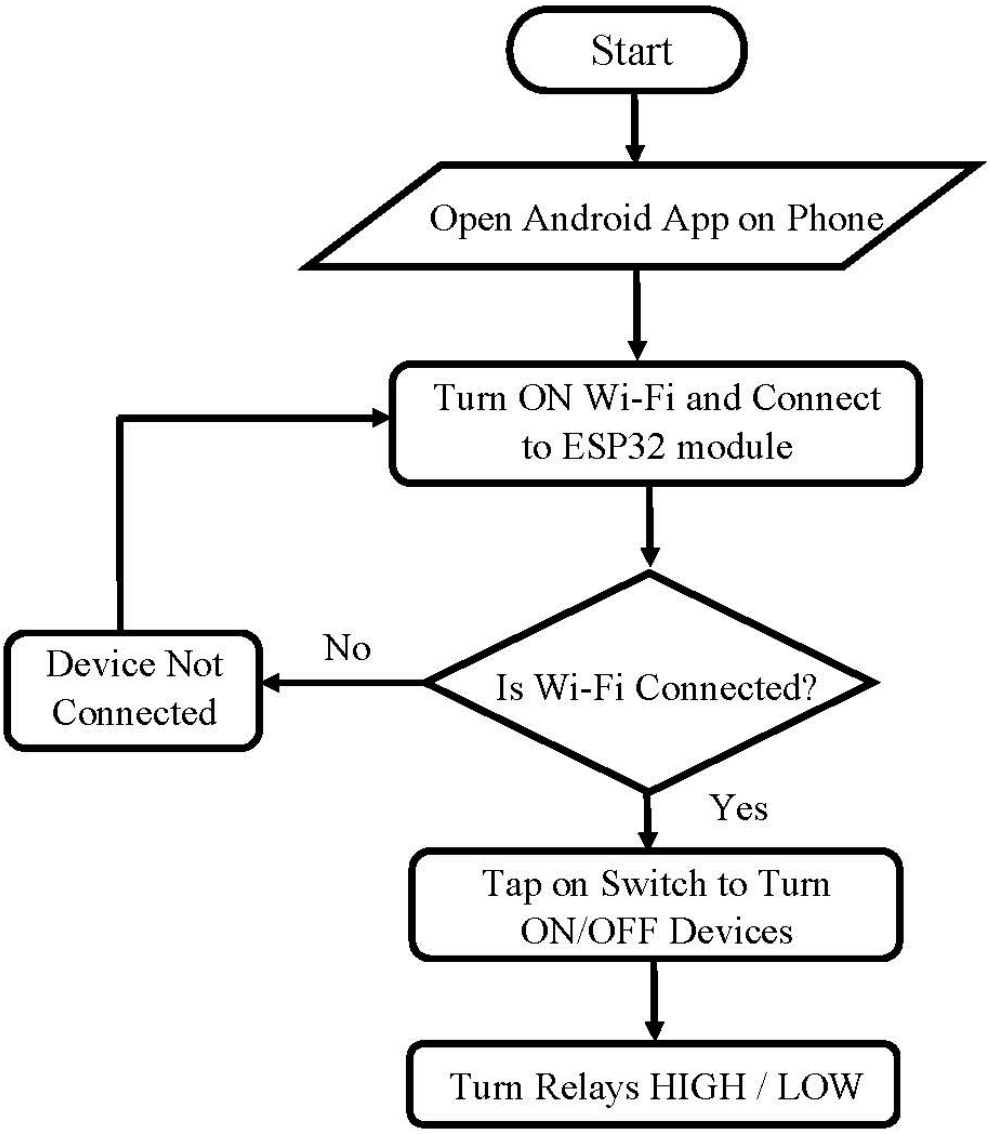


Figure 19: Flowchart for Wi-Fi Controlled Home Automation

The above flowchart in figure 19 describes about controlling home appliances over Wi-Fi with any internet connection. First we have to turn ON our phone's Wi-Fi and open the Android App. Now connect the phone to ESP32 module using Wi-Fi, if it's connected, then it will allow us to control our gadgets, else it will

ask to connect again with ESP32 module. On single tapping the switch to ON/OFF position on Android App, it will trigger the relays to HIGH / LOW.

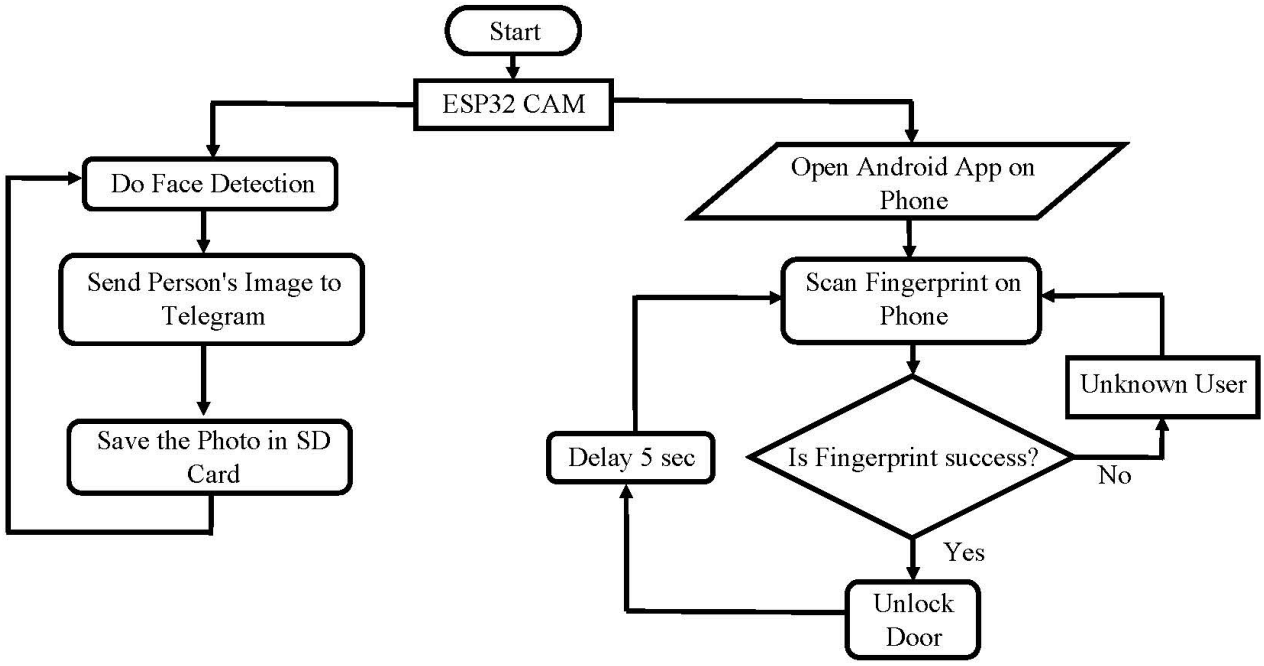


Figure 20: Flowchart for Face Detection and Door Unlock

The above flowchart in figure 20 describes about face detection and unlocking the main door. Here the ESP32 CAM does two tasks. First, the ESP32 CAM does face detection and if detects a person, it will an image to user's telegram account and also it save that same image in the SD card of ESP32 CAM. This process will be ON continuously. Second, to open main door of the home, the user has to open the Android App (ESP32 CAM must be connected to phone through Wi-Fi as mentioned earlier), the user has to tap on 'Scan Finger' button and upon prompting scan his finger. If the authorized user scan's fingerprint and if it matches, it will unlock the door for 5 secs and then the solenoid lock de-energizes, else will prompt fingerprint did not matched, scan the fingerprint again.

3.3 Circuit Diagrams

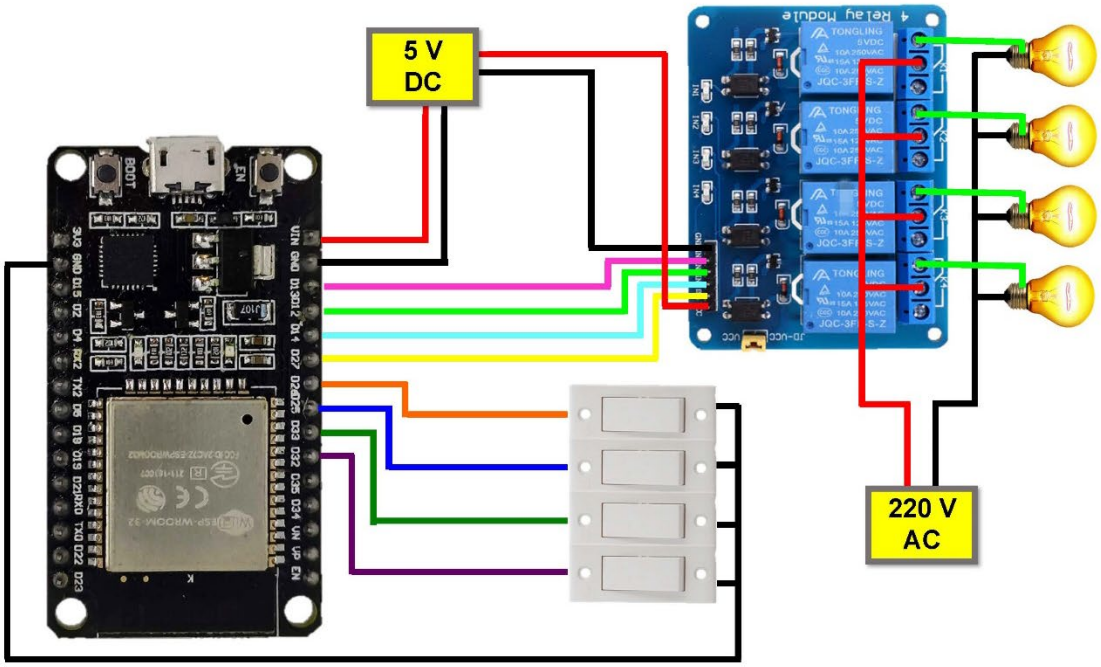


Figure 21: Circuit Diagram for Home Automation

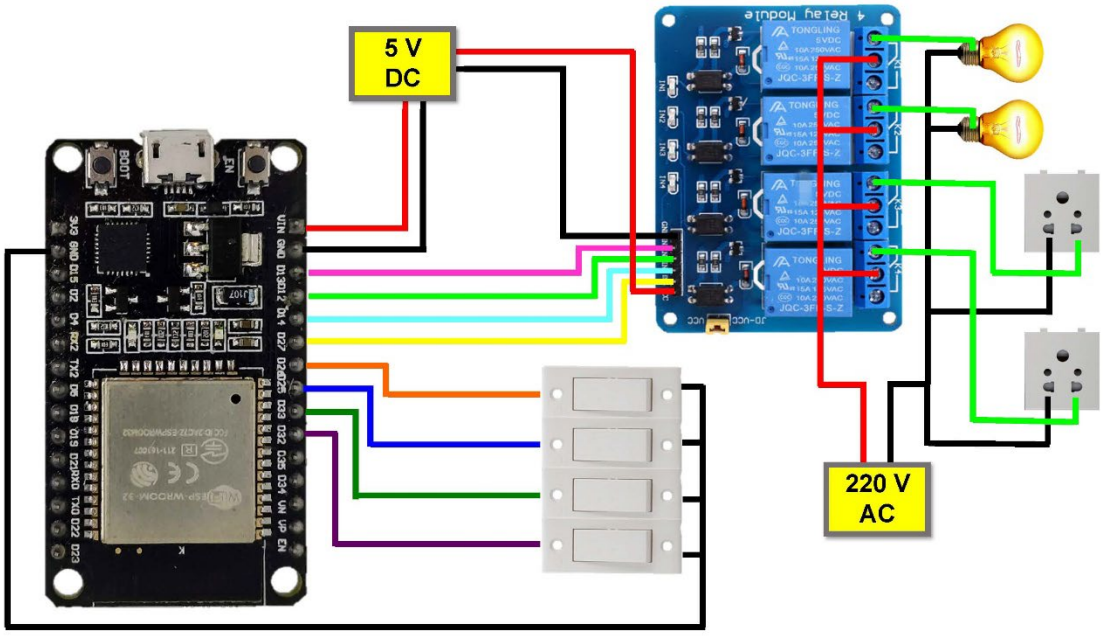


Figure 22: Circuit Diagram for Home Automation

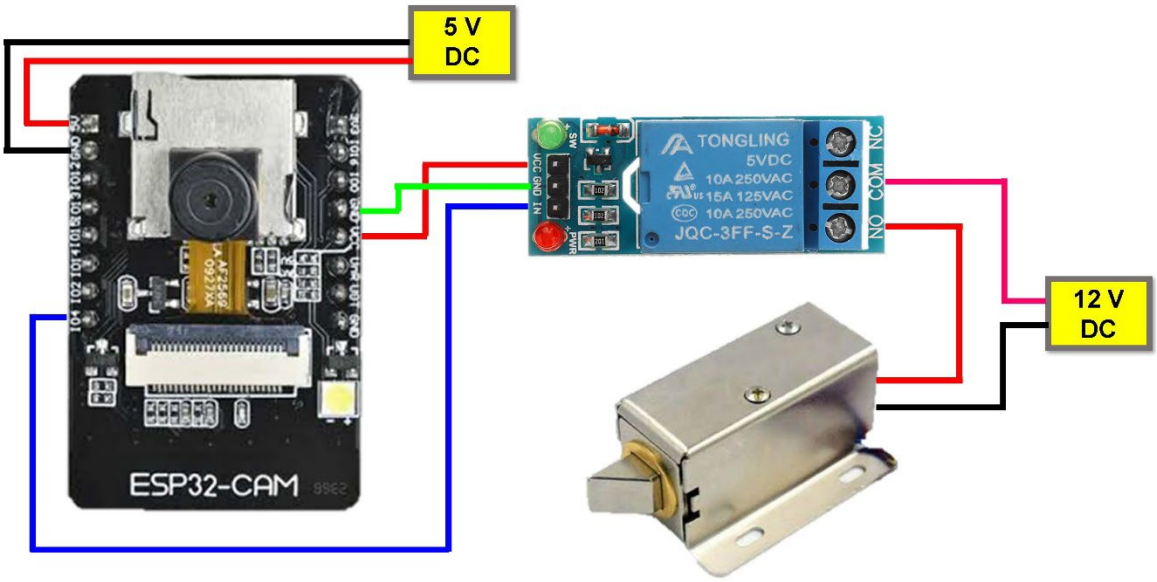


Figure 23: Circuit Diagram for Solenoid Lock

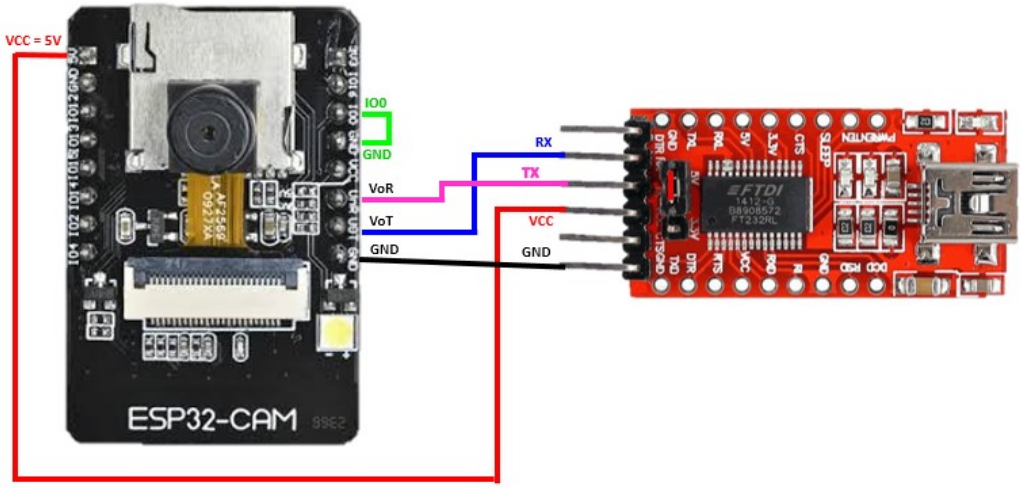


Figure 24: Circuit Diagram for Programming ESP32 CAM with FTDI Programmer

3.4 Telegram APIs

They provide two types of APIs to developers. The Bot API makes it simple to construct programs that utilize Telegram messages as an interface. You may create your own customized Telegram clients using Telegram and TDLib. Both APIs are completely free to use. Telegram Widgets may also be added to your website. Designers are invited to design Animated Stickers or Custom Telegram Themes. You may use this API to create your own customized Telegram clients. It is completely available to any developers that want to create Telegram apps on this platform.

3.4.1 Bot API

You may use this API to link bots to our system. Telegram Bots are unique accounts that do not require the creation of an extra phone number. These accounts act as a user interface for programming that is executing somewhere on your server.

You don't need to understand how the MTProto encryption protocol works to utilize it; their intermediate server will handle all encryption and communication with the Telegram API for you. You contact with this server over a basic HTTPS interface that provides a streamlined version of the Telegram API.

3.4.2 Bots: An Overview for Developers

Bots are third-party apps that operate within Telegram. Bots may be interacted with by sending messages, commands, and inline requests. Your bots are controlled through HTTPS calls to Telegram's Bot API.

3.4.3 What may bots be used for?

Bots might be used for a variety of purposes, such as:

- **Get customized notifications and news:** A bot can function as a smart newspaper, delivering relevant articles to you as soon as it is released.
- **Integrate with other services:** A bot can add material from third-party services to Telegram chats. Gmail bot, GIF bot, IMDB bot, Wiki bot, Music bot, Youtube bot, GitHubBot are few examples.
- **Accept payments from Telegram users:** A bot might provide paid services or function as a virtual shop. Demo Shop Bot, Demo Store, and so forth.
- **Create custom tools:** Alerts, weather predictions, translations, formatting, and other services may be provided by a bot. Markdown bot, Sticker bot, Vote bot, and Like bot are some examples.
- **Build single- and multiplayer games:** From basic arcades and puzzles to 3D shooters and real-time strategic games, a bot may provide complex HTML5 experiences. For example, GameBot and Gamee.
- **Build social services:** A bot might link people seeking discussion companions based on shared interests or location.

3.4.4 How do bots work?

Telegram Bots, at their core, are special accounts that do not require an additional phone number to set up. There are two methods for users to connect with bots:

- Open a conversation with bots or add them to groups to send them messages and orders.

- Requests can be sent straight from the input field by entering the bot's @username and a query. This allows inline bots to submit material immediately into any chat, group, or channel.

User's messages, commands, and requests are sent to the software executing on your servers. All encryption and communication with the Telegram API is handled by our intermediate server. You contact with this server over a basic HTTPS interface that provides a streamlined version of the Telegram API. That interface is referred to as our Bot API.

3.4.5 What differs bots from humans?

- Bots have no online status and no last seen timestamps; instead, the interface displays the label 'bot'.
- Because bots have limited cloud storage, older messages may be deleted by the server quickly after they have been processed.
- Bots are unable to begin discussions with users. First, a user must add them to a group or send them a message. People may locate your bot by using t.me/bot username> links or a username search.
- Bot usernames must always end in 'bot' (for example, @TriviaBot, @GitHub bot).
- When added to a group, bots do not automatically get all messages.
- Bots do not eat, sleep, or whine (unless expressly programmed otherwise).

3.4.6 Bot perks

Telegram bots are unique in many ways — we offer two kinds of keyboards, additional interfaces for default commands and deep linking as well as text formatting, integrated payments and more.

3.4.7 BotFather

BotFather is the king of all bots. It will assist you in creating new bots and changing the parameters of current ones.

3.4.8 Creating a new bot

- To make a new bot, use the /newbot command. The BotFather will ask you for your new bot's name and username before generating an authentication token.
- Your bot's name appears in contact information and elsewhere.
- The Username is a brief name that will appear in remarks and t.me links. Usernames can be between 5 and 32 characters long and are case insensitive, although they can only contain Latin letters, digits, and underscores. The username of your bot must finish in 'bot,' for example, 'tetris bot' or 'TetrisBot'.
- The token, which looks like 110201543:AAHdqTcvCH1vGWJxfSeofSAs0K5PALDsaw, is necessary to authorize the bot and submit queries to the Bot API. Keep your token safe and secure; it may be used to operate your bot by anybody.

3.4.9 Getting an authentication token

If your old token has been hacked or you have misplaced it, use the /token command to generate a new one.

3.4.10 BotFather Commands

The rest of the instructions are self-explanatory:

/mybots — provides a list of your bots, along with options to change their settings.

/mygames — does the same function for your games.

Edit bots

/setname — Change the name of your bot.

/setdescription — Modify the bot's description, which is a brief text of up to 512 characters that describes your bot. Users will see this text, headed 'What can this bot do?' at the start of their chat with the bot.

/setabouttext — Change the bot's about information, which can be as short as 120 characters. This text will appear on the bot's profile page. This text is provided along with the link when they share your bot with someone.

Change the bot's profile image using **/setuserpic**. It's always good to be able to put a face to a name.

/setcommands — modify the list of commands that your bot supports. When users type / in the conversation with your bot, they will see these instructions as suggestions. Each command includes a name, arguments, and a text description (must begin with a slash '/', alphanumeric plus

underscores, no more than 32 characters, case-insensitive). When users input '/' in a chat with your bot, they will see a list of commands.

/deletebot — Delete your bot to release its username.

Edit settings

/setinline — Enable or disable inline mode for your bot.

/setinlinegeo - Request location data to offer inline location results.

/setjoingroups — toggles whether or not your bot may be added to groups. Any bot must be able to process private messages, but you may deactivate this if your bot was not built to function in groups.

Set which messages your bot will receive when added to a group with **/setprivacy**. When privacy mode is turned off, the bot receives all communications. We recommend that you leave privacy mode activated. For this modification to take effect, you must re-add the bot to existing groups.

Manage games

Create a new game with **/newgame**.

Get a list of your games with **/listgames**.

/editgame — Change the settings of a game.

/deletegame — Get rid of an existing game.

PROTOCOLS AND ALGORITHM

Chapter 4

4.1 OSI Model

The ISO-OSI model is a conceptual framework that partitions any networked communication device into seven layers of abstraction, each performing distinct tasks based on the underlying technology and internal structure of the hosts. These seven layers, from bottom-up, are as follows: Physical layer, Data link layer, Network layer, Transport layer, Session layer, Presentation layer, and Application layer.

(i) **Physical Layer:** This is a media layer and is also referred to as layer 1 of the OSI model. The physical layer is responsible for taking care of the electrical and mechanical operations of the host at the actual physical level. These operations include or deal with issues relating to signal generation, signal transfer, voltages, the layout of cables, physical port layout, line impedances, and signal loss. This layer is responsible for the topological layout of the network (star, mesh, bus, or ring), communication mode (simplex, duplex, full duplex), and bit rate control operations. The protocol data unit associated with this layer is referred to as a symbol.

(ii) **Data Link Layer:** This is a media layer and layer 2 of the OSI model. The data link layer is mainly concerned with the establishment and termination of the connection between two hosts, and the detection and correction of errors during communication between two or more connected hosts. IEEE 802 divides the OSI layer 2 further into two sub-layers: Medium access control (MAC) and logical link control (LLC). MAC is responsible for

access control and permissions for connecting networked devices; whereas LLC is mainly tasked with error checking, flow control, and frame synchronization. The protocol data unit associated with this layer is referred to as a frame.

(iii) **Network Layer:** This layer is a media layer and layer 3 of the OSI model. It provides a means of routing data to various hosts connected to different networks through logical paths called virtual circuits. These logical paths may pass through other intermediate hosts (nodes) before reaching the actual destination host. The primary tasks of this layer include addressing, sequencing of packets, congestion control, error handling, and Internet working. The protocol data unit associated with this layer is referred to as a packet.

