CONTENTS

Preface

This project proposes a near-end strategy optimization method for Industrial IoT intrusion detection system using computer vision and honeypot technology. The system integrates network gateway, Blynk cloud, and Android application to present relevant information. Deep KNN, feature selection based on Light GBM and ReLU activation function are used to construct the intrusion detection model. The proposed system achieves a high detection rate of 97% for various parameter assaults and outperforms existing intrusion detection systems based on deep learning models. Our system provides a robust and efficient solution for detecting network assaults in real-time.

1

Acknowledgement

We would like to express our gratitude to all the team members who have worked tirelessly on this project, contributing their time, skills, and knowledge to make it a success. The project would not have been possible without their hard work, dedication, and collaboration.

Additionally, we would like to thank our project supervisor and all the faculty members who have provided us with the necessary guidance, support, and resources to carry out this project. Their expertise, insights, and feedback have been invaluable in shaping our ideas and guiding us throughout the development process.

Furthermore, we would like to acknowledge the support and encouragement we received from our friends and family, who provided us with the motivation and inspiration to keep pushing forward. Their unwavering support and belief in us have been a constant source of strength and encouragement.

Finally, we would like to express our heartfelt appreciation to everyone who has contributed to this project in any way, big or small. We are proud of what we have accomplished, and we are grateful for the opportunity to work on this project and learn from it. Thank you all for your support and encouragement.

Roles and Responsibilities

Our project team comprises of four members, each with unique roles and responsibilities. The following is a detailed breakdown of each member's roles and responsibilities:

Aadhaar Koul - Networking and IOT Engineer (Team Leader)
* Worked on the design and implementation of the honeypot system.
* Developed the Android application for the project.
* Worked on the integration of various modules into the Android application.
* Led the team in project planning, implementation, and testing.
* Coordinated with other team members to ensure timely completion of project milestones.
* Conducted regular meetings to discuss project progress and resolve any issues.

Baseer Fatima - IOT, Cloud, and Integrations Engineer
* Worked on the integration of various IOT sensors into the system.
* Designed and implemented the cloud infrastructure for the project.
* Developed the integration between the cloud and the Android application.
* Coordinated with Novneet Kaur to ensure successful integration of different modules.
* Conducted testing and troubleshooting of the IOT and cloud components.

Noneet Kaur - IOT and Cloud Integrations Engineer
* Worked on the integration of various IOT sensors into the system.
* Developed the integration between the cloud and the Android application.
* Coordinated with Baseer Fatima to ensure successful integration of different modules.
* Conducted testing and troubleshooting of the IOT and cloud components.

Arjun Charak - AI/ML/AR Engineer
* Worked on the computer vision module using AI/ML techniques.
* Hosted the computer vision module on the cloud infrastructure.
* Developed the integration between the computer vision module and the Android application.
* Conducted testing and troubleshooting of the computer vision module.

In summary, each team member played a critical role in the development of the project, with clear responsibilities and areas of expertise. The team leader ensured effective coordination and communication among the team members. The successful collaboration and coordination among team members led to the successful completion of the project.

Abstract

The project is an Intrusion Detection System (IDS) that leverages the power of honeypot systems, computer vision, and various modules integrated into a single framework. The system consists of different modules, including a network gateway, honeypot module, IOT module, Blynk cloud module, computer vision module, and Android app modules. These modules work together seamlessly, with an Android app acting as the interface to access all the functionality.

The network gateway module serves as the entry point for all network traffic into the system. It inspects all incoming traffic and sends it to the appropriate module for further processing. The honeypot module, on the other hand, simulates vulnerable services to attract potential attackers and gathers information about their tactics and techniques. The IOT module monitors devices connected to the network and sends alerts if any unusual activity is detected.

The Blynk cloud module provides a cloud-based infrastructure for the system, enabling remote access and control. The computer vision module uses machine learning algorithms to analyze video feeds from security cameras and detect potential intrusions or security breaches. Finally, the Android app module provides a user-friendly interface that allows the system administrator to access all the functionalities of the system from a single place.

Whenever any suspicious activity is detected on any of the modules, the notification system promptly alerts the administrator, who can then take appropriate action. The notification system sends push notifications to the administrator's device, prompting them to pay attention to the intrusion and investigate further.

In conclusion, the IDS project based on honeypot systems and computer vision is a comprehensive solution for detecting and preventing security breaches in a networked environment. The integration of various modules and the Android app interface makes the system easy to use and manage, even for non-technical users. With its advanced features and notification system, the IDS project is an excellent tool for securing networks against potential cyber-attacks.

4

Motivation

The increasing dependence of individuals, businesses, and governments on technology has led to a rise in cybersecurity threats and attacks. These attacks can lead to significant financial losses, theft of sensitive information, and damage to reputation. Therefore, it has become crucial to develop effective tools and techniques to detect and prevent such attacks.

Our project aims to address this issue by developing an Intrusion Detection System (IDS) based on honeypot systems and computer vision. This system detects and alerts the user of any suspicious activity or intrusion attempts in real-time, allowing for quick and effective action to be taken.

Moreover, our IDS system integrates various modules, including network gateway, honeypot, IOT sensors, cloud infrastructure, computer vision, and Android application, to create a comprehensive and robust solution. This integration enables the system to be adaptable, scalable, and effective in detecting and preventing a wide range of cybersecurity threats.

The motivation for our project is to create a reliable and efficient IDS system that can be used by individuals, businesses, and governments to safeguard their assets and sensitive information. Our project's main goal is to make the world a safer place by developing innovative and effective solutions that address real-world cybersecurity challenges.

In conclusion, our project's motivation stems from the need to develop effective tools and techniques to detect and prevent cybersecurity attacks. We believe that our IDS system based on honeypot systems and computer vision, with its comprehensive integration of various modules, is a step in the right direction towards achieving this goal.

Background and Context

The project began with the initial idea of building a computer vision module that would use machine learning to classify users and allow or deny access to a particular system. The team successfully trained the machine learning model and implemented the basic functionality of the module on a machine rig. However, the team soon realized the need to expand the scope and vision of the project to include more robust security implementations.

With team members' expertise in areas such as IOT, cybersecurity, AI/ML, and networking, a stream of new module ideas emerged. To streamline the project and prevent scope creep, the team decided to set an endpoint for additional module implementations.

The team then developed a framework that would allow different modules to work together in a coordinated manner. The final set of modules included Network Gateway, Honeypot Module, IOT Module, Blynk Cloud Module, Computer Vision Module, Surveillance Module, and Android App Module. The team estimated the project requirements and cost within their budget and expertise and worked on creating the first functional model and workflow.

To make the information easily accessible, the team decided to deploy the entire system onto various hosting platforms and display relevant information on an android application in real-time. This approach provided users/admins with a single platform for all network and premises security implementations, making the IDS a comprehensive and effective solution.

Overall, the team's focus on expertise, cost, and implementation led to the successful development of a highly functional and efficient IDS system that can be used in various settings, including homes, businesses, and government organizations.
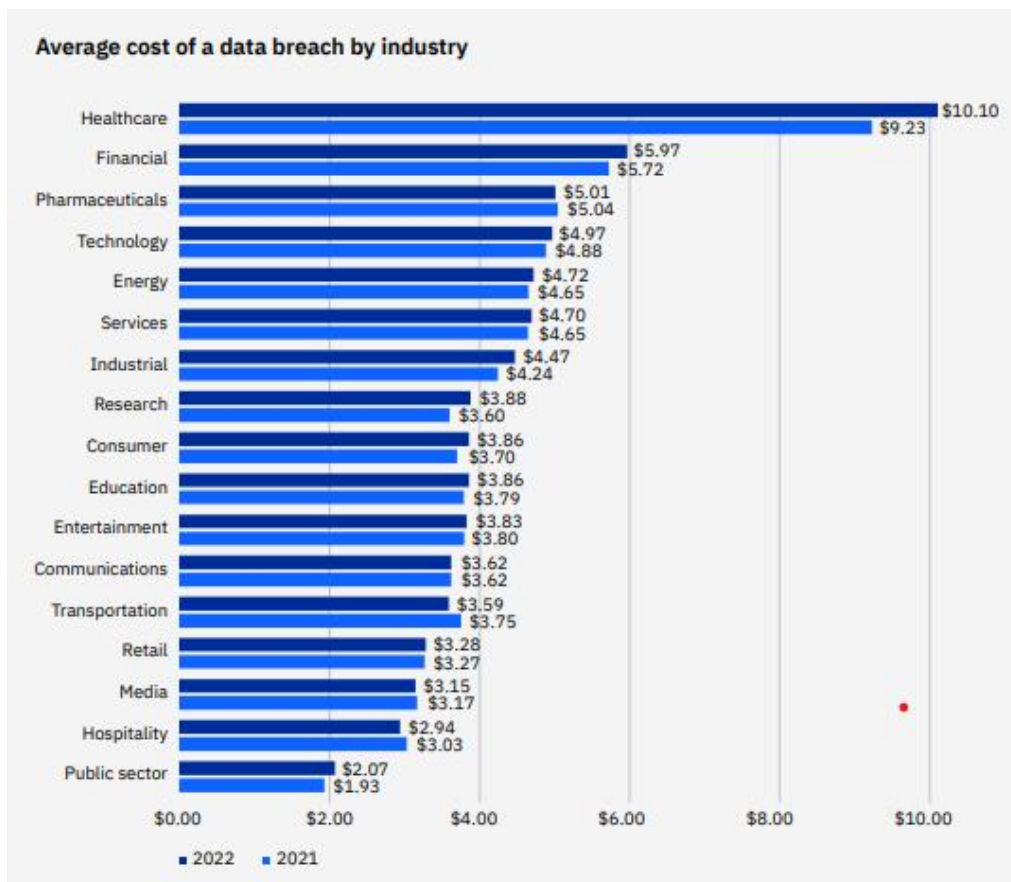
Problem Statement and Justification

Small-scale industries, general public, and unaware users are particularly vulnerable to cyber attacks due to their limited resources, lack of awareness, and reliance on technology. Small-scale industries, in particular, often lack the cybersecurity infrastructure and resources of larger organizations, making them more susceptible to attacks. According to a recent survey, 43% of cyber attacks target small businesses.

General public and unaware users are also at risk, as they may not be familiar with basic cybersecurity practices and may unknowingly expose themselves to cyber threats. This includes using weak passwords, clicking on suspicious links, and downloading malicious software.

Furthermore, many people are unaware of the importance of regularly updating their software, antivirus, and firewalls. This leaves them vulnerable to attacks that exploit known vulnerabilities in outdated software and systems.

The consequences of a cyber attack on small-scale industries, general public, and unaware users can be devastating, including financial losses, reputational damage, and theft of sensitive data. It is therefore essential that everyone takes proactive steps to protect themselves against cyber attacks, such as educating themselves on basic cybersecurity practices, using strong passwords, keeping their software up to date, and being vigilant against suspicious activity.

Average cost of a data breach by industry

Industry	2022	2021
Healthcare	$10.10	$9.23
Financial	$5.97	$5.72
Pharmaceuticals	$5.01	$5.04
Technology	$4.97	$4.88
Energy	$4.72	$4.65
Services	$4.70	$4.65
Industrial	$4.47	$4.24
Research	$3.88	$3.60
Consumer	$3.86	$3.70
Education	$3.86	$3.79
Entertainment	$3.83	$3.80
Communications	$3.62	$3.62
Transportation	$3.59	$3.75
Retail	$3.28	$3.27
Media	$3.15	$3.17
Hospitality	$2.94	$3.03
Public sector	$2.07	$1.93

■ 2022　■ 2021

* According to a report by Cybersecurity Ventures, cybercrime is predicted to cost the world $10.5 trillion annually by 2025, up from $3 trillion in 2015.

* The same report also estimates that ransomware attacks will occur every 11 seconds by 2021, up from every 14 seconds in 2019.

* In 2020, the FBI reported a 400% increase in cybercrime complaints due to the COVID-19 pandemic, with losses totaling over $4.2 billion.

* The 2021 SonicWall Cyber Threat Report found that ransomware attacks increased by 62% globally in 2020, with the average ransom demand increasing to $170,404.

* A study by Verizon found that 70% of breaches in 2020 were caused by external actors, while 52% involved hacking.

* According to the 2021 Cost of a Data Breach Report by IBM Security and Ponemon Institute, the average cost of a data breach was $4.24 million in 2021, up 10% from 2020.

Average cost and frequency of data breaches by initial attack vector
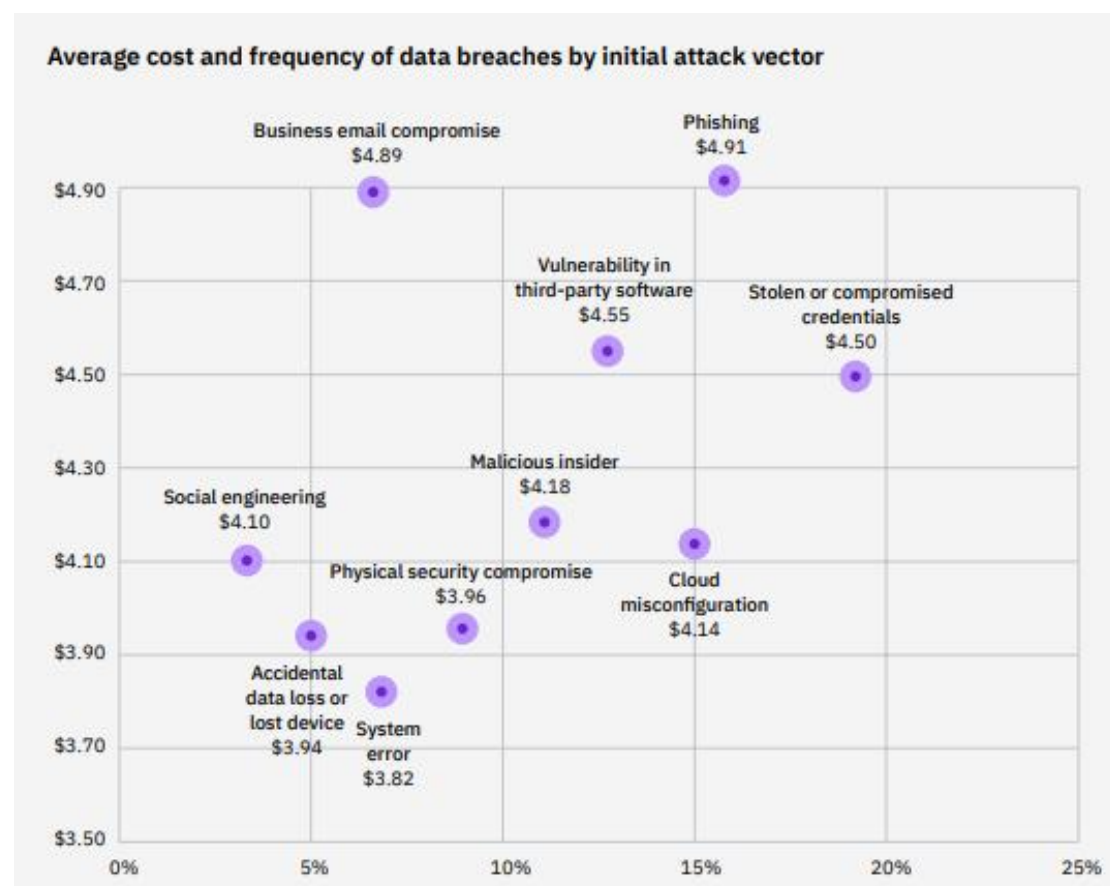
Global study at a glance

Countries	2022 sample	Percent	Years studied	Currency
United States	64	12%	17	USD
India	49	9%	11	INR
United Kingdom	43	8%	15	GBP
Brazil	43	8%	10	BRL
Germany	38	7%	14	EUR
Japan	35	6%	11	JPY
France	33	6%	13	EUR
Middle East[1]	31	6%	9	SAR
South Korea	30	5%	5	KRW
Australia	26	5%	13	AUD
Canada	25	5%	8	CAD
Italy	24	4%	11	EUR
ASEAN[2]	23	4%	6	SGD
Latin America[3]	23	4%	3	MXN
South Africa	21	4%	7	ZAR
Scandinavia[4]	20	4%	4	NOK
Turkey	20	4%	5	TRY
Total	**550**	**100%**		

In the second figure we can see that the phishing attacks are most frequent types of attacks to which the naive users are the most susceptible to. There by the general public is most vulnerable to these types of attacks.

Solution

There needs to be a hybrid intruder detection system that works on both the network and premises level and handles all the security parameter using the state-of-the-art artificial intelligent model specifically designed for the maximum output ad results in the cyber world. A home automation system needs to be devised that would possess the IOT modules like the Laser sensors, Ultrasonic Sensors, PIR Sensors etc which would work in coordinated manner around a framework which tracks each and every sensor and the network activity and generates the flag based triggered notification for the user.

The Solution need to be cheap and efficient that would simulate the real world IDS's. In order to achieve these cheap microcomputers and microcontroller need to be integrated into the framework such as the Raspberry pi 4 Model, Arduino pro micro, ESP 32, 8266 etc. These Components would work in a coordinated manner to generate flags whenever an IOT module triggers a signal.

A live feed for the IOT reading need to be rendered to a cloud module which would give an ability to render the reading to an android app or a website using an integratable API key

Solution Objectives
* Cut down costs for maximum affordability.
* Make the product seamless and easy to use.
* Cover most of the security concerns to deliver the best product
* Deliver software components that work on most of the architectures and devices.
* Bring the product to the market as soon as possible and make it available to everyone as soon as possible in order to make this world a better place.

Expected Results
* Deliver all the modules as planned
* Deliver a high quality and efficient product
* Ability to penetrate local markets as soon as possible
* Ability to scale up as per the market requirement.
* The consumer should be totally satisfied with the ethics and the quality as promised.

Introduction

What is an IDS ?

A system called an intrusion detection system (IDS) observes network traffic for malicious transactions and sends immediate alerts when it is observed. It is software that checks a network or system for malicious activities or policy violations. Each illegal activity or violation is often recorded either centrally using a SIEM system or notified to an administration. IDS monitors a network or system for malicious activity and protects a computer network from unauthorized access from users, including perhaps insiders. The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between 'bad connections' (intrusion/attacks) and 'good (normal) connections'.
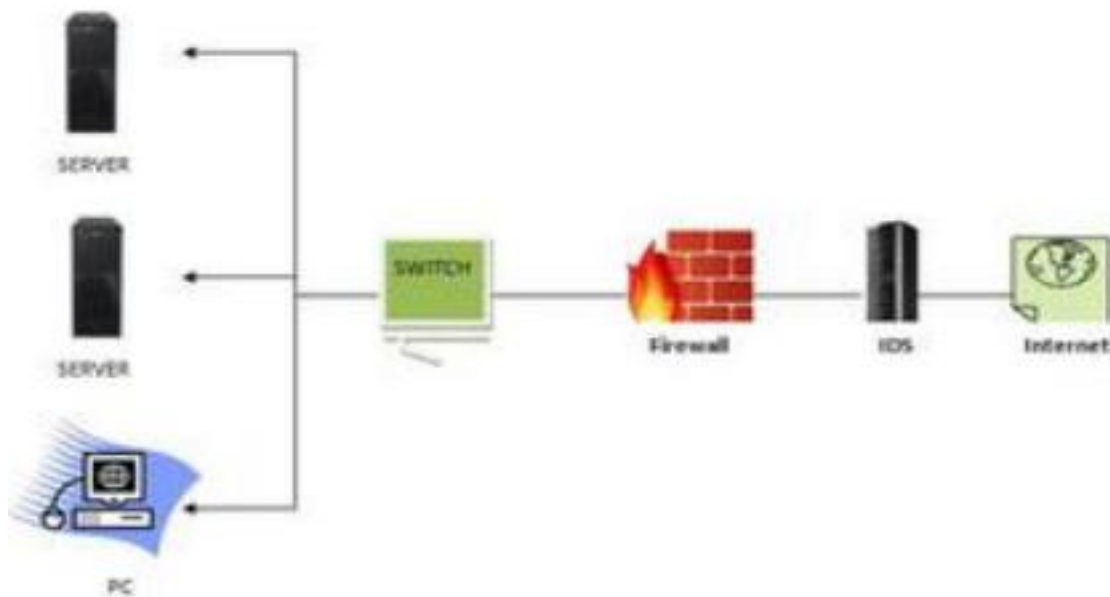

How does an IDS work ?

* An IDS (Intrusion Detection System) monitors the traffic on a computer network to detect any suspicious activity.
* It analyzes the data flowing through the network to look for patterns and signs of abnormal behavior.
* The IDS compares the network activity to a set of predefined rules and patterns to * identify any activity that might indicate an attack or intrusion.
* If the IDS detects something that matches one of these rules or patterns, it sends an alert to the system administrator.
* The system administrator can then investigate the alert and take action to prevent any damage or further intrusion.