(iv) **Transport Layer:** This is layer 4 of the OSI model and is a host layer. The transport layer is tasked with end-to-end error recovery and flow control to achieve a transparent transfer of data between hosts. This layer is responsible for keeping track of acknowledgments during variable-length data transfer between hosts. In case of loss of data, or when no acknowledgment is received, the transport layer ensures that the particular erroneous data segment is re-sent to the receiving host. The protocol data unit associated with this layer is referred to as a segment or datagram.

(v) **Session Layer:** This is the OSI model's layer 5 and is a host layer. It is responsible for establishing, controlling, and terminating of

communication between networked hosts. The session layer sees full utilization during operations such as remote procedure calls and remote sessions. The protocol data unit associated with this layer is referred to as data.

(vi) **Presentation Layer:** This layer is a host layer and layer 6 of the OSI model. It is mainly responsible for data format conversions and encryption tasks such that the syntactic compatibility of the data is maintained across the network, for which it is also referred to as the syntax layer. The protocol data unit associated with this layer is referred to as data.

(vii) **Application Layer:** This is layer 6 of the OSI model and is a host layer. It is directly accessible by an end-user through software APIs (application program interfaces) and terminals. Applications such as file transfers, FTP (file transfer protocol), e-mails, and other such operations are initiated from this layer. The application layer deals with user authentication, identification of communication hosts, quality of service, and privacy. The protocol data unit associated with this layer is referred to as data. A networked communication between two hosts following the OSI model is shown in figure 25.

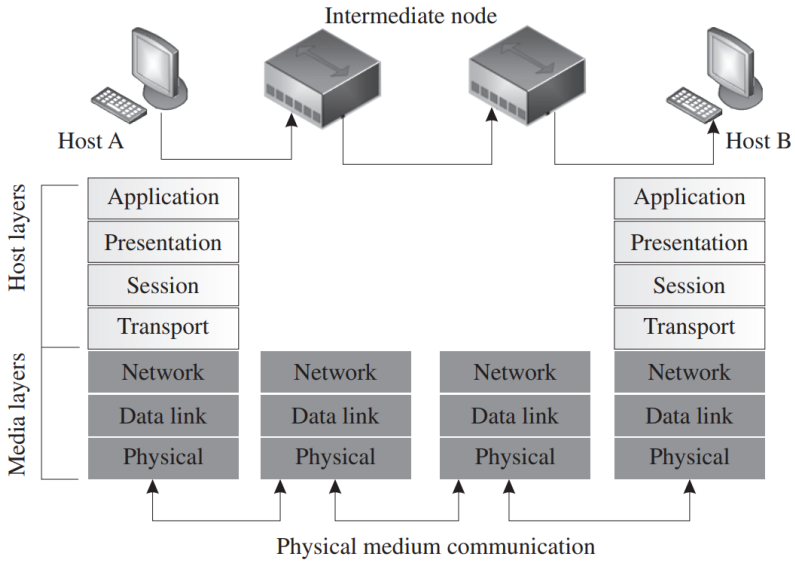


Figure 25: Networked communication between two hosts following the OSI model [20]

4.2 Network Layer Addressing

Network layer addressing is also termed as IP-based addressing or logical addressing. IPv4 addressing uses 32-bits long addresses, whereas IPv6 uses addresses that are 128 bits long. These addresses can identify the source or destination addresses from the address itself. The mapping of a device/host's logical address to its hardware address is done through a mechanism called address resolution protocol (ARP). During transmission of a packet from a host, the IPv4 sends an IPv4 packet, the next-hop address, and the next-hop interface to the ARP.

Direct delivery is performed by the ARP if the destination address for delivery matches the next-hop address. In contrast, if the addresses do not match, the ARP performs an indirect delivery by forwarding the packet to a router or an intermediate node. The resolution of the mapping of a packet's next-hop address

to its MAC address is made using broadcasting ARP requests. The returning ARP reply frame to the sender contains the MAC address corresponding to the packet's next-hop address.

In the context of addressing, we will discuss the structure of IPv4 and IPv6 packets, which will provide a much clearer understanding of the workings of these two protocols.

(i) **IPv4:** The IPv4 header packet shown in Figure 1.5 has 13 distinct fields, the functions of which are enumerated as follows.

- **VER:** It is 4 bits long and represents the version of IP. In the example given in Figure 26, it is 4 bits (binary: 0100).

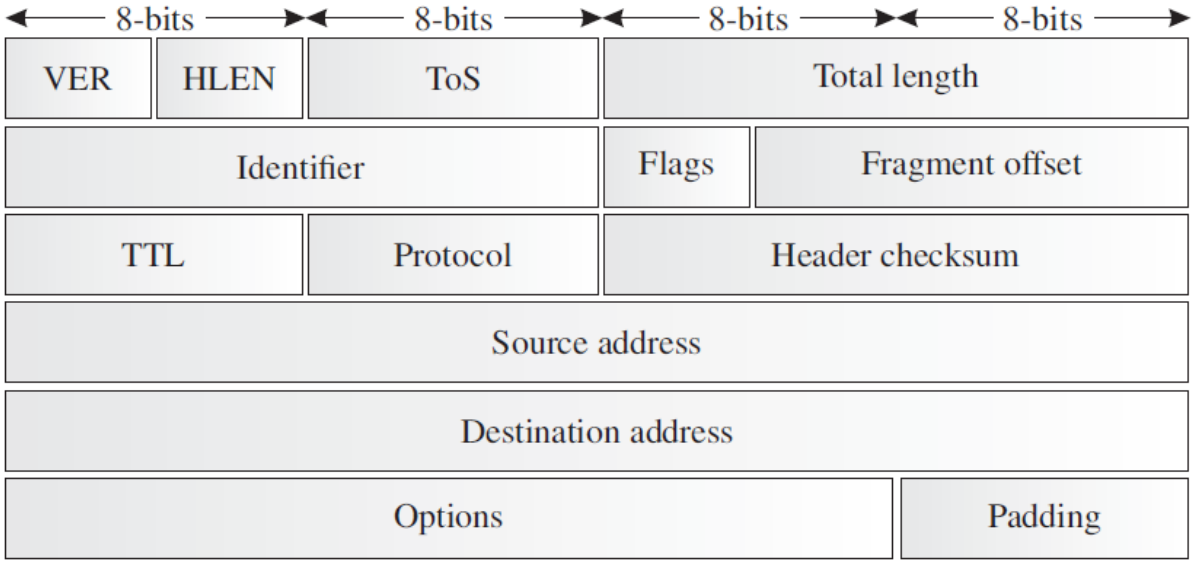


Figure 26: An IPv4 packet header structure [20]

- **HLEN:** It is 4 bits long and denotes the length of the IPv4 packet header.
- **ToS:** It is 8 bits long. The first six most significant bits represent the differentiated services code point (DSCP) to be provided to this

packet (by the routers). Explicit congestion notification (ECN), which gives information about the congestion witnessed in the network, is handled by the last 2 bits.

- **TOTAL LENGTH:** It is 16 bits long and identifies the length of the entire IPv4 packet, including the header and the payload.
- **IDENTIFIER:** It is 16 bits long and used to identify the original packets in case of packet fragmentation along the network.
- **FLAGS:** It is a 3-bit field with the most significant bit always set to 0. FLAGS indicates whether a packet can be fragmented or not in case the packet is too big for the network resources.
- **FRAGMENT OFFSET:** It identifies the exact offset or fragment position of the original IP packet and is 13 bits long.
- **TTL:** It is 8 bits long and prevents a packet from looping infinitely in the network. As it completes a link, its value is decremented by one.
- **PROTOCOL:** It is 8 bits long. This field identifies the protocol of the packet as user datagram protocol, UDP (17), transmission control protocol, TCP (6), or Internet control message protocol, ICMP (1). The identification is made at the network layer of the destination host.

- **HEADER CHECKSUM:** It is 16 bits long and used for identifying whether a packet is error-free or not.
- **SOURCE ADDRESS:** It indicates the origin address of the packet and is 32 bits long.
- **DESTINATION ADDRESS:** It indicates the destination address of the packet and is 32 bits long.
- **OPTIONS and PADDING:** It is an optional field, which may carry values for security, time stamps, route records, and others.

(ii) **IPv6:** The IPv6 header packet shown in Figure 1.6 has eight distinct fields, the functions of which are enumerated as follows.

- **VER:** It is 4 bits long and represents the version of IP. In the example given in Figure 27, it is 6 (binary: 0110).

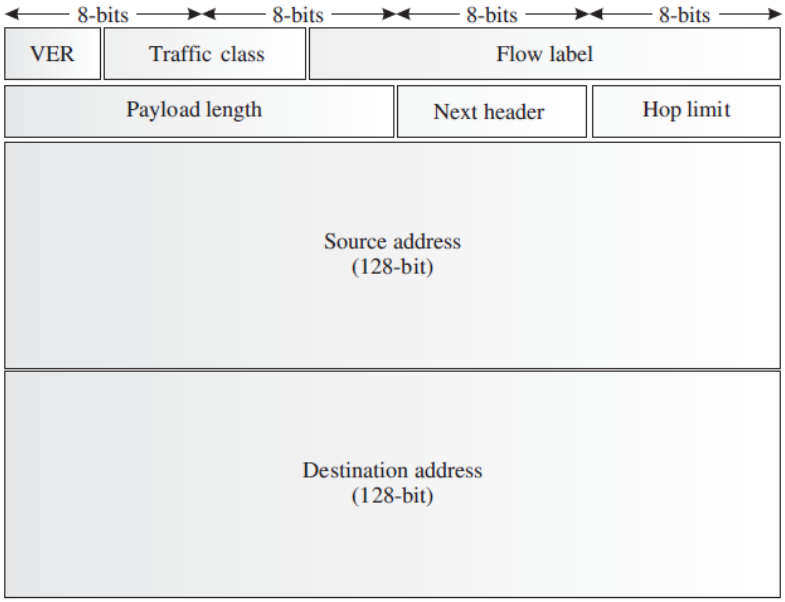


Figure 27: An IPv6 packet header structure [20]

- **TRAFFIC CLASS:** It is 8 bits long. The first six most significant bits represent the type of service to be provided to this packet (by the routers); explicit congestion notification (ECN) is handled by the last 2 bits.
- **FLOW LABEL:** It is 20 bits long and designed for streaming media or real time data. The FLOW LABEL allows for information flow ordering; it also avoids packet resequencing.
- **PAYLOAD LENGTH:** It is 16 bits long and provides a router with information about a packet's payload length or the amount of data contained in the packet's payload.
- **NEXT HEADER:** It is 8 bits long and informs the router about the type of extension header the packet is carrying. Some of the extension headers and their corresponding values are as follows: Hop-by-hop options header (0), routing header (43), fragment header (44), destination options header (60), authentication header (51), and encapsulating security payload header (50). In case an extension header is absent, it represents the upper layer protocol data units (PDUs).
- **HOP LIMIT:** It is 8 bits long and prevents a packet from looping infinitely in the network. As it completes a link, the limit's value is decremented by one.

- **SOURCE ADDRESS:** It is 128 bits long and indicates the origin address of the packet.
- **DESTINATION ADDRESS:** It is 128 bits long and indicates the destination address of the packet.

4.3 Internet Protocol Version 6 (IPv6)

The Internet Protocol Version 6 or IPv6, as it is commonly known, is a resultant of the developments on and beyond IPv4 due to fast depleting address ranges in IPv4. The IPv4 was not designed to handle the needs of the future Internet systems, making it cumbersome and wasteful to use for IoT-based applications. The needs of massive scalability and limited resources gave rise to IPv6, which was developed by the IETF (Internet Engineering Task Force); it is also termed as the Internet version 2. Similar to IPv4, IPv6 also works on the OSI layer 3 (network layer). However, in contrast to IPv4 (which is 32 bits long and offers around 4,294,967,296 addresses), IPv6 has a massive logical address range (which is 128 bits long). Additional features in IPv6 include auto-configuration features, end-to-end connectivity, inbuilt security measures (IPSec), provision for faster routing, support for mobility, and many others. These features not only make IPv6 practical for use in IoT but also makes it attractive for a majority of the present-day and upcoming IoT-based deployments. Interestingly, as IPv6 was designed entirely from scratch, it is not backward compatible; it cannot be made

to support IPv4 applications. Figure 26 shows the differences between IPv4 and IPv6 packet structures.

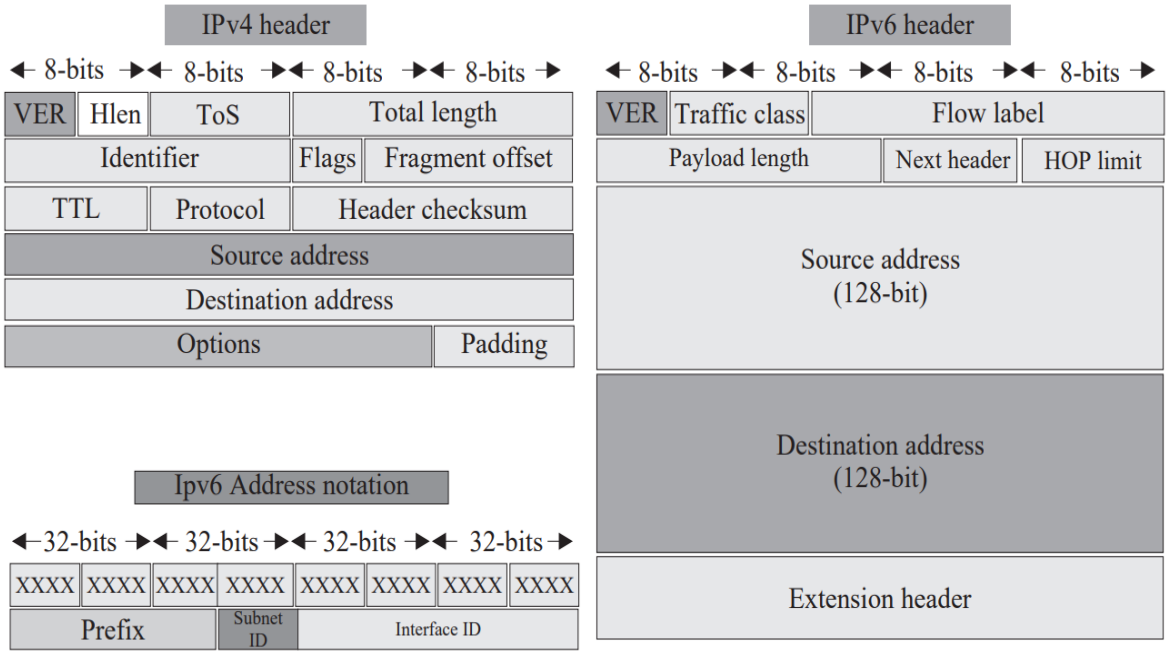


Figure 28: Differences between IPv4 and IPv6 packets and the IPv6 address notation [20]

4.3.1 IPv6 Addressing

The IPv6 addressing scheme has a crucial component: the interface identifier (IID). IID is made up of the last 64 bits (out of the 128 bits) in the IPv6 address. IPv6 incorporates the MAC (media access control) address of the system for IID generation. As a device's MAC address is considered as its hardware footprint and is globally unique, the use of MAC makes IID unique too. The IID is auto-configured by a host using IEEE's extended unique identifier (EUI-64) format. IPv6 supports three types of unicasting: Global unicast address (GUA), link local address (LL), and unique local address (ULA). The GUA is synonymous with IPv4's static addresses (public IP). It is globally identifiable and uniquely

addressable. The global routing prefix is designated by the first (most significant) 48 bits. The first three bits of this routing prefix is always set to 001; these three bits are also the most significant bits of this prefix. In contrast, LLs are auto-configured IPv6 addresses, whose communication is limited to within a network segment only (under a gateway or a router). The first 16 bits of LL addresses are fixed and equals FE80 in hexadecimal. The subsequent 48 bits are set to 0. As these addresses are not routable, the LLs' scope is restricted to within the operational purview of a router or a gateway. Finally, ULAs are locally global and unique. They are meant for use within local networks only. Packets from ULAs are not routed to the Internet. The first half of an ULA is divided into four parts and the last half is considered as a whole. The four parts of the first part are the following: Prefix, local bit, global ID, and subnet ID, whereas the last half contains the IID. ULA's prefix is always assigned as FD in hexadecimal (1111 110 in binary). If the least significant bit in this prefix is assigned as 1, it signifies locally assigned addresses.

4.4 TCP/IP Transport Layer

The transport layer is the third layer in the TCP/IP protocol suite and is an important connectivity entity as it acts as the interlocutor for the clients and the servers in a client–server paradigm. This layer forms the core of the TCP/IP protocol suite as it provides logical mechanisms for data exchanges between two or more points over the Internet. As mentioned in the previous sections, the transport layer engages in networking functionalities such as process-to-process

communication, encapsulation and de-capsulation of data, multiplexing and de-multiplexing of virtual pathways, flow control, error control, and congestion control.

From a broader perspective, the transport layer provides two types of services:

1) Connectionless and 2) connection-oriented.

These service types at the transport layer determine the degree of interdependence between transmitted packets. The application layer for both service types first divides a message into smaller chunks, which are then acceptable by the transport layer for further transmission. These chunks are sequentially forwarded from the application layer to the transport layer. Upon receiving these chunks, the transport layer encapsulates these into packets for transmission. Generally, the packets in a connectionless transport layer service are independent of one another; whereas the packets in a connection-oriented service are dependent on one another.

4.5 Wi-Fi

Wi-Fi or Wi-Fi is technically referred to by its standard, IEEE 802.11, and is a wireless technology for wireless local area networking of nodes and devices built upon similar standards (Figure 29).Wi-Fi utilizes the 2.4 GHz ultra-high frequency (UHF) band or the 5.8 GHz super high frequency (SHF) ISM radio bands for communication. For operation, these bands in Wi-Fi are subdivided into multiple channels. The communication over each of these channels is achieved by multiple devices simultaneously using time-sharing based TDMA multiplexing. It uses CSMA/CA for channel access.

Various versions of IEEE 802.11 have been popularly adapted, such as a/b/g/n. The IEEE 802.11a achieves a data rate of 54 Mbps and works on the 5 GHz band using OFDM for communication. IEEE 802.11b achieves a data rate of 11 Mbps and operates on the 2.4 GHz band. Similarly, IEEE 802.11g also works on the 2.4 GHz band but achieves higher data rates of 54 Mbps using OFDM. Finally, the newest version, IEEE802.11n, can transmit data at a rate of 140 Mbps on the 5 GHz band.

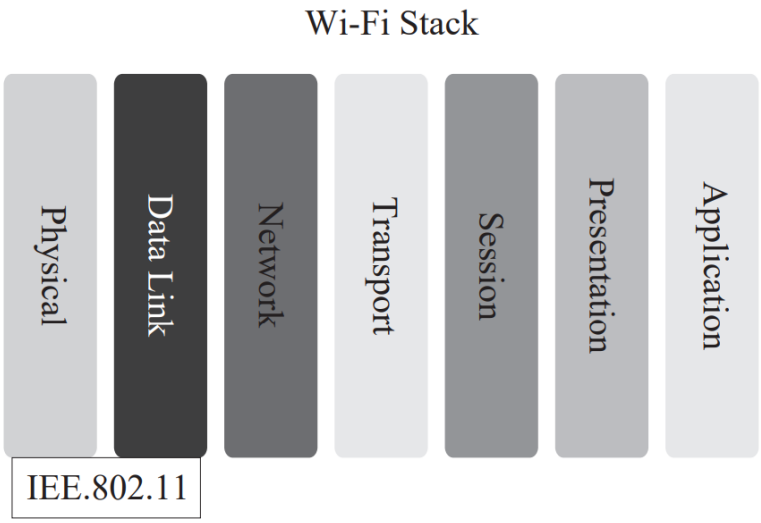


Figure 29: The IEEE 802.11 Wi-Fi stack [20]

Wi-Fi devices can network using a technology referred to as wireless LAN (WLAN), as shown in Figure 30. A Wi-Fi enabled device has to connect to a wireless access point, which connects the device to the WLAN. WLAN is then responsible for forwarding the messages from the devices to and from between the devices and the Internet. Wi-Fi devices can network using a technology referred to as wireless LAN (WLAN). A Wi-Fi enabled device has to connect to a wireless access point, which connects the device to the WLAN. WLAN is then

responsible for forwarding the messages from the devices to and from between the devices and the Internet.

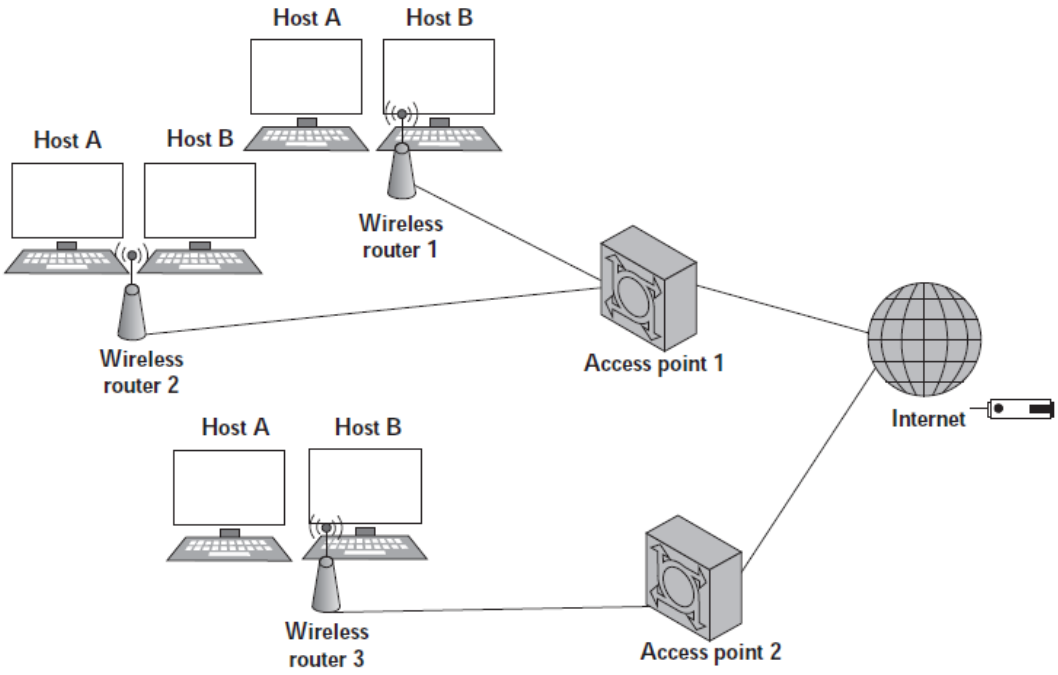


Figure 30: The Wi-Fi Deployment Architecture [20]

4.6 HTTP Protocol

HTTP is a protocol that is used to retrieve resources such as HTML pages. It is the backbone of all data transmission on the Web and is a client-server protocol, which implies that requests are made by the recipient, which is typically the Web browser. A full document is reconstructed from the many sub-documents retrieved, such as text, layout description, photos, videos, scripts, and others.

Clients and servers communicate by sending and receiving individual messages (as opposed to a stream of data). Requests are messages sent by the client, which is commonly a Web browser, and replies are messages delivered by the server as an answer.

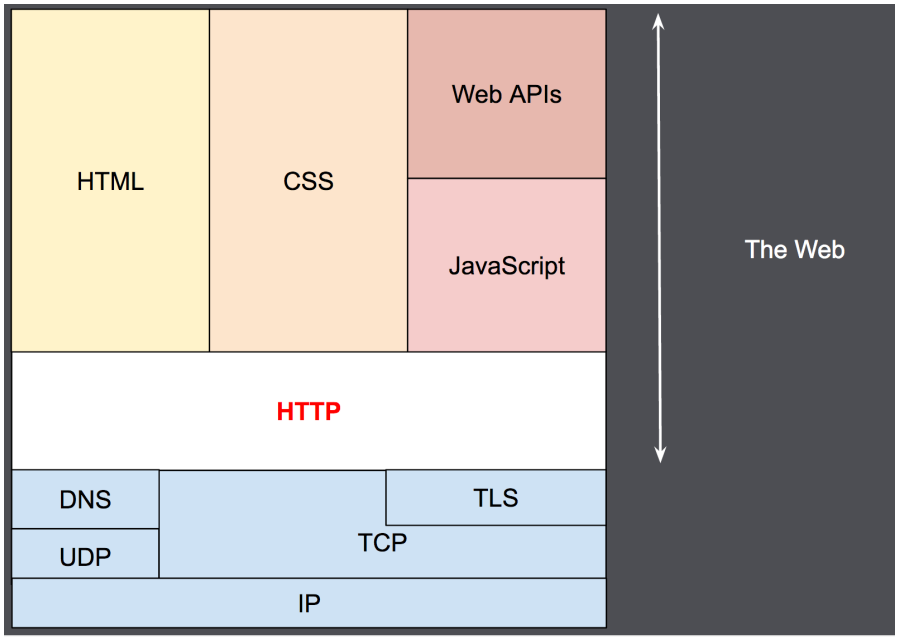


Figure 31: Layers of HTTP [54]

HTTP, which was created in the early 1990s, is an adaptable protocol that has changed through time. It is an application layer protocol that is communicated over TCP or a TLS-encrypted TCP connection, however it could potentially be sent over any trustworthy transport protocol. Because of its extensibility, it is used to retrieve not just hypertext pages, but also pictures and videos, as well as to publish material to servers, as with HTML form results. HTTP may also be used to get portions of documents in order to dynamically update Web sites.

4.6.1 HTTP Components

HTTP is a client-server protocol, which means that requests are submitted by a single entity, the user-agent (or a proxy on behalf of it). The user-agent is often a Web browser, although it can be anything, such as a robot that crawls the Web to populate and maintain a search engine index.

Each request is sent to a server, which processes it and returns an answer known as the response. Between the client and the server, there are multiple entities known as proxies that provide various functions such as gateways or caches.

In fact, there are additional computers between a browser and the server processing the request: routers, modems, and other devices. These are buried in the network and transport layers due to the Web's tiered nature. At the application layer, HTTP is on top. Although useful for troubleshooting network issues, the underlying layers are usually irrelevant to the explanation of HTTP.

- **Client: The user-agent**

  Any instrument that works on behalf of the user is referred to as a user-agent. This function is typically done by the Web browser, but it may also be performed by tools used to debug applications by engineers and Web developers.

  The browser is always the one who makes the request. It is never the server's fault (though some mechanisms have been added over the years to simulate server-initiated messages).

  The browser sends an initial request to fetch the HTML document that represents the page in order to display it. It then parses this file, making further requests for execution scripts, layout information (CSS) to display, and page sub-resources (usually images and videos). The Web browser then assembles these resources to display the entire content, the Web page. Scripts run by the

browser might retrieve more resources later on, and the browser will update the Web page accordingly.

A hypertext document is a Web page. This implies that part of the shown content is a link that may be triggered (typically with a mouse click) to retrieve a new Web page, allowing the user to direct their user-agent and surf the Web. The browser converts these instructions into HTTP requests and then interprets the HTTP answers to provide a clear response to the user.

❖ **Web Server**

The server, on the other end of the communication connection, serves the document requested by the client. Virtually, a server appears to be a single machine; however, it may be a collection of servers sharing the load (load balancing), or a complex piece of software interrogating other computers (such as cache, a database server, or e-commerce servers), completely or partially generating the document on demand.

A server is not always a single machine; many server software instances might run on the same hardware. They may even share the same IP address using HTTP/1.1 and the Host header.

4.7 WebSocket

Websocket is an IETF (Internet Engineering Task Force) standardized full-duplex communication protocol. WebSockets (WS), an OSI layer seven protocol, enables reliable and full-duplex communication channels over a single TCP connection. Figure 32 shows the position of a WebSocket layer in a stack. The

WS relies on the OSI layer 4 TCP protocol for communication. Despite being different from the HTTP protocol, WS is compatible with HTTP and can work over HTTP ports 80 and 443, enabling support for network mechanisms such as the HTTP proxy, which is usually present during organizational Internet accesses through firewalls.

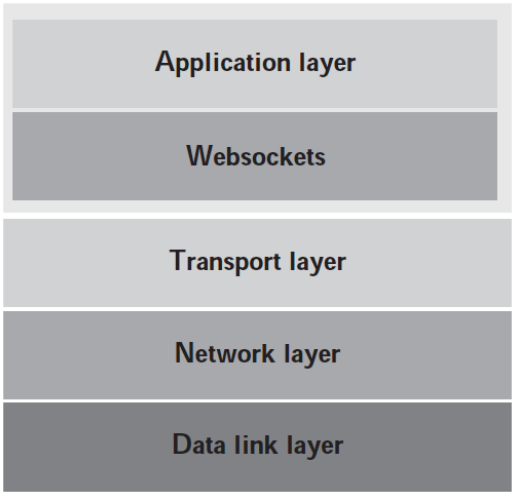


Figure 32: A representation of the position of WebSockets in a stack [20]

WS enables client–server interactions over the Web. Web servers and clients such as browsers can transfer real-time data between them without incurring many overheads. Upon establishment of a connection, servers can send content to clients without the clients requesting them first. Messages are exchanged over the established connection, which is kept open, in a standardized format. Support for WS is present in almost all modern-day browsers; however, the server must also include WS support for the communication to happen.

The full-duplex communication provided by WS is absent in protocols such as HTTP. Additionally, the use of TCP (which supports byte stream transfers) is also enhanced by enabling it to provide message stream transfers using WS. Before

the emergence of WS, comet channels were used for attaining full-duplex communication over port 80. However, comet systems were very complicated and incurred significant overheads, which made their utility limited for constrained application scenarios mainly associated with IoT.

Websocket (WS) and WebSocket secure (WSS) have been specified as uniform resource identifier (URI) schemes in the WS specification, which are meant for unencrypted and encrypted connections, respectively. The WS handshake process and the frames can be quickly inspected using browser development tools.

4.7.1 Operational Principle of WebSocket

A client initiates the WS connection process by sending a WS handshake request. In response, a WS server responds with a WS handshake response. As the servers have to incorporate both HTTP and WS connections on the same port, the handshaking is initiated by an HTTP request/response mechanism. Upon establishment of a connection between the client and server, the WS communication separates as a bidirectional protocol that is non-conformant with the HTTP protocol. The WS client sends an update header and a sec-WebSocket-Key header, which contains base64 encoded random bytes. The server responds to the client's request using a hash of the key included in the Sec-WebSocket-Accept header. This allows the WS to overcome a caching proxy's efforts to resend previous WS communication. A fixed string, 258EAFA5- E914-47DA-95CA-C5AB0DC85B11, is appended to the un-decoded value from the Sec-WebSocket-Key header by a hashing function using the SHA (secure hash

algorithm)-1, which is finally encoded using base64 encoding. Once the WS full-duplex connection is established, minimally framed data (small header and a payload), which may be data or text, can be exchanged.

The WS transmissions or messages can be further split into multiple data frames whenever the full message length is not available during message transfer. This feature is occasionally exploited to include/multiplex several simultaneous streams, using simple extensions to the WS protocol. This multiplexing avoids the monopoly of a single large payload over the WS port.

4.7.2 WebSocket Protocol

The WebSocket protocol is a form of framed protocol that uses discrete chucks with each packet. It also uses a frame type, data section, and payload length to ensure correct operation. Knowing the basic blocks of the WebSocket protocol is essential for a thorough comprehension of the protocol. The main points are listed below.

- The Fin Bit is the foundation of the WebSocket. It will be produced automatically after the connection is established.
- RSV1, RSV2, and RSV3 bits are retained for future chances.
- Opcode is a component of every frame that describes the process of understanding the payload data of that frame. Common opcode values include 0x00, 0x0, 0x02, 0x0a, 0x08, and many more.
- When one bit is set to 1, the mask bit is activated.

For all payload data, WebSocket requires the usage of a client-selected random key. When paired with payload data, the masking key facilitates payload data sharing in an XOR operation. Masking prevents cache misinterpretation or cache poisoning, which is critical for application API security.

Payload length

In WebSocket, this is utilized for the total length encoding of the payload data. When the encoded data length is less than 126 bytes, payload len is shown. When the payload data length exceeds 126 bytes, extra fields are utilized to describe the length.

Masking key

Each frame sent by the client to the server is masked with a 32-bit value. When the mask bit is set to 1, the masking key appears. The masking key will be zero if the mask bit is set to 0.

Payload data

Payload data includes any arbitrary application data and extension data. This data is utilized in the early WebSocket handshakes between the client and server for negotiation.

4.8 Object Detection

Object detection model aimed at localizing and identifying multiple objects in a single image. This model is a TensorFlow.js port of the COCO-SSD model. This model recognizes items in the COCO dataset, a large-scale object detection,

segmentation, and captioning dataset. The model can detect 80 different types of objects. (SSD is an abbreviation for Single Shot MultiBox Detection.)

This TensorFlow.js model does not need any prior knowledge of machine learning. It can take any browser-based image element as input (for example, img>, video>, canvas> elements) and return an array of bounding boxes with class name and confidence level.

4.8.1 What is COCO?

COCO is a large-scale dataset for object identification, segmentation, and captioning. COCO has several features:

- Object segmentation
- Recognition in context
- Superpixel stuff segmentation
- 330K images (>200K labeled)
- 1.5 million object instances
- 80 object categories
- 91 stuff categories
- 5 captions per image
- 250,000 people with keypoints

4.8.2 Object Detector – COCO SSD

The problem of detecting one or more items in an image, drawing bounding boxes around them, and determining the class of the objects has long been a basic difficulty in computer vision. The TensorFlow Object Identification API is a

TensorFlow-based open source framework that makes it simple to build, train, and deploy object detection models.

This framework supports a variety of object detection architectures, including SSD (Single Shot Detector) and Faster R-CNN (Faster Region-based Convolutional Neural Network), as well as feature extractors such as MobileNet and Inception. There are several architectures and extractors to choose from, but which one to employ depends on the use-case - what accuracy and speed is required.

The multimodal toolbox includes a real-time object identification implementation based on the COCO SSD model from Tensorflow. The application accepts a video as input (by camera or uploaded) and then recognizes all the items visible in each frame, returning their positions, class, and confidence score.

Figure 33: Example of object detection with confidence score [46]

4.8.3 The Process

The pre-trained COCO-SSD model is among the fastest and most accurate tensorflow architectures, allowing it to be utilized in a browser. COCO refers to the "Common Objects in Context" dataset, which was used to train the model. This picture collection is mostly used for object identification, segmentation, and captioning, and it contains about 200k annotated photos categorized as "person," "bus," "zebra," and "tennis racket." SSD, which stands for Single Shot Detector, is a neural network architecture comprised of a single feed-forward convolutional neural network that predicts the labels of the image's objects as well as their location during the same action. The opposite of this "single-shot" feature is an architecture that employs a "proposal generator," a component that searches for regions of interest within a picture.

After identifying the regions of interest, the next step is to extract the visual properties of these regions and determine whether objects are present in them, a process known as "feature extraction." The default feature extractor for COCO-SSD is lite MobileNet v2, which is based on the MobileNet architecture.

Simply choose the 'Real Time Object Detection' option from the tools page to activate the real time object detection capability. When the website loads, make sure the webcam is turned on, and the function will recognize all the items in the video frame that have been trained in the COCO-SSD model and report information for each object identified. Users may also capture or post videos, as well as extract data from the objects. If the model discovers one or more items in

each frame it monitors, it will report the class/category (person, sofa, etc), statistics of the bounding box, and confidence score of the object.

4.8.4 80 Classes Names in COCO Dataset

Person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, traffic light, fire hydrant, stop sign, parking meter, bench, bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe, backpack, umbrella, handbag, tie, suitcase, frisbee, skis, snowboard, sports ball, kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket, bottle, wine glass, cup, fork, knife, spoon, bowl, banana, apple, sandwich, orange, broccoli, carrot, hot dog, pizza, donut, cake, chair, couch, potted plant, bed, dining table, toilet, TV, laptop, mouse, remote, keyboard, cell phone, microwave, oven, toaster, sink, refrigerator, book, clock, vase, scissors, teddy bear, hair drier, toothbrush.

4.8.5 TensorFlow Object Detection API

The TensorFlow object detection API (Application Programming Interface) provides a platform for building a deep learning network that can detect objects. They have pre-trained models in their framework, which they call Model Zoo. This comprises a set of pre-trained models that have been trained on the COCO dataset.

4.9 SSD: Single Shot MultiBox Detector in TensorFlow

SSD is a unified object detection framework that operates on a single network. The original Caffe code has been re-implemented in TensorFlow in this repository. It now only implements VGG-based SSD networks (with 300 and 512

inputs), but the project's design is modular, which should make the implementation and training of other SSD versions simple (ResNet or Inception based for instance). The current TF checkpoints have been transferred directly from SSD Caffe models.

The organisation is modelled around the TF-Slim models repository, which contains implementations of popular designs (ResNet, Inception and VGG). As a result, it is divided into three major sections:

- Datasets: interface to common datasets (Pascal VOC, COCO...) as well as scripts to convert those to TF-Records.
- Networks: SSD network definition, as well as standard encoding and decoding methods
- Pre-processing: methods for pre-processing and data augmentation inspired by the original VGG and Inception implementations.

4.9.1 How Does the SSD Work?

4.9.1.1 Image classification and Object detection

In computer vision, image classification predicts the item in an image, whereas object detection not only predicts the object but also detects its position in terms of bounding boxes. For example, when we create a swimming pool classifier, we take an input image and predict if it has a pool, but an object detection model would also tell us where the pool is located.

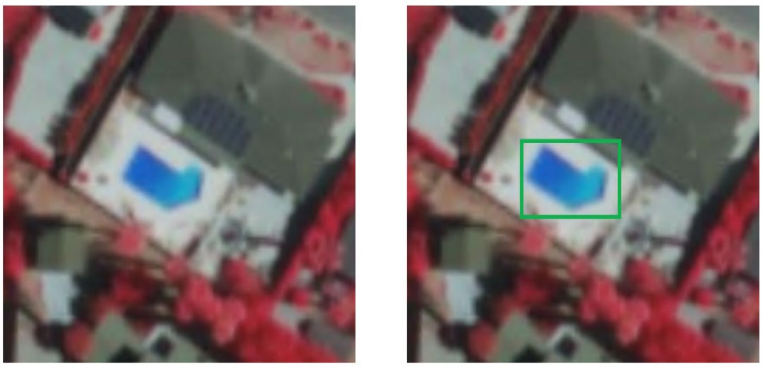


Figure 34: Difference between classification and object detection [44]

As an example, assuming just one class and one object in an image, the output of an object identification model should include:

- Possibility of finding an object.
- The bounding box's height.
- The bounding box's width.
- Horizontal coordinate of the bounding box's center point.
- The vertical coordinate of the bounding box's center point.

This is simply one of the output specification conventions. Different models and implementations may use different forms, but the aim is to output the probability and position of the item.

4.9.1.2 Why would a sliding window method not work?

It makes sense to develop an object recognition model on top of an image classification model. Once we have a decent image classifier, we can easily detect objects by dragging a 'window' across the image and classifying whether the image in that window (cropped out portion of the image) is of the required kind. Sounds easy! Well, there are at least two problems:

- How do you determine what size window to use such that the object is always visible? Different sorts of items (for example, a palm tree and a swimming pool) and even the same type of object (for example, a tiny building and a huge structure) can have varied sizes.
- Aspect ratio (the ratio of height to breadth of a bounding box) (the ratio of height to width of a bounding box). Many items can exist in diverse forms, such as a building footprint having a different aspect ratio than a palm tree.

To solve these problems, we would have to experiment with various sizes/shapes of sliding windows, which would be very computationally intensive, especially with deep neural networks.

Figure 35: Example of sliding window approach [44]

In practice, dominant object identification algorithms fall into two categories. R-CNN and Fast(er) R-CNN algorithms employ a two-step strategy, first identifying regions where objects are predicted to be located, and then detecting objects solely in those regions using ConvNet. YOLO (You Only Look Once) and SSD

(Single-Shot Detector) algorithms, on the other hand, use a fully convolutional approach in which the network is able to find all objects within an image in one pass (hence 'single-shot' or 'look once') through the ConvNet. Region proposal algorithms often have somewhat greater accuracy but are slower to execute, whereas single-shot methods are more efficient and have comparable accuracy, and this is what we will focus on in this section.

4.9.1.3 Single Shot Detector (SSD)

SSD is made up of two parts: a backbone model and an SSD head. As a feature extractor, the backbone model is typically a pre-trained image classification network. This is typically a ResNet-trained network that has had the final fully connected classification layer removed. As a result, we have a deep neural network that can extract semantic meaning from an input image while keeping its spatial structure, although at a lesser resolution. The backbone for ResNet34 produces 256 7x7 feature maps for an input image. The SSD head is just one or more convolutional layers added to this backbone, with the outputs read as bounding boxes and classifications of objects in the spatial position of the final layer activations.

The first few layers (white boxes) in figure 35 represent the backbone, while the last few layers (blue boxes) represent the SSD head.

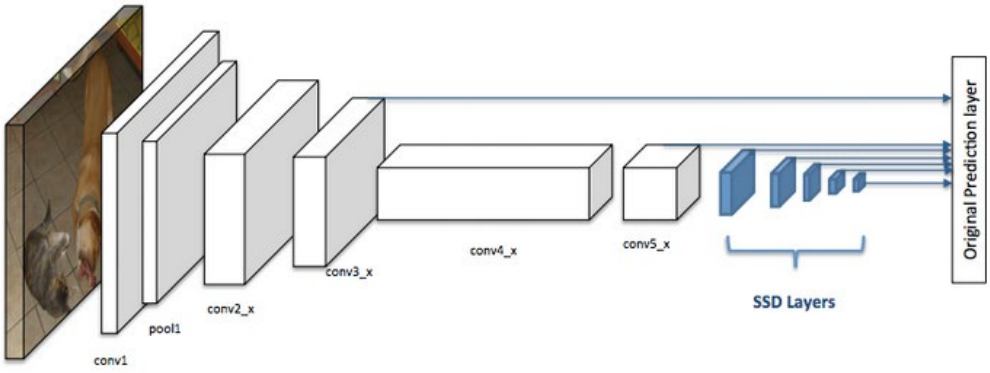


Figure 36: Architecture of a convolutional neural network with a SSD detector [44]

4.9.1.4 Grid Cell

Instead of a sliding window, SSD splits the picture with a grid and assigns each grid cell to identify objects in that section of the image. Detecting objects is simply predicting the type and location of an object within that region. If there is no object, we take it to be the background class, and the location is omitted. In the example below, for example, we may use a 4x4 grid. Each grid cell can output the position and geometry of the item contained within it.

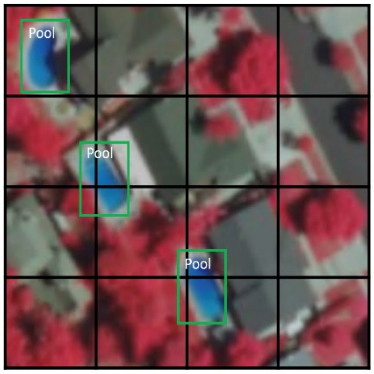


Figure 37: Example of a 4x4 grid [44]

Now you might be wondering what if there are multiple objects in one grid cell or we need to detect multiple objects of different shapes. There is where anchor box and receptive field come into play.

4.9.1.5 Anchor Box

Each grid cell in SSD can have several anchor/prior boxes attached to it. These anchor boxes are pre-defined, and each one is in charge of the size and shape of a grid cell. In the illustration below, for example, the swimming pool correlates to the taller anchor box, while the structure belongs to the broader box.

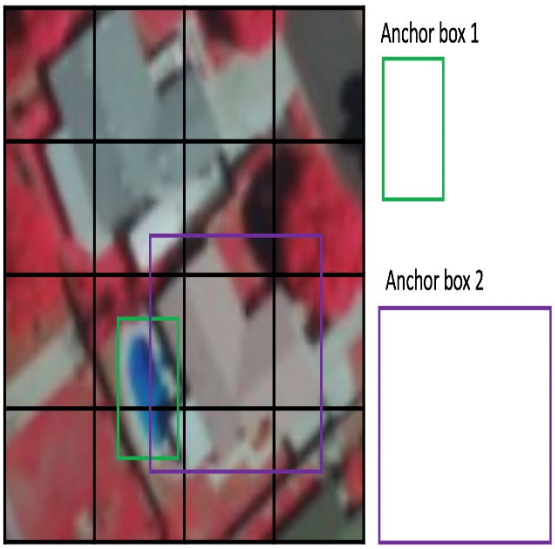


Figure 38: Example of two anchor boxes [44]

During training, SSD employs a matching phase to match the proper anchor box with the bounding boxes of each ground truth item inside an image. Essentially, the anchor box with the greatest degree of overlap with an item is in charge of predicting the class and position of that object. This attribute is utilized to train the network and, once trained, to forecast the identified objects and their positions. In practice, each anchor box has its own aspect ratio and zoom level.

4.9.1.6 Aspect Ratio

Not everything has a square form. Some are longer and broader to varied degrees. To cater for this, the SSD architecture provides for pre-defined aspect ratios of

the anchor boxes. At each zoom/scale level, the ratios parameter may be used to describe the different aspect ratios of the anchor boxes associated with each grid cell.

Figure 39: The bounding box of building 1 is higher, while the bounding box for building 2 is wider [44]

4.9.1.7 Zoom Level

The anchor boxes do not have to be the same size as the grid cells. We can be looking for smaller or larger things within a grid cell. The zooms option specifies how much the anchor boxes should be scaled up or down in relation to each grid cell. Buildings are often larger than swimming pools, as seen by the anchor box example.

4.9.1.8 Receptive Field

The region in the input space that a certain CNN's feature is looking at is referred to as the receptive field (i.e. be affected by). We shall use the terms "feature" and "activation" interchangeably here and regard them as the linear combination of the preceding layer at the relevant position (often followed by an activation

function to increase non-linearity). The convolution operation causes features at various layers to represent varying widths of regions in the input picture. The size indicated by a feature grows greater as it goes deeper. In the following example, we begin with the bottom layer (5x5) and then apply a convolution to produce the middle layer (3x3), with one feature (green pixel) representing a 3x3 section of the input layer (bottom layer). The convolution is then applied to the middle layer to produce the top layer (2x2), where each feature corresponds to a 7x7 area on the input picture. These green and orange 2D arrays are also known as feature maps, which refer to a collection of features formed by applying the same feature extractor to different parts of the input map in a sliding window fastion. The same feature map's features have the same receptive field and look for the same pattern, but at various places. This creates the spatial invariance of ConvNet.

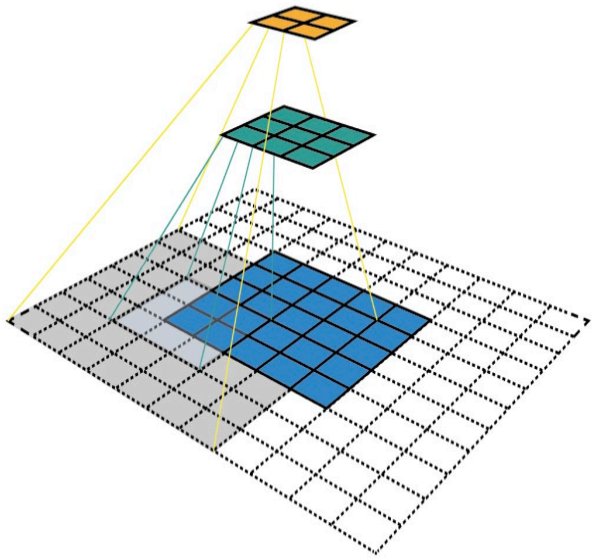


Figure 40: Visualizing CNN feature maps and receptive field [44]

The key concept of the SSD design is the receptive field, which allows us to identify objects at different sizes and produce a tighter bounding box. For an input

picture, the ResNet34 backbone generates 256 7x7 feature maps. If we supply a 4x4 grid, the most straightforward technique is to simply apply a convolution on this feature map and transform it to 4x4. This strategy can work to some extent and is exactly the concept of YOLO (You Only Look Once). SSD goes above and beyond by applying additional convolutional layers to the backbone feature map and having each of these convolution layers produce an object detection result. Predictions from previous layers aid in dealing with smaller sized items since earlier layers with smaller receptive fields may represent smaller sized objects.