11

Classification of Intrusion Detection System

IDS are classified into 5 types

* Network Intrusion Detection System (NIDS): Network intrusion detection systems (NIDS) are set up at a planned point within the network to examine traffic from all devices on the network. It performs an observation of passing traffic on the entire subnet and matches the traffic that is passed on the subnets to the collection of known attacks. Once an attack is identified or abnormal behavior is observed, the alert can be sent to the administrator. An example of a NIDS is installing it on the subnet where firewalls are located in order to see if someone is trying to crack the firewall.
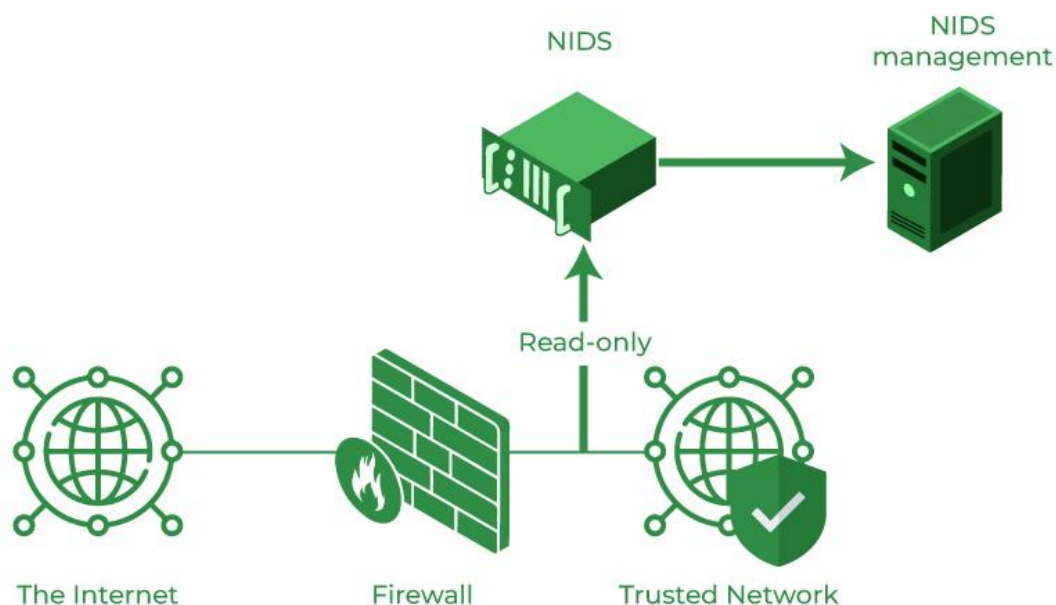
* Host Intrusion Detection System (HIDS): Host intrusion detection systems (HIDS) run on independent hosts or devices on the network. A HIDS monitors the incoming and outgoing packets from the device only and will alert the administrator if suspicious or malicious activity is detected. It takes a snapshot of existing system files and compares it with the previous snapshot. If the analytical system files were edited or deleted, an alert is sent to the administrator to investigate. An example of HIDS usage can be seen on mission-critical machines, which are not expected to change their layout.
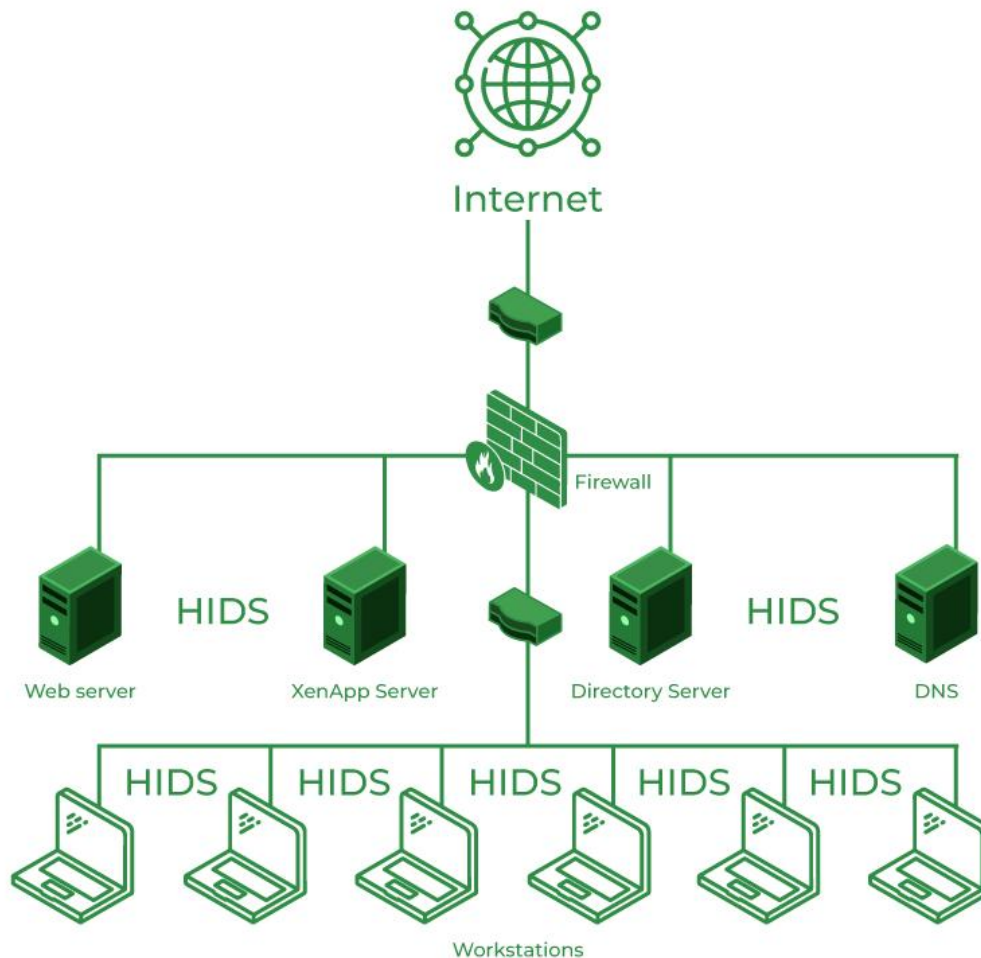
* Protocol-based Intrusion Detection System (PIDS): Protocol-based intrusion detection system (PIDS) comprises a system or agent that would consistently reside at the front end of a server, controlling and interpreting the protocol between a user/device and the server. It is trying to secure the web server by regularly monitoring the HTTPS protocol stream and accepting the related HTTP protocol. As HTTPS is unencrypted and before instantly entering its web presentation layer then this system would need to reside in this interface, between to use the HTTPS.

* Application Protocol-based Intrusion Detection System (APIDS): An application Protocol-based Intrusion Detection System (APIDS) is a system or agent that generally resides within a group of servers. It identifies the intrusions by monitoring and interpreting the communication on application-specific protocols. For example, this would monitor the SQL protocol explicitly to the middleware as it transacts with the database in the web server.

* Hybrid Intrusion Detection System: Hybrid intrusion detection system is made by the combination of two or more approaches to the intrusion detection system. In the hybrid intrusion detection system, the host agent or system data is combined with network information to develop a complete view of the network system. The hybrid intrusion detection system is more effective in comparison to the other intrusion detection system. Prelude is an example of Hybrid IDS.

Benefits of IDS

* Detects malicious activity: IDS can detect any suspicious activities and alert the system administrator before any significant damage is done.

* Improves network performance: IDS can identify any performance issues on the network, which can be addressed to improve network performance.

* Compliance requirements: IDS can help in meeting compliance requirements by monitoring network activity and generating reports.

* Provides insights: IDS generates valuable insights into network traffic, which can be used to identify any weaknesses and improve network security.

Detection Method of IDS

1. Signature-based Method: Signature-based IDS detects the attacks on the basis of the specific patterns such as the number of bytes or a number of 1s or the number of 0s in the network traffic. It also detects on the basis of the already known malicious instruction sequence that is used by the malware. The detected patterns in the IDS are known as signatures. Signature-based IDS can easily detect the attacks whose pattern (signature) already exists in the system but it is quite difficult to detect new malware attacks as their pattern (signature) is not known.

2. Anomaly-based Method: Anomaly-based IDS was introduced to detect unknown malware attacks as new malware is developed rapidly. In anomaly-based IDS there is the use of machine learning to create a trustful activity model and anything coming is compared with that model and it is declared suspicious if it is not found in the model. The machine learning-based method has a better-generalized property in comparison to signature-based IDS as these models can be trained according to the applications and hardware configurations.

Comparison of IDS with Firewalls

IDS and firewall both are related to network security but an IDS differs from a firewall as a firewall looks outwardly for intrusions in order to stop them from happening. Firewalls restrict access between networks to prevent intrusion and if an attack is from inside the network it doesn't signal. An IDS describes a suspected intrusion once it has happened and then signals an alarm.

Intrusion Detection System (IDS) is a powerful tool that can help businesses in detecting and prevent unauthorized access to their network. By analyzing network traffic patterns, IDS can identify any suspicious activities and alert the system administrator. IDS can be a valuable addition to any organization's security infrastructure, providing insights and improving network performance.

What is a Honeypot ?

Honeypot is a network-attached system used as a trap for cyber-attackers to detect and study the tricks and types of attacks used by hackers. It acts as a potential target on the internet and informs the defenders about any unauthorized attempt to the information system.

Honeypots are mostly used by large companies and organizations involved in cybersecurity. It helps cybersecurity researchers to learn about the different type of attacks used by attackers. It is suspected that even the cybercriminals use these honeypots to decoy researchers and spread wrong information.

The cost of a honeypot is generally high because it requires specialized skills and resources to implement a system such that it appears to provide an organization's resources still preventing attacks at the backend and access to any production system.

A honeynet is a combination of two or more honeypots on a network.
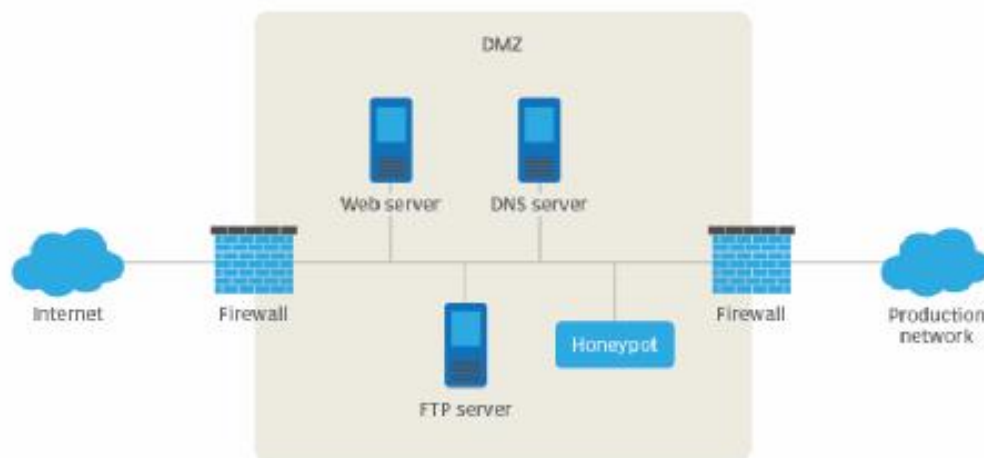
Types of Honeypot

Honeypots are classified based on their deployment and the involvement of the intruder.
Based on their deployment, honeypots are divided into :

1. Research honeypots- These are used by researchers to analyze hacker attacks and deploy different ways to prevent these attacks.

2. Production honeypots- Production honeypots are deployed in production networks along with the server. These honeypots act as a frontend trap for the attackers, consisting of false information and giving time to the administrators to improve any vulnerability in the actual system.

Based on interaction, honeypots are classified into:

1. Low interaction honeypots:Low interaction honeypots gives very little insight and control to the hacker about the network. It simulates only the services that are frequently requested by the attackers. The main operating system is not involved in the low interaction systems and therefore it is less risky. They require very fewer resources and are easy to deploy. The only disadvantage of these honeypots lies in the fact that experienced hackers can easily identify these honeypots and can avoid it.

2. Medium Interaction Honeypots: Medium interaction honeypots allows more activities to the hacker as compared to the low interaction honeypots. They can expect certain activities and are designed to give certain responses beyond what a low-interaction honeypot would give.

3. High Interaction honeypots:A high interaction honeypot offers a large no. of services and activities to the hacker, therefore, wasting the time of the hackers and trying to get complete information about the hackers. These honeypots involve the real-time operating system and therefore are comparatively risky if a hacker identifies the honeypot. High interaction honeypots are also very costly and are complex to implement. But it provides us with extensively large information about hackers.

Advantages of Honeypot

1. Acts as a rich source of information and helps collect real-time data.
2. Identifies malicious activity even if encryption is used.
3. Wastes hackers' time and resources.
4. Improves security.

Disadvantages of Honeypot

1. Being distinguishable from production systems, it can be easily identified by experienced attackers.
2. Having a narrow field of view, it can only identify direct attacks.
3. A honeypot once attacked can be used to attack other systems.
4. Fingerprinting(an attacker can identify the true identity of a honeypot).

16

What is Computer Vision ?

Computer vision means the extraction of information from images, text, videos, etc. Sometimes computer vision tries to mimic human vision. It's a subset of computer-based intelligence or Artificial intelligence which collects information from digital images or videos and analyze them to define the attributes.

The entire process involves image acquiring, screening, analyzing, identifying, and extracting information. This extensive processing helps computers to understand any visual content and act on it accordingly. Computer vision projects translate digital visual content into precise descriptions to gather multi-dimensional data. This data is then turned into a computer-readable language to aid the decision-making process. The main objective of this branch of Artificial intelligence is to teach machines to collect information from images.

Applications of Computer Vision

* Medical Imaging: Computer vision helps in MRI reconstruction, automatic pathology, diagnosis, and computer aided surgeries and more.

* AR/VR: Object occlusion, outside-in tracking, and inside-out tracking for virtual and augmented reality.

* Smartphones: All the photo filters (including animation filters on social media), QR code scanners, panorama construction, Computational photography, face detectors, image detectors like (Google Lens, Night Sight) that we use are computer vision applications.

* Internet: Image search, Mapping, photo captioning, Ariel imaging for maps, video categorization and more.

17

What is IOT ?

IoT stands for Internet of Things. It refers to the interconnectedness of physical devices, such as appliances and vehicles, that are embedded with software, sensors, and connectivity which enables these objects to connect and exchange data. This technology allows for the collection and sharing of data from a vast network of devices, creating opportunities for more efficient and automated systems.

Internet of Things (IoT) is the networking of physical objects that contain electronics embedded within their architecture in order to communicate and sense interactions amongst each other or with respect to the external environment. In the upcoming years, IoT-based technology will offer advanced levels of services and practically change the way people lead their daily lives. Advancements in medicine, power, gene therapies, agriculture, smart cities, and smart homes are just a very few of the categorical examples where IoT is strongly established.

IoT is network of interconnected computing devices which are embedded in everyday objects, enabling them to send and receive data.

Over 9 billion 'Things' (physical objects) are currently connected to the Internet, as of now. In the near future, this number is expected to rise to a whopping 20 billion.

Main components used in IoT:

* Low-power embedded systems: Less battery consumption, high performance are the inverse factors that play a significant role during the design of electronic systems.

* Sensors : Sensors are the major part of any IoT applications. It is a physical device that measures and detect certain physical quantity and convert it into   signal which can be provide as an input to processing or control unit for analysis purpose.

1.Different types of Sensors :
2.Temperature Sensors
3. Image Sensors
4. Gyro Sensors
5. Obstacle Sensors
6. RF Sensor
7. IR Sensor
8. MQ-02/05 Gas Sensor
9. LDR Sensor
10. Ultrasonic Distance Sensor

* Control Units : It is a unit of small computer on a single integrated circuit containing microprocessor or processing core, memory and programmable

input/output devices/peripherals. It is responsible for major processing work of IoT devices and all logical operations are carried out here.

* Cloud computing: Data collected through IoT devices is massive and this data has to be stored on a reliable storage server. This is where cloud computing comes into play. The data is processed and learned, giving more room for us to discover where things like electrical faults/errors are within the system.

* Availability of big data: We know that IoT relies heavily on sensors, especially in real-time. As these electronic devices spread throughout every field, their usage is going to trigger a massive flux of big data.

* Networking connection: In order to communicate, internet connectivity is a must where each physical object is represented by an IP address. However, there are only a limited number of addresses available according to the IP naming. Due to the growing number of devices, this naming system will not be feasible anymore. Therefore, researchers are looking for another alternative naming system to represent each physical object.

There are two ways of building IoT:

1. Form a separate internetwork including only physical objects.

2. Make the Internet ever more expansive, but this requires hard-core technologies such as rigorous cloud computing and rapid big data storage (expensive).

IOT Enablers:

* Collect and Transmit Data : For this purpose sensors are widely used they are used as per requirements in different application areas.

* Actuate device based on triggers produced by sensors or processing devices : If certain condition is satisfied or according to user's requirements if certain trigger is activated then which action to performed that is shown by Actuator devices.

* Receive Information : From network devices user or device can take certain information also for their analysis and processing purposes.

* Communication Assistance : Communication assistance is the phenomena of communication between 2 network or communication between 2 or more IoT devices of same or different Networks. This can be achieved by different communication protocols like : MQTT , Constrained Application Protocol, ZigBee, FTP, HTTP etc.

19
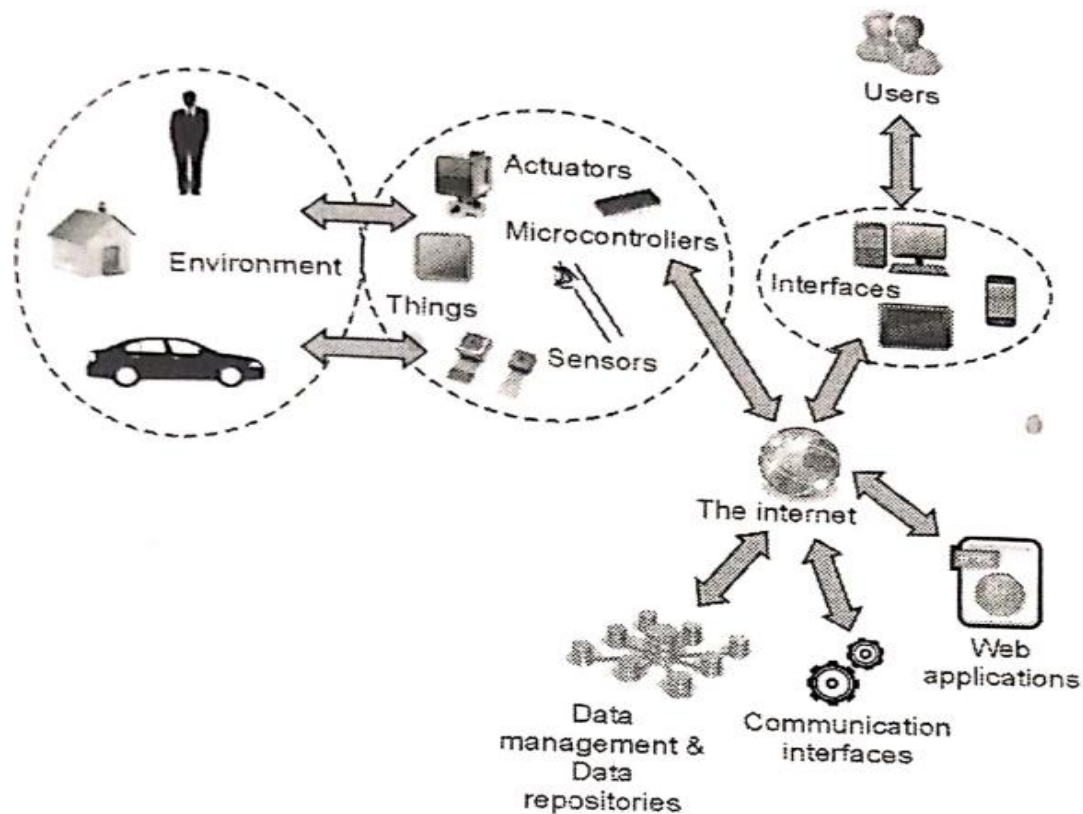
Characterstics of IOT

* Massively scalable and efficient
* IP-based addressing will no longer be suitable in the upcoming future.
* An abundance of physical objects is present that do not use IP, so IoT is made possible.
* Devices typically consume less power. When not in use, they should be automatically programmed to sleep.
* A device that is connected to another device right now may not be connected in another instant of time.
* Intermittent connectivity – IoT devices aren't always connected. In order to save bandwidth and battery consumption, devices will be powered off periodically when not in use. Otherwise, connections might turn unreliable and thus prove to be inefficient.

Definitions , Acronyms , Abbreviations

- **IDS - Intrusion Detection System**IDS stands for Intrusion Detection System. It is a security technology that monitors network traffic and system activities to identify and detect unauthorized access, suspicious behavior, or potential security breaches within a computer network or system.
- **AI - Artificial Intelligence**Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.
- **ML - Machine Learning:**Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.
- **AR - Augmented Reality**(AR) is the integration of digital information with the user's environment in real time.
- **IoT - Internet of Things**The Internet of Things (IoT) describes the network of physical objects— "things"—that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet.
- **CV - Computer Vision**Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information.
- **NMS - Network Management System**Network management system (NMS) is an application or set of applications used to monitor and administer the network, helping admins gain in-depth visibility and control over the network architecture.
- **API - Application Programming Interface**API stands for Application Programming Interface. In the context of APIs, the word Application refers to any software with a distinct function. Interface can be thought of as a contract of service between two applications
- **GUI - Graphical User Interface:**a visual way of interacting with a computer using items such as windows, icons, and menus, used by most modern operating systems.
- **CLI - Command Line Interface**A command-line interface (CLI) is a text-based user interface (UI) used to run programs, manage computer files and interact with the computer
- **HTTP** - Hypertext Transfer Protocol: An application protocol used for transmitting hypertext (web) documents on the internet.
- **HTTPS** - Hypertext Transfer Protocol Secure: An extension of HTTP that provides secure communication over a computer network by encrypting the data.
- **DNS** - Domain Name System: The system that translates human-readable domain names into IP addresses, allowing users to access websites using domain names.
- **IP** - Internet Protocol: A protocol responsible for addressing and routing data packets across the internet.
- **JSON** - JavaScript Object Notation: A lightweight data interchange format that is easy for humans to read and write and easy for machines to parse and generate.
- **CSS** - Cascading Style Sheets: A style sheet language used for describing the presentation of a document written in HTML or XML.
- **HTML** - Hypertext Markup Language: The standard markup language for creating web pages and web applications.
- **JS** - JavaScript: A high-level programming language that is widely used for creating interactive and dynamic web content.
- **XML** - Extensible Markup Language: A markup language that defines rules for encoding documents in a format that is both human-readable and machine-readable.
- **WiFi** - Wireless Fidelity: A wireless communication technology that allows devices to connect to a local area network (LAN) and access the internet.
- **TCP/IP** - Transmission Control Protocol/Internet Protocol: The suite of communication protocols used for transmitting data packets across networks, including the internet.
- **GPIO** - General Purpose Input/Output: A generic pin on a microcontroller that can be configured to either input or output signals.
- **IDE** - Integrated Development Environment: A software application that provides comprehensive tools for writing, testing, and debugging code.
- **SDK** - Software Development Kit: A set of tools, libraries, and documentation that developers use to create software applications for specific platforms or frameworks.

- **URL** - Uniform Resource Locator: The address used to access a resource on the internet, such as a web page or file.
- **API Key** - Application Programming Interface Key: A unique identifier or authentication token used to access and interact with an API.
- **FPS** - Frames Per Second: The number of frames (images) displayed or processed per second in a video or animation.
- **RAM** - Random Access Memory: A type of computer memory that allows data to be read from and written to by the processor.
- **HTTP POST** - Hypertext Transfer Protocol POST: An HTTP method used to send data to the server for processing or storage.
- **HTTP GET** - Hypertext Transfer Protocol GET: An HTTP method used to retrieve data from a server.
- **IDE** - Intrusion Detection and Prevention System: A security mechanism that not only detects but also prevents unauthorized access or malicious activities in a network or system.
- **SSL** - Secure Sockets Layer: A cryptographic protocol that ensures secure communication over a computer network, commonly used to secure HTTPS connections.
- **MQTT** - Message Queuing Telemetry Transport: A lightweight messaging protocol designed for efficient communication in IoT devices and networks.
- **API Documentation** - Documentation that provides detailed information, instructions, and examples on how to use
- **SQL** - Structured Query Language: A programming language used for managing and manipulating relational databases.
- **CDN** - Content Delivery Network: A distributed network of servers that delivers web content to users based on their geographic location, improving website performance and availability.
- **VPN** - Virtual Private Network: A secure and encrypted connection that allows users to access a private network over the internet.
- **SSL/TLS** - Secure Sockets Layer/Transport Layer Security: Protocols that provide secure communication and data encryption over a computer network.
- **API Gateway** - A server or service that acts as an intermediary between an application and an API, handling requests, authentication, and traffic management.
- **CLI** - Command Line Interface: A text-based interface used for executing commands and interacting with a computer system or software.
- **SDN** - Software-Defined Networking: A network architecture that separates the control plane from the data plane, allowing centralized management and programmability of network resources.
- **REST** - Representational State Transfer: An architectural style for designing networked applications, often used in web services development, emphasizing scalability, simplicity, and statelessness.
- **MVC** - Model-View-Controller: A software architectural pattern that separates the representation of information (model), user interface (view), and application logic (controller) in an application.
- **CDN** - Content Delivery Network: A distributed network of servers that caches and delivers web content to users, reducing latency and improving website performance.
- **SSH** - Secure Shell: A cryptographic network protocol that provides secure remote access and secure file transfer between networked devices.
- **OOP** - Object-Oriented Programming: A programming paradigm that organizes data and behavior into reusable objects, focusing on concepts such as encapsulation, inheritance, and polymorphism.
- **JVM** - Java Virtual Machine: A virtual machine that allows Java bytecode to be executed on different hardware platforms, providing platform independence.
- **API Rate Limiting** - A technique used to control and limit the number of requests made to an API within a specified time frame to prevent abuse or overload.
- **XSS** - Cross-Site Scripting: A security vulnerability that allows malicious users to inject malicious scripts into web pages viewed by other users.
- **CSRF** - Cross-Site Request Forgery: A security vulnerability that tricks users into performing unintended actions on a website without their knowledge or consent.
- **JWT** - JSON Web Token: A compact and self-contained token format used for securely transmitting information between parties as a JSON object.
- **ORM** - Object-Relational Mapping: A technique that maps objects from an object-oriented programming language to a relational database, allowing seamless interaction between the two.