4.10 Convolutional Neural Networks

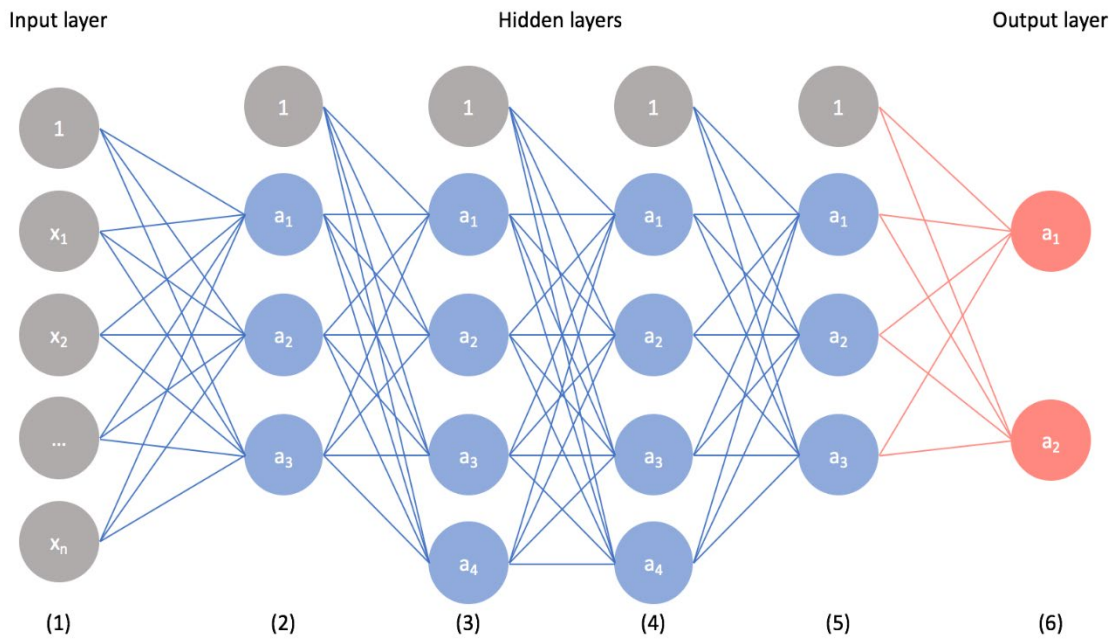


Figure 41: Feed-forward network [43]

As it turns out, there are several neural network topologies, each with their unique set of advantages. The architecture is determined by the layers we implement and

how they are connected. The neural network seen above is a feed-forward network (also known as a multilayer perceptron), which consists of a succession of completely linked layers.

Convolutional neural networks are widely employed in machine learning image recognition applications. Convolutional neural networks have an advantage over feed-forward networks in that they may address feature localization.

Consider the following scenario: we want to develop a neural network that can detect handwritten digits. For instance, given a 4 by 4 pixel picture as input, our neural network should categorize it as a "1."

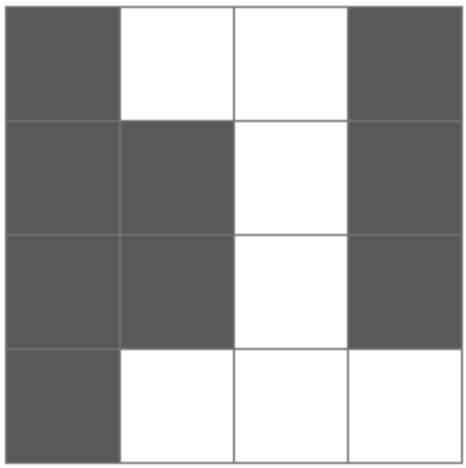


Figure 42: 4 x 4 Pixel Image [43]

Images are essentially a matrix of values relating to the intensity of light at each pixel value (white for maximum intensity, black for lowest intensity). Color pictures are commonly represented by light intensity values for red, green, and blue for each pixel value, whereas grayscale images contain a single value for each pixel. Thus, the dimensions of a 400 by 400 pixel picture are [4004001] [4004001] for a grayscale image and [4004003] [4004003] for a color image.

When we use photos for machine learning models, we usually rescale the light intensity values such that they are between 0 and 1.

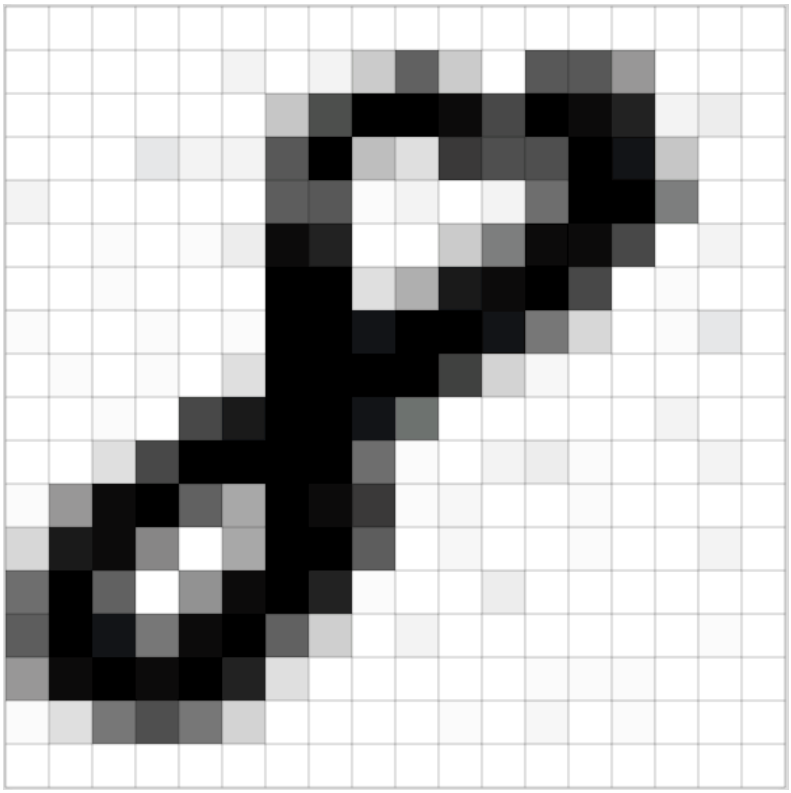


Figure 43: Pixel Image [43]

As a result, SSD enables us to construct a hierarchy of grid cells at various tiers. We could, for example, use a 4x4 grid to discover little things, a 2x2 grid to find medium-sized objects, and a 1x1 grid to find objects that span the whole image.

4.10.1 Revisiting feed-forward networks

First, consider how this might appear with a feed-forward network and identify any flaws with this technique. First and foremost, we cannot feed this picture straight into the neural network. Because a feed-forward network requires a vector of inputs, we must convert our 2D array of pixel values into a vector.

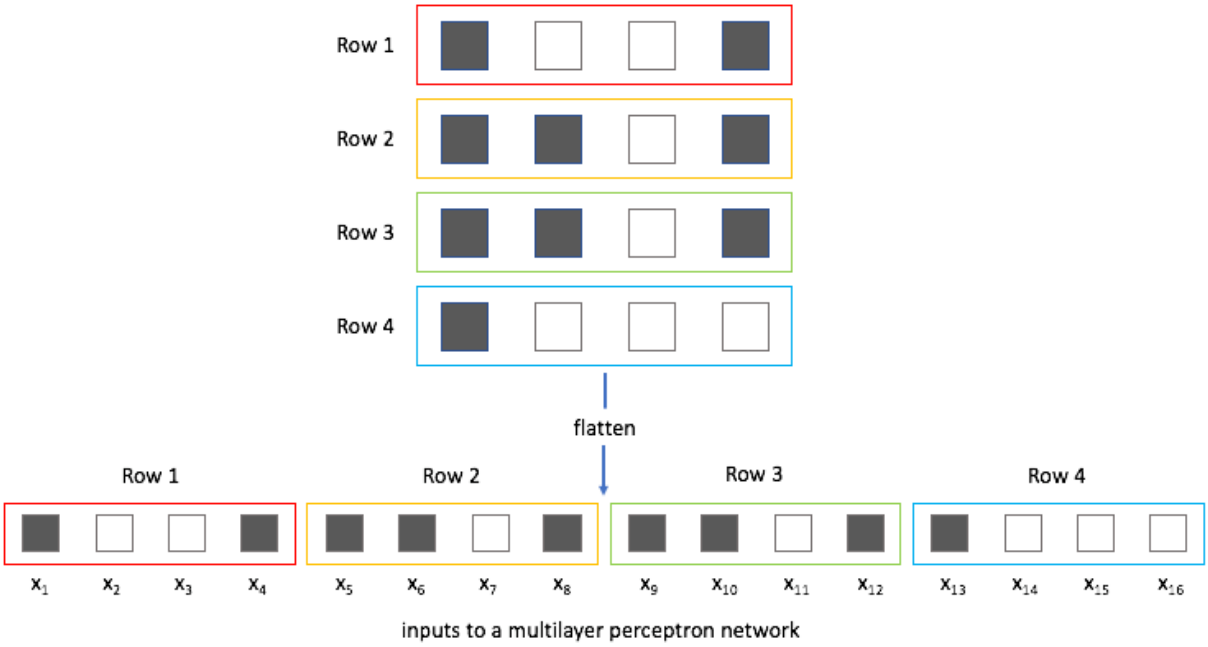


Figure 44: Multilayer network [43]

We can now use each pixel value as a feature and input it into the neural network.

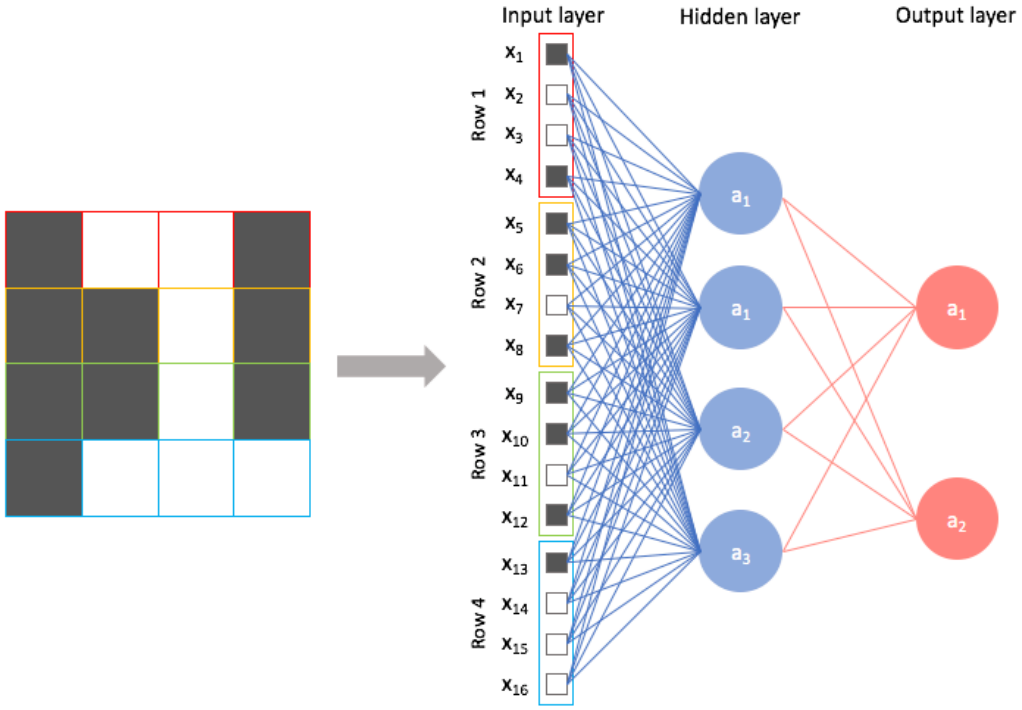


Figure 45: Neural Network Layers [43]

Unfortunately, when we transform the 2D array of pixel values into a vector, we lose a lot of information about the image; especially, we lose the spatial connections within the data.

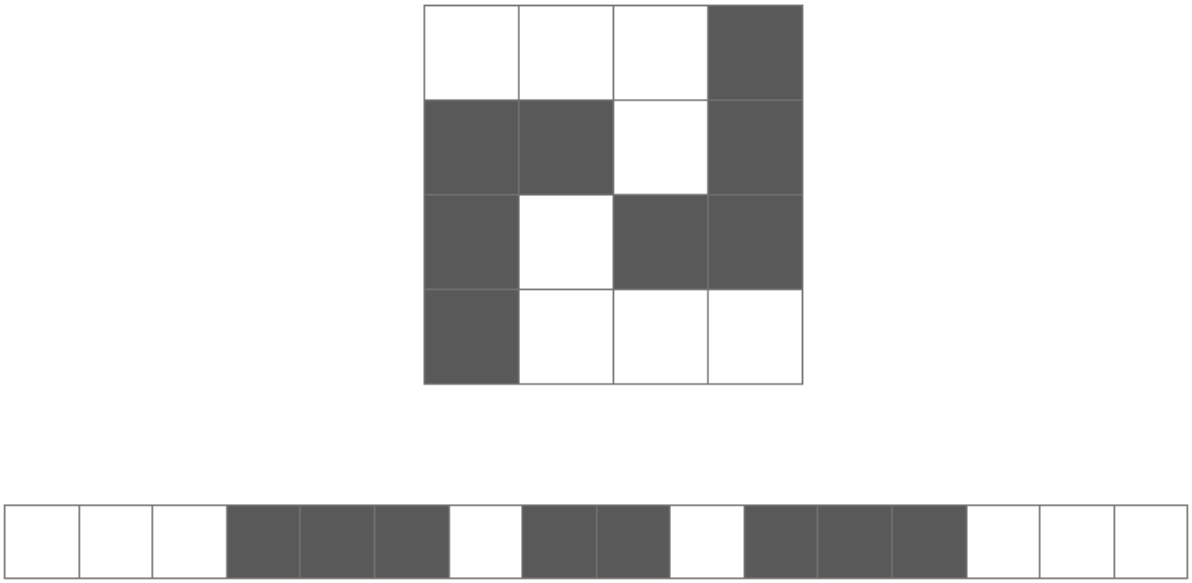


Figure 46: Numbers in flattened vector [43]

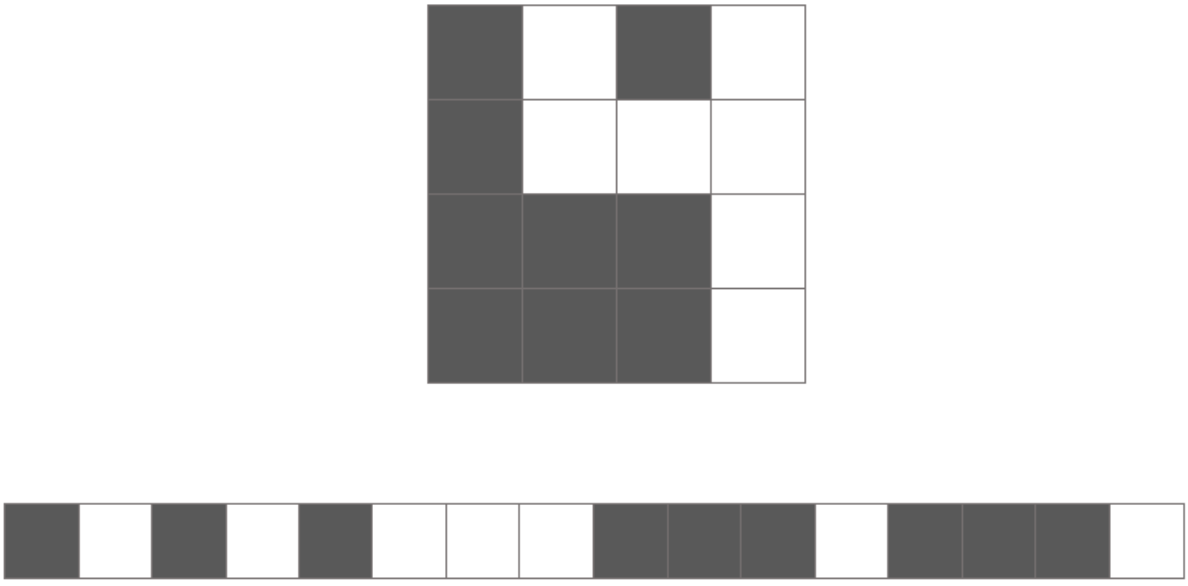


Figure 47: Numbers in flattened vector [43]

4.10.2 Convolution Layers

A convolution layer defines a window through which we analyze a portion of the picture before scanning the complete image via this window. We may parameterize the window to seek for certain characteristics (e.g. edges) inside a picture, as seen below. This window is also known as a filter because it generates an output picture that focuses primarily on the portions of the image that displayed the characteristic it was looking for. The result of a convolution is known as a feature map.

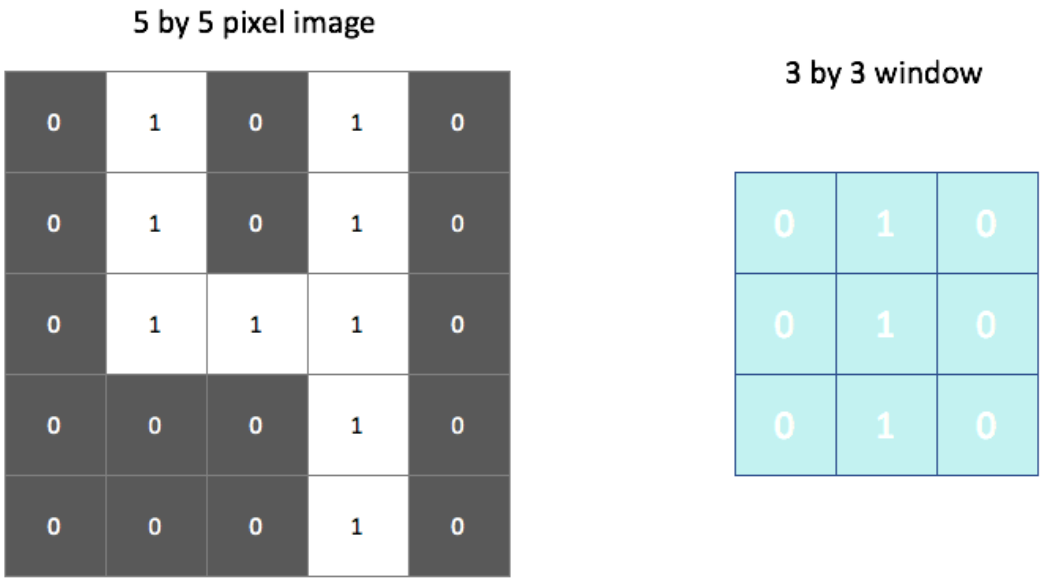


Figure 48: 5 x 5 Pixel Image and 3 x 3 Window [43]

This filter is superimposed over the picture, and the pixel and filter values are linearly combined (and then activated). The result, seen on the right, indicates parts of the picture that include a vertical line. To accurately define all of the properties of a "4", we may use a similar technique with a filter tailored to locate horizontal edges.

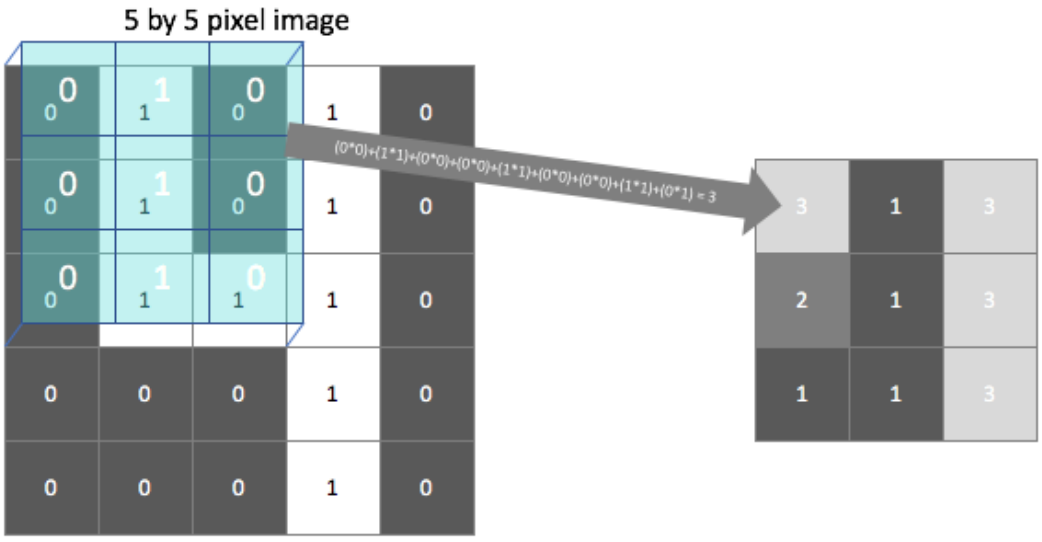


Figure 49: 5 x 5 Pixel Image and 3 x 3 Window [43]

At each step, we'll scan each filter across the picture, computing the linear combination (and subsequent activation). The input picture is shown in blue in the figure below, and the convolved image is shown in green. Consider each pixel in the convolved layer to be a neuron that receives all of the pixel values now in the window as inputs, which are then linearly combined with the relevant weights in our filter.

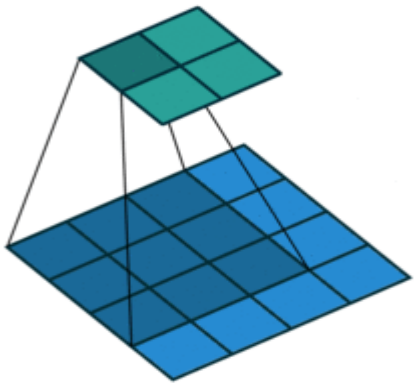


Figure 50: Input image is represented in blue and convolved image is represented in green [43]

You may also use 0-valued pixels to pad the edges of your photos so that the original image is fully scanned and its dimensions are preserved.

A typical convolutional layer, as illustrated in the example above, will apply numerous filters, each of which will return a feature mapping of the input indicating the (spatial) regions where a feature is present. Notice how certain feature mappings are more beneficial than others; in the example below, the "right diagonal" feature mapping, which looks for right diagonals in the picture, results in a dark image, indicating that the feature is not there.

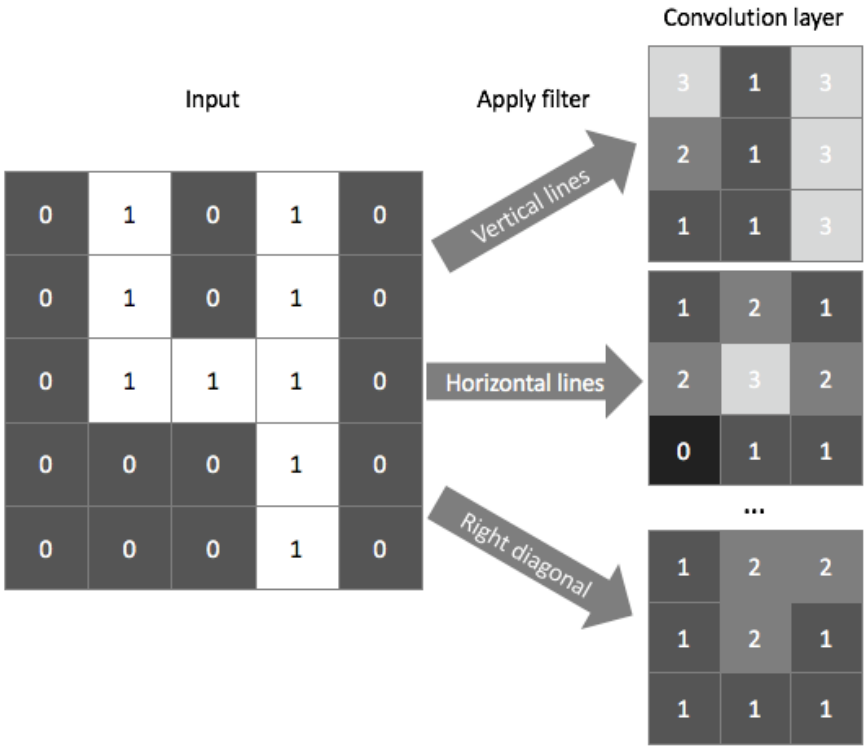


Figure 51: Feature mapping [43]

In fact, we don't directly describe the filters that will be used by our convolutional layer; instead, we parameterize the filters and let the network determine which ones to employ during training. We do, however, specify the number of filters to be used at each tier.

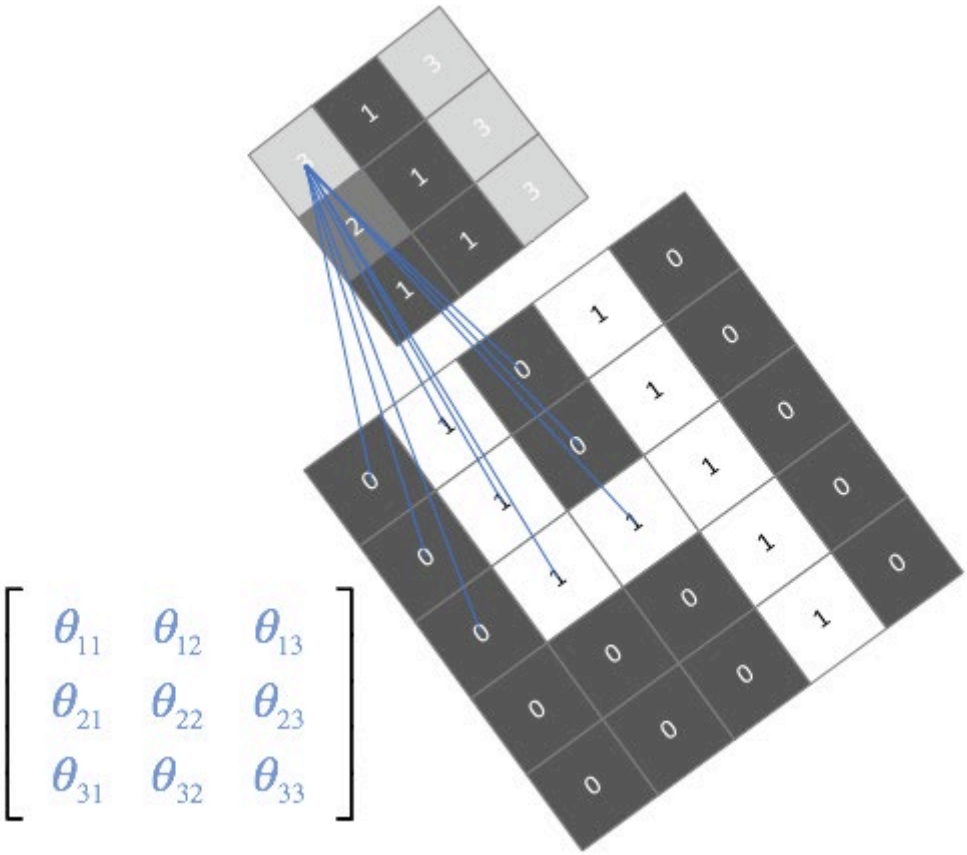


Figure 52: Convolutions on Convolutions [43]

To learn more intricate patterns within the features mapped in the previous layer, we can stack layers of convolutions together (i.e. perform convolutions on convolutions). This enables our neural network to identify broad patterns in the early layers and then zero in on patterns within patterns in the later layers.

Let's go over the feed-forward architecture again and compare it to convolutional layers now that we know more about what's going on inside them.

A feed-forward network connects each pixel to each node in the layer below, ignoring any spatial information in the image.

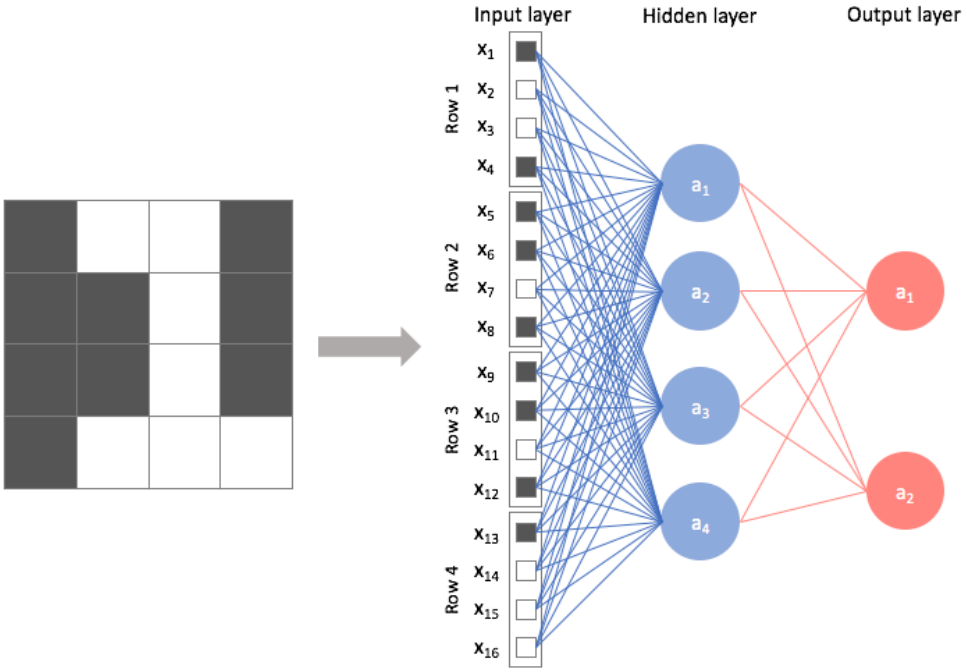


Figure 53: A feed-forward network [43]

A convolutional architecture, on the other hand, examines particular portions of the picture. In this scenario, a 2 by 2 filter with a stride of 2 is scanned across the picture to produce four nodes, each of which has localized information about the image.

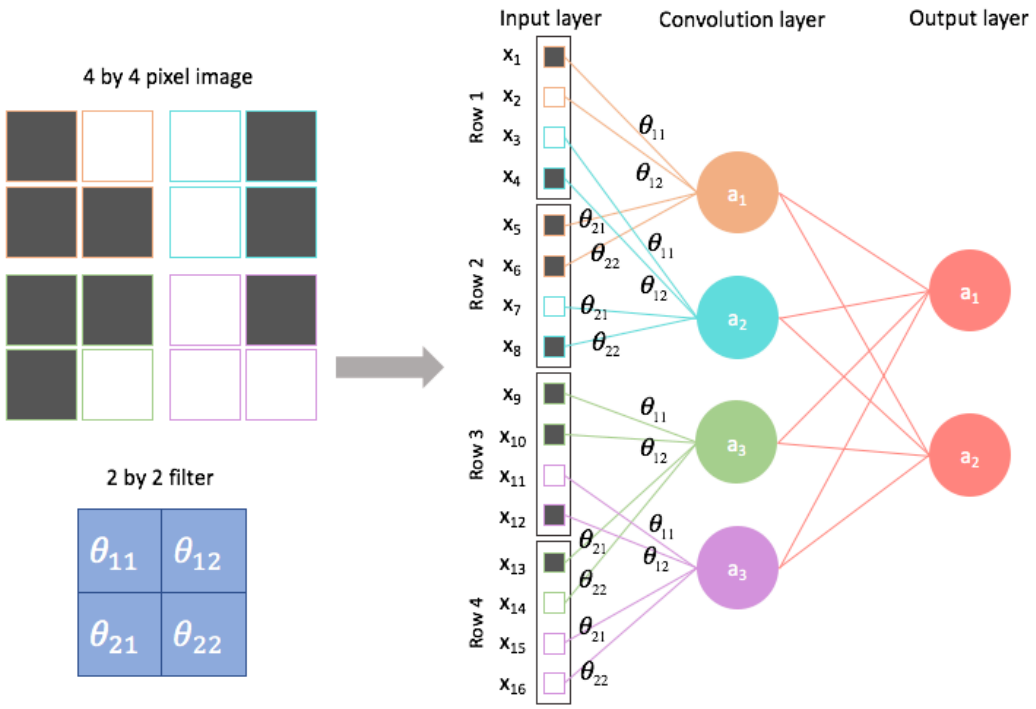


Figure 54: 2 x 2 filter with a stride of 2 [43]

The four nodes may be joined to produce the "pixels" of a feature map, where the extracted feature is determined by the filter settings. Spatial information can be saved here.

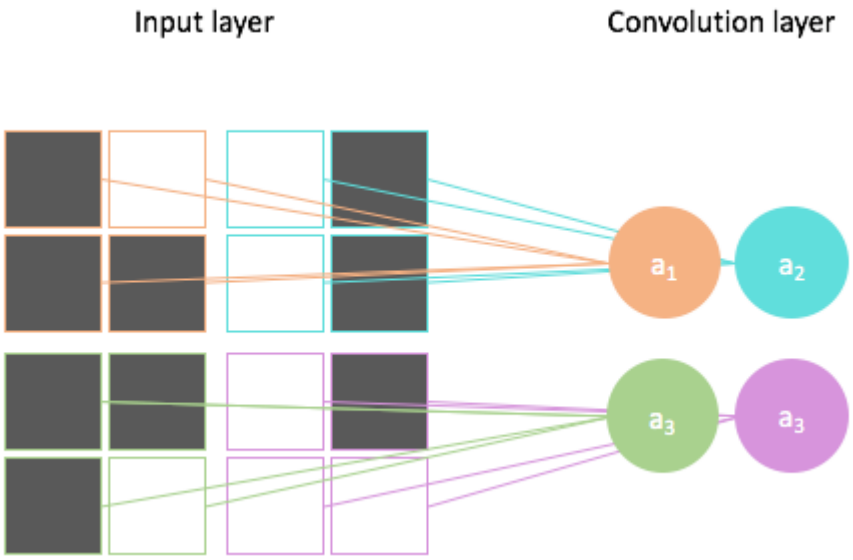


Figure 55: The four nodes joined to produce the “pixels” of a feature map [43]

We can scan the image with multiple filters to generate a variety of feature mappings. Each feature mapping will expose the sections of the picture that express the given feature as described by our filter's settings.

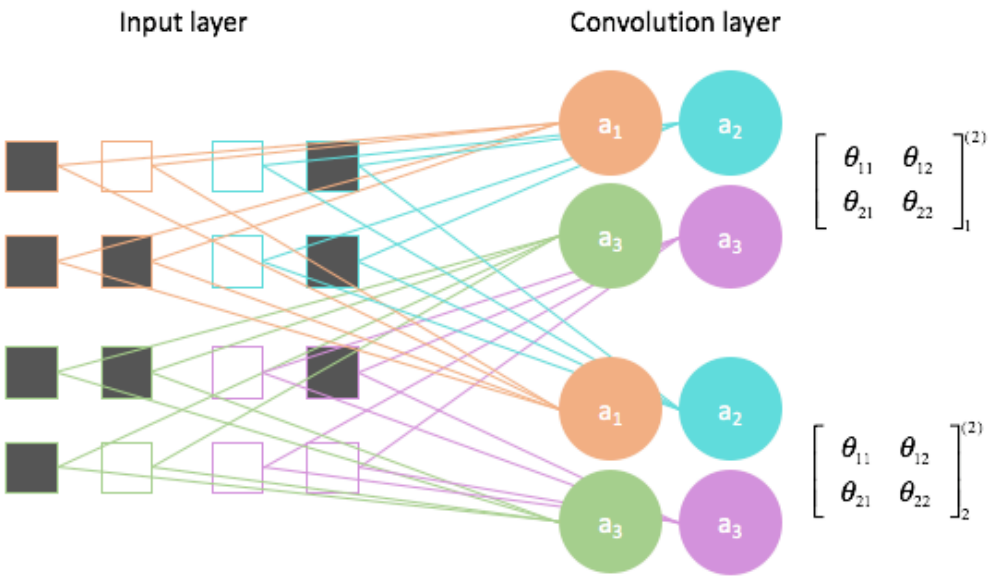


Figure 56: Multiple feature mapping of the image [43]

A convolution layer, in general, will turn one input into a stack of feature mappings of that input. You may or may not reduce the size of your input depending on how you configure your padding. The depth of the feature map stack is determined by the number of filters defined for each layer.

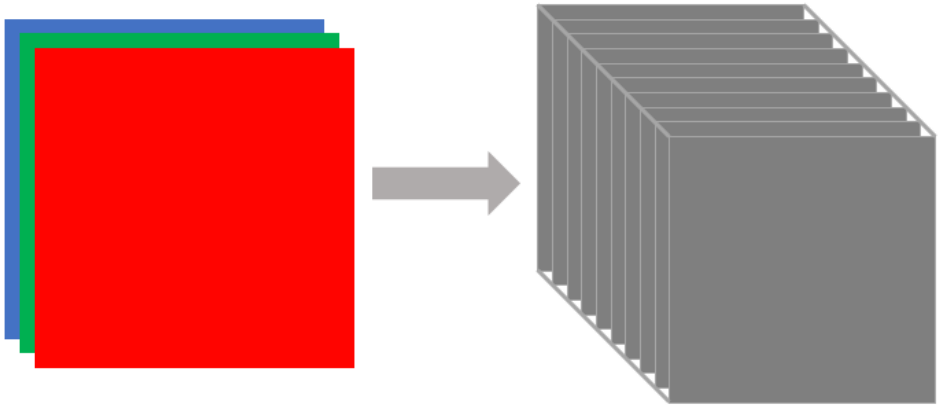


Figure 57: Input color image into a stack of feature image [43]

Each filter will have a predetermined width and height, but the depth of the filter should be the same as the depth of the input. Our filter had a depth=1 in previous cases where we looked at grayscale photos. However, if we were to convolve a color image, we would require a filter of depth=3, which includes a filter for each of the three RGB color channels.

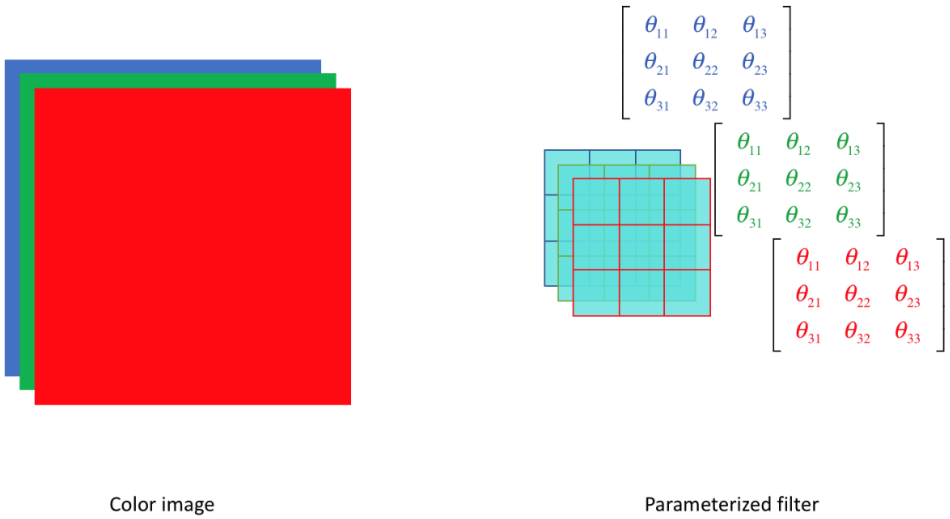


Figure 58: Input color image to parameterized filter [43]

The same phenomena occurs when we visualize convolutional networks. The network learns to extract low-level characteristics from prior layers and then combine these elements into increasingly rich data representations. This means that specialized feature maps are more likely to be found in subsequent levels of the network.

We have demonstrated how we can combine successive convolutional procedures to identify the existence of a square in the original input. Although the filter values in this example are hard-coded, the filter values in a convolutional layer are learnt.

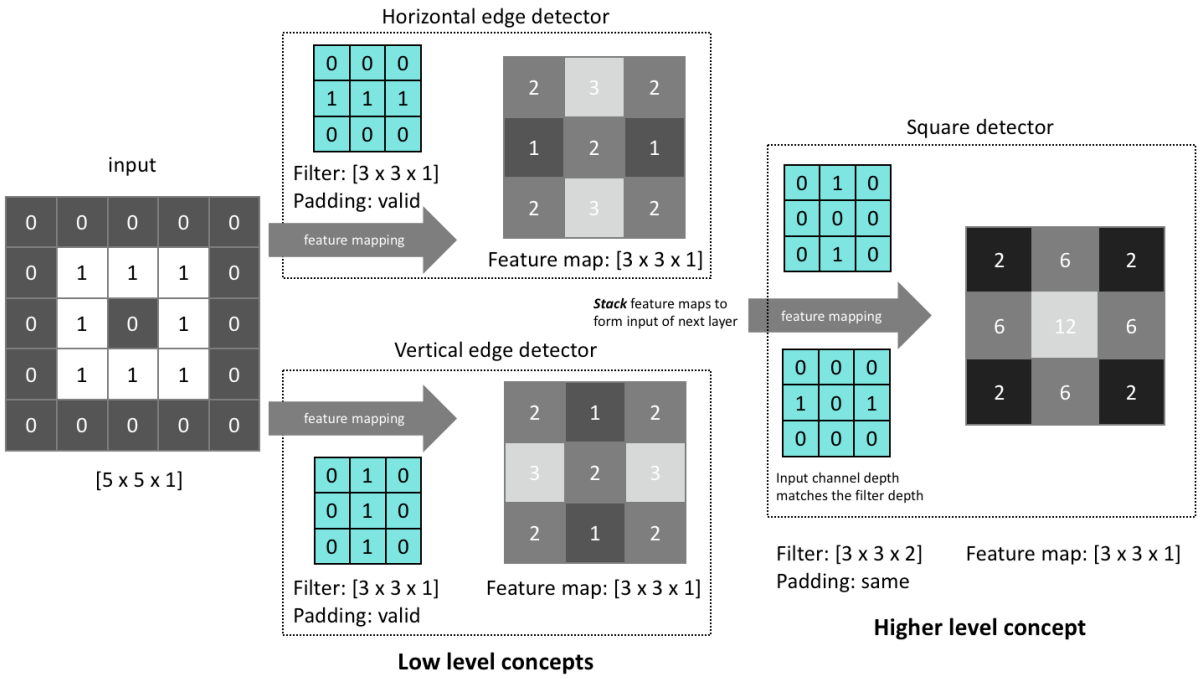


Figure 59: The filter values in a convolution layer are learned [43]

Layers in a convolutional network operate on the same fundamental concept, transforming input into meaningful representations via parameterized filters.

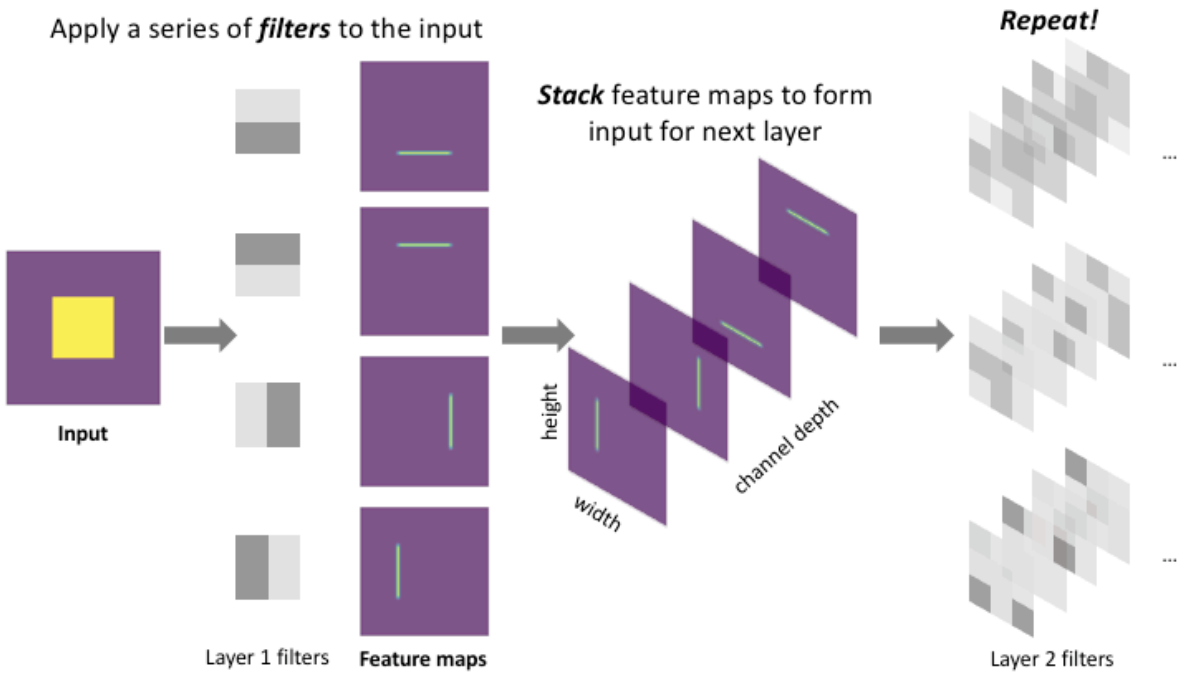


Figure 60: Layers in a Convolution Network [43]

It's very common to see an increase in channel depth (number of feature maps) as you progress into deeper layers of a network; as feature maps become more specialized, representing higher-level abstract concepts, we simply need more feature maps to represent the input. Consider how a small set of lines can be combined to form a large number of shapes, which can then be combined to form an even larger number of objects. If we want to build a model with broad representational power, we'll need to expand our feature maps accordingly.

4.10.3 Pooling Layers

To compress the spatial information of our feature mappings, we can use a pooling layer. We'll still use a window to scan across the image, but this time our goal is to compress information rather than extract specific features. We'll define the window size and stride in the same way we did with convolutional layers. Padding is not commonly used for layer pooling.

Max pooling

We simply return the maximum value (highlighted in yellow) present inside the window for each scanning location in max pooling.

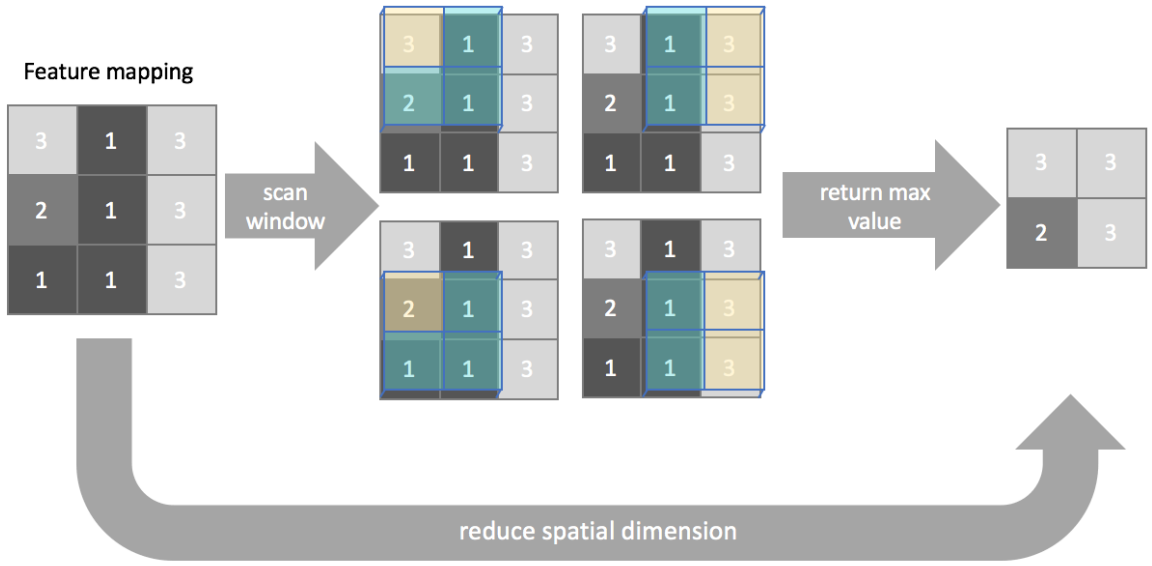


Figure 61: Max Pooling [43]

When necessary, you can define layers for average pooling or minimum pooling, though max pooling is the most commonly used pooling layer.