Theory

Technical Stack

The project utilizes a diverse set of frameworks, technologies, operating systems, and programming languages to deliver its functionality. Below is the comprehensive technical stack employed in the project:

Front-end:
HTML5
CSS3
JavaScript
Bootstrap framework
XML

Back-end:
Java
Python

Database:
Java
Python

Hardware and IoT:
ESP32 CAM module
Blynk Cloud platform
Arduino IDE

Networking:
TCP/IP
HTTP/HTTPS protocols

Framework:
Django

Web Server:
Apache

Operating System:
Ubuntu Linux

Additional Tools and Libraries:
OpenCV for computer vision processing
TensorFlow for machine learning capabilities
Blynk mobile app for remote control
GitHub for version control and collaboration

Languages

JavaScript: JavaScript is the primary language used for implementing the frontend of the project, including the user interface, interactivity, and dynamic rendering of network information. It is widely supported by web browsers and allows for seamless client-side scripting.

HTML/CSS: HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) are used for structuring and styling the project's web pages. HTML is responsible for defining the page structure and content, while CSS is used for visual presentation, including layout, colors, and typography.

Arduino (C++): Arduino, which uses a simplified version of C++, is employed for programming the ESP32 CAM module. It enables low-level control of the camera module's functions, configuration, and interactions with other components.

Python: Python is utilized for the backend server implementation, including the Blynk cloud module and handling data transmission between the network gateway and the admin control application. Python's versatility and extensive libraries make it suitable for server-side programming tasks.

Java: Java is used for developing the admin control application, which provides an interface for managing and controlling the network gateway. Java's object-oriented programming paradigm and rich ecosystem make it well-suited for building robust desktop applications.

XML: XML (eXtensible Markup Language) is used for data representation and configuration in various parts of the project. It provides a standardized format for storing and exchanging structured data, ensuring compatibility and ease of integration between different components.

The combination of these programming languages forms the core technology stack for the project, enabling the implementation of different modules and functionalities across the frontend, backend, embedded systems, and desktop application layers

Computer Vision Module

Understanding about YOLOv8:

YOLOv8, like other YOLO models, is an acronym for "You Only Look Once," referring to its ability to perform object detection tasks in a single forward pass of the neural network. However, YOLOv8 is much more than just a faster version of its predecessor. It is the newest state-of-the-art model in the YOLO family, featuring a new backbone network, a new loss function, a new anchor-free detection head, and other architectural improvements.

The task of segmenting an image into regions of interest, or image segmentation, has several uses in computer vision, including object identification and recognition, medical imaging, and autonomous vehicles. Deep learning-based algorithms have produced impressive results recently, whereas conventional picture segmentation methods still rely on thresholding and clustering techniques. YOLOv8, a state-of-the-art deep learning model, is at the forefront of cutting-edge picture segmentation technology and is capable of performing object identification, classification, and instance segmentation tasks with amazing speed and accuracy. The YOLOv8 model, which offers a number of enhancements and features over its predecessors, was created by Ultralytics, the same group responsible for the hugely successful YOLOv5 model.

Image Segmentation performed by YOLOv8:

Image segmentation is the task of partitioning an image into multiple regions or segments, where each segment corresponds to a specific object or region of interest. YOLOv8 is a popular object detection algorithm that can be adapted for image segmentation.

In the context of YOLOv8, the network is initially trained to perform object detection. It learns to predict the location and class of objects in an image by dividing the image into a grid and assigning bounding boxes to objects present in each grid cell. This training process involves optimizing the network's parameters to minimize the difference between predicted bounding boxes and the ground truth bounding boxes.

25

To generate a segmentation mask, the predictions made by YOLOv8 are used. Each bounding box prediction consists of the coordinates of the bounding box and the predicted class of the object within it. These predictions are then utilized to determine the pixel-level boundaries of each object.

One common approach is to associate each pixel within a bounding box with the corresponding object class. This can be achieved by assigning a specific value or label to all pixels within the bounding box region. For example, if the object within a bounding box is a car, all pixels within that box are labelled as "car" in the segmentation mask.

However, a single bounding box may not cover the entire extent of an object, especially if the object is large or irregularly shaped. To handle this, additional techniques can be applied to refine the segmentation mask. For instance, post-processing methods such as contour detection or region growing algorithms can be utilized to expand the segmented regions based on the boundaries defined by the bounding boxes.

YOLOv8 Instance Segmentation with Pre-Trained Models and the COCO128-seg Dataset:

Overall, by training YOLOv8 for object detection and utilizing its predictions, it is possible to generate a segmentation mask that delineates the pixel-level boundaries of each object in an image. This mask can be useful for various computer vision tasks, such as instance segmentation, object tracking, or semantic segmentation.

Instance segmentation is indeed a subfield of image segmentation that goes beyond identifying object boundaries and aims to assign unique labels to each instance of an object in an image. It provides a more detailed understanding of the scene by distinguishing between individual objects of the same class and capturing their precise shapes.

YOLOv8, which is primarily an object detection algorithm, also offers pre-trained segmentation models with the "-seg" suffix, such as "yolov8n-seg.pt". These models are specifically trained for instance segmentation and can be utilized to perform instance-level analysis on images.

The pre-trained segmentation models for YOLOv8, such as "yolov8n-seg.pt," are trained on the COCO128-seg dataset. The COCO128-seg dataset is a variant of the

popular COCO (Common Objects in Context) dataset, which contains a large number of images with 80 different object classes. However, the COCO128-seg dataset is a subset of COCO that includes only 128 object classes.

During the training process, the YOLOv8 model is trained on the COCO128-seg dataset for 100 epochs. An epoch refers to a complete iteration over the entire training dataset. The image size used during training is 640 pixels. This size determines the resolution at which the images are processed during both training and inference stages. By training on the COCO128-seg dataset and using the pre-trained "yolov8n-seg.pt" model, you can leverage the power of YOLOv8 for instance segmentation tasks. This means that the model will not only identify objects and their boundaries but also differentiate between instances of the same object class and assign unique labels to each instance. This enables more detailed analysis and understanding of the objects and their shapes within the image.

The Benefits of YOLOv8 for Image Segmentation

1) Speed is indeed one of the significant advantages of YOLOv8 compared to other deep learning models, making it a popular choice for various real-time applications. According to Ultralytics, a company specializing in computer vision research and development, YOLOv8 demonstrates exceptional performance in terms of speed.

When it comes to image segmentation, YOLOv8 exhibits impressive speed, capable of processing frames at a rate of 81 frames per second. This speed far surpasses other state-of-the-art models like Mask R-CNN, which typically achieves around 6 frames per second for image segmentation tasks. This significant difference in processing speed makes YOLOv8 highly desirable for real-time applications where timely and efficient analysis of visual data is crucial.

The high speed of YOLOv8 is particularly advantageous for tasks such as self-driving cars, security cameras, and video analysis. In the context of self-driving cars, real-time image segmentation is essential for perceiving the surrounding environment, detecting objects, and making instant decisions based on the segmentation results. The fast processing capability of YOLOv8 ensures that the car can quickly and accurately interpret the scene, enabling timely responses and enhancing safety.

Similarly, in security camera systems, real-time image segmentation enables the detection and tracking of objects of interest, such as intruders or suspicious activities.

YOLOv8's speed allows for immediate analysis of the video feed, enabling rapid alerts and responses to potential security threats.

Moreover, for video analysis applications, where large amounts of data need to be processed in real-time, YOLOv8's speed is a significant advantage. It enables efficient and quick extraction of object-level information, facilitating tasks such as object tracking, behavior recognition, and content analysis.

The exceptional speed of YOLOv8 can be attributed to its architecture and design choices, such as its efficient backbone network and the use of anchor boxes for bounding box predictions. These optimizations allow for faster inference times without compromising on accuracy, making YOLOv8 an ideal choice for real-time image segmentation tasks.

2) Accuracy is a crucial aspect of any computer vision model, and YOLOv8 excels in this regard, demonstrating impressive performance in object detection and image segmentation tasks. Despite its high speed, YOLOv8 maintains a remarkable level of accuracy, making it a compelling choice for various applications.

The mean average precision (mAP) score is a widely used metric to evaluate the accuracy of object detection and image segmentation models. YOLOv8 achieves a superior mAP score compared to other state-of-the-art models like Detectron2, with a notable improvement of up to 44%. For instance, YOLOv8 achieves an mAP of 63.2% on the challenging COCO (Common Objects in Context) dataset, demonstrating its ability to accurately detect and segment objects.

The improved accuracy of YOLOv8 can be attributed to its advanced architecture and refined loss function. YOLOv8 utilizes a deep neural network architecture that combines various techniques, including a feature pyramid network, anchor boxes, and multi-scale predictions. These components allow YOLOv8 to capture and represent objects of different sizes and scales effectively, enhancing its accuracy in detecting objects across the image.

Moreover, YOLOv8 incorporates an improved loss function that addresses the challenges of false positives and false negatives. The loss function optimizes the model's parameters by minimizing the discrepancies between predicted bounding boxes and ground truth bounding boxes. This optimization process helps reduce false positives (incorrectly identifying non-existent objects) and false negatives (failing to

detect actual objects), leading to improved accuracy in object detection and segmentation.

By balancing speed and accuracy, YOLOv8 provides a compelling solution for various applications. Its accuracy ensures reliable and precise detection and segmentation results, while its high-speed processing enables real-time or near-real-time analysis of images and videos.

It is worth noting that while YOLOv8 achieves impressive accuracy, the specific mAP score may vary depending on the dataset and the specific configuration of the model. Different training strategies, datasets, and hyperparameter choices can influence the accuracy achieved by YOLOv8. Thus, it is essential to consider the specific evaluation metrics and experimental conditions when assessing its accuracy for a particular task or dataset.

3) Flexibility is a key attribute of YOLOv8, as it offers a unified framework that can handle various image segmentation tasks within a single model. This versatility enables YOLOv8 to be applied to a wide range of applications that require multiple tasks to be performed simultaneously or interchangeably.

One of the notable advantages of YOLOv8's unified framework is its ability to perform object detection. Object detection involves identifying and localizing objects within an image. YOLOv8 excels in this task by dividing the image into a grid and predicting bounding boxes for objects present in each grid cell. This capability is highly valuable in applications where detecting and locating objects of interest is critical, such as autonomous vehicles that need to identify pedestrians, vehicles, and traffic signs.

In addition to object detection, YOLOv8 can also perform instance segmentation. Instance segmentation goes beyond object detection by not only identifying object boundaries but also assigning unique labels to each instance of an object. This is particularly useful when differentiating between multiple instances of the same object class is necessary. For example, in video surveillance, YOLOv8 can accurately segment and label individuals within a crowd, allowing for detailed tracking and analysis.

Furthermore, YOLOv8's flexibility extends to image classification tasks. Image classification involves assigning a label or category to an entire image. With its unified framework, YOLOv8 can not only detect and segment objects but also

classify the entire image, providing a comprehensive understanding of the visual content. This flexibility makes YOLOv8 suitable for applications such as image search engines, where the model needs to recognize and classify images based on their content.

The ability to handle multiple tasks within a single model makes YOLOv8 highly efficient and resource-friendly. Rather than training and deploying separate models for each segmentation task, YOLOv8 enables a unified approach, reducing computational complexity and memory requirements. This is particularly beneficial in resource-constrained scenarios, where efficient utilization of computing resources is crucial.

4) Pre-trained models provided by YOLOv8 are a significant advantage for developers as they offer ready-to-use models trained on large-scale datasets such as COCO (Common Objects in Context) and VOC (Visual Object Classes). These pre-trained models are already equipped with knowledge gained from extensive training on diverse and representative data, making them a valuable resource for various segmentation tasks.

For object detection, YOLOv8 provides pre-trained models that are capable of accurately detecting and localizing objects within images. These models have learned to recognize a wide range of object classes from the COCO dataset, which contains thousands of labeled images spanning 80 different categories. By utilizing a pre-trained object detection model, developers can save significant time and effort that would otherwise be required for training from scratch.

Instance segmentation pre-trained models offered by YOLOv8 go beyond object detection by providing the ability to differentiate between instances of the same object class. These models have been trained on datasets such as COCO, where instances are labeled individually, allowing for precise segmentation at the instance level. By starting with a pre-trained instance segmentation model, developers can benefit from the expertise of the model in accurately segmenting objects and assigning unique labels to each instance.
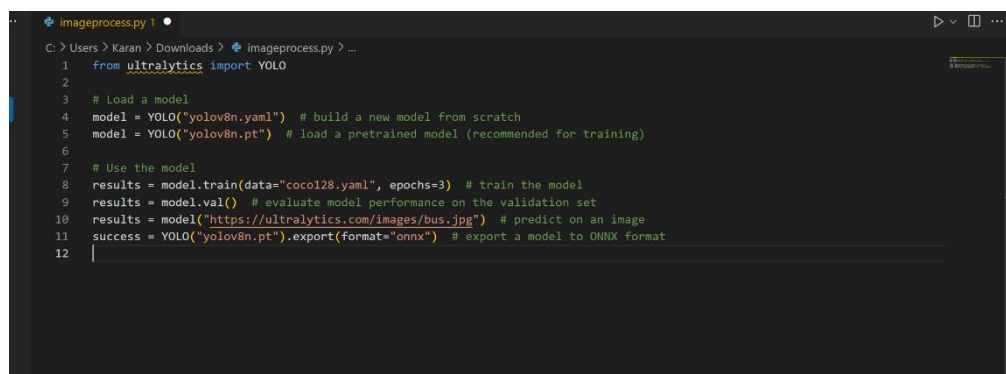
In addition to object detection and instance segmentation, YOLOv8 also offers pre-trained models for image classification. These models are trained on datasets like COCO and VOC, which comprise images from various categories. By utilizing a pre-trained image classification model, developers can perform high-level categorization

of images without the need for additional training. This can be particularly useful in scenarios where quick analysis or content-based filtering is required.

The availability of pre-trained models in YOLOv8 significantly reduces the time and resources needed for training from scratch. Developers can leverage the knowledge captured by these models on large and diverse datasets, saving valuable time and computational resources. Additionally, pre-trained models serve as a starting point for fine-tuning, allowing developers to adapt the models to their specific use cases or target domains. Fine-tuning involves retraining the pre-trained model on a smaller, domain-specific dataset to achieve even better performance.

YOLOv8 Python Package:

In addition to the CLI tool available, YOLOv8 is now distributed as a PIP package. This makes local development a little harder, but unlocks all of the possibilities of weaving YOLOv8 into your Python code

```python
from ultralytics import YOLO

# Load a model
model = YOLO("yolov8n.yaml")  # build a new model from scratch
model = YOLO("yolov8n.pt")  # load a pretrained model (recommended for training)

# Use the model
results = model.train(data="coco128.yaml", epochs=3)  # train the model
results = model.val()  # evaluate model performance on the validation set
results = model("https://ultralytics.com/images/bus.jpg")  # predict on an image
success = YOLO("yolov8n.pt").export(format="onnx")  # export a model to ONNX format
```

Deep Dive into our Computer Vision Model:

Introduction:

The Real-Time Intruder Detection and Security System is a computer vision model designed to monitor a camera feed in real-time and detect potential intruders. The system employs advanced image processing techniques to identify individuals and classify them as known or unknown based on comparison with a pre-existing database of relatives and friends. If an unknown person is detected, the system initiates a series of security measures to ensure the safety of the premises and its occupants.

Operation Workflow The Real-Time Intruder Detection and Security System follows the following workflow:

a. **Camera Feed Processing:** The live camera feed is continuously analyzed by the computer vision model. The model processes the frames, identifies individuals, and extracts relevant features for further analysis.

b. **Person Detection and Classification:** Using advanced image processing techniques, the model performs person detection by identifying human shapes and silhouettes within the camera feed. Detected individuals are then classified as known or unknown.

c**. Database Comparison:** If an individual is classified as known, the system compares their features with the information stored in the database. If a match is found, the system allows entry to the authorized person.

d. **Intruder Identification:** If an unknown person is detected, the system flags them as a potential intruder and proceeds to analyse their behaviour for any signs of unethical activity. The model can detect weapon-like objects or unusual movements that may indicate a threat.

e. **Security Measures Activation:** Upon identifying an unknown person engaged in suspicious behaviour, the system activates the security measures. The locking mechanism of the house is engaged to restrict access, and a high-beam alarm system is triggered to deter the intruder and attract attention.

f. **Real-time Notifications:** Simultaneously, the system sends real-time notifications to the nearest neighbours, alerting them about the potential intrusion. This ensures that they are promptly informed and can take appropriate action.

System's features and benefits:

Accurate Intruder Detection: The computer vision model used by the system employs advanced deep learning techniques, resulting in highly accurate person detection and classification. This means that the system can reliably distinguish between individuals present in the camera feed, minimizing false positives and false negatives.

Efficient Database Comparison: The system integrates with a comprehensive database that stores the information of authorized individuals such as relatives and friends. When an unknown person is detected, the system swiftly compares their features with the database to determine if they are a known individual. This efficient

database comparison enables effective access control, reducing the risk of false alarms and ensuring that authorized individuals are granted access without delay.
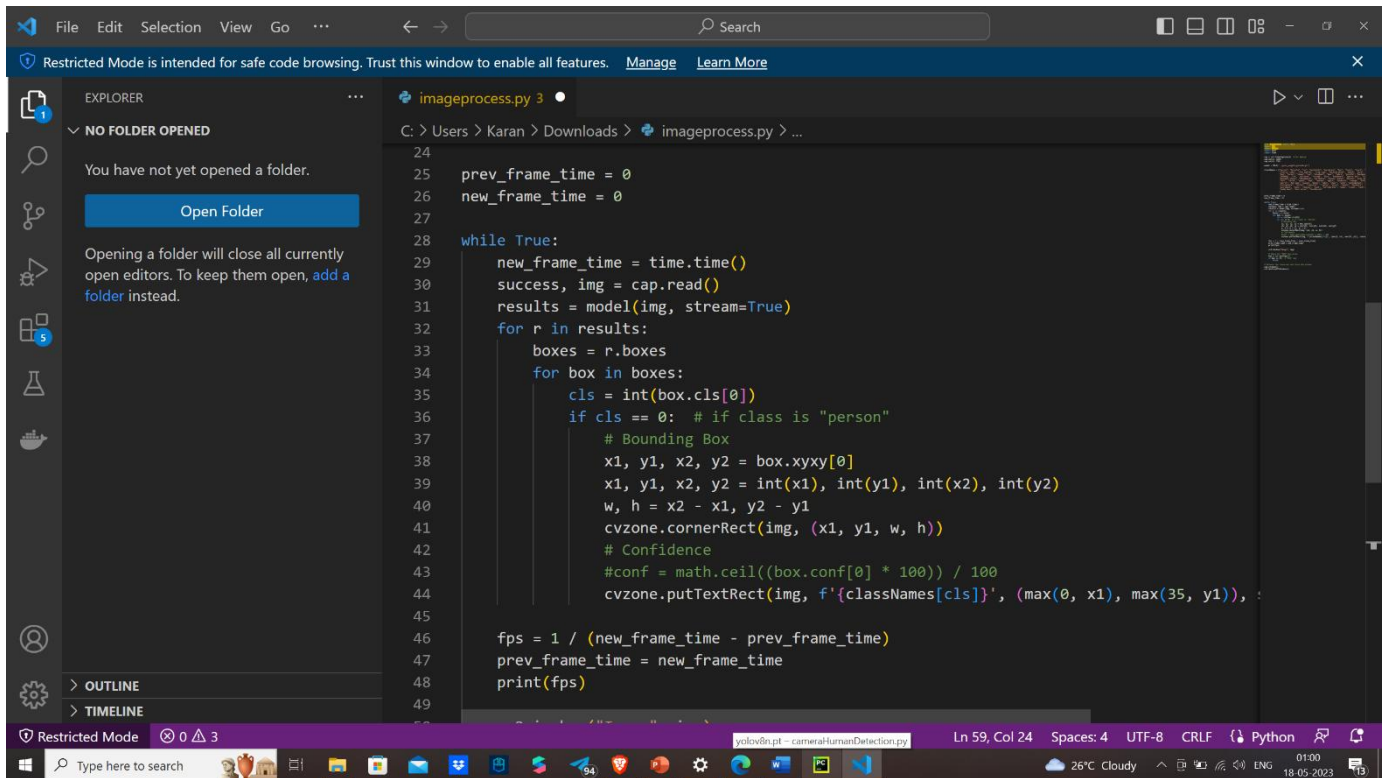
Robust Security Measures: Upon detecting an unknown person engaged in suspicious behaviour, the system activates a series of security measures to protect the premises. The locking mechanism of the house is initiated, preventing further access to unauthorized individuals. Additionally, a high-beam alarm system is triggered, which serves as a deterrent and draws attention to the potential intrusion. These robust security measures work in tandem to enhance the safety and security of the premises and its occupants.

Real-time Alerts: One of the key strengths of the system is its ability to provide real-time notifications. When an unknown intruder is detected and the security measures are activated, the system immediately sends alerts to the nearest neighbours. These real-time notifications enable the neighbours to be promptly informed about the potential intrusion, empowering them to take appropriate actions such as contacting authorities or providing assistance. The quick dissemination of information significantly enhances the overall security response time.

Comprehensive Monitoring: The system continuously monitors the camera feed in real-time, ensuring constant vigilance over the premises. It analyses the behaviour and movements of individuals to detect any signs of unethical or suspicious activity. This comprehensive monitoring capability allows the system to identify potential threats beyond just the presence of an unknown person, such as the detection of weapon-like objects. By detecting and addressing such behaviours, the system contributes to maintaining a secure environment.

Customizability and Scalability: The Real-Time Intruder Detection and Security System can be customized and scaled according to specific requirements. It can accommodate different camera setups and configurations, making it adaptable to various premises and surveillance needs. The system's architecture allows for seamless integration with additional security components or technologies, enabling further enhancements to the overall security infrastructure.

#Code:


```
24
25    prev_frame_time = 0
26    new_frame_time = 0
27
28    while True:
29        new_frame_time = time.time()
30        success, img = cap.read()
31        results = model(img, stream=True)
32        for r in results:
33            boxes = r.boxes
34            for box in boxes:
35                cls = int(box.cls[0])
36                if cls == 0:  # if class is "person"
37                    # Bounding Box
38                    x1, y1, x2, y2 = box.xyxy[0]
39                    x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
40                    w, h = x2 - x1, y2 - y1
41                    cvzone.cornerRect(img, (x1, y1, w, h))
42                    # Confidence
43                    #conf = math.ceil((box.conf[0] * 100)) / 100
44                    cvzone.putTextRect(img, f'{classNames[cls]}', (max(0, x1), max(35, y1)),
45
46        fps = 1 / (new_frame_time - prev_frame_time)
47        prev_frame_time = new_frame_time
48        print(fps)
49
```
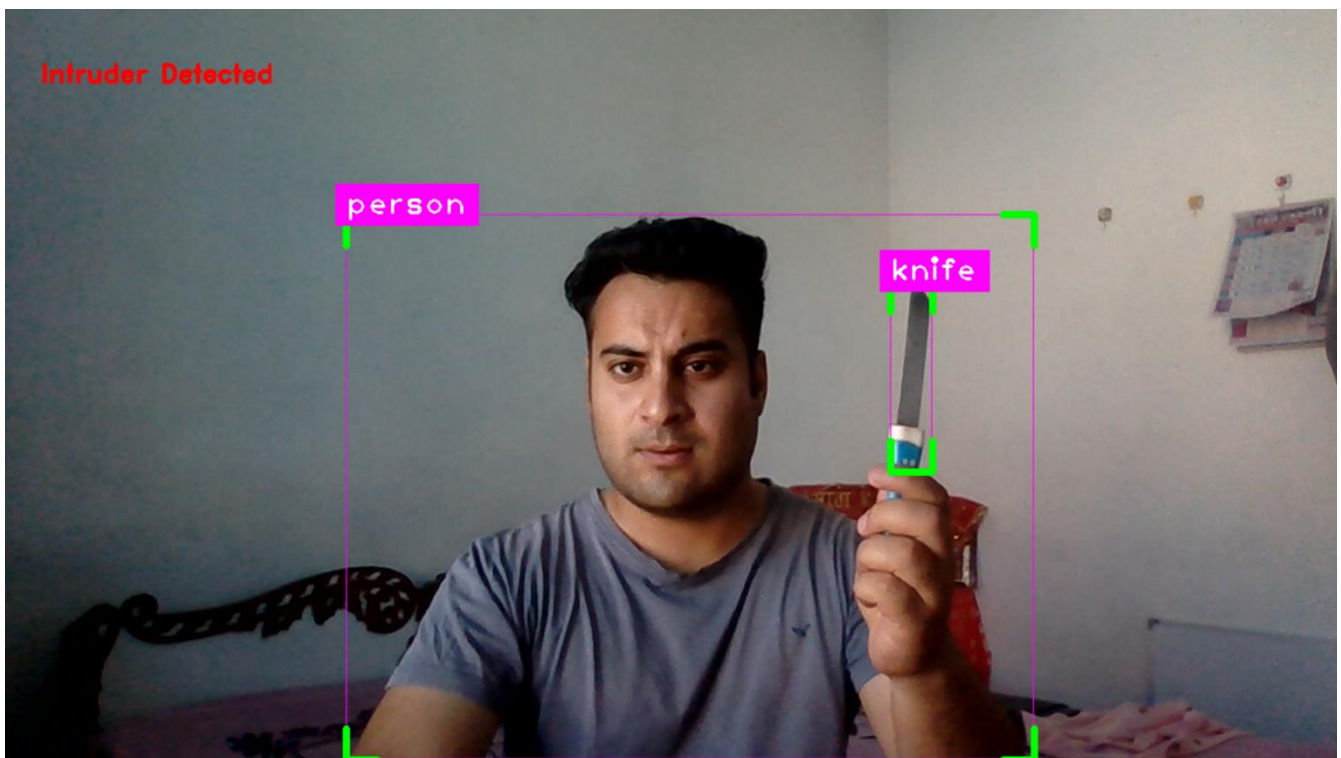

#Output:

34

Surveillance Module

Implementing a full fledged camera surveillance system coversup most of the premiss areas but it comes with a pretty nifty cost andmaintainece . Which is why goig with a $10 ESP-32 Cam modul is the best way to go.

The ESP32-CAM offers several advantages over traditional company surveillance cameras, making it a compelling choice for various applications. Firstly, the ESP32-CAM is highly cost-effective compared to commercial surveillance cameras, making it a more affordable option for individuals or organizations on a limited budget. Additionally, the ESP32-CAM is compact and lightweight, allowing for discreet placement in various locations where traditional cameras may not be suitable or easily installed.

Another significant advantage of the ESP32-CAM is its versatility. It is a programmable device that can be customized and integrated into existing systems or projects, providing flexibility in functionality. With its built-in Wi-Fi capabilities, the ESP32-CAM can connect to a network, stream video footage, and support real-time monitoring and remote access via mobile devices or web interfaces.

Furthermore, the ESP32-CAM offers the convenience of easy setup and configuration. It can be quickly installed and configured using the Arduino IDE, simplifying the development process for users with programming experience. Its compatibility with various libraries and APIs allows for seamless integration with other devices or platforms.The pictorial representation of the ESP-32 Cam is depicted below.

Fig : EP-32 Cam

Features

Here is a list with the ESP32-CAM features:

1. The smallest 802.11b/g/n Wi-Fi BT SoC module
2. Low power 32-bit CPU,can also serve the application processor
3. Up to 160MHz clock speed, summary computing power up to 600 DMIPS
4. Built-in 520 KB SRAM, external 4MPSRAM
5. Supports UART/SPI/I2C/PWM/ADC/DAC
6. Support OV2640 and OV7670 cameras, built-in flash lamp
7. Support image WiFI upload
8. Support TF card
9. Supports multiple sleep modes
10. Embedded Lwip and FreeRTOS
11. Supports STA/AP/STA+AP operation mode
12. Support Smart Config/AirKiss technology
13. Support for serial port local and remote firmware upgrades (FOTA)
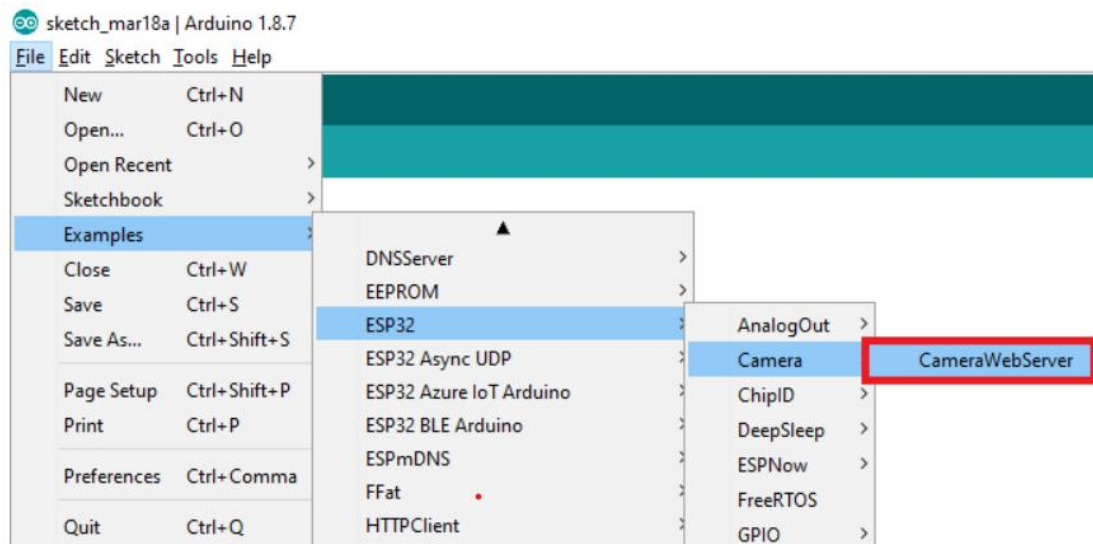
Circuit
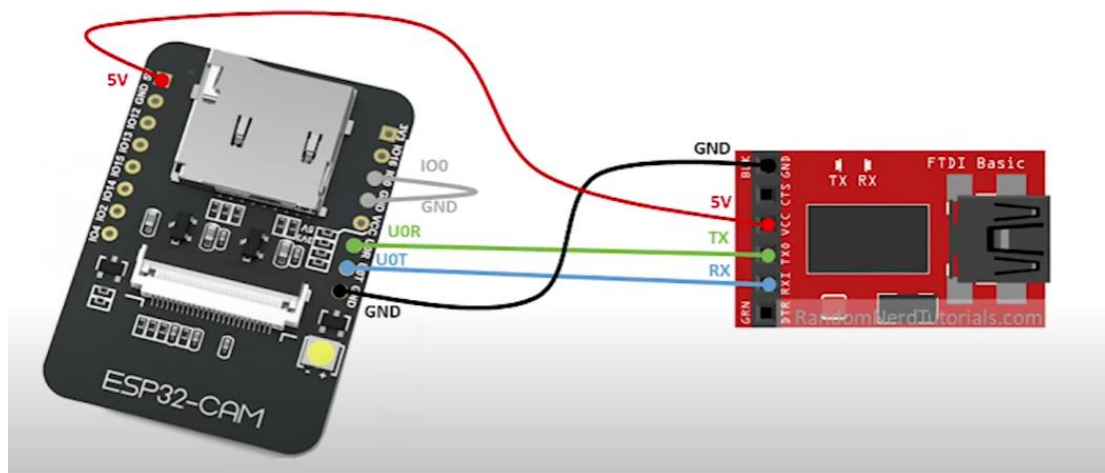
Video Streaming Server

Developing the script:

1. Install the ESP32 add-on

Configuring the ide section can be found in the Honeypot module section

In your Arduino IDE, go to **File** > **Examples** > **ESP32** > **Camera** and open the **CameraWebServer** example.

Code:

```
#include "esp_camera.h"
     #include <WiFi.h>

//
// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
//          Ensure ESP32 Wrover Module or other board with PSRAM is selected
//          Partial images will be transmitted if image exceeds buffer size
//

// Select camera model
     #define CAMERA_MODEL_WROVER_KIT // Has PSRAM
//#define CAMERA_MODEL_ESP_EYE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
//#define CAMERA_MODEL_AI_THINKER // Has PSRAM
//#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM

     #include "camera_pins.h"

     const char* ssid = "*********";
     const char* password = "*********";

     void startCameraServer();

     void setup() {
     Serial.begin(115200);
     Serial.setDebugOutput(true);
     Serial.println();

     camera_config_t config;
     config.ledc_channel = LEDC_CHANNEL_0;
     config.ledc_timer = LEDC_TIMER_0;
     config.pin_d0 = Y2_GPIO_NUM;
     config.pin_d1 = Y3_GPIO_NUM;
     config.pin_d2 = Y4_GPIO_NUM;
     config.pin_d3 = Y5_GPIO_NUM;
     config.pin_d4 = Y6_GPIO_NUM;
```

```cpp
    config.pin_d5 = Y7_GPIO_NUM;
    config.pin_d6 = Y8_GPIO_NUM;
    config.pin_d7 = Y9_GPIO_NUM;
    config.pin_xclk = XCLK_GPIO_NUM;
    config.pin_pclk = PCLK_GPIO_NUM;
    config.pin_vsync = VSYNC_GPIO_NUM;
    config.pin_href = HREF_GPIO_NUM;
    config.pin_sscb_sda = SIOD_GPIO_NUM;
    config.pin_sscb_scl = SIOC_GPIO_NUM;
    config.pin_pwdn = PWDN_GPIO_NUM;
    config.pin_reset = RESET_GPIO_NUM;
    config.xclk_freq_hz = 20000000;
    config.pixel_format = PIXFORMAT_JPEG;

    // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
    //                   for larger pre-allocated frame buffer.
    if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
    } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
    }

    #if defined(CAMERA_MODEL_ESP_EYE)
    pinMode(13, INPUT_PULLUP);
    pinMode(14, INPUT_PULLUP);
    #endif

    // camera init
    esp_err_t err = esp_camera_init(&config);
    if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
    }

    sensor_t * s = esp_camera_sensor_get();
    // initial sensors are flipped vertically and colors are a bit saturated
    if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1); // flip it back
    s->set_brightness(s, 1); // up the brightness just a bit
    s->set_saturation(s, -2); // lower the saturation
    }
    // drop down frame size for higher initial frame rate
    s->set_framesize(s, FRAMESIZE_QVGA);

    #if                     defined(CAMERA_MODEL_M5STACK_WIDE) ||
defined(CAMERA_MODEL_M5STACK_ESP32CAM)
    s->set_vflip(s, 1);
    s->set_hmirror(s, 1);
    #endif

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
```

```
    startCameraServer();

    Serial.print("Camera Ready! Use 'http://");
    Serial.print(WiFi.localIP());
    Serial.println("' to connect");
}

void loop() {
// put your main code here, to run repeatedly:
delay(10000);
}
```

The code begins by selecting the camera model by uncommenting the corresponding #define statement. The available camera models include WROVER_KIT, ESP_EYE, M5STACK_PSRAM, M5STACK_V2_PSRAM, M5STACK_WIDE, M5STACK_ESP32CAM, AI_THINKER, and TTGO_T_JOURNAL.

The SSID and password for the Wi-Fi network are defined as variables. These credentials will be used to connect the ESP32-CAM module to the desired network.

The setup() function is then defined. It starts the serial communication, configures debug output, and initializes the camera configuration structure, "config," with the appropriate pin mappings and settings. The "config" structure specifies various GPIO pins for different camera functions, such as data lines, clock signals, synchronization signals, and power-related pins. The pixel format is set to PIXFORMAT_JPEG, indicating that the captured images will be in JPEG format.

If the PSRAM (pseudo-static random access memory) is detected, the configuration is adjusted to support higher resolution (UXGA) and lower JPEG quality with a larger frame buffer. If PSRAM is not present, the configuration uses SVGA resolution and higher JPEG quality with a single frame buffer.

Additional pin configurations and settings specific to certain camera models are included within conditional compilation blocks.

The esp_camera_init(&config) function is called to initialize the camera with the provided configuration. If the initialization fails, an error message is printed.

Further adjustments are made to the sensor settings to account for specific camera models, such as flipping the image vertically, adjusting brightness, and saturation. The frame size is set to QVGA for a higher initial frame rate.

The code then proceeds to connect to the specified Wi-Fi network using the provided SSID and password. It waits until the connection is established and displays a message indicating a successful connection.

The startCameraServer() function is called to start the camera server, which enables the ESP32-CAM module to serve a web interface for accessing and viewing the camera stream.

39

Finally, the serial console prints the IP address of the ESP32-CAM module, indicating that it is ready for use.

In the loop() function, there is a delay of 10 seconds, allowing for the execution of any additional code that may be added in the future.
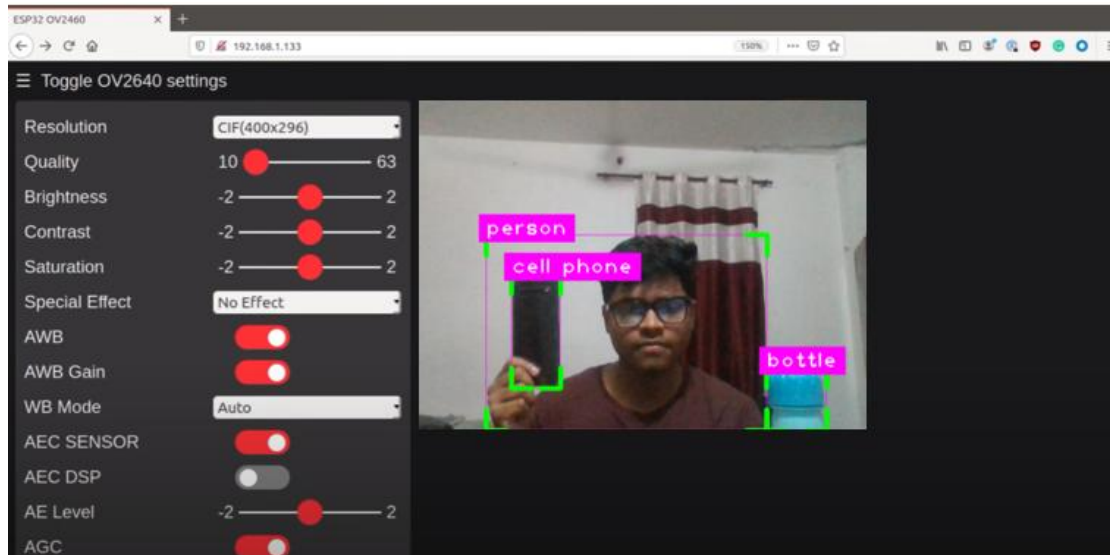
With our Computer vision module we were able to get the live ai feed and implement esp 32 features on top of that in order to make the detections while in transit. The optimun output of the module was rendered as follows:

IOT Module

Building a cost effective and industry grade product requires top notch state of the art components which comes with a heavy price. Developing something that would meet the requirements of the general public like relaibility , cost effectiveness and extensibility requires reengineering the concepts and bind together an alternate solution that would mimic the industry grade product. Keeping inmind the general publics ideoligy and requirement components like Raspberry Pi , Arduino , ESP32 etc make the best fit. The same components have been used in this project to achieve many of the impportant modules one of them being the IOT.

More information on the indivisual components is given below:

ESP 8266:

The ESP8266 is a low-cost, low-power, and highly integrated Wi-Fi SOC (system on a chip) developed by Espressif Systems. It combines a microcontroller unit (MCU) and Wi-Fi functionality, making it capable of connecting to Wi-Fi networks and communicating with other devices over the internet. It is an highly integrated chip designed to provide full internet connectivity in a small package.

Features:

CPU: The ESP8266 integrates a 32-bit Tensilica microcontroller unit (MCU) with a clock speed of up to 160 MHz.

Wi-Fi Connectivity: It supports 802.11 b/g/n Wi-Fi standards and can function as a Wi-Fi client or create its own access point (AP).

Memory: The module typically includes flash memory for program storage and RAM for data storage, ranging from 512 KB to 4 MB.

GPIO: General Purpose Input/Output (GPIO) pins allow the module to interface with various sensors, actuators, and external devices.

Serial Communication: It supports UART (Universal Asynchronous Receiver-Transmitter) communication for serial data transfer with other devices.

Power Management: The module operates at low power consumption, making it suitable for battery-powered IoT applications.

Programming: The ESP8266 can be programmed using the Arduino IDE or other development platforms, making it accessible to a broad range of developers.
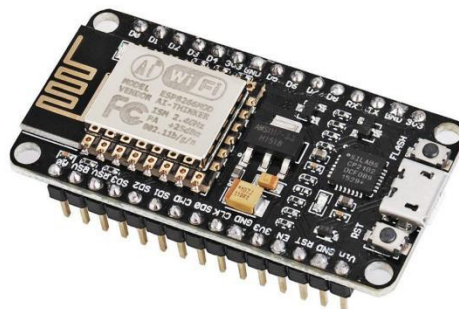
The Raspberry Pi 4 Model B:

The Raspberry Pi 4 Model B is a single-board computer developed by the Raspberry Pi Foundation. It is the fourth-generation model in the Raspberry Pi series and offers significant improvements in performance and features compared to its predecessors.

Specifications:

CPU: The Raspberry Pi 4 is equipped with a quad-core ARM Cortex-A72 CPU, running at 1.5GHz. This offers a significant performance boost compared to previous models.

RAM: It comes with different RAM options, including 2GB, 4GB, or 8GB LPDDR4-3200 SDRAM, providing increased memory capacity for improved multitasking and performance.

GPU: The board features a VideoCore VI graphics processor, supporting 4K video playback and OpenGL ES 3.0 graphics acceleration.
Connectivity: The Raspberry Pi 4 includes dual-band 2.4GHz and 5GHz wireless

LAN, as well as Bluetooth 5.0. It also offers Gigabit Ethernet for high-speed wired connections.
USB Ports: There are two USB 2.0 ports and two USB 3.0 ports, allowing for various peripheral connections.

Video Outputs: It provides two micro HDMI ports with support for up to 4K resolution, making it suitable for multimedia applications and dual-monitor setups.
Storage: The board features a microSD card slot for the operating system and storage, with improved data transfer rates compared to previous models.
GPIO Pins: The Raspberry Pi 4 retains the 40-pin GPIO header, allowing for hardware interfacing and expansion using add-on boards and HATs (Hardware Attached on Top)
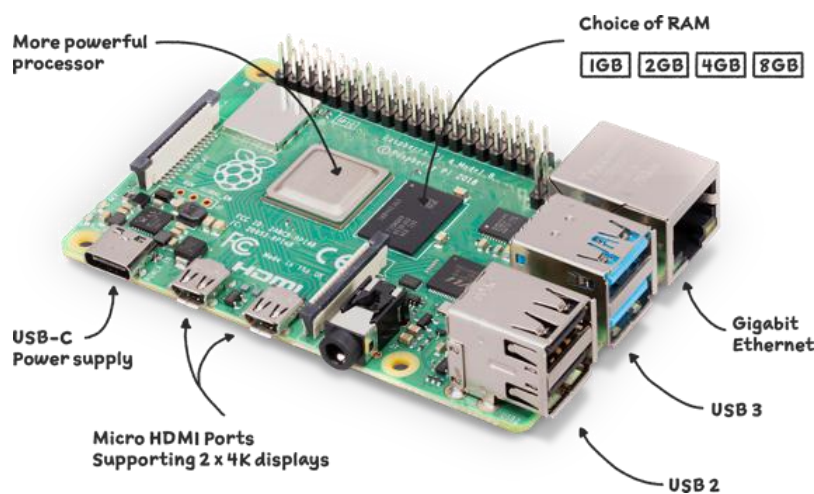
Raspberry Pi 4 Model B

In our project, we utilized the Raspberry Pi 4 Model B in conjunction with a computer vision (CV) module to enable advanced visual processing and analysis capabilities. The Raspberry Pi 4, with its powerful CPU and GPU, provides the computational resources necessary for real-time image and video processing.

The CV module, which can be an external camera or an integrated camera module connected to the Raspberry Pi, captures visual input from the environment. This visual input can include images, video streams, or even live feeds from surveillance cameras.

The Raspberry Pi 4 processes the captured visual data using various CV algorithms and techniques. These algorithms can include image recognition, object detection, facial recognition, motion tracking, and more. The board's processing power allows for quick and efficient analysis of visual data, enabling near real-time responses.
Raspberry Pi 4's connectivity options, such as Wi-Fi and Ethernet, allow for seamless integration with cloud-based CV services or remote access for monitoring and control. This expands the possibilities for our project, such as leveraging cloud-based machine learning models for more advanced CV tasks or remotely accessing and managing the CV module.