Global pooling

A more extreme case is global pooling, in which we define our window to match the dimensions of the input, effectively compressing the size of each feature mapping to a single value. This is useful for building convolutional neural networks in which need to be able to accept a variety of input sizes, as it compresses your feature representation from an arbitrary w × h × cw × h × c into a fixed 1 × 1 × c1 × 1 × c feature map.

Strided convolutions

Strided convolutions are an alternative method for down sampling the spatial resolution of our feature mappings, allowing for some learned down sampling rather than explicitly defining how to summarize a window (as is the case with max/average/min pooling).

A convolution's stride length determines how many steps we take when sliding our window (filter) across an image. The stride length of a conventional convolution is one, thus we move the filter one unit as we scan through the input.

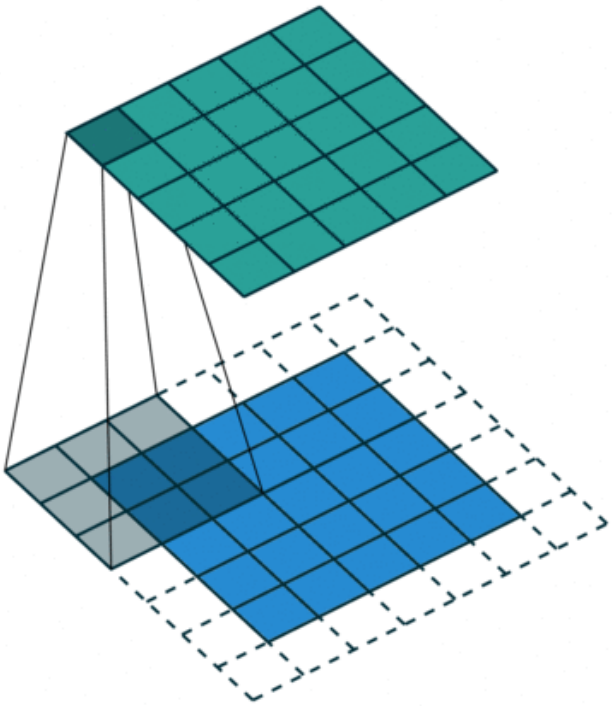


Figure 62: Strided Convolutions [43]

If we defined a stride length of two, we would move the filter by two units for each calculation.

Take note of how this reduces the dimensions of the output feature map in comparison to the input.

The dimensions of a feature map may be calculated using the formula below:

$$\left(\frac{n_{width} - f_{width} + 2p_{width}}{s_{width}} + 1, \frac{n_{height} - f_{height} + 2p_{height}}{s_{height}} + 1\right)$$

where n is the input dimension, f is the filter size, p is the padding, s is the stride length.

4.10.4 Putting everything together

Convolutional neural networks (also known as ConvNets) are typically made up of convolutional layers with periodic down sampling (either through pooling or strided convolutions). Convolutional layers generate feature mappings that serve to explain the input in various ways, whereas pooling layers compress the spatial dimensions, reducing the number of parameters required to extract features in subsequent layers. We finally reduce this representation of our original input to a deep stack of 1 by 1 (single value) representations of the original input for picture classification. Now that we have this information, we can input it into a few fully-connected layers to calculate the chance of each output class being present in the image.

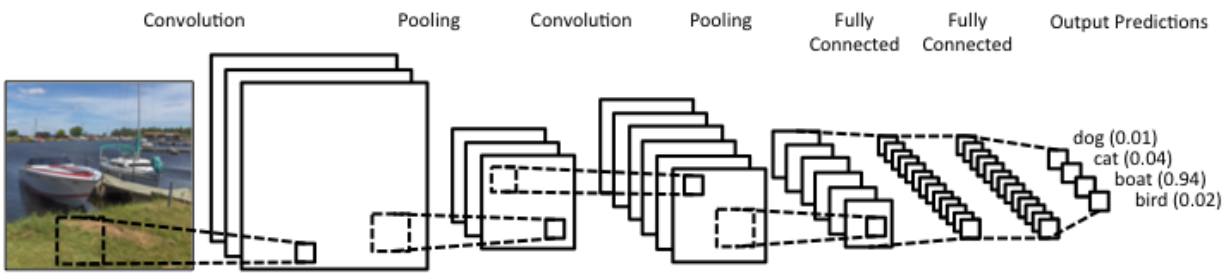


Figure 63: Input image to output predictions [43]

RESULTS AND DISCUSSION

Chapter 5

This project was successful in controlling house appliances and the main entrance using a smartphone. The platforms used to give commands (Touch and Voice) to manage home appliances over internet are Google Home, Amazon Alexa, and Sinric Pro App. We have created an Android software that allows us to operate household appliances without using the internet and unlock the main door by scanning our fingerprint on our smartphone. The Android App can also stream live video with facial detection. The ESP32 CAM module can transmit intruder detected photographs to an authorized user's Telegram account, as well as save the images to the SD card.

Demo House Model:

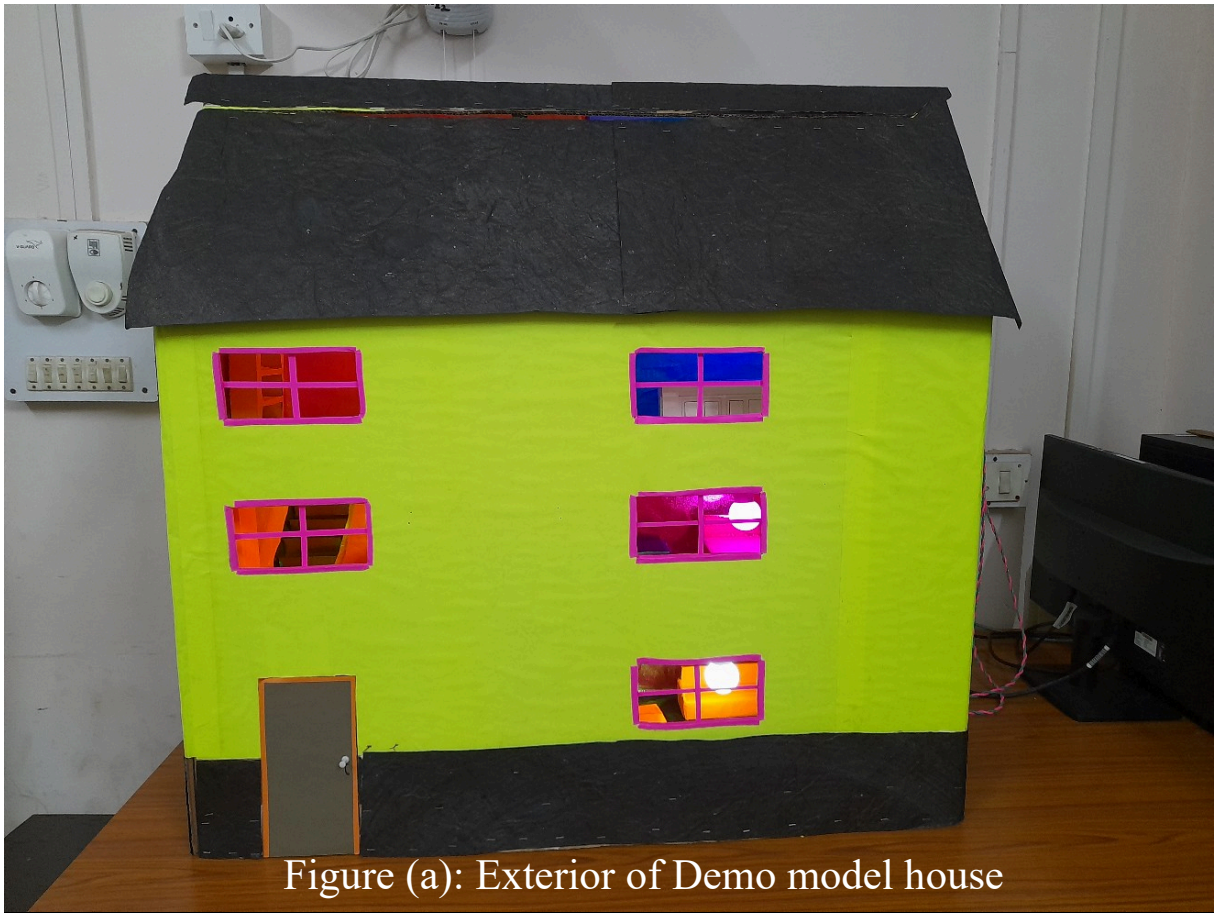


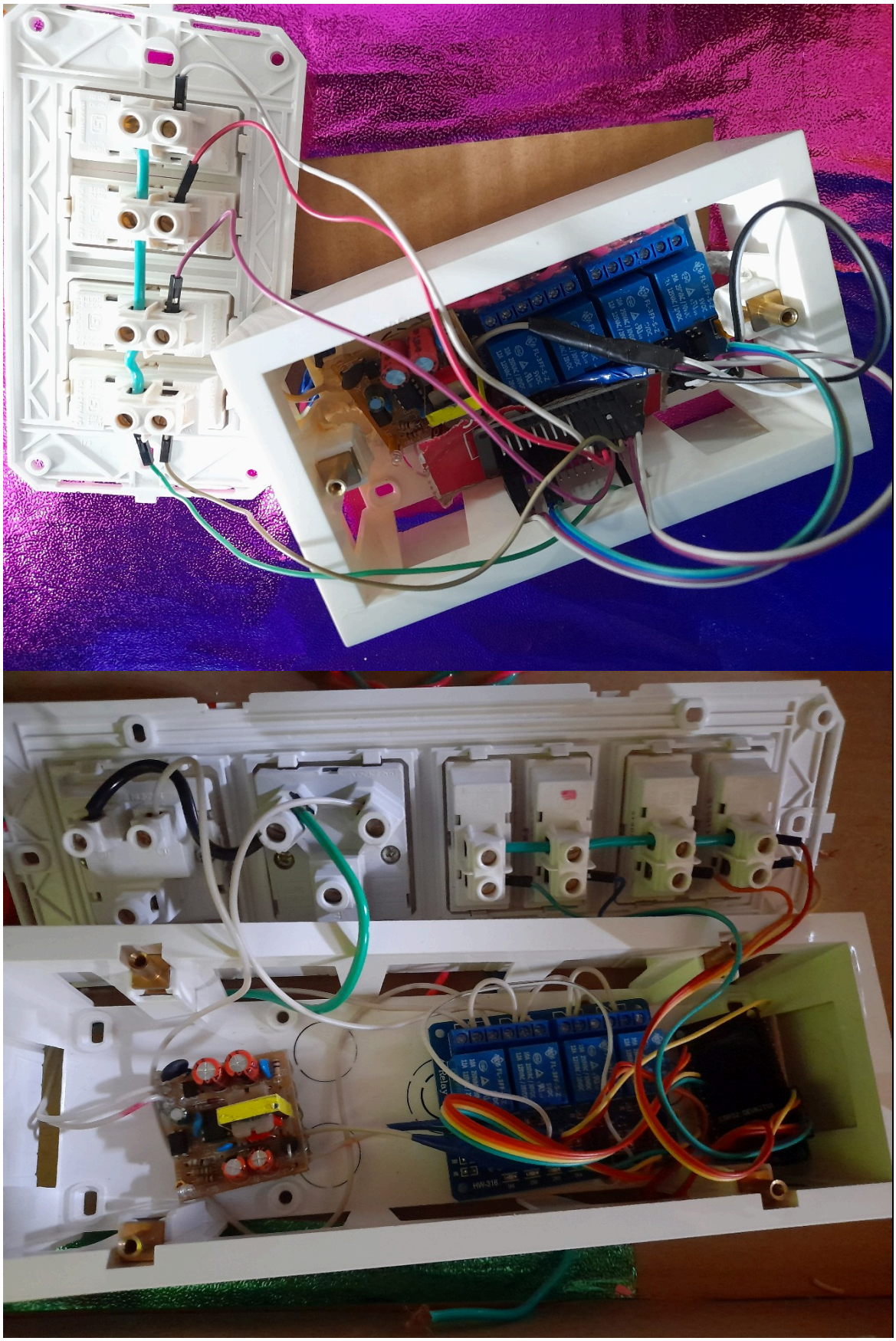
Figure (a): Exterior of Demo model house

Figure (b): Exterior of Demo model house

Figure (c): Interior of Demo model house

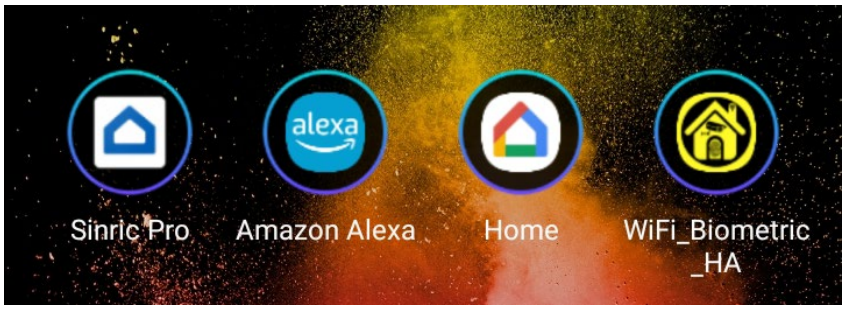
Figure (d): Interior rooms of Demo model house

Internal Connections for Home Automation:



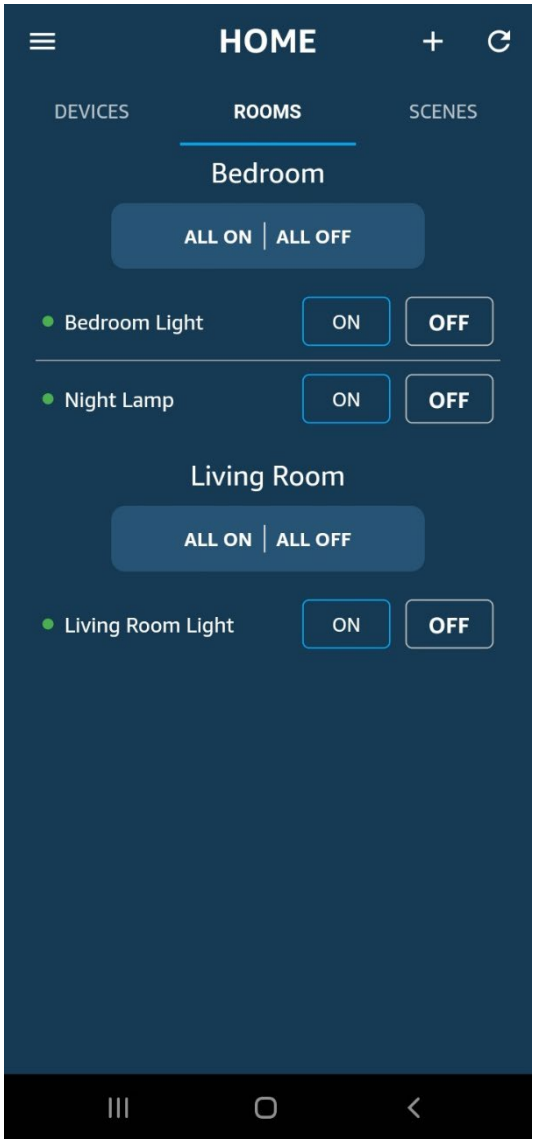
Different Apps used in our Project:

Sinric Pro, Amazon Alexa, Google Home and Android App



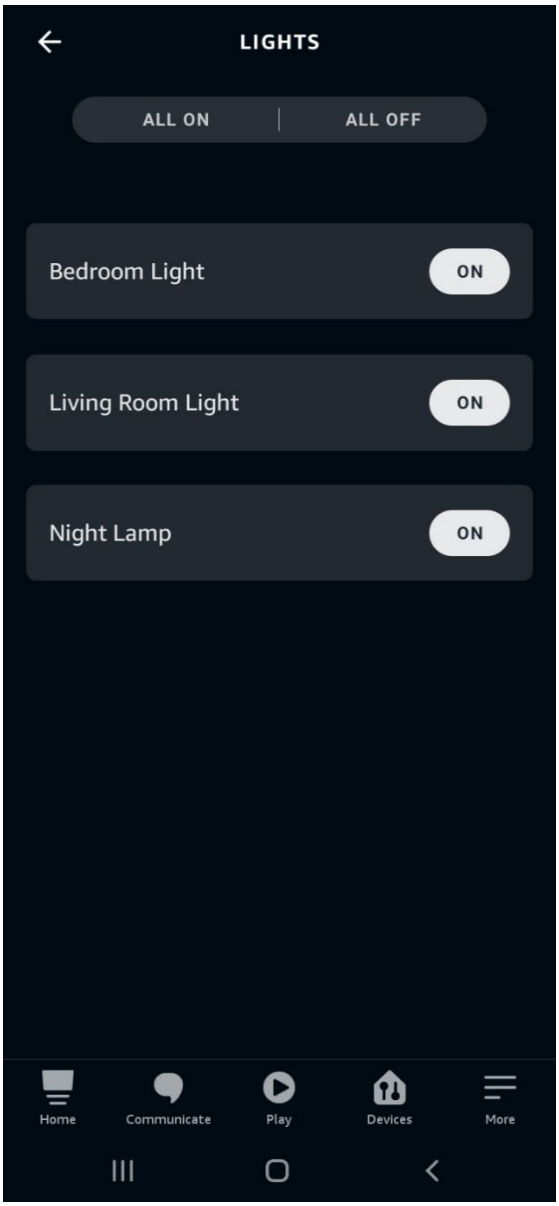
User Interface of Sinric Pro App:

Here we have created 2 rooms, Bedroom and Living room. In Bedroom, we added two lights and in Living room there is one light.



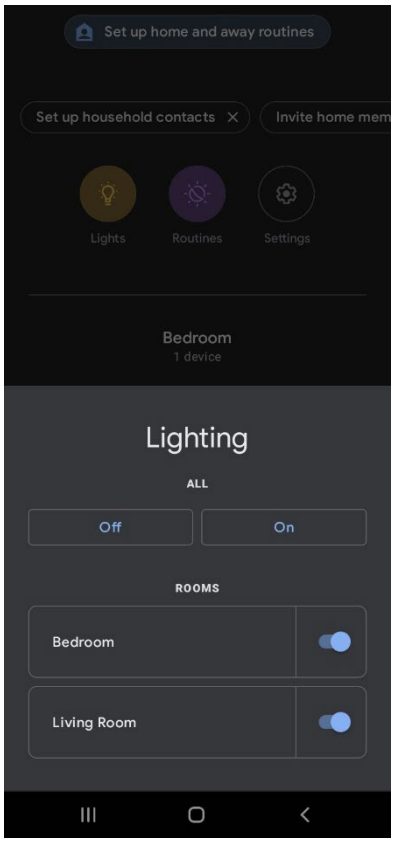
User Interface of Amazon Alexa App:

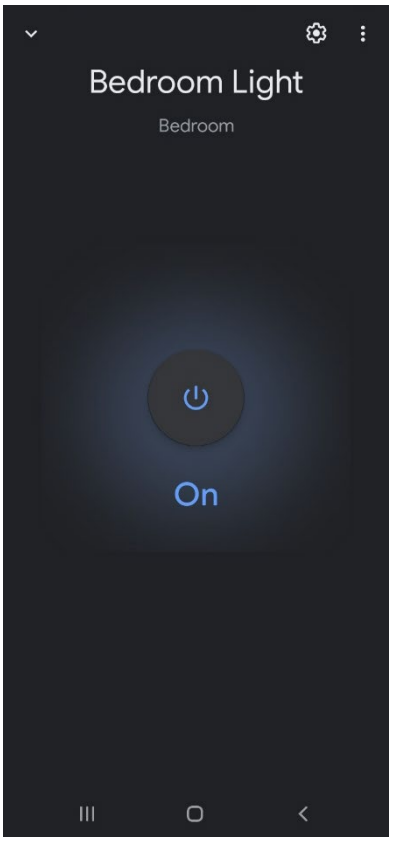
Similarly, we have added 3 lights in Amazon Alexa app, Bedroom Light, Living Room Light and Night Lamp.

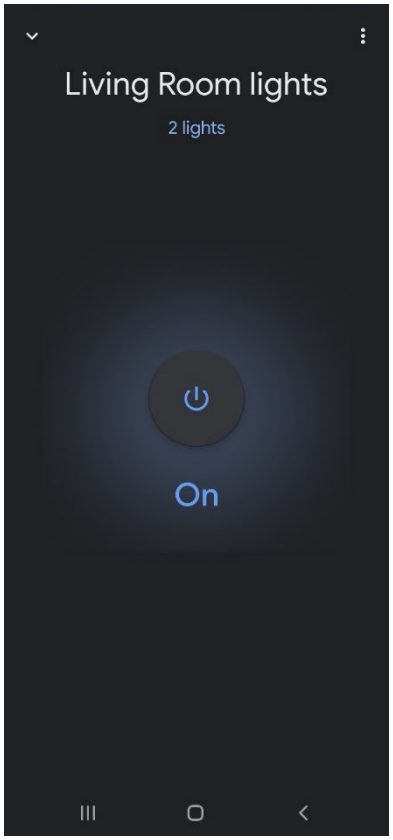


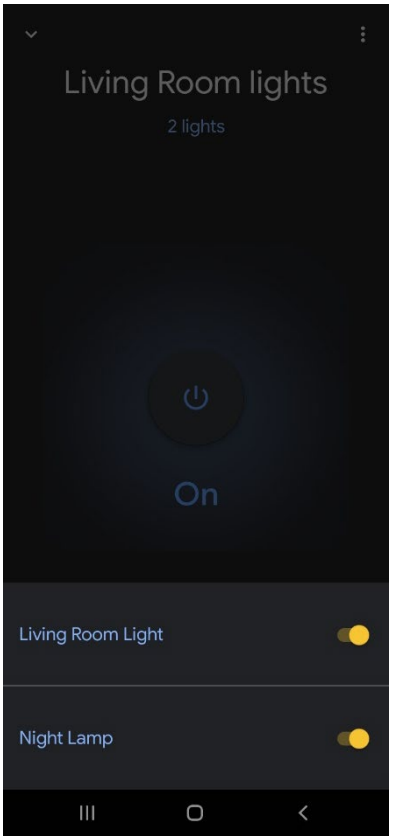
User Interface of Google Home App:

Similarly, we have created 2 rooms in Google Home app, Bedroom and Living room and added the lights accordingly.









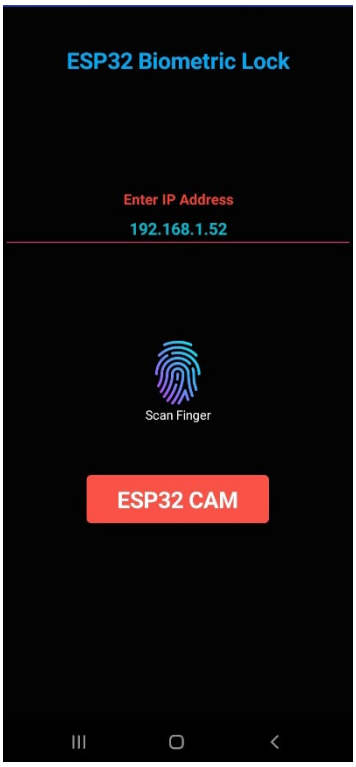
User Interface of Android App:

We have designed an Android App with two rooms, Room 1 and Room 2 and four device switches in each.