Camera Module:

The camera module used in our project, integrated with the Raspberry Pi, serves as a crucial component for intruder detection and surveillance applications. The camera module, specifically designed for the Raspberry Pi, captures high-resolution images and video streams, enabling real-time monitoring and analysis of the surrounding environment.

The camera module is a small, lightweight device that connects to the Raspberry Pi's dedicated camera port. It provides a wide range of features and capabilities, such as adjustable focus, customizable exposure settings, and the ability to capture both still images and video footage.
In our project, we utilized the camera module in combination with the Raspberry Pi to implement an intruder detection system. The camera captures images or video frames of the monitored area at regular intervals or continuously, depending on the project requirements. These visual data are then processed and analyzed by the Raspberry Pi in real-time using computer vision algorithms Once an intruder is detected, the Raspberry Pi can trigger appropriate actions, such as sending notifications or alerts to connected devices.

The camera module's integration with the raspberry Pi allows for seamless control and customization. Through programming, we can adjust camera settings, such as resolution, frame rate, and exposure, to optimize image quality and adapt to varying lighting conditions.

Furthermore, the Raspberry Pi's connectivity options, such as Wi-Fi or Ethernet, enable remote access and control of the camera module. This allows us to monitor the live camera feed remotely, receive real-time notifications, or even access stored images or video footage from anywhere with an internet connection.

Camera Module
IOT SENSORS USED:
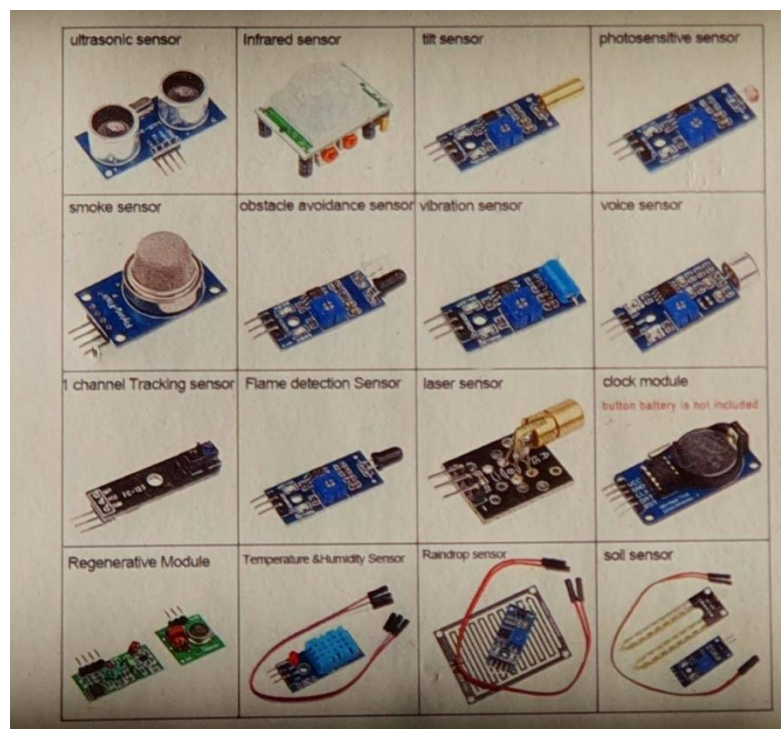
What Is Blynk Cloud

Blynk enables you to create mobile apps to interact with your hardware projects without the need for extensive mobile app development knowledge. You can design buttons, sliders, graphs, and other UI elements in the Blynk app, which can then communicate with your hardware through the Blynk cloud server.
The Blynk platform relies on cloud-based services to facilitate the communication between the mobile app and the hardware device. It uses the Internet of Things (IoT) technology to establish a connection and exchange data between the two.

FEATURES OF BLYNK CLOUD

- Supports Hardware Platforms: Blynk supports a wide range of hardware platforms commonly used in IoT projects, including Arduino, Raspberry Pi,

ESP8266, ESP32, Particle, and many more. Each platform typically requires a compatible library or firmware that enables it to communicate with the Blynk cloud server.

- Blynk Libraries: Blynk provides libraries and firmware for various hardware platforms, making it easier to establish a connection between the hardware and the Blynk cloud server. These libraries offer pre-built functions and methods to handle communication, data transfer, and interaction with the Blynk app.

- Blynk App: The Blynk mobile app serves as the user interface for your IoT project. You can design a custom dashboard using the Blynk app, adding buttons, sliders, graphs, and other UI elements to control and monitor your devices. The app establishes a connection with the Blynk cloud server to send and receive data.

- Blynk Cloud Server: The Blynk cloud server acts as a bridge between the Blynk app and your IoT devices. It handles the communication between the two, allowing you to send commands from the app to the devices and receive data from the devices back to the app. The cloud server acts as a central hub for data transmission and synchronization.

- Communication Protocols: Blynk supports various communication protocols to interact with IoT devices. The most common protocols include Wi-Fi, Bluetooth, Ethernet, and GSM. Depending on your hardware platform and connectivity options, you can choose the appropriate protocol to establish a connection between your devices and the Blynk cloud server.

- Virtual Pins: Blynk introduces the concept of virtual pins, which are used to facilitate communication between the hardware and the app. You can link virtual pins on the Blynk app with specific functions or variables on your IoT device. When the virtual pin is triggered or updated on the app, the corresponding function or variable on the hardware is invoked or modified

IOT devices used

1. Ultra sonic sensor
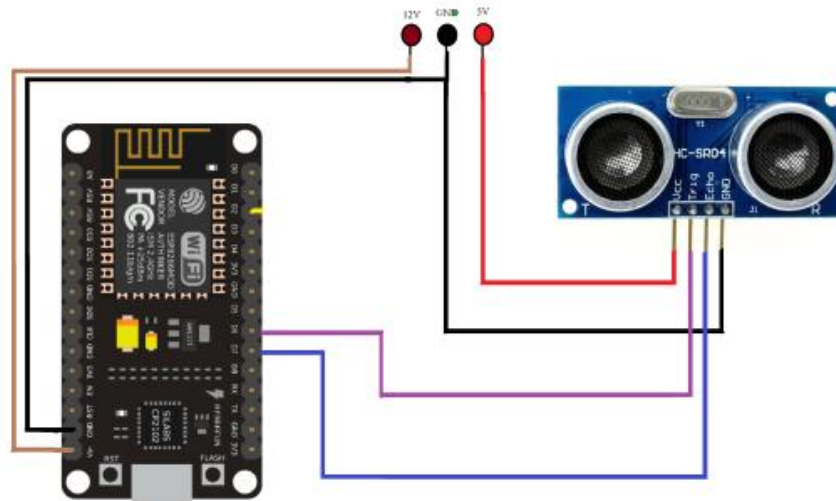2. Humidity and temperature sensor
3. Esp8266
4.

Distance measurement using ultrasonic sensor with cloud blynk

- Set up your hardware:
- Connect the ultrasonic sensor to your microcontroller according to the sensor's specifications.
- Ensure that the microcontroller is properly connected to your computer or network, depending on the board type.
- Install necessary libraries:

- Install the Blynk library for your microcontroller board. You can find the library and installation instructions on the official Blynk website or the library manager of your integrated development environment (IDE).
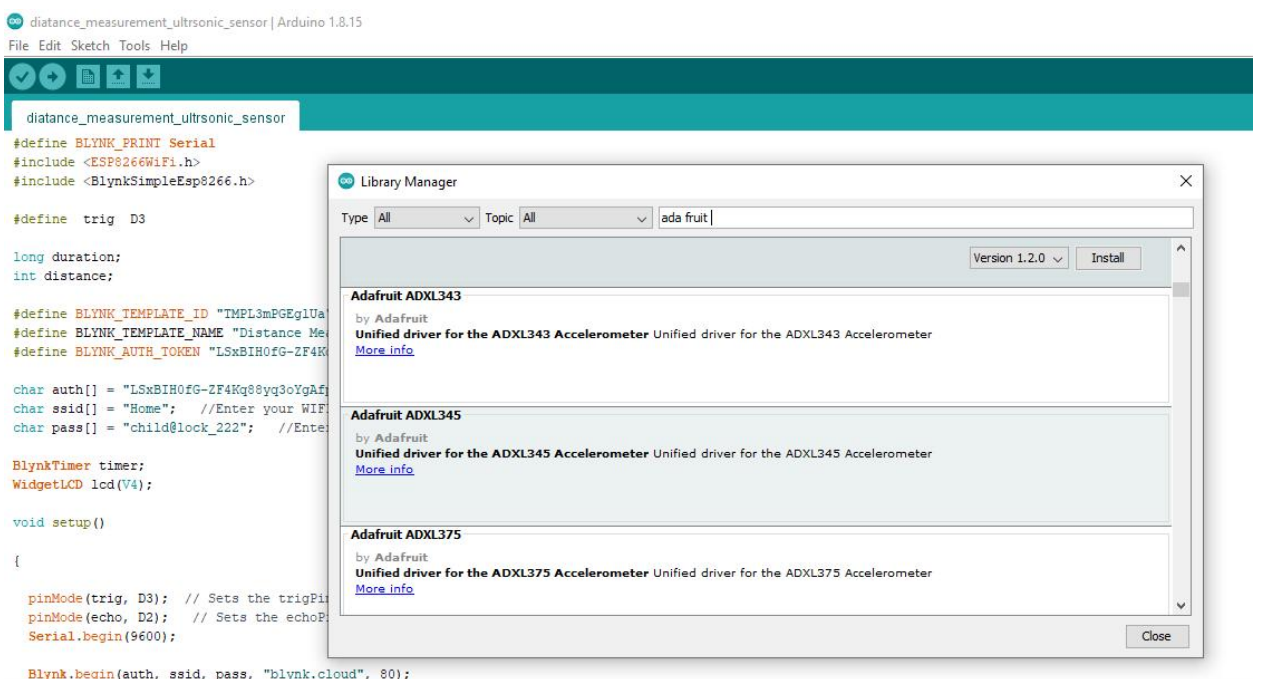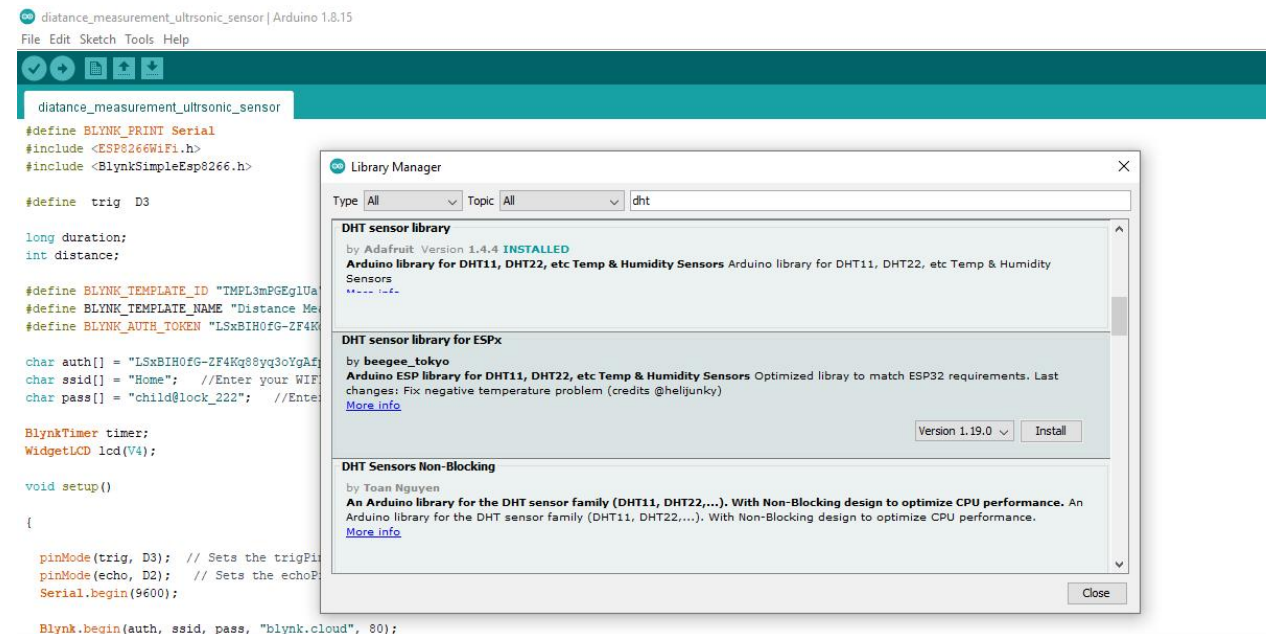

- Create a Blynk project and obtain an auth token:
- Open the Blynk app and create a new project.
- Add a Value Display widget to display the distance measurement.
- Note down the auth token, which is required to establish a connection between your hardware and your app.

Humidity and temperature measuring using esp8266 with cloud blynk

Install necessary libraries:

- DHT Sensor Library: This library is specifically designed for DHT series sensors such as DHT11, DHT21, and DHT22. It provides functions to read temperature and humidity values from the sensor. You can find the library by searching for "DHT sensor library" in the library manager of the Arduino IDE.

- Adafruit Unified Sensor Library: This library is a prerequisite for using various Adafruit sensor libraries, including the DHT sensor library mentioned above. It provides a common interface and sensor abstraction layer for Adafruit sensors. You can find it in the Arduino Library Manager by searching for "Adafruit Unified Sensor."

- Adafruit DHT Library: Adafruit offers its own library for DHT series sensors. It provides similar functionality to the DHT Sensor Library but with some additional features. This library can also be found in the Arduino Library Manager by searching for "Adafruit DHT.
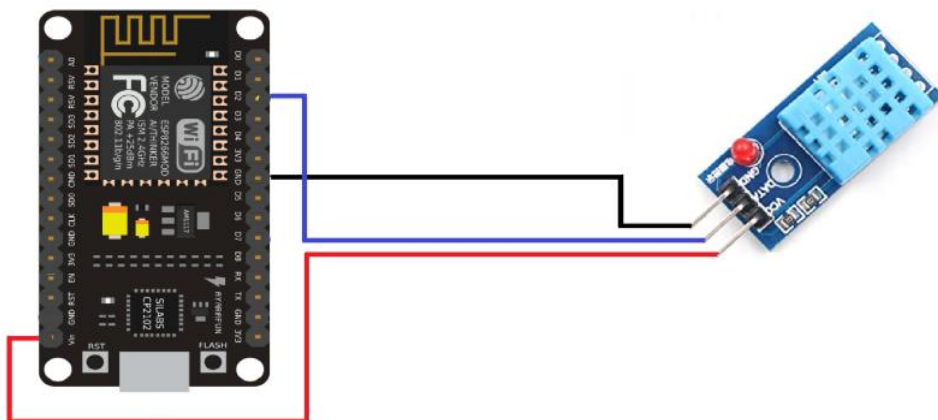
```cpp
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

#define  trig  D3

long duration;
int distance;

#define BLYNK_TEMPLATE_ID "TMPL3mPGEg1Ua";
#define BLYNK_TEMPLATE_NAME "Distance Measurement Ultrasonic sensor";
#define BLYNK_AUTH_TOKEN "LSxBIH0fG-ZF4Kq88yq3oYgAfpz5x13A";

char auth[] = "LSxBIH0fG-ZF4Kq88yq3oYgAfpz5x13A";       // You should get Auth Token in the Blynk App.
char ssid[] = "Home";   //Enter your WIFI name
char pass[] = "child@lock_222";   //Enter your WIFI password

BlynkTimer timer;
WidgetLCD lcd(V4);

void setup()

{

  pinMode(trig, D3);  // Sets the trigPin as an Output
  pinMode(echo, D2);   // Sets the echoPin as an Inpu
  Serial.begin(9600);

  Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
```

```cpp
  Serial.begin(9600);

  Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);

  timer.setInterval(1000L, sendSensor);

}
void loop()

{

  Blynk.run();

  timer.run();

}

void sendSensor()

{

  digitalWrite(trig, LOW);   // Makes trigPin low

  delayMicroseconds(2);        // 2 micro second delay
  digitalWrite(trig, HIGH);  // tigPin high
  delayMicroseconds(10);        // trigPin high for 10 micro seconds
  digitalWrite(trig, LOW);   // trigPin low
```

This code utilizes the ESP8266WiFi and Blynk libraries.

"your_blynk_authentication_token"is replaced with the authentication token generated in the Blynk app for your project. Also, "your_wifi_network_name" and "your_wifi_password"is updated with actual Wi-WiFi network credentials.

The code defines the trigger and echo pins for the ultrasonic sensor. In the setup() function, it initializes the serial communication, connects to the Blynk cloud using the provided authentication token and Wi-Fi credentials, and sets the pin modes.

In the loop() function, it triggers the ultrasonic sensor by sending a pulse, measures the duration of the echo using pulseIn(), calculates the distance in centimeters, and sends the distance value to the serial monitor and the Blynk app using Blynk.virtualWrite().

The delay(1000) introduces a one-second delay between each distance measurement. This value is adjusted as per requirements.

```
humidity_and_temperature
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include<HCSR04.h>
#define trig D3
#define echo D2
HCSR04 hc(trig,echo);

char auth[] = "5IdFwfK_pKS6Nkhv0iIy2na7yEDwC-P2 ";
char ssid[] = " Home";   // your ssid
char pass[] = "child@lock_222"; // your pass


BlynkTimer timer;

void sendSensor()
{
  int c = hc.dist();
  if ( c == 0 ) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  Blynk.virtualWrite(V4, c);  // select virtual pin (v5) in blynk app


}
BlynkTimer timer;

void sendSensor()
{
  int c = hc.dist();
  if ( c == 0 ) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  Blynk.virtualWrite(V4, c);  // select virtual pin (v5) in blynk app


}
void setup()
{
  // Debug console
  Serial.begin(9600);

  Blynk.begin(auth, ssid, pass);

timer.setInterval(1000L, sendSensor);
}

void loop()
{
  Blynk.run();
  timer.run();     }
```

"your_blynk_authentication_token" is replaced with the authentication token generated in the Blynk app for your project. Also, update "your_wifi_network_name" and "your_wifi_password" with your actual Wi-Fi network credentials.

The code defines the DHT sensor pin and type. In the setup() function, it initializes the serial communication, connects to the Blynk cloud using the provided authentication token and Wi-Fi credentials, and initializes the DHT sensor.

In the loop() function, it reads the temperature and humidity values from the DHT sensor using dht.readTemperature() and dht.readHumidity(). If the readings are valid, it sends the values to the serial monitor and the Blynk app using Blynk.virtualWrite().

The delay(2000) introduces a two-second delay between each measurement. Adjust this value as per your requirements.

Blynk Cloud Module

Blynk is a comprehensive software suite that enables the prototyping, deployment, and remote management of connected electronic devices at any scale.
Whether it's personal IoT projects or commercial connected products in the millions, Blynk empowers users to connect their hardware to the cloud and create iOS, Android, and web applications, analyze real-time and historical data from devices, remotely control them from anywhere, receive important notifications, and much more.

Components of the Blynk IoT Platform

Blynk.Console is a feature-rich web application catering to different [types of users](). Its key functionalities include:

- Configuration of connected devices on the platform, including application settings.
- Device, data, user, organization, and location management.
- Remote monitoring and control of devices
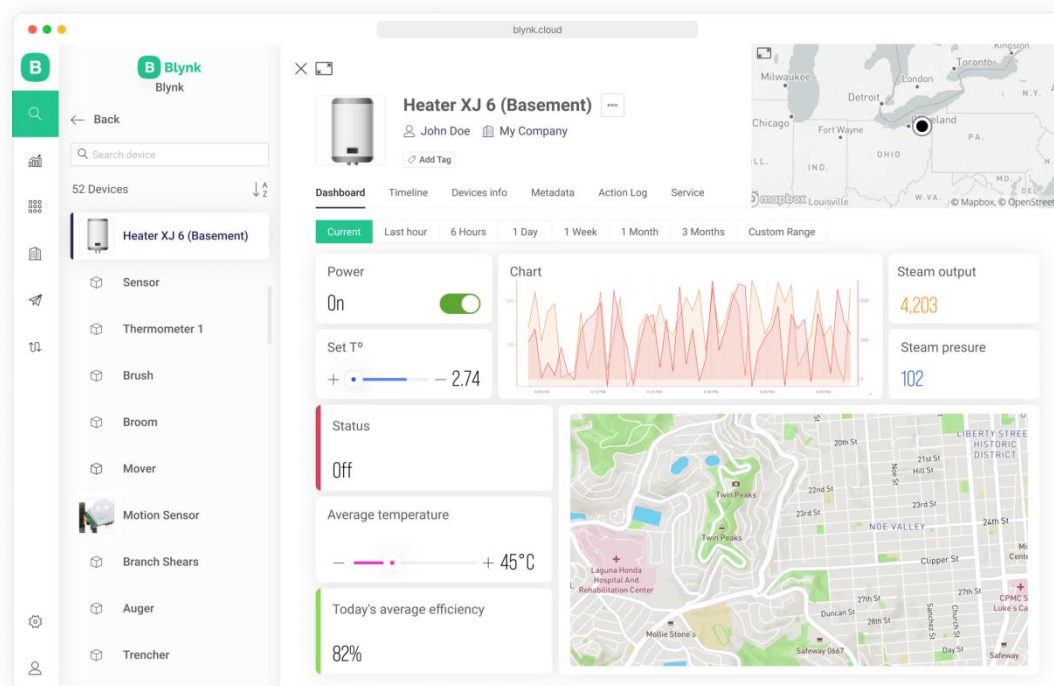
Blynk Library is a user-friendly and portable C++ library, that comes pre-configured to work with hundreds of development boards. It implements a streaming connection protocol, allowing for low-latency and bi-directional communication.

How Data Flows From Device to Blynk

With Blynk you can send raw or processed data from any sensor or actuator connected to the MCU board

When you send data to Blynk it flows through a Datastream using Blynk protocol. Then every value is automatically timstamped and stored in the Blynk.Cloud database (you can also send batches of timestamped data if needed).
Datastream is a channel that tells Blynk what type of data is flowing through it.

With Blynk you can send any raw or processed data from any sensor or actuator.

Virtual Pins in Blynk allow for data exchange between hardware and the Blynk app. Unlike physical pins, they provide hardware-independent functionality, making it easier to transition between different hardware platforms. With Virtual Pins, you can send data from the app, process it on the microcontroller, and send it back to the smartphone. This abstraction enables various interactions such as triggering functions, reading I2C devices, controlling motors, and interfacing with external libraries. Additionally, Virtual Pins offer more control over widget behavior and stability compared to manipulating digital pins. When using Virtual Pins, there is no direct correlation between them and the physical GPIO pins on your hardware; you need to implement the code to link them. The data sent through Virtual Pins can be stored as raw data or averaged based on the chosen plan, and the Chart Widget in Blynk.Console can be used to visualize and store the data.

Send Data With Blynk Library Firmware API

This method utilizes Blynk Protocol and it's the most common and easy-to-use method when you need to send data in real-time.

First you need to do is setup a template with a datastream to configure what type of data your hardware will be sending.

When you have the datastream set, use its Virtual Pin number further.

```
This is an example code on how to send data every second with a timer:
// Declaring a global variabl for sensor data
    int sensorVal;

// This function creates the timer object. It's part of Blynk library
    BlynkTimer timer;

    void myTimer()
    {
    // This function describes what will happen with each timer tick
    // e.g. writing sensor value to datastream V5
    Blynk.virtualWrite(V5, sensorVal);
    }
```

```
void setup()
{
//Connecting to Blynk Cloud
Blynk.begin(auth, ssid, pass);

// Setting interval to send data to Blynk Cloud to 1000ms.
// It means that data will be sent every second
timer.setInterval(1000L, myTimer);
}

void loop()
{
// Reading sensor from hardware analog pin A0
sensorVal = analogRead(A0);

// Runs all Blynk stuff
Blynk.run();

// runs BlynkTimer
timer.run();
}
```

Network Gateway Module

The network gateway module serves as a crucial component of the project, providing transparent network information and facilitating seamless administration through the admin control application. This section highlights the key features and functionalities of the network gateway module, which utilizes JavaScript to render and display essential network details.

DNS Name: The module retrieves and displays the DNS name associated with the network. The DNS name provides a human-readable identifier for the network, making it easier for administrators to recognize and manage multiple networks if applicable.

IP Address: The network gateway module retrieves the IP address assigned to the network. This information is vital for identifying and accessing devices within the network and allows administrators to establish connections and perform network-related tasks.

ISP Provider: By querying the network, the module retrieves and presents the name of the Internet Service Provider (ISP). This information helps administrators identify the company responsible for providing internet connectivity to the network and aids in troubleshooting network issues if required.

Network Latency: The module measures and displays the network latency, which refers to the time it takes for data packets to travel from the source to the destination and back. Network latency is crucial for assessing network performance and identifying potential bottlenecks or connectivity issues.

Network Speed: The module measures and showcases the network speed, indicating the data transfer rate between devices within the network. This information helps administrators monitor network performance and optimize data transmission for efficient communication.

Router Name: The module retrieves and presents the name of the router used in the network. The router name is helpful for identifying and distinguishing between different network configurations or access points, particularly in complex network setups.

By leveraging JavaScript functionality, the network gateway module dynamically updates and renders these network details in real-time within the admin control application. This provides administrators with comprehensive visibility into the network environment and empowers them to monitor, manage, and troubleshoot network-related issues effectively.

Code:

```javascript
function pingHost() {
const host = document.getElementById("ping-host").value;
const xhr = new XMLHttpRequest();
  xhr.open("GET", "https://" + host, true);
```

```javascript
    xhr.timeout = 10000;
    xhr.onload = function ()
    {
    const pingTime = new Date() - startTime;
    const pingResult = document.getElementById("ping-result");
        pingResult.innerHTML = "Ping time: " + pingTime + "ms";
        pingResult.classList.add("slide-in");
    };
    xhr.onerror = function ()
    {
        document.getElementById("ping-result").innerHTML = "Ping failed";
    };
  const startTime = new Date();
    xhr.send();
  }

  function measureNetworkSpeed()
  {
    const startTime = performance.now();
    const fileSize = 1; // 5MB file size for testing
    fetch('https://example.com/file.bin', {
        method: 'GET',
      cache: 'no-cache',
      headers: {
    'Content-Type': 'application/octet-stream',
        'Content-Length': `${fileSize}`
  }
    }).then(response => {
     const endTime = performance.now();
     const duration = (endTime - startTime) / 1000; // convert to seconds
     const speedMbps = (fileSize / duration / 1024 / 1024 * 8).toFixed(2); // calculate Mbps
    document.getElementById("download-speed").innerHTML = speedMbps + " Mbps";
    }).catch(error => {
      console.log(error);
    });
  }


  function checkNetworkStatus() {
  const statusIcon = document.getElementById("network-status-icon");
  const statusText = document.getElementById("network-status-text");
    if (navigator.onLine) {
        statusIcon.innerHTML = "&#x2714";
        statusIcon.style.color = "#1abc9c";
        statusText.innerHTML = "Online";
    } else {
        statusIcon.innerHTML = "&#x2716;";
        statusIcon.style.color = "#e74c3c";
        statusText.innerHTML = "Offline";
    }
  }



  // IP
  var xhr = new XMLHttpRequest();
xhr.open('GET', 'https://api.ipify.org', true);
    xhr.onload = function() {
    if (this.status === 200) {
    document.getElementById('ip').innerHTML = this.responseText;
    }
    };
    xhr.send();
```

```
// ISP
    var xhr2 = new XMLHttpRequest();
    xhr2.open('GET', 'https://api.ipify.org/?format=json', true);
    xhr2.onload = function() {
    if (this.status === 200) {
    var data = JSON.parse(this.responseText);
    var ip = data.ip;
    var xhr3 = new XMLHttpRequest();
    xhr3.open('GET', 'https://ipwhois.app/json/' + ip, true);
    xhr3.onload = function() {
    if (this.status === 200) {
    var data2 = JSON.parse(this.responseText);
    document.getElementById('isp').innerHTML = data2.isp;
    }
    };
    xhr3.send();
    }
    };
    xhr2.send();


// Router
    document.getElementById('router').innerHTML = location.hostname;

// DNS
    var dns = 'example.com';
    var xhr = new XMLHttpRequest();
    xhr.open('GET', 'https://dns.google/resolve?name=' + dns, true);
    xhr.onload = function() {
    if (this.status === 200) {
    var data = JSON.parse(this.responseText);
    var ipAddress = data.Answer[0].data;
    console.log('DNS: ' + ipAddress);
    document.getElementById('dns').innerHTML = ipAddress;
    }
    };
    xhr.send();


//speed
// Speed test
    function measureNetworkSpeed() {
    const startTime = Date.now();
    const fileSize = 1024 * 1024 * 10; // 10 MB
    const url = "https://speed.hetzner.de/10GB.bin"; // replace with your preferred file URL
    const xhr = new XMLHttpRequest();
    xhr.open("GET", url + "?r=" + Math.random(), true);
    xhr.responseType = "arraybuffer";

    xhr.onload = function(e) {
    const endTime = Date.now();
    const duration = (endTime - startTime) / 1000;
    const bitsLoaded = fileSize * 8;
    const speedBps = bitsLoaded / duration;
    const speedMbps = (speedBps / (1024 * 1024)).toFixed(2);
    console.log("Download speed: " + speedMbps + " Mbps");
    document.getElementById("speed").innerHTML = speedMbps;
    };

    xhr.onerror = function(e) {
    console.error("Error fetching file", e);
    };
```

```
    xhr.send();
    }



//latency
    var xhr = new XMLHttpRequest();
    var startTime, latency;

    xhr.onreadystatechange = function() {
    if (this.readyState === 4) {
    var endTime = new Date().getTime();
    latency = endTime - startTime;
    console.log("Latency: " + latency + "ms");
    document.getElementById("latency").innerHTML = latency + "ms";
    }
    };

    startTime = new Date().getTime();
    xhr.open("HEAD", "https://www.google.com", true);
    xhr.send();

// Initialize widgets on page load
    measureNetworkSpeed();
    checkNetworkStatus();
```

The provided code snippet demonstrates several JavaScript functions that perform network-related operations and retrieve network information.

The pingHost() function allows the user to ping a specific host or URL. It utilizes the XMLHttpRequest object to send a GET request to the specified URL. Upon receiving a response, it calculates the ping time by subtracting the start time from the current time. The result is then displayed in the designated HTML element.

In the measureNetworkSpeed() function, the network speed is measured by fetching a file from a given URL. Using the fetch API, a GET request is sent to the URL, and the duration of the request is calculated by subtracting the start time from the end time. Based on the file size and duration, the network speed is calculated in Mbps and displayed in the corresponding HTML element.

The checkNetworkStatus() function determines the network status by checking the value of the navigator.onLine property. If the property evaluates to true, indicating an online connection, a checkmark icon is displayed, and the text is set to "Online." Otherwise, a cross icon is displayed, and the text is set to "Offline."

The remaining sections of the code involve retrieving specific network information using XMLHttpRequest. The IP address is fetched from the "api.ipify.org" API, while the ISP (Internet Service Provider) is obtained by making additional requests to "ipwhois.app." The router name is extracted from the location.hostname property. Lastly, the DNS information is retrieved by sending a GET request to the "dns.google/resolve" API.

Additionally, the code includes functions for measuring network latency and initializing the widgets on page load. The network latency is calculated by sending a

HEAD request to "www.google.com" using XMLHttpRequest, and the duration is calculated based on the start and end times. The network speed and status are measured and displayed using the corresponding functions.
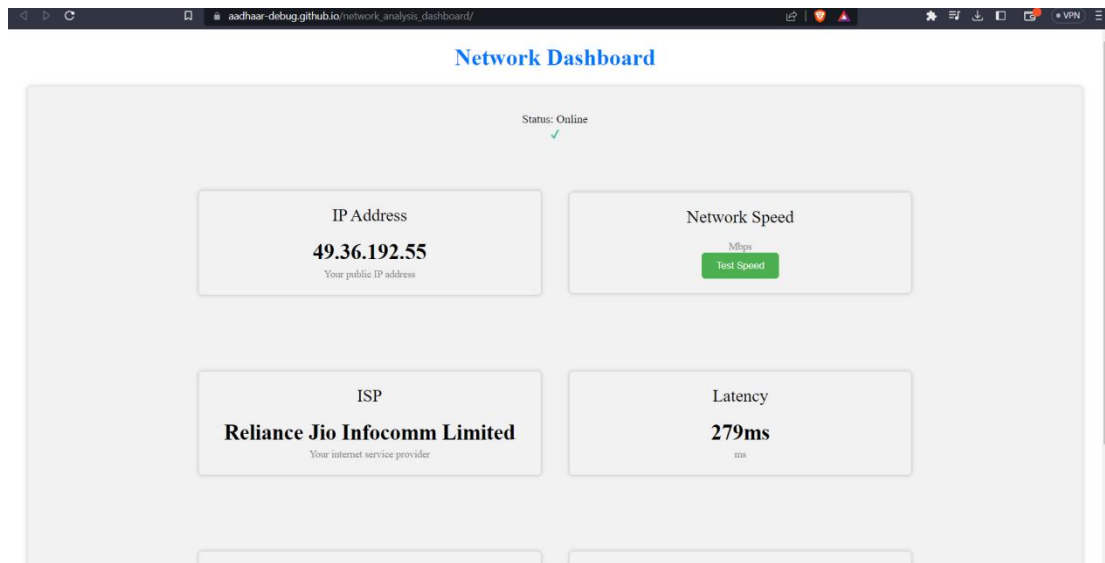
The network gateway section in the project documentation elucidates how the network information is retrieved and how JavaScript is utilized to render and display the data in a user-friendly manner. This transparency and accessibility foster efficient network administration and facilitate informed decision-making for maintaining optimal network performance.

Honeypot Module

Instead of relying on the complex frameworks like django and flask which would have brought a boat to cost not only in the development sector but in the maintainence sector as well. This is why the team decided to us the ESP8266 from the IOT module to build a network interfared honeypot system which would render the network the information onto the admin's device as well. The pictorial representation of which can be seen below in the given figure.

Fig : 7 - ESP8266

The ESP8266 microcontroller is a versatile device that can be utilized in various applications.The ESP8266 module enables microcontrollers to connect to 2.4 GHz Wi-Fi, using IEEE 802.11 bgn. It can be used with ESP-AT firmware to provide Wi-Fi connectivity to external host MCUs, or it can be used as a self-sufficient MCU by running an RTOS-based SDk One of its intriguing uses is setting up a seemingly vulnerable WiFi network with an open service server to detect any unauthorized attempts to connect.

Features of ESP8266

- 802.11 b/g/n protocol
- Wi-Fi Direct (P2P), soft-AP
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLL, regulators, and power management units
- +19.5dBm output power in 802.11b mode
- Integrated temperature sensor
- Supports antenna diversity
- Power down leakage current of < 10uA
- Integrated low power 32-bit CPU could be used as application processor
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)

59

By flashing an Arduino script onto the ESP8266, we can create a WiFi network that acts as a honeypot, luring potential hackers.

Flashing code to ESP8266

In order to properly configure the script, we must ensure that the necessary ESP8266 libraries are added to the Arduino IDE. Additionally, the ESP8266 ports need to be installed in the boards section by using the ESP8266 by Community package.
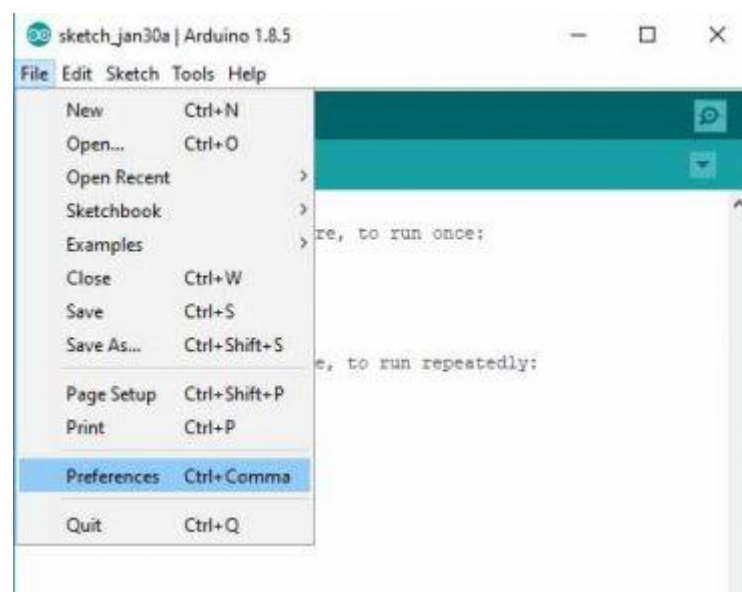
Installing ESP8266 Board in Arduino IDE

ESP8266 community created an add-on for the Arduino IDE. So before starting with ESP8266 first install Arduino IDE.

Installing ESP8266 to Arduino IDE

1) Open Arduino IDE,

Open **preferences** window from Arduino IDE. Go to File -> Preferences.

2) Enter the URL

"http://arduino.esp8266.com/stable/package_esp8266com_index.json" into Additional Board Manager URLs field and click the "OK" button

If you already have a URL in there, and want to keep it, you can separate multiple URLs by placing a comma between them.
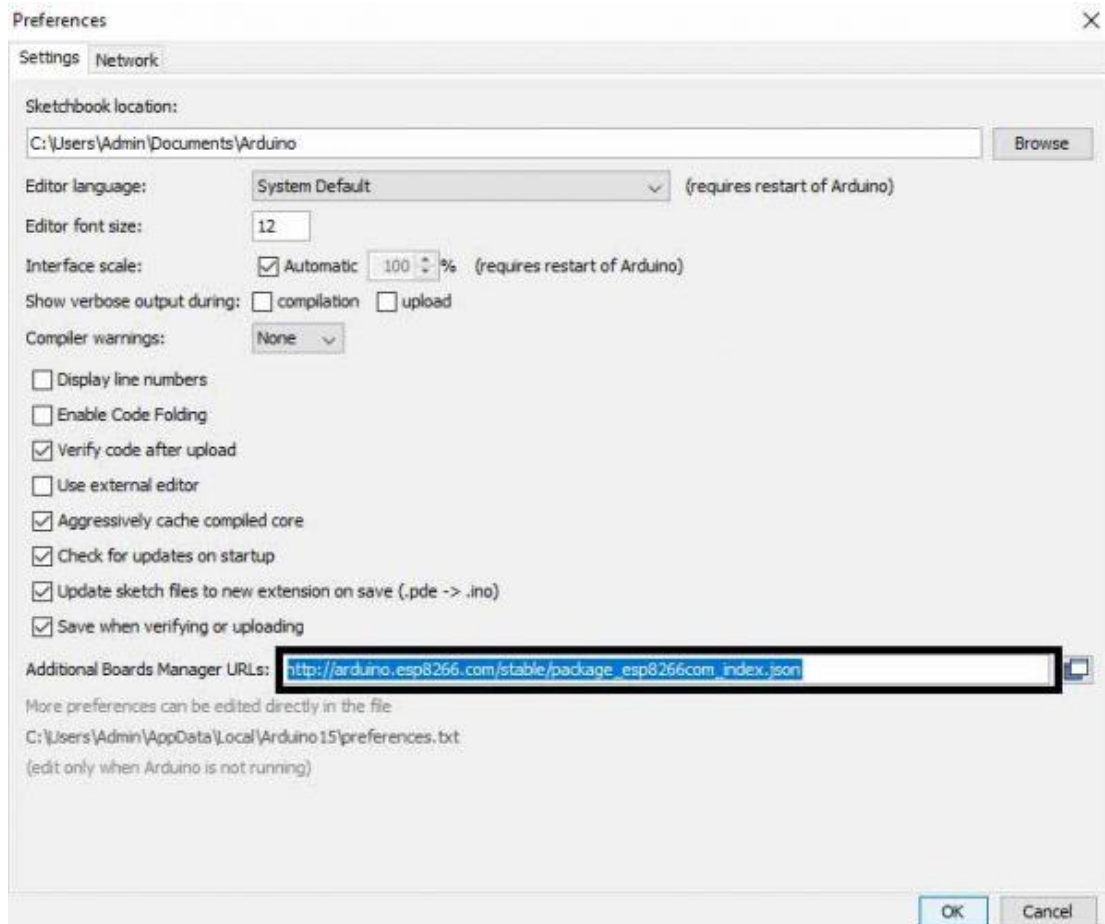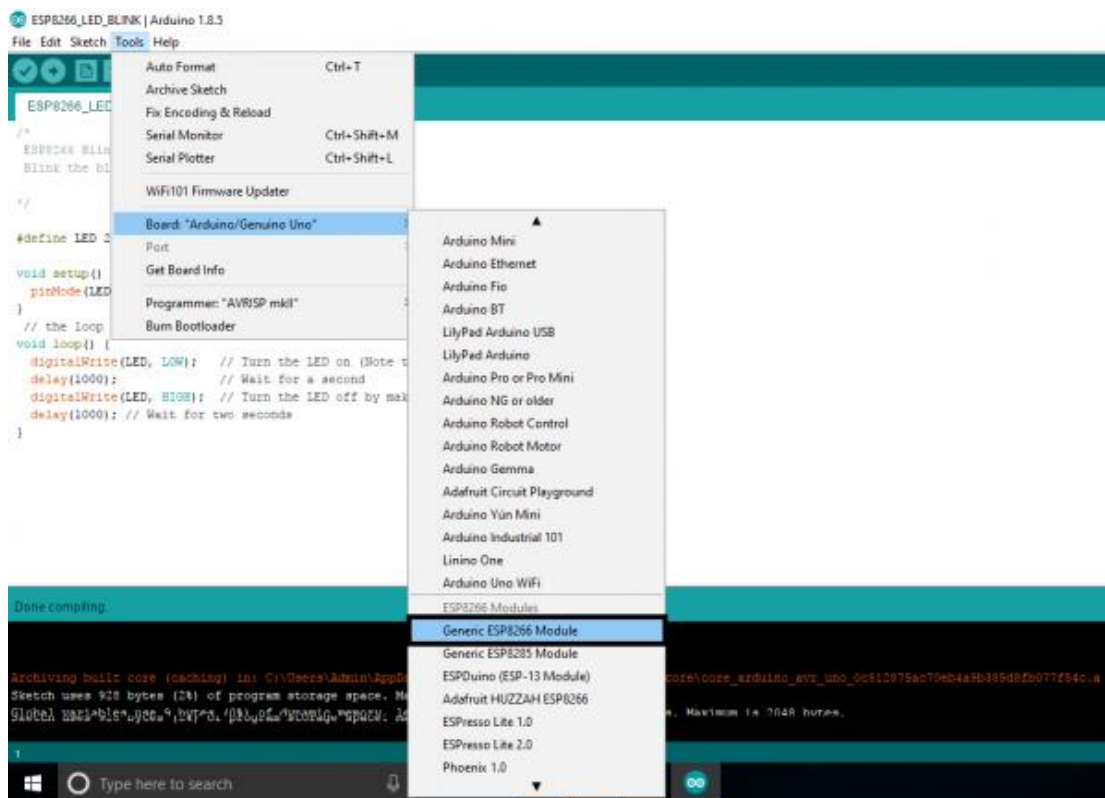
3) Open Boards Manager. Go to Tools -> Board -> Boards Manager
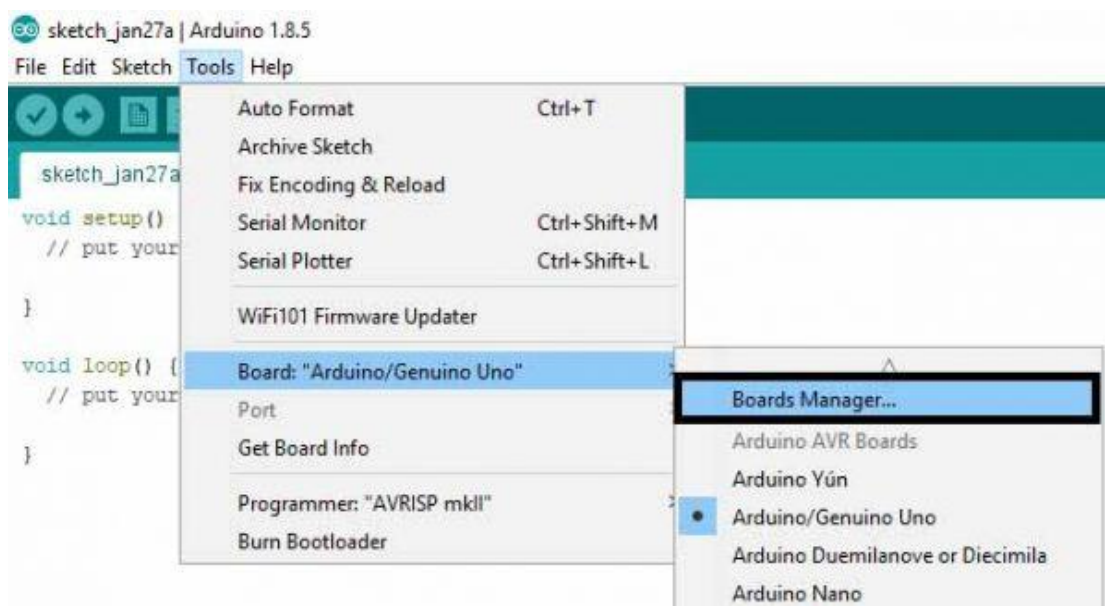
4) Search ESP8266 board menu and install "esp8266 platform"

5) Choose your ESP8266 board from Tools > Board > Generic ESP8266 Module

6) Selected Board details with Port selection
7) Re-open your Arduino IDE

Flashing Steps

The module can enter a number of bootloader modes depending on GPIO pin states. To flash NodeMCU (or any other firmware) you'll need to connect the following pins:

* GPIO 0: LOW
* GPIO 2: HIGH
* GPIO 15: LOW

Apply 3.3V and GND and use a 3.3V UART to connect the device to a computer.