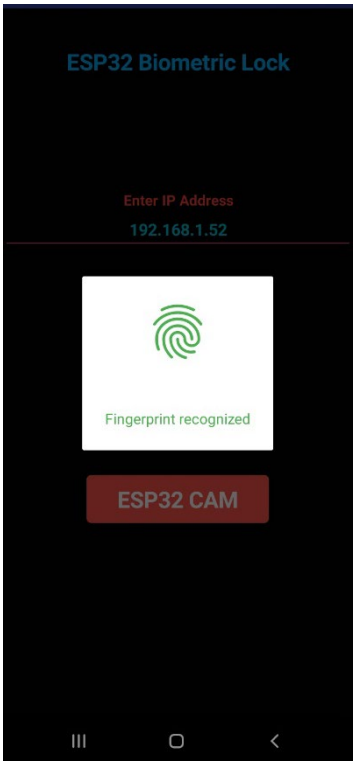
When All Devices are turned to ON position, the switch thumb shows GREEN Color and when turned to OFF position, the switch thumb shows RED Color:

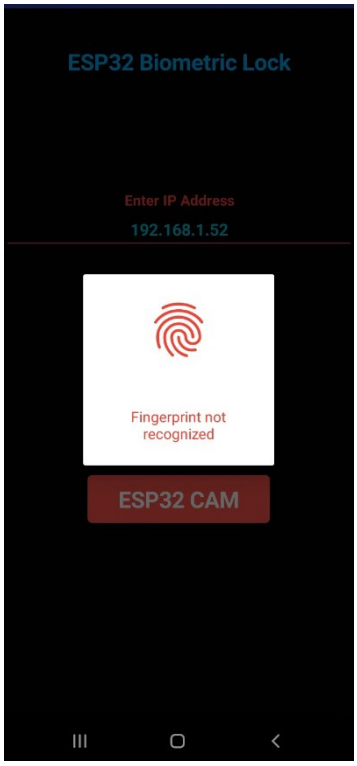
Screen 2 of Android App:

User Interface to Unlock main door of Home:



On pressing “Scan Finger” Button, if fingerprint is recognized then the door will unlock, else the door remains locked:





Door Closed:

Figure (e): Solenoid lock fixed at the main door of the house

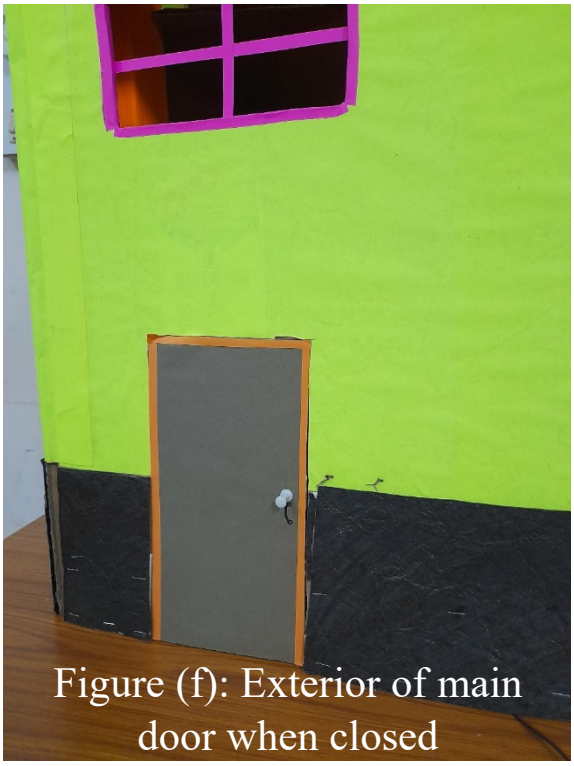


Figure (f): Exterior of main door when closed

Door Opened:

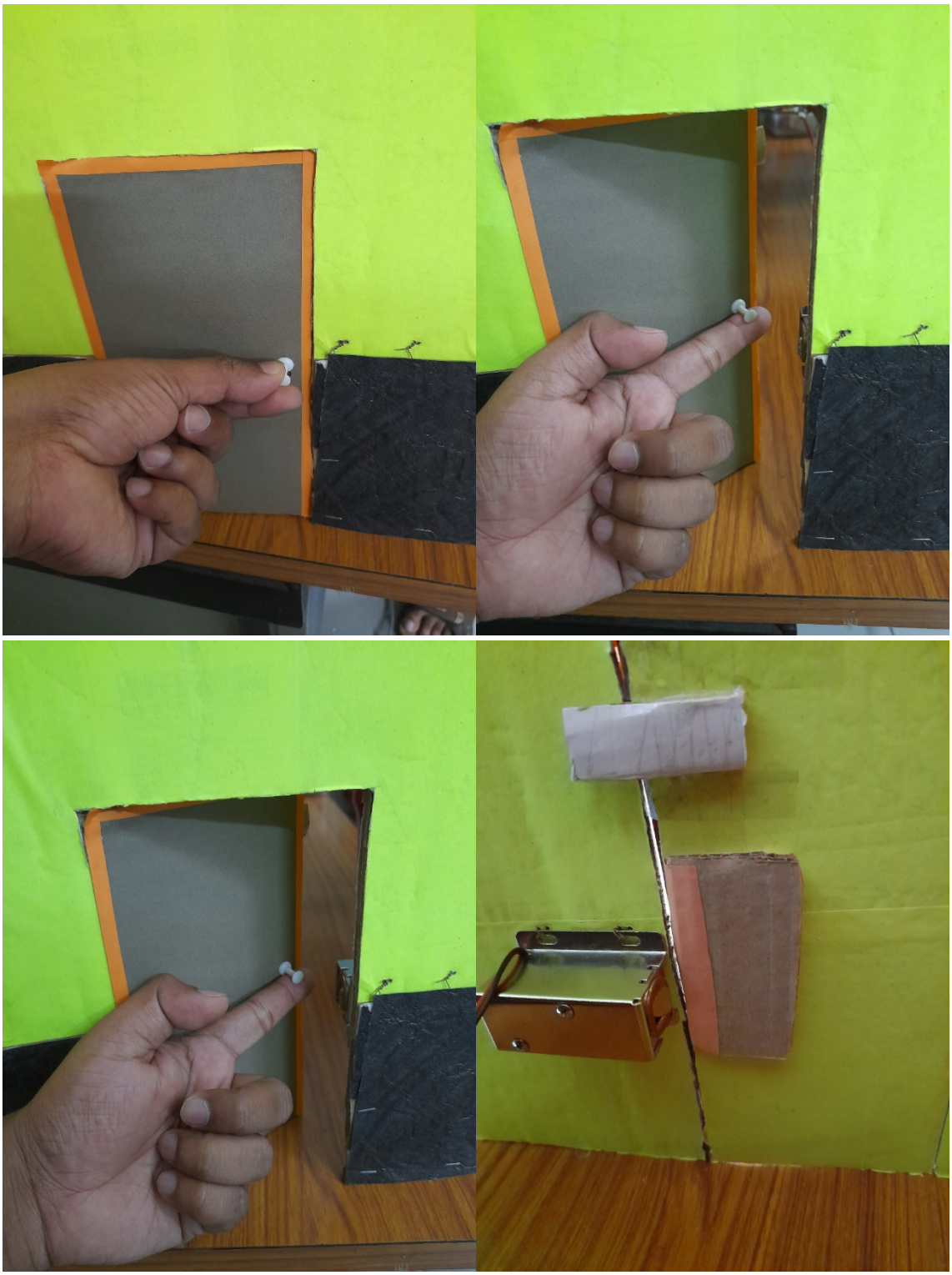


Figure (g): Main door when open

ESP32 Camera:

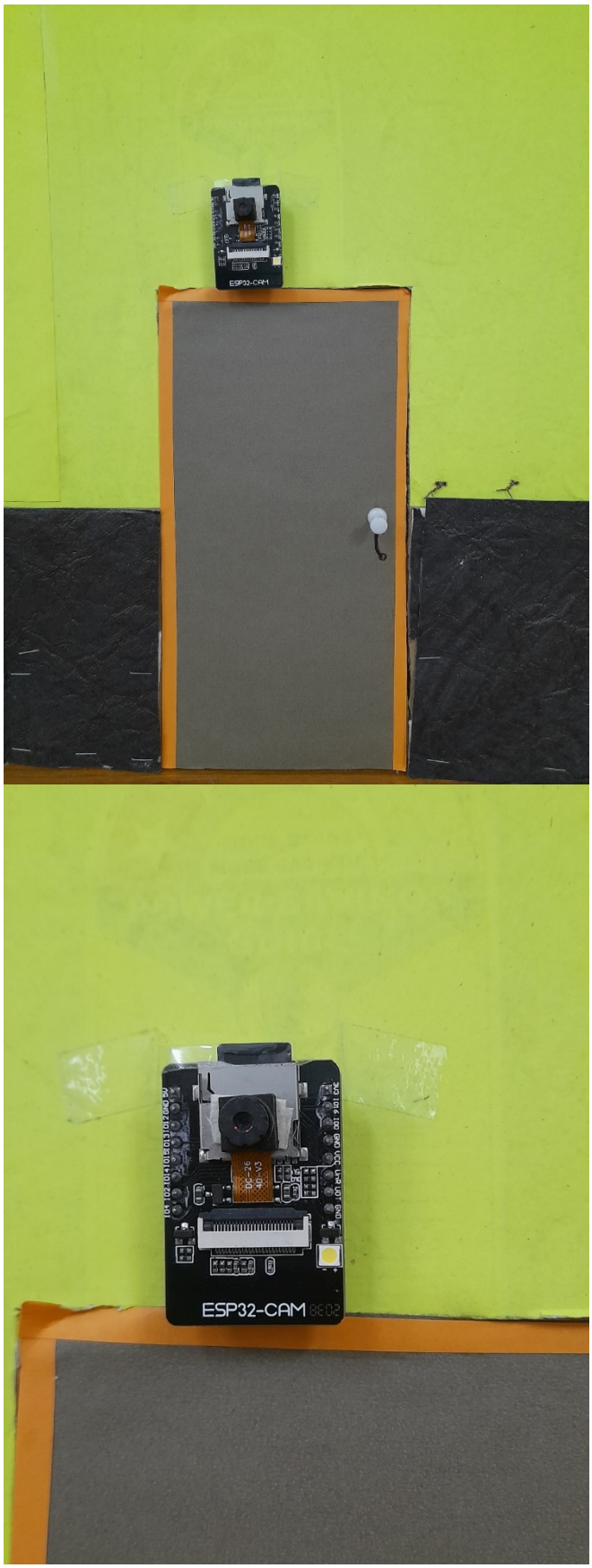
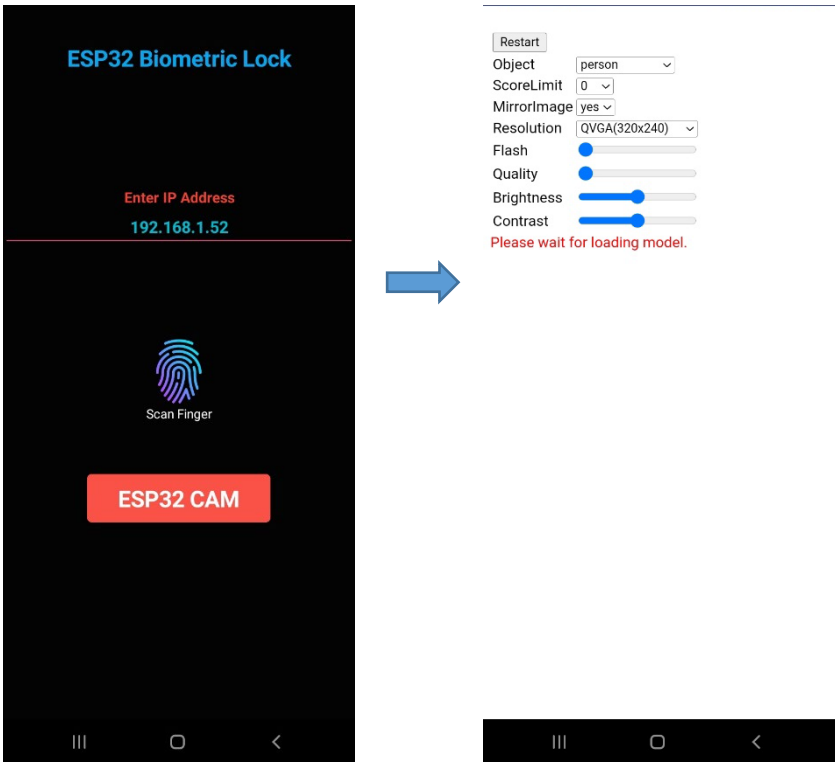


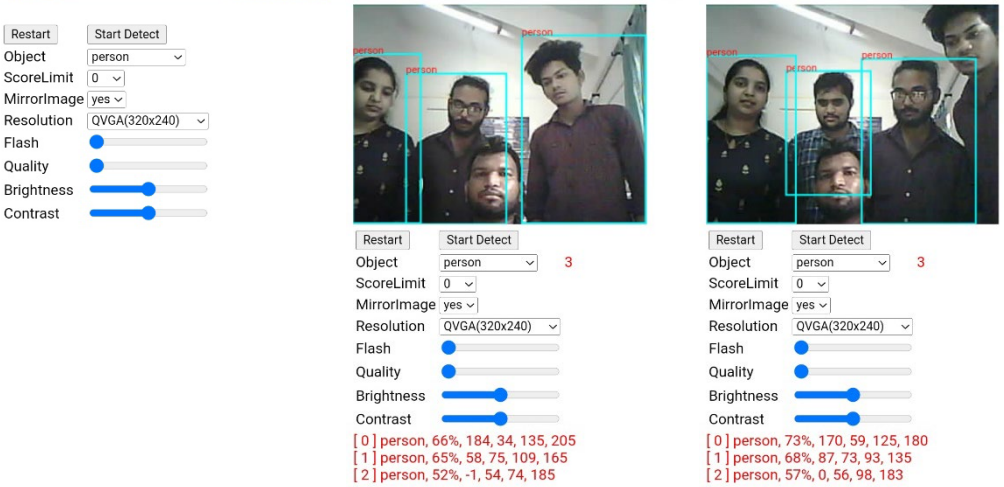
Figure (h): Camera is placed outside the main door

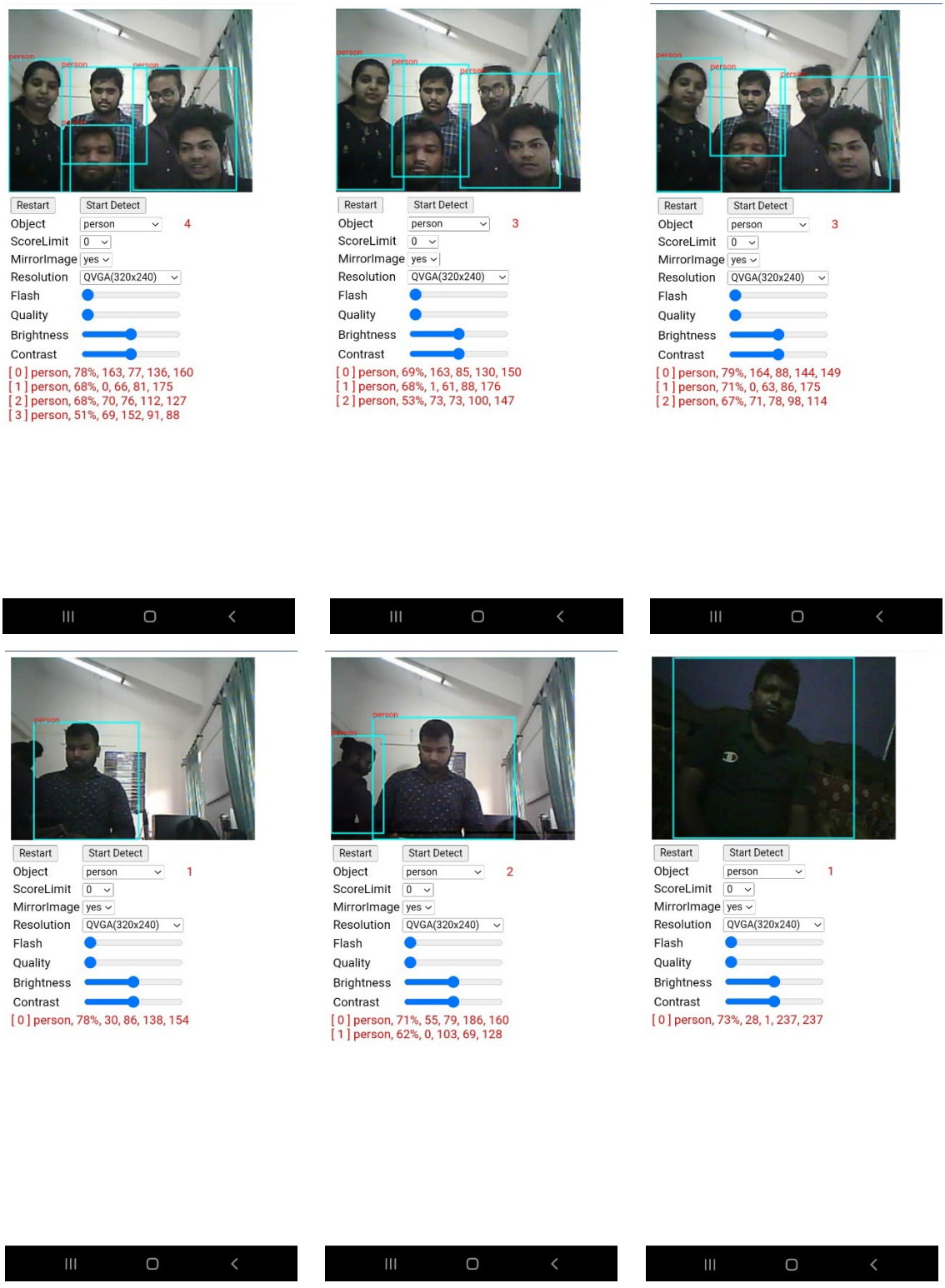
Screen 3 of Android App:

We have created a button in Screen 2 named “ESP32 CAM”, on clicking “ESP32 CAM” it will open a web page in the app itself for Video Streaming with Face Detection. We have to wait of few seconds till the model loads.



Once the model loads, click on “Start Detect” button. The Face Detection Starts.

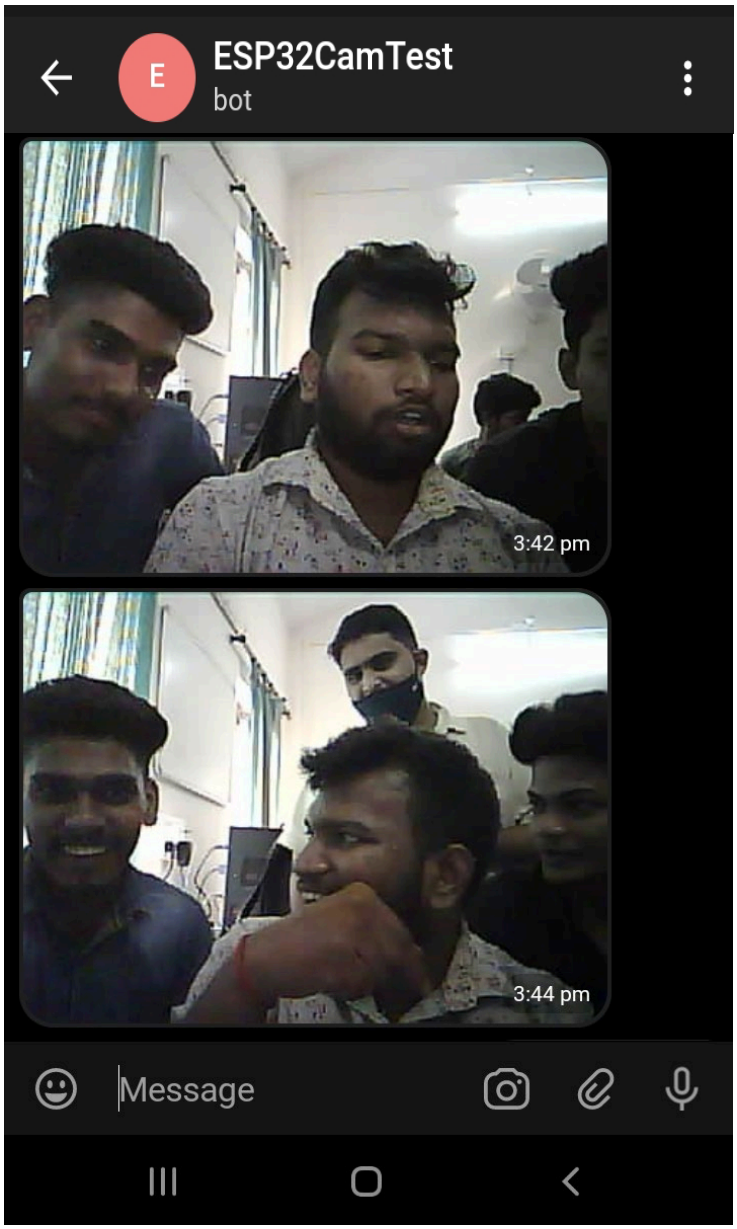




We used a pre-trained COCO-SSD model for face detection:

Class (Object)	Number of Images	Accuracy
Person	200	79%

Telegram Bot sending images to the authorized user:



CONCLUSION AND FUTURE WORKS

Chapter 6

6.1 Conclusion

In this work, we focused on several processes for remotely monitoring and managing electrical and electronic equipment using the ESP32. Because of technological advancements, Wi-Fi networks are easily available in all areas such as homes, office buildings, and industrial buildings, thus the proposed wireless network may be simply operated using any Wi-Fi network. Using touch controls and voice commands in the Google Home or Amazon Alexa App, we can operate and monitor our gadgets from anywhere in the world. The ESP32 may be connected to an Android App we designed over Wi-Fi, allowing the user to control domestic appliances without an internet connection.

Using the ESP32 CAM module, we enhanced the current security mechanism. The ESP32 CAM module sends intruder photographs with facial detection to the authenticated user's Telegram account and stores them to the SD card placed in the ESP32 CAM. We may also access our front door by using the Android App we designed to scan the fingerprint sensor on our phone. The ESP32 CAM is installed outside the house, and we can watch live video streaming on any web browser as well as our Android app.

This project is strongly recommended for everyone in the world, especially for users with disabilities and householders. This proposal will lead to a greener future by saving and lowering power expenditures. Furthermore, it will assist and guide the disabled person to work independently and to manage their house safety in a more orderly manner.

6.2 Cost of this proposed system

Sr. No.	Components	Quantity	Cost
1.	ESP32 Module	2	1210/-
2.	ESP32 CAM Module	1	850/-
3.	4 Channel Relay Module	2	538/-
4.	Solenoid Lock	1	499/-
5.	Jumper Wires	1	60/-
6.	Modular Switches	8	520/-
7.	8 Modular PVC Box	1	235/-
8.	4 Modular PVC Box	1	116/-
9.	8 Modular Box Plate	1	245/-
10.	4 Modular Box Plate	1	118/-
11.	0.5W LED Bulb	5	275/-
12.	Bulb Holder	5	150/-
13.	Electrical Wire	12 meters	150/-
14.	20A Modular Socket	2	508/-
TOTAL			**5474/-**

6.3 Future Work

- This project may be developed further by including Face Recognition utilizing the ESP32 CAM, allowing it to detect and authenticate the user's identity.
- Sensors such as Glass break detectors, Door and Window Contacts, Passive Infrared (PIR) motion detectors, Photoelectric, Heat and Smoke Detectors can be utilized to improve and strengthen the security system.
- LUX sensors, humidity sensors, Temperature sensors, air composition sensors, water level sensors, pressure sensors, vibration sensors, and ultrasonic sensors are examples of sensors that can be used in home automation.
- Voice Commands based on Regional Languages for Home Automation.