Follow the below steps to put device in serial programming mode before clicking on upload icon

To upload program to ESP on some boards you need to hold flash button while uploading and some boards require reset with holding of flash button.

1. Press and hold RST button.
2. Press and HOLD FLASH button while RST is in pressed condition.
3. Release RST while FLASH is in pressed condition or on some boards hold it during program upload.
4. Release FLASH after releasing RST

To facilitate this setup, we begin by acquiring a canary token, a unique identifier that will be triggered if someone interacts with the WiFi network. The canary token serves as an alert mechanism, providing valuable information such as the IP address of the person attempting to breach the network.
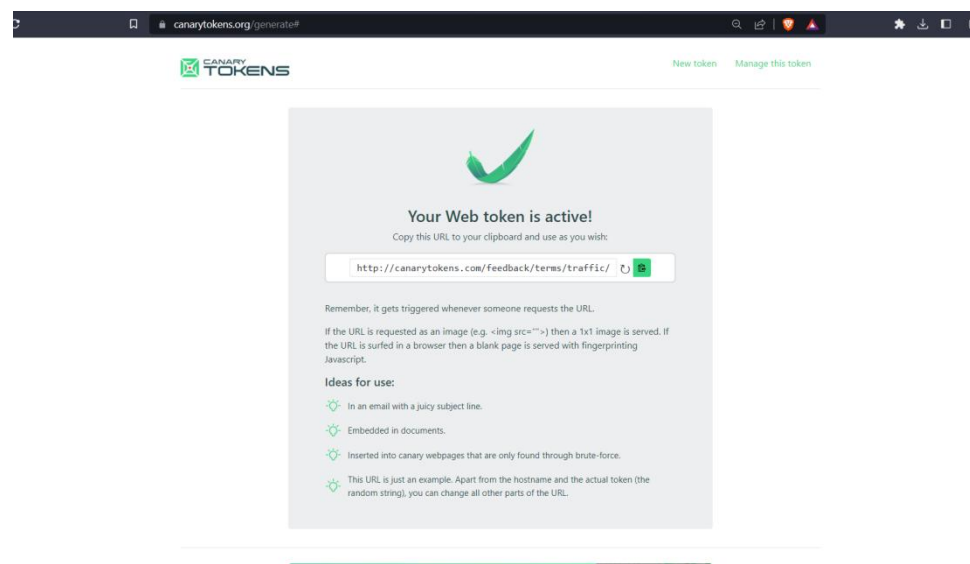
Developing the Honeypot Script

Code :

```
// NAPT example released to public domain

#if LWIP_FEATURES && !LWIP_IPV6

    #define HAVE_NETDUMP 0

    #ifndef STASSID
    #define STASSID "Your_Wifi_Network_Name" // set the SSID (name) of the Wi-Fi network the
ESP8266 will connect to for internet
    #define STAPSK  "Your_Wifi_Network_Password" // set the password of the Wi-Fi network the
ESP8266 will connect to for internet
    #define NEWSSID "honeypot_Wifi_Name" // set the name (SSID) of the Wi-Fi network the ESP8266
will create
    #define NEWPASS "honeypot_Wifi_Password" // set the password of the Wi-Fi network the ESP8266
will create
    #endif

    #include <ESP8266WiFi.h>
    #include <lwip/napt.h>
    #include <lwip/dns.h>
    #include <dhcpserver.h>
    #include <ESPCanary.h>

    #define NAPT 1000
    #define NAPT_PORT 10

    #if HAVE_NETDUMP

    #include <NetDump.h>

    void dump(int netif_idx, const char* data, size_t len, int out, int success) {
    (void)success;
    Serial.print(out ? F("out ") : F(" in "));
    Serial.printf("%d ", netif_idx);

    // optional filter example: if (netDump_is_ARP(data))
    {
    netDump(Serial, data, len);
    //netDumpHex(Serial, data, len);
    }
    }
    #endif

    String  canary  =  "Your_Canarytoken_URL";    //grab  FREE  web  bug/URL  tokens  at
http://canarytokens.org
    String ftp_user = "admin";   //if you replace this with "%" it will accept ANY username
    String ftp_pass = "password"; //if you replace this with "%" it will accept ANY password
    bool append_ip = false;     //if you are using a canary token, leave this as false
    String append_char = "?";    //if you are using a canary token, this doesn't matter
    //if you are using your own webhook,with a bunch of GET
    //parameters then you would want this to be "&" so the IP
    //address becomes the final GET parameter

    FtpServer ftpSrv;  //set #define FTP_DEBUG in ESPCanary.h to see ftp verbose on serial


    void setup() {
    Serial.begin(115200);
```

```cpp
  Serial.printf("\n\nNAPT Range extender\n");
  Serial.printf("Heap on start: %d\n", ESP.getFreeHeap());

  #if HAVE_NETDUMP
  phy_capture = dump;
  #endif

  // first, connect to STA so we can get a proper local DNS server
  WiFi.mode(WIFI_STA);
  WiFi.begin(STASSID, STAPSK);
  while (WiFi.status() != WL_CONNECTED) {
  Serial.print('.');
  delay(500);
  }
  Serial.printf("\nSTA: %s (dns: %s / %s)\n",
  WiFi.localIP().toString().c_str(),
  WiFi.dnsIP(0).toString().c_str(),
  WiFi.dnsIP(1).toString().c_str());

  // give DNS servers to AP side
  dhcps_set_dns(0, WiFi.dnsIP(0));
  dhcps_set_dns(1, WiFi.dnsIP(1));

  WiFi.softAPConfig(  // enable AP, with android-compatible google domain
  IPAddress(172, 217, 28, 254),
  IPAddress(172, 217, 28, 254),
  IPAddress(255, 255, 255, 0));
  WiFi.softAP(NEWSSID, NEWPASS);
  Serial.printf("AP: %s\n", WiFi.softAPIP().toString().c_str());

  Serial.printf("Heap before: %d\n", ESP.getFreeHeap());
  err_t ret = ip_napt_init(NAPT, NAPT_PORT);
  Serial.printf("ip_napt_init(%d,%d):   ret=%d   (OK=%d)\n",   NAPT,   NAPT_PORT,   (int)ret,
(int)ERR_OK);
  if (ret == ERR_OK) {
  ret = ip_napt_enable_no(SOFTAP_IF, 1);
  Serial.printf("ip_napt_enable_no(SOFTAP_IF): ret=%d (OK=%d)\n", (int)ret, (int)ERR_OK);
  if (ret == ERR_OK) {
  Serial.printf("WiFi  Network  '%s'  with  password  '%s'  is  now  NATed  behind  '%s'\n", NEWSSID,
NEWPASS, STASSID);
  }
  }
  Serial.printf("Heap after napt init: %d\n", ESP.getFreeHeap());
  if (ret != ERR_OK) {
  Serial.printf("NAPT initialization failed\n");
  }

  /////FTP Setup, ensure SPIFFS is started before ftp; /////////
  if (SPIFFS.begin()) {
  Serial.println("SPIFFS opened!");
  ftpSrv.begin(ftp_user,ftp_pass,canary,append_ip,append_char);      //username, password for ftp. set
ports in ESPCanary.h  (default 21, 50009 for PASV)
  }
  }

  #else

  #error "NAPT not supported in this configuration"

  void setup() {
  Serial.begin(115200);
  Serial.printf("\n\nNAPT not supported in this configuration\n");
  }
```

```
#endif

void loop() {
ftpSrv.handleFTP();
}
```

The provided code snippet demonstrates the implementation of an ESP8266-based NAPT (Network Address and Port Translation) range extender, showcasing its functionality as a WiFi honeypot. Initially, the required network configurations are defined, such as the SSID and password for the existing WiFi network (STASSID and STAPSK, respectively) and the desired name and password for the honeypot network (NEWSSID and NEWPASS). Additionally, a canary token URL is specified (canary) to capture information about potential intrusion attempts.

The code leverages various libraries, including the ESP8266WiFi library for WiFi-related functionalities and the lwIP (lightweight IP) library for NAPT and DNS operations. The setup() function is responsible for initializing the system. It first connects to the existing WiFi network in station (STA) mode and obtains the local DNS server's IP address. This IP address is then assigned to the access point (AP) side to provide DNS services. The ESP8266 is set up as an AP with the specified honeypot network credentials.

NAPT functionality is enabled by calling ip_napt_init() with the desired number of NAPT entries (NAPT) and the port range (NAPT_PORT). If the initialization is successful, the code enables NAPT for the softAP interface. The serial console outputs relevant information about the network configurations and the success of NAPT initialization.

Additionally, the code includes FTP server functionality using the FtpServer library. The FTP server is set up with the specified username (ftp_user), password (ftp_pass), and canary token settings. If SPIFFS (SPI Flash File System) initialization is successful, the FTP server is started, allowing FTP interactions and handling them using the handleFTP() function in the loop().

It is important to note that the code's functionality relies on specific libraries, configurations, and network setups. Proper configuration of these parameters and addressing any potential errors or compatibility issues are essential for the successful deployment and operation of this ESP8266-based system.

Once everything is set up, the network honeypot, named "honeypot," will be connected to the Testnet, offering a connection on this new network. When potential hackers connect to the WiFi, they will notice a slower version of the network. If they attempt to scan the network or explore its ports, they will discover a vulnerable FTP server listed as an open service on port 21.

As soon as they try to connect to the FTP server or perform an nmap scan, the canary token is triggered, capturing important information about the person. This information includes their IP address, DNS, ISP, location, and any Tor jumps they may have made.
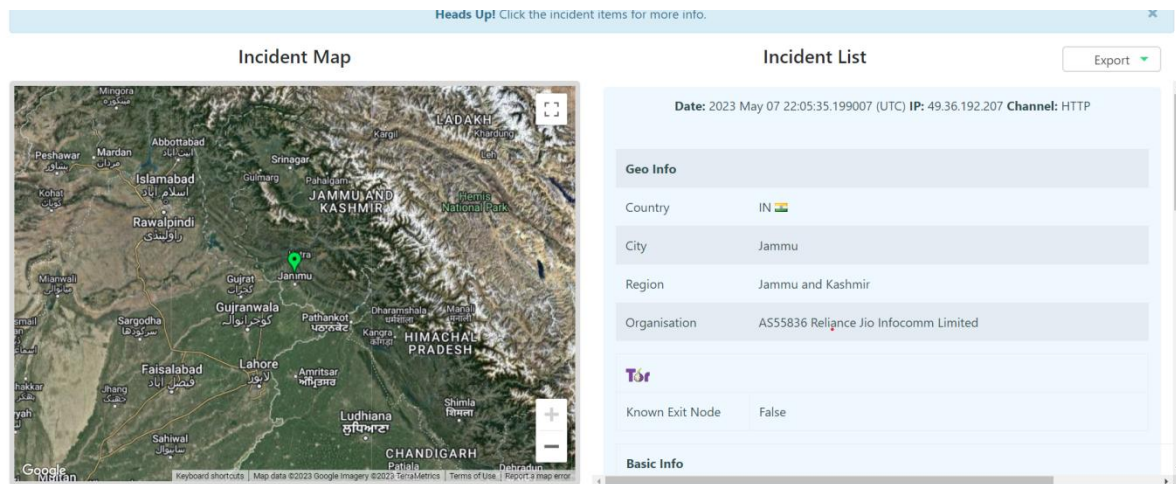
By leveraging the ESP8266 microcontroller and the canary token system, our IDS provides an effective means of identifying and tracking unauthorized access attempts. It allows administrators to gather valuable insights about potential threats and take appropriate measures to secure their network and systems.

Admin Control Application

The requirement of an application was a must in order to bring out the best out of the retrieved abundance of data. For the project to work a complex platform application was not a requirement as the whole motoo of developing the application was just to render the data reterieved from the IDS sensors ,the gateways and the cloud platforms.

Developing an application with such a motive could easily be achieved by traditional application development methods and programming languages. In this case the developers decided to go with Java and XML.

Java presenting as the backede and the logic for the application and XMl presentin as the interface.

Many reasons for choosing to develop the application in java were :

* **Platform Independence**: Java is a platform-independent language, which means the application can run on various operating systems, including Windows, macOS, and Linux. This portability allows you to reach a broader audience and ensures compatibility across different devices.

* **Extensive Library Support**: Java has a vast collection of libraries and frameworks that can expedite the development process. These libraries provide ready-made solutions for common functionalities, reducing the need for reinventing the wheel and saving development time.

* **Rich Development Ecosystem**: Java has a mature and robust development ecosystem with comprehensive documentation, a large developer community, and abundant online resources. This ecosystem facilitates faster development, troubleshooting, and access to expert advice when encountering challenges.

* **XML for Layout Design**: XML is widely used in Android development to define the application's layout structure. XML offers a declarative approach, separating the presentation layer from the application logic. It provides flexibility, ease of understanding, and better maintenance of UI design.

* **Android Framework Integration**: Java is the official language for Android development, supported by the Android Software Development Kit (SDK) and Android Studio. Building the application in Java allows seamless integration with the Android framework and access to its extensive features and functionalities.

* **Code Reusability**: Java's object-oriented programming (OOP) paradigm promotes code reusability, making it easier to maintain and update the application. Additionally, Java's inheritance and modular structure support code organization, enabling efficient collaboration among team members.

* **Community Support and Documentation**: Java has a vast and active developer community that continuously contributes to the improvement of the language and its *

* **associated tools**. This community support ensures access to updated documentation, tutorials, forums, and open-source libraries that can simplify development and problem-solving.

* **Stability and Performance**: Java is known for its stability and performance. The language's mature and optimized runtime environment helps deliver reliable and efficient applications, ensuring smooth performance and responsiveness.

* **Enterprise-Grade Development**: Java has been extensively used for enterprise-level application development due to its scalability, security features, and support for multi-threading. If your project aims for scalability or integration with enterprise systems, Java provides a solid foundation.

* **Familiarity and Developer Availability**: Java is one of the most widely taught languages in universities and has been prevalent in the software development industry for decades. As a result, finding skilled Java developers and accessing knowledge resources is relatively easier compared to newer or less popular languages like Swift or Kotlin.

The development part of the application began with choosing the best IDE, for thisb case the developers decided to go with the Android Studio. Specifically talking

Android Studio V._{2022.2.1(Flamingo) / 13 April 2023 Stable.}

Developing an app on the Android Studio IDE offers several benefits that contribute to a streamlined and efficient development process. First and foremost, Android Studio is the official Integrated Development Environment (IDE) for Android app development, making it highly compatible and optimized for building Android applications. The IDE provides a comprehensive set of tools, including code editing, debugging, and testing features, all tailored specifically for Android development. This enables developers to write clean and efficient code, easily navigate through project files, and quickly identify and fix any issues. Android Studio also integrates

seamlessly with the Android Software Development Kit (SDK) and offers extensive documentation and resources, making it easier for developers to access the latest APIs, libraries, and best practices. The built-in Gradle build system simplifies the management of project dependencies and allows for easy customization and configuration. Additionally, Android Studio provides powerful UI design tools, such as the Layout Editor, which enables developers to create visually appealing user interfaces using drag-and-drop functionality. The real-time layout preview helps ensure that the UI elements are correctly positioned and responsive on different screen sizes and orientations. Furthermore, Android Studio offers robust emulation and testing capabilities with its built-in Android Emulator, allowing developers to test their apps on various virtual devices with different Android versions and configurations. The integration with Firebase and Google Cloud services also facilitates easy integration of backend functionality and services into the app. Overall, Android Studio provides a feature-rich, efficient, and developer-friendly environment that empowers developers to create high-quality Android applications with ease.
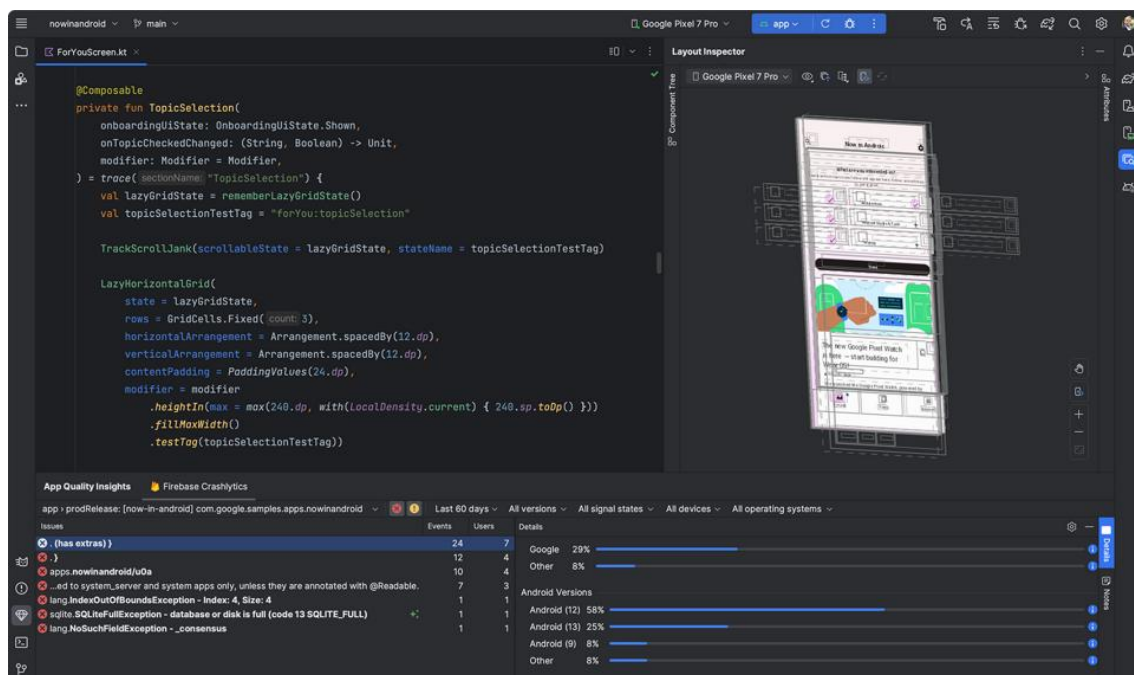
Application development became even easier on IDE as much as it could have been but the application structure and user experience still needed to be developed at the developers end which required a stable yet apealing activity / directory flow.

As per the requirement a well planned activity flow was created , the traces of which and the pictorial diagram can be viewed as below.
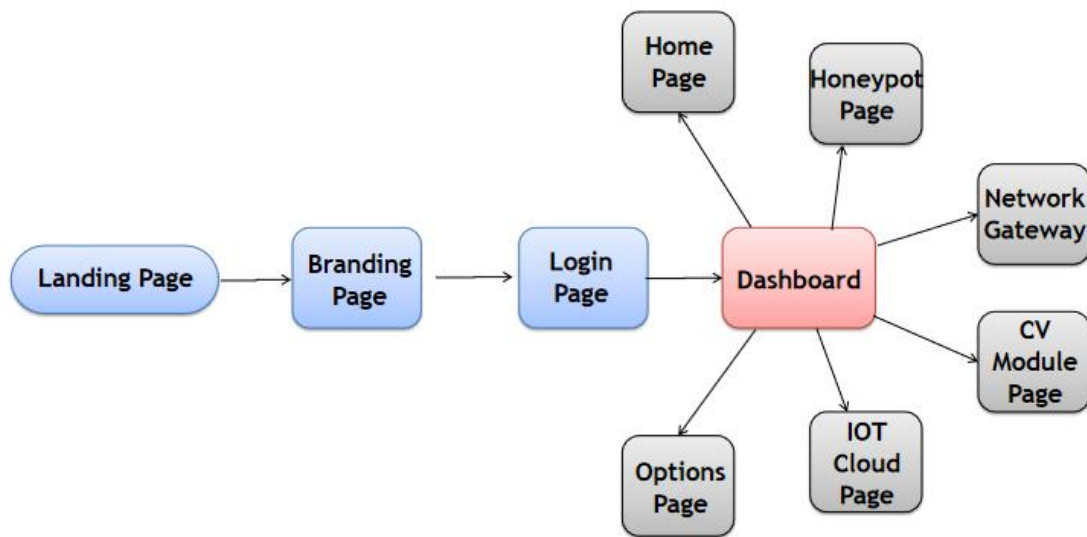
The user interface for the application can be found below :
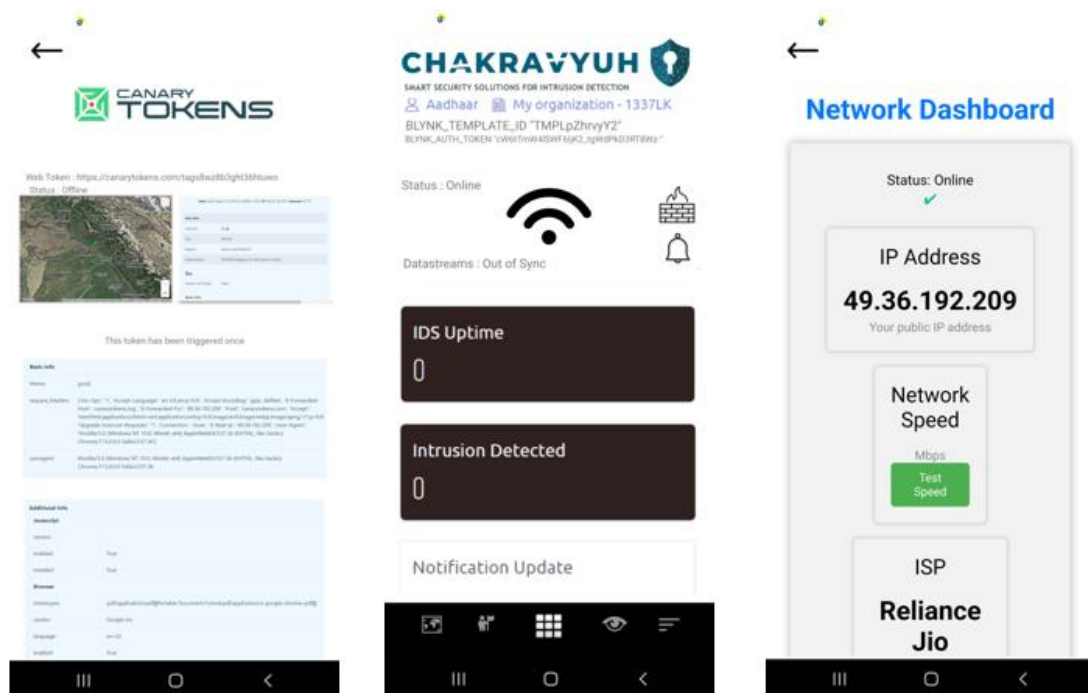
Implementation

Libraries Used :

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Toast;
import java.util.Timer;
import java.util.TimerTask;
import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.net.ConnectivityManager;
import android.net.Network;
import android.net.NetworkCapabilities;
import android.net.NetworkInfo;
import android.os.Build;
import android.os.Bundle;
import android.support.v4.app.NotificationCompat;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import android.annotation.SuppressLint;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.webkit.WebSettings;
import android.webkit.WebView;
```

Use:

The code snippet provided includes several import statements that import different classes and libraries in Android development. Here is an explanation of each library mentioned:

android.content.Intent: This library allows you to work with intents, which are used for inter-component communication within an Android application. Intents can be used to start activities, services, or broadcast messages between different components.

android.support.v7.app.AppCompatActivity: This library provides support for backward compatibility with older versions of Android. It allows you to extend the AppCompatActivity class, which is a base class for activities in Android. It provides additional features and compatibility support for various UI elements and themes.

android.os.Bundle: This library contains classes to work with bundles, which are used for passing data between activities or fragments. Bundles can store key-value pairs and are commonly used for passing data during activity transitions or fragment transactions.

android.widget.Toast: This library provides the Toast class, which allows you to display short-lived messages or notifications to the user. Toast messages are typically used to provide simple feedback or notifications in response to user actions.

java.util.Timer and java.util.TimerTask: These libraries provide classes for scheduling and executing tasks at specified intervals. They are used for creating timers and scheduling recurring actions or background tasks.

android.annotation.SuppressLint: This library is used to apply annotations to code elements, providing additional information or instructions to the compiler or tools. It helps in improving code quality and ensuring proper usage of certain features or APIs.

android.app.Notification, android.app.NotificationChannel, android.app.NotificationManager, android.support.v4.app.NotificationCompat: These libraries are used for creating and managing notifications in Android. They provide classes and methods to construct and display notifications to the user, set notification channels for Android 8.0 and above, and customize the appearance and behavior of notifications.

android.content.Context: This library provides classes and interfaces for accessing global information about an application's environment. It allows you to retrieve system services, access resources, and perform other operations related to the application context.

android.net.ConnectivityManager, android.net.Network, android.net.NetworkCapabilities, android.net.NetworkInfo: These libraries are used for network connectivity-related operations. They provide classes and methods to check network availability, monitor network changes, and retrieve information about network connections.

android.graphics.Color: This library provides utility methods and constants for working with colors in Android. It allows you to create and manipulate colors, define color values, and perform color-related operations.

android.widget.ImageView, android.widget.TextView, android.widget.Button, android.widget.EditText: These libraries provide UI elements for creating image views, text views, buttons, and editable text fields (edit texts) in Android applications. These classes allow you to display and interact with different types of user interface components.

import android.annotation.SuppressLint;
This import statement is used to include the SuppressLint class from the android package. It provides annotations that can be used to suppress lint warnings in Android code.

import android.support.v7.app.AppCompatActivity;
This import statement is used to include the AppCompatActivity class from the android.support.v7.app package. It is a base class for activities that use the support library action bar features. AppCompatActivity provides compatibility for older versions of Android.

import android.os.Bundle;
This import statement is used to include the Bundle class from the android.os package. The Bundle class is used to pass data between Android components (such as activities or fragments) using key-value pairs.

import android.webkit.WebSettings;
This import statement is used to include the WebSettings class from the android.webkit package. The WebSettings class provides various settings and configuration options for a WebView component, such as enabling JavaScript, setting cache modes, and controlling the layout.

import android.webkit.WebView;
This import statement is used to include the WebView class from the android.webkit package. The WebView class is a view that displays web pages or web content within an Android application. It allows developers to embed a browser-like functionality into their apps.

Each library mentioned above serves a specific purpose in Android development, providing ready-made classes and methods to simplify various tasks related to UI, intents, notifications, networking, and more.

Module Code snippets:

Login Page:
Code :

```java
package com.example.chakravyuh;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;


import com.example.chakravyuh.R;

public class login extends AppCompatActivity {

    private EditText usernameEditText;
    private EditText passwordEditText;
    private Button loginButton;

    @SuppressLint("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        // Find views by their corresponding IDs
        usernameEditText = findViewById(R.id.usernameEditText);
        passwordEditText = findViewById(R.id.passwordEditText);
        loginButton = findViewById(R.id.loginButton);

        // Set click listener for the login button
        loginButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Get the entered username and password
                String username = usernameEditText.getText().toString();
                String password = passwordEditText.getText().toString();


                // Validate the username and password
                if (username.equals("admin") && password.equals("password")) {
                    // Authentication successful
                    Toast.makeText(login.this, "Login successful", Toast.LENGTH_SHORT).show();
                    // Proceed to the next activity or perform other actions

                    Intent intent1 = new Intent(login.this, dashboard.class);
                    startActivity(intent1);
                    finish();


                } else {
                    // Authentication failed
```

```
            Toast.makeText(login.this, "Invalid credentials", Toast.LENGTH_SHORT).show();
        }
    }
    });
  }
}
```

The provided code represents an Android activity class named login in the package com.example.chakravyuh. This class extends the AppCompatActivity class and is responsible for handling the login functionality of the application.

Inside the onCreate method, the layout for the activity is set using setContentView to associate it with the XML layout file activity_login.xml. The UI elements such as the usernameEditText, passwordEditText, and loginButton are retrieved using the findViewById method and assigned to corresponding member variables.

A click listener is set on the loginButton, and when it is clicked, the code inside the onClick method is executed. Within this method, the entered username and password values are retrieved from the respective EditText fields.

The entered username and password are then validated, and if they match the expected values ("admin" and "password" in this case), a success message is displayed using a toast notification. Additionally, an intent is created to start the dashboard activity, and the current activity is finished.

On the other hand, if the entered credentials do not match the expected values, an error message is displayed using a toast notification.

Overall, this code implements the login functionality of the application, allowing users to enter their credentials, perform validation, and navigate to the dashboard activity upon successful authentication.

Dashboard Page:
Code :

```
package com.example.chakravyuh;

import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.net.ConnectivityManager;
import android.net.Network;
import android.net.NetworkCapabilities;
import android.net.NetworkInfo;
import android.os.Build;
import android.os.Bundle;
import android.support.v4.app.NotificationCompat;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
```

```java
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;


public class dashboard extends AppCompatActivity {

    private static final String CHANNEL_ID = "my_channel";
    private static final int NOTIFICATION_ID = 1;
    private ImageView image;
    private ImageView image1;
    private ImageView image2;
    private ImageView image3;
    private ImageView image15;
    private ImageView image12;
    private TextView TextView1;
    private ImageView noInternetImage;

    public dashboard() {
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_dashboard);

        // Find the ImageView widgets in the layout
        image = findViewById(R.id.imageView8);
        image1 = findViewById(R.id.imageView10);
        image2 = findViewById(R.id.imageView11);
        image3 = findViewById(R.id.imageView14);
        noInternetImage = findViewById(R.id.imageView37);
        TextView1 = findViewById(R.id.textView2);
        image15 = findViewById(R.id.imageView15);
        image12 = findViewById(R.id.imageView12);

        // Hide the no internet image by default
        noInternetImage.setVisibility(View.GONE);

        // Set an OnClickListener for the ImageView
        image.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Create an Intent to start the new activity
                Intent intent = new Intent(dashboard.this, aiLivefeed.class);
                startActivity(intent);
                finish();
            }
        });


        // Set an OnClickListener for the ImageView
        image15.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Create an Intent to start the new activity
                Intent intent = new Intent(dashboard.this, networkGateway.class);
                startActivity(intent);
                // Show notification
                showNotification();
            }
        });
```

```java
        // Set an OnClickListener for the ImageView
        image12.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Create an Intent to start the new activity
                Intent intent = new Intent(dashboard.this, firewall.class);
                startActivity(intent);
                // Show notification
                showNotification();
            }
        });


        // Set an OnClickListener for the ImageView
        image1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Create an Intent to start the new activity
                Intent intent = new Intent(dashboard.this, dashboard.class);
                startActivity(intent);
                finish();
            }
        });

        // Set an OnClickListener for the ImageView
        image2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Create an Intent to start the new activity
                Intent intent = new Intent(dashboard.this, honeypot.class);
                startActivity(intent);
                finish();
            }
        });

        // Set an OnClickListener for the ImageView
        image3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Create an Intent to start the new activity
                Intent intent = new Intent(dashboard.this, notificationActivity.class);
                startActivity(intent);
                // Show notification
                showNotification();
            }
        });

        // Register network callback to monitor internet connectivity status
        ConnectivityManager connectivityManager =
                (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
        if (connectivityManager != null) {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
                connectivityManager.registerDefaultNetworkCallback(new
ConnectivityManager.NetworkCallback() {
                    @Override
                    public void onAvailable(Network network) {
                        // Internet connectivity is available, hide the no internet image
                        runOnUiThread(new Runnable() {
                            @Override
                            public void run() {
                                noInternetImage.setVisibility(View.GONE);
                                Toast.makeText(dashboard.this, "System Online", Toast.LENGTH_SHORT).show();
                                TextView1.setText("Status : Online");
```

```java
                    }
                });
            }

            @Override
            public void onLost(Network network) {
                // Internet connectivity is lost, show the no internet image
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        noInternetImage.setVisibility(View.VISIBLE);
                        Toast.makeText(dashboard.this, "System Offline", Toast.LENGTH_SHORT).show();
                        TextView1.setText("Status : Offline");
                    }
                });
            }
        });
    }
}

private void showNotification() {
    NotificationManager notificationManager =
            (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

    // Check if Android version is Oreo or higher
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        // Create NotificationChannel
        NotificationChannel channel = new NotificationChannel(
            CHANNEL_ID,
            "My Channel",
            NotificationManager.IMPORTANCE_HIGH);
        channel.setDescription("My Channel Description");
        channel.enableLights(true);
        channel.setLightColor(Color.RED);
        channel.enableVibration(true);
        notificationManager.createNotificationChannel(channel);
    }

    // Check if device is connected to internet
    if (isConnectedToInternet()) {
        // Remove the image from the ImageView
        image3.setImageDrawable(null);
    } else {
        // Add the image to the ImageView
        image3.setImageResource(R.drawable.imageView37);
    }

    NotificationCompat.Builder builder =
            new NotificationCompat.Builder(this, CHANNEL_ID)
                .setSmallIcon(R.drawable.beatapplogo)
                .setContentTitle("Alert !")
                .setContentText("Intrusion Detected on the Honeypot system");

    notificationManager.notify(NOTIFICATION_ID, builder.build());
}


private boolean isConnectedToInternet() {
    ConnectivityManager            connectivityManager            =            (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo activeNetworkInfo = connectivityManager.getActiveNetworkInfo();
    return activeNetworkInfo != null && activeNetworkInfo.isConnected();
```

```
    }
}
```

The provided code represents an Android activity class named dashboard in the package com.example.chakravyuh. This class extends the AppCompatActivity class and is responsible for managing the dashboard functionality of the application.

Within the onCreate method, the layout for the activity is set using setContentView to associate it with the XML layout file activity_dashboard.xml. Several ImageView and TextView widgets are retrieved using findViewById and assigned to corresponding member variables.

Several click listeners are set on the ImageViews, and when clicked, they trigger specific actions. For example, clicking on the image ImageView starts a new activity named aiLivefeed, while clicking on image15 starts the networkGateway activity and shows a notification. Similarly, other ImageViews start different activities upon being clicked.

The dashboard class also registers a network callback to monitor the internet connectivity status. This is done using the ConnectivityManager class, which checks if the device is connected to the internet. Based on the connectivity status, the visibility of the noInternetImage ImageView is toggled, and a corresponding toast message is displayed. The TextView1 is also updated with the status information.

The showNotification method creates and displays a notification using the NotificationManager and NotificationCompat.Builder classes. The method checks the Android version and creates a notification channel if the version is Oreo or higher. Depending on the internet connectivity status, the image in image3 ImageView is either removed or set to display a specific image. The notification is then built and displayed.

Additionally, the isConnectedToInternet method checks if the device is connected to the internet using the ConnectivityManager and returns a boolean value indicating the connectivity status.

Overall, this code implements the functionality of the dashboard activity, allowing users to interact with various ImageViews, monitor internet connectivity, and receive notifications based on certain conditions.

The similar code snippets have been used to achieve the other aspects of the application as well.All the integrations and the cloud data rendering allowed us to present the admin all the relevant data in realtime.

Proposed Framework

A Framework in which the modules computer vision , surveillance , Blynk cloud , IOT , network gateway , Honeypot , android application work in a coordinated manner . the framework would comprises of a Network and home automation control panel that is receiving real-time responses from different components like wifi , detection sensors , arduino , home gateway , media server . The wifi render the information from a centralized home security system that comprises of a VPN , Network Sniffer , IP surveillance and a honeypot system. The detection sensors receive information from the ultrasonic sensors , laser sensors , motion sensors , PIR sensors in real-time. The arduino renders the real-time information from electricity appliances. the home gateway renders the information from the home gateway dashboard and a media server the is connected to the network storage. ..... whenever at any instant a sensor or an activity happens on any of the modules of the framework a Smart notification system triggers a push notification in the admin control application telling them that there has been an intrusion on a particular sensor.
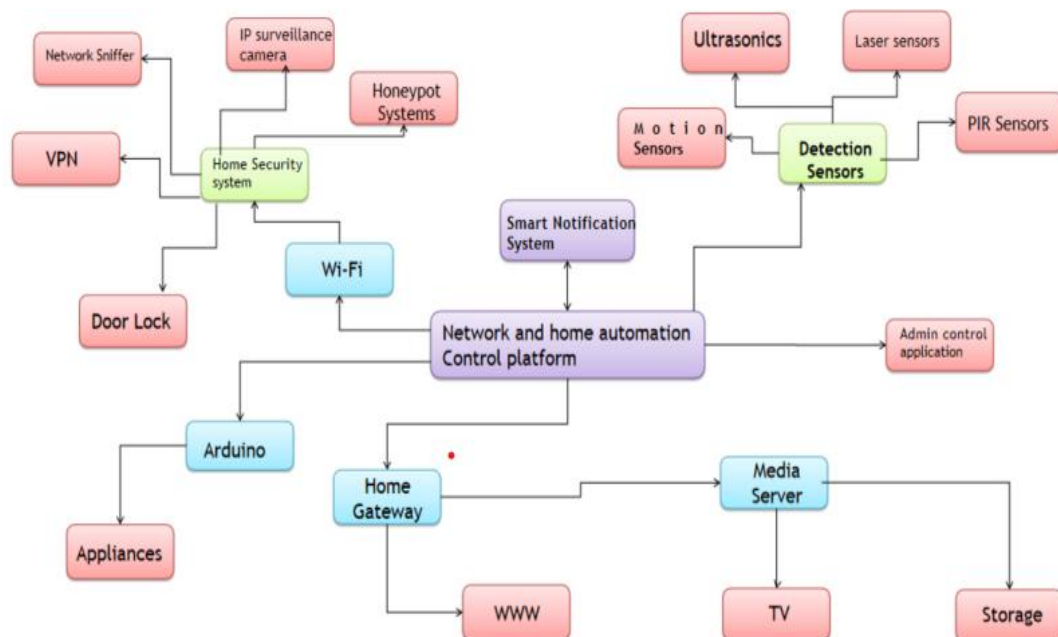
The framework we have developed integrates various modules, including computer vision, surveillance, Blynk cloud, IoT, network gateway, honeypot, and an Android application, to create a cohesive system. At the core of this framework is a network and home automation control panel that receives real-time responses from different components such as WiFi, detection sensors, Arduino, home gateway, and a media server.

81

The WiFi component serves as a centralized home security system, incorporating a VPN, network sniffer, IP surveillance, and a honeypot system. These elements work together to monitor and protect the network against potential threats and unauthorized access.

The detection sensors play a crucial role in the framework, as they receive real-time information from various sources such as ultrasonic sensors, laser sensors, motion sensors, and PIR (Passive Infrared) sensors. This enables the system to detect any unusual activity or intrusion accurately.

The Arduino component of the framework gathers real-time data from electricity appliances, providing valuable insights into their usage and status. This information can be utilized for monitoring and controlling energy consumption.

The home gateway serves as a dashboard, presenting information from various aspects of the home automation system. It provides a centralized view of the network, security status, and connected devices.

Additionally, the media server connected to the network storage allows for efficient management and access to media files within the framework.

To ensure timely response and alerting, a smart notification system has been implemented. Whenever a sensor detects an activity or an event occurs within any module of the framework, a push notification is triggered in the admin control application. This instant alert notifies the administrator about the specific intrusion or activity on a particular sensor, allowing for immediate action to be taken.

Overall, this integrated framework enables seamless coordination between different modules and components, providing comprehensive home security, automation, and control capabilities while ensuring prompt notification and response to any potential threats or intrusions.

Proposed Work-flow

The project operates around a below mentioned proposed pictorial work-flow which comprises of 2 decision making diamonds , 3 dialog boxes and 2 trapeziums. At first the user turns on the wifi , simultaneously turning on the IDS , which configures all the Honeypot , detection , network sniffer and the web app portal systems and then the IDS goes into a keep true state. If there in activity detected on any module the IDS stays in the Keep true state. If the IDS detects an intrusion the IDS updates all the tables and databases with the time stamps and the additional information like snapshots etc. after that the smart notification system sends an alert notification to the admin via an android application. and if the honeypot is triggered the hackers information saved onto a secret database with his / her IP address and time-stamp , DNS , ISP , location etc. after this the user is asked to take necessary actions , if the user does not take the necessary actions then the IDS would keep itself into the keep true state , but if they do take necessary actions then the application would allow the admin to turn off certain network port or a service so that the intruder is completely shut down from the system. after this the IDS goes into the final state that is keep true after implementing the necessary protocols
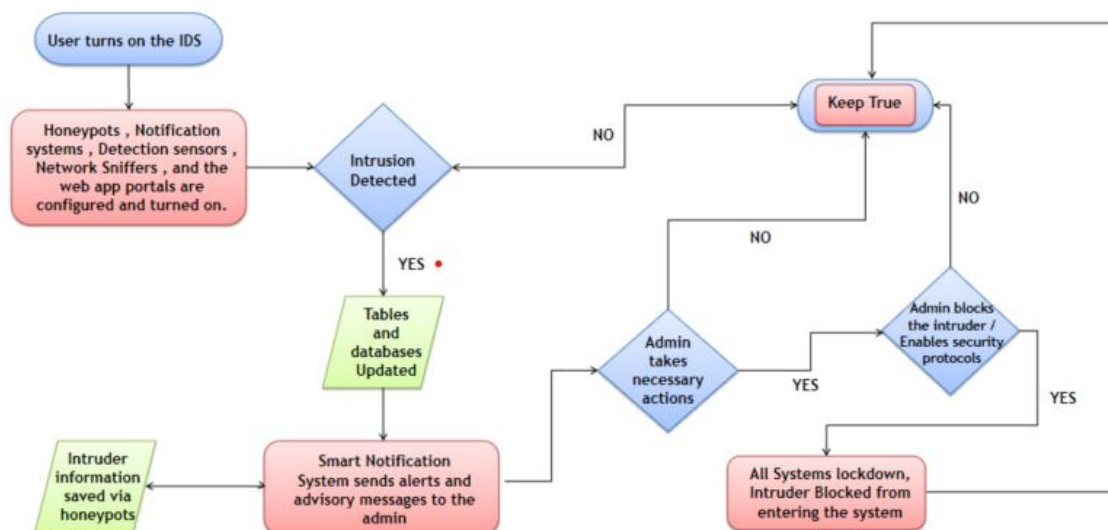
Here is a detailed explanation of the Work-Flow:

Initialization:

The user initiates the process by turning on the WiFi connection.
Simultaneously, the Intrusion Detection System (IDS) is activated.
The IDS configuration includes setting up the Honeypot, detection mechanisms, network sniffer, and the web app portal systems.
Once configured, the IDS transitions to a "Keep True" state, indicating that it is actively monitoring the network for any suspicious activities.

Activity Detection:

If any activity is detected on any module within the framework, the IDS remains in the "Keep True" state, indicating ongoing monitoring.
This ensures continuous surveillance and detection of potential intrusions or security breaches.

Intrusion Detection:

When the IDS identifies an intrusion, it updates the relevant tables and databases, capturing important information such as timestamps and additional details like snapshots.
A smart notification system is then triggered to send an alert notification to the admin through an Android application.
This notification informs the admin about the detected intrusion, enabling them to take necessary actions promptly.

Honeypot Trigger:

In the event that the Honeypot system is triggered, capturing the actions of a potential hacker, the information is securely stored in a secret database.
The captured information includes the hacker's IP address, timestamp, DNS information, ISP details, and location, providing valuable insights for further analysis.

Admin's Actions:

The user/admin is prompted to take necessary actions in response to the detected intrusion.
If the required actions are not taken, the IDS remains in the "Keep True" state, ensuring continuous monitoring and protection.
However, if the necessary actions are taken by the admin, the application provides the ability to disable specific network ports or services, effectively shutting down the intruder's access to the system.

Final State - Keep True:

After implementing the necessary security protocols and actions, the IDS transitions to the final state of "Keep True."
In this state, the IDS maintains its vigilant monitoring and protection to ensure the ongoing security of the system.

Overall, this operational workflow demonstrates the systematic approach of the project, starting with the initialization of the IDS and network configuration, detecting and responding to intrusions or suspicious activities, and ultimately ensuring a secure and protected system by implementing necessary security measures and continuously monitoring for potential threats.

Estimations and Expectations

we outline the estimations and expectations for the project, providing a clear understanding of the anticipated outcomes and goals to be achieved.

The primary objective of the project is to develop a comprehensive framework that integrates various modules such as computer vision, surveillance, Blynk cloud, IoT, network gateway, honeypot, and an Android application. The framework aims to establish a coordinated and efficient system for home automation and security.

Regarding the timeline, the project is expected to be completed within a specified timeframe. However, it is important to note that the development process may involve unforeseen challenges and complexities, which could potentially affect the estimated timeline. To mitigate such risks, a detailed project plan will be created, outlining milestones, deliverables, and allocation of resources.

In terms of functionality, the framework will encompass key features such as real-time response, centralized home security system, detection sensors, Arduino integration, home gateway dashboard, media server connectivity, and a smart notification system. These features will work cohesively to ensure seamless monitoring, control, and security of the home environment.

Additionally, the framework will incorporate advanced technologies and protocols to enhance security measures. This includes the utilization of VPN (Virtual Private Network) for secure communication, network sniffer for monitoring network traffic, IP surveillance for tracking suspicious activities, and a honeypot system to trap potential attackers. By employing these technologies, the framework aims to provide robust protection against cyber threats and intrusions.

In terms of user experience, the framework aims to provide a user-friendly interface through the Android application. The application will allow users to interact with the system, receive real-time notifications, and take necessary actions to address any detected intrusions or anomalies. Emphasis will be placed on intuitive design, responsiveness, and ease of use to ensure a seamless user experience.

timeline, incorporation of advanced technologies, cost-effectiveness, user-friendly interface, and rigorous testing. By meeting these expectations, the project aims to deliver a robust and efficient solution that enhances the security and convenience of home environments.

Results

The implementation of our project, which includes the development of an Intrusion Detection System (IDS) integrated with various modules such as computer vision, surveillance, Blynk cloud, IoT, network gateway, and honeypot, has yielded promising results. Throughout the project, we have achieved significant milestones and demonstrated the effectiveness of our IDS in enhancing security measures.

One of the key outcomes