BIBLIOGRAPHY

Chapter 7

7.1 Papers Referred:

[1] Shanskar Rai, Depshika Thapa, Odzer Zangpo Bhutia, Supriya Rai, Sarashwati Gurung, Ms. Sujala Pradhan, Diploma in Computer Science and Technology CCCT Government polytechnic, Chisopani Namchi, South Sikkim, India, Vol 8, Issue 7, July 2021.

[2] Saurabh Singh, Harjeet Matharu & Dr. Sangeeta Mishra, Department of Electronics and Communication, Mumbai University, Thakur College of Engineering and Technology, Vol 6, Issue 11, Nov 2017.

[3] Ms. Priyanka, Mr. Parveen Kantha M. Tech. Scholar, Department of Computer Science Engineering, B.R.C.M.C.E.T., Bahal, Haryana, India, Vol 7, Issue 10, Oct 2020.

[4] Shaik Abdul Nabi, Shaik Jilani Bhasa, Eluri Nimisha, Thota Greeshma, Tiriveedula Priyanka, K Vasudevan, Department of EEE, QIS College of Engineering and Technology, Ongole, India, Vol 12, Issue 1, 2021.

[5] Harshada Bamnote, Ashirwad Rakhonde, Shweta Ghagare, Sandhya Dharpure, Kiran Kapse, Prof. Sarvesh Warjurkar Department of Information and Technology, Tulsiramji Gaikwad-Patil College of Engineering and Technology, Nagpur, India, Vol 3, Issue 2.

[6] A. Lavanya, Boge Praharsha, Banda Sai Shivani, Bagary Archana, Badini Sai Kiran Goud, Electronics and Communication Engineering, TKR College of Engineering and Technology, Medbowli, Meerpet, Telangana, India, Vol 9, Issue 6, June 2021.

[7] Vinay Sagar K N, Kusuma S M, Digital Communication Engg, Department of Telecommunication, MSRIT, Bangalore, India, Vol 2, Issue 3, Jan 2015.

[8] Priyanka Gaurkhede, Prasanna Titarmare, Ashish Polke, Ankita Tupte, Rani vaidya, Tushar Nawkar, Akshay Ashtankar, Kiran wadekar, Department of Electrical Engineering, Suryoday College of Engg. & Technology, Nagpur, India, Vol 7, Issue 3, 2021.

[9] Janardhana Thammineni, Computer science and engineering, GVP-SITAM, Vizianagaram, Vol 10, Issue 9, Sept 2019.

[10] Girish Yadav B.Tech (CSE) IV Year, CVR College of Engineering, Hyderabad M. Sathya Devi, Assistant Professor, Dept. of CSE, CVR College of Engineering, Hyderabad, Special Issue, April 2017.

[11] Olutosin Taiwo and Absalom E. Ezugwu School of Mathematics, Statistics and Computer Science, University of Kwazulu-Natal, Westville Campus, Private Bag X54001, Durban 4000, South Africa, Vol 2021, Article ID 9928254.

[12] Gyan Prakash Pandey, Kirti Singh, Jyoti Kumari, Rajneesh, “IoT based Home Automation”, International Journal of Scince, Engineering and Technology, 2018, Volume 6, Issue 2.

[13] Lalit Mohan Satapathy, Samir Kumar Bastia, Nihar Mohanty, “Arduino based Home Automation using Internet of Things (IoT)”, International Journal of Pure and Applied Mathematics, Volume 118, No. 17, 2018, 769-778.

[14] B. Harshini, T. Hitendra Sarma, R. Dharma Reddy, V. Venkata Krishna, "Low-cost Home Automation using IoT" Department of Information Technology Vasavi College of Engineering, Ibrahimbagh, Hyderabad, India.

[15] José David Esquicha-Tejada, Juan Carlos, Copa Pineda Universidad Nacional de San Agustin de Arequipa, Arequipa, Peru.

[16] "Voice based Smart Home Automation using Internet of Things (IoT)", Journal of Engineering Sciences, Vol 10, Issue 9, Sept 2019.

[17] Olutosin Taiwo, Absalom E. Ezugwu, Olaide N. Oyelade and Mubarak S. Almutairi, "Enhanced Intelligent Smart Home Control and Security System based on Deep Learning Model", Volume 2022, Article ID 9307961.

[18] Olutosin Taiwo and Absalom E. Ezugwu, "Internet of Things-Based Intelligent Smart Home Control System", Volume 2021, Article ID 9928254.

[19] Iyer, Ritvik & Sharma, Antara. (2019). IoT based Home Automation System with Pattern Recognition. International Journal of Recent Technology and Engineering. 8. 3925-3929. 10.35940/ijrte.B2060.078219.

7.2 Books Referred:

[20] "Introduction to IoT", By Sudip Misra, Anandarup Mukherjee and Arjit Roy, Indian Institute of Technology, Kharagpur, April 2021, ISBN: 9781108842952.

7.3 Websites Referred:

[21] https://iotcircuithub.com/nodemcu-alexa-home-automation-system/

[22] https://www.yosnalab.com/article?project=Home-Automation-with-SINRIC

[23] https://easyelectronicsproject.com/esp32-projects/home-automation-project-esp32-alexa-google/

[24] https://www.instructables.com/Home-Security-System-Using-ESP32-CAM-and-Telegram-/

[25] https://circuitdigest.com/microcontroller-projects/esp32-cam-face-recognition-door-lock-system

[26] https://github.com/sinricpro/esp8266-esp32-sdk

[27] https://github.com/geeksville/Micro-RTSP

[28] https://www.oracle.com/in/internet-of-things/what-is-iot/#:~:text=What%20is%20IoT%3F,and%20systems%20over%20the%20internet.

[29] https://data-flair.training/blogs/how-iot-works/

[30] https://www.hiotron.com/iot-architecture-layers/

[31] https://www.scaler.com/topics/components-of-iot/

[32] https://intellipaat.com/blog/future-scope-of-iot/

[33] https://lanars.com/blog/iot-home-automation

[34] https://www.digitaltrends.com/home/google-home-defined/

[35] https://theassistant.io/guide/what-is-alexa/

[36] https://www.digitaltrends.com/home/what-is-amazons-alexa-and-what-can-it-do/

[37] https://docs.kodular.io/#creator

[38] https://waynealarm.com/types-of-alarm-sensors/

[39] https://www.simform.com/blog/home-automation-using-internet-of-things/

[40] https://www.twinschip.com/ESP32_WROOM-32_Development_Board

[41] https://components101.com/switches/5v-four-channel-relay-module-pinout-features-applications-working-datasheet

[42] https://protosupplies.com/product/solenoid-electric-door-lock-12v-0-9a/

[43] https://www.jeremyjordan.me/convolutional-neural-networks/

[44] https://developers.arcgis.com/python/guide/how-ssd-works/

[45] https://cocodataset.org/#home

[46] https://mmla.gse.harvard.edu/tools/coco-ssd-object-detector/

[47] https://www.npmjs.com/package/@tensorflow-models/coco-ssd

[48] https://github.com/balancap/SSD-Tensorflow

[49] https://core.telegram.org/bots

[50] https://core.telegram.org/api

[51] https://www.ibiblio.org/e-notes/ml/detect/coco-ssd.htm

[52] https://components101.com/cables/ftdi-cable-usb-to-rs232-converter

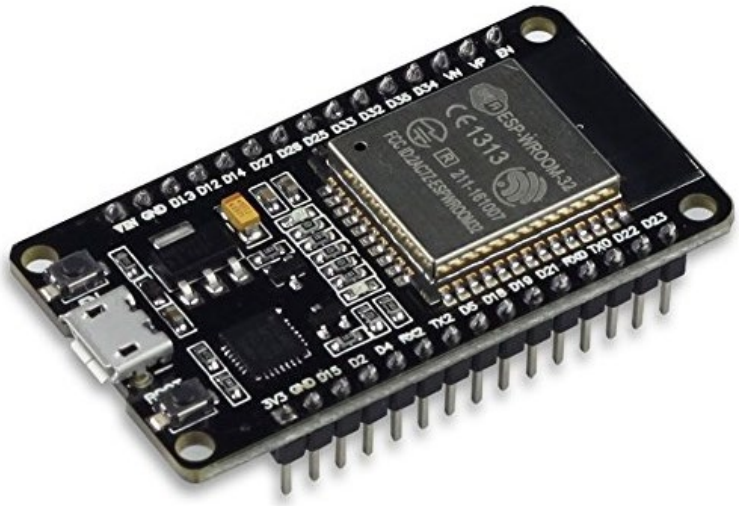
[53] https://www.wallarm.com/what/a-simple-explanation-of-what-a-websocket-is

[54] https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

APPENDIX

Chapter 8

8.1 ESP32 ESP-WROOM-32 Development Board – Wi-Fi + Bluetooth



Features:

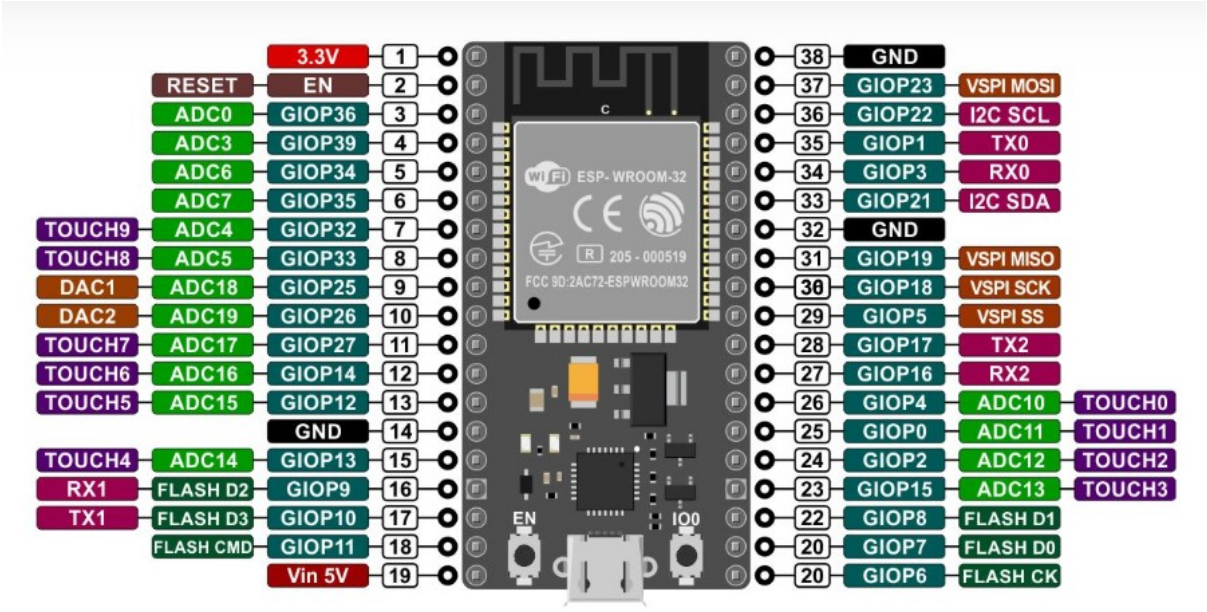
1. Two CPU cores with separate control and changeable clock frequencies ranging from 80 MHz to 240 MHz.
2. The antenna's +19.5 dBm output provides a strong physical range.
3. Legacy Bluetooth, which supports L2CAP, SDP, GAP, SMP, AVDTP, AVCTP, A2DP (SNK), and AVRCP (CT).
4. Support for Bluetooth Low Energy (BLE) profiles such as L2CAP, GAP, GATT, SMP, and GATT-based profiles such as BluFi and SPP-like, among others.
5. Bluetooth Low Energy (BLE) links to smartphones and sends out low-energy beacons for simple detection.
6. Because the sleep current is less than 5 A, it is suited for battery-powered and wearable-electronics applications.
7. Includes a 4MB flash memory.

8. Capacitive touch sensors, Hall sensors, low-noise sensing amplifiers, SD card interface, Ethernet, high-speed SPI, UART, I2S, and I2C are among the peripherals.
9. Fully certified, complete with integrated antenna and software stacks.

Specifications:

- 520 KB integrated SRAM
- Wi-Fi and Bluetooth hybrid
- High degree of integration
- Extremely low-power management
- 4 MB Flash memory
- Antenna on the PCB

Pinouts:



8.2 ESP32 CAM Module

Features:

1. Clock speed of up to 160 MHz, total computing power of up to 600 DMIPS.
2. 520 KB SRAM built-in, external 4MPSRAM.
3. Supports UART, SPI, I2C, PWM, ADC, and DAC.
4. Support for OV2640 and OV7670 cameras, as well as a built-in flash light.
5. Allow picture Wi-Fi upload.
6. TF card support.
7. Allows enabling numerous sleep modes.
8. Lwip and FreeRTOS embedded.
9. STA / AP / STA+AP operating modes are supported.
10. Smart Config / AirKiss technology support.
11. Serial port firmware updates, both local and remote (FOTA).

Specifications:

- SPI Flash: 32Mbit by default
- RAM: 520KB built-in + external 4MBPSRAM
- Measurements: 27*40.5*4.5 (0.2) mm/1.06*1.59*0.18*
- Bluetooth: 4.2 BR/EDR and BLE standards
- 802.11 b/g/n/e/i Wi-Fi
- Interfaces supported include UART, SPI, I2C, and PWM
- TF card support: maximum capacity 4GB
- IO ports: 9
- Serial Port Baud-rate: 115200 bps by default
- Image Output Formats: JPEG (only supported by the OV2640), BMP, Grayscale
- Frequency Range: 2412-2484MHz
- Antenna: Onboard PCB antenna with a gain of 2dBi
- Transmit Power: 802.11b: 17 ± 2 dBm (@ 11Mbps);

  802.11g: 14 ± 2 dBm (@ 54Mbps);

  802.11n: 13 ± 2 dBm (@ MCS7)
- Receiving Sensitivity: CCK, 1Mbps: -90dBm;

  CCK, 11Mbps: -85dBm;

  6 Mbps (1/2 BPSK): 88dBm;

  54 Mbps (3/4 64-QAM): -70dBm;

  MCS7 (65 Mbps, 72.2Mbps): -67dBm

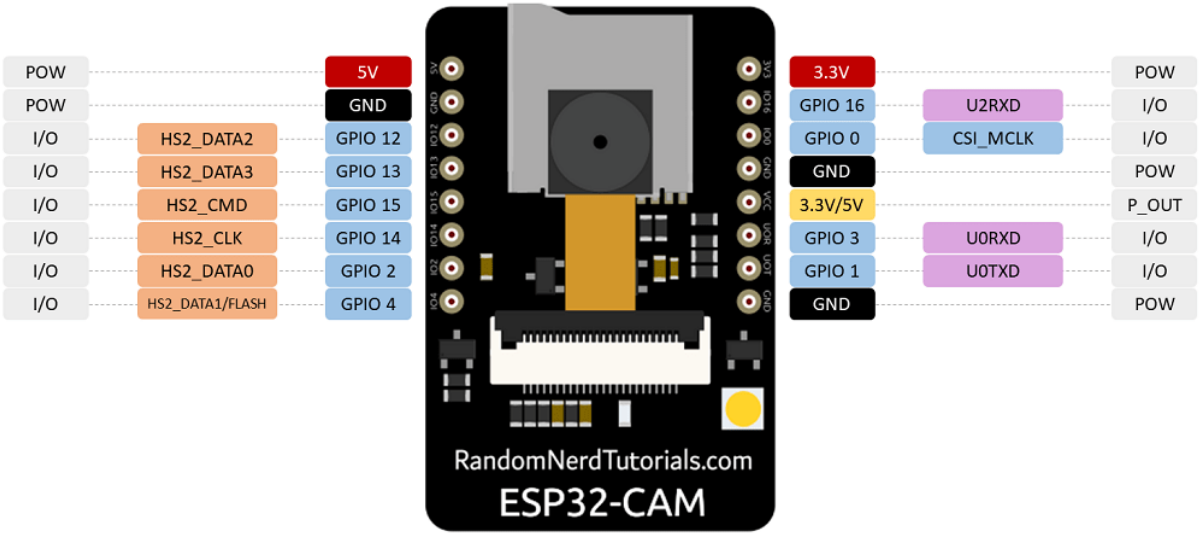
- Power consumption when the flash is turned off: 180mA @ 5V,

  Turn on the flash and increase the brightness to the highest setting: 5V,

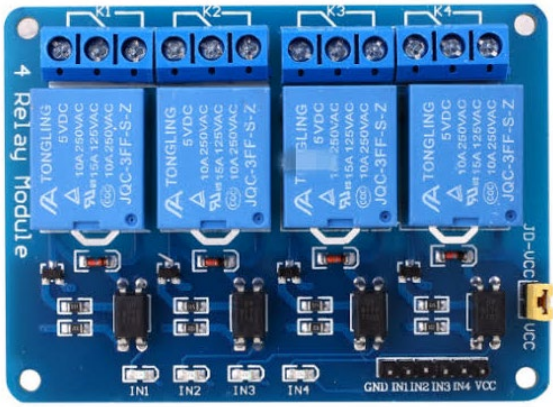
  310mA Deep sleep: the minimum power usage is 6mA @ 5V,

  Up to 20mA @ 5V for medium sleep Up to 6.7mA @ 5V for light sleep.
- WPA / WPA2 / WAP2-Enterprise / WPS security
- 5V power supply range
- Working temperature: -20o C to 85o C
- Storage conditions: -40o C to 90o C, 90% RH
- 90g in weight

Pinouts:



8.3 Four Channel Relay Module



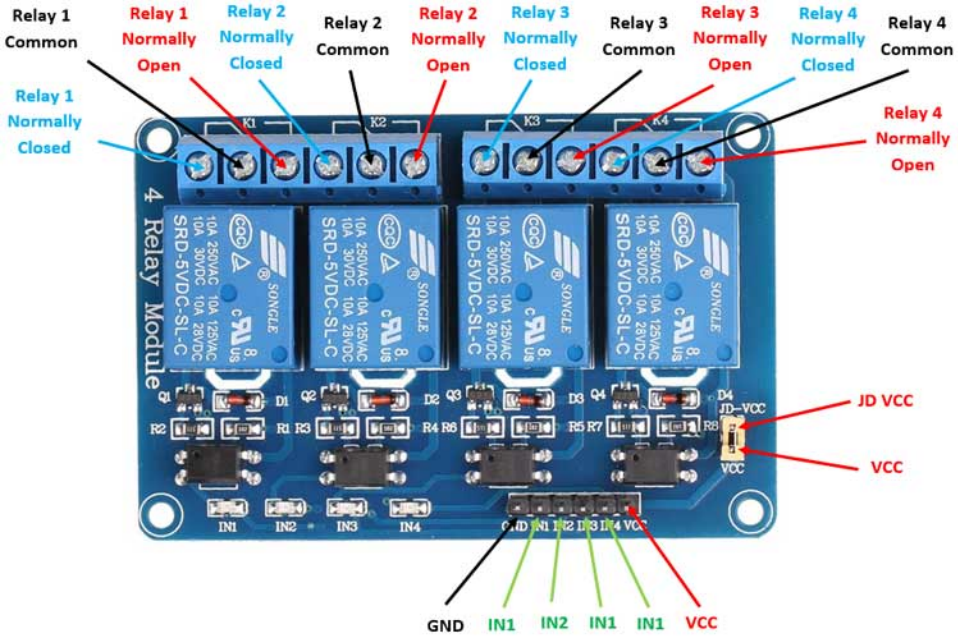
Features:

1. Relay Maximum output: DC 30V/10A, AC 250V/10A.
2. 4 Channel Relay Module with Opto-Coupler. LOW Level Trigger expansion board, which is compatible with Arduino control board.
3. Standard interface that can be controlled directly by microcontroller (8051, AVR, *PIC, DSP, ARM, ARM, MSP430, TTL logic).
4. Relay of high quality low noise relays SPDT. A common terminal, a normally open, one normally closed terminal.
5. Opto-Coupler isolation, for high voltage safety and prevent ground loop with microcontroller.

Specifications:

- Voltage range: 3.7V to 6V
- 5mA trigger current
- Current when the relay is turned on: 70mA (single), 300mA (all four)
- Maximum contact voltage of the relay: 250V AC, 30V DC
- Maximum relay current: 10A

Pinouts:



8.4 Solenoid Lock

Features:

1. 9-12V Operation
2. 0.9A / 10W power draw
3. LOW Level Trigger expansion board, which is compatible with Arduino control board.
4. Standard interface that can be controlled directly by microcontroller (8051, AVR, PIC, DSP, ARM, MSP430, TTL logic).

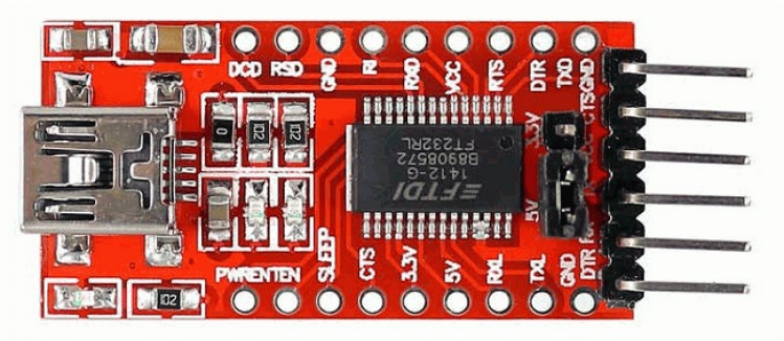
5. Relay of high quality low noise relays SPDT. A common terminal, a normally open, one normally closed terminal.
6. Opto-Coupler isolation, for high voltage safety and prevent ground loop with microcontroller.
7. The locking latch may be rotated in all four directions.
8. Heavy-duty stainless steel structure that is completely enclosed.
9. It features a slanted cut slug and an excellent mounting bracket.

Specifications:

- 12V DC (you can use 9-12V DC, but lower voltage results in weaker/slower operation).
- Draws 650mA at 12V, 500mA at 9V when activated.
- Designed for 1-10 seconds long activation time.
- Max Dimensions: 41.85mm / 1.64” x 53.57mm / 2.1” x 27.59mm / 1.08”
- Dimensions: 22.57mm / 0.92” x 67.47mm / 2.65” x 27.59mm / 1.08”
- Wire length: 222.25mm / 8.75”
- Weight: 147.71g

Pinouts:

8.5 FTDI Programmer



Features:

1. TTL data transfer speeds ranging from 300 baud to 3M baud.
2. RS485 and RS232 Support.
3. USB to UART conversion on a Single chip.
4. There is no need for extra firmware programming.
5. Receive and Transmit LED drive signals.
6. USB 2.0 high-speed compatibility.
7. Minimal USB bandwidth usage.
8. Internal EEPROM with writable area for the user.
9. The output can be linked directly to the microcontroller's UART or I/O.
10. The circuitry in the cable manage the whole USB protocol.
11. Power – On – Reset circuit integrated.
12. Option for inverting the UART signal.
13. TTL input and true 5V / 3.3V / 2.8V / 1.8V CMOS drive output.
14. Temperature range of operation: -40ºC to +85ºC.
15. 128-byte receive buffer and 256-byte transmit buffer, using buffer smoothing technology high data speed.

Specifications:

- USB to TTL Serial Adapter Module
- FT232RL Chip
- Support 3.3V – 5 Volts
- Mini USB Interface
- RXD / TXD Transceiver Communication Indicator
- Overcurrent safety is provided by a 500mA self-restore fuse in USB power
- DTR, RXD, TX, VCC, CTS, GND are the pin definitions
- Pitch: 2.54mm
- Dimensions (L x W): 36mm x 18mm

Pinouts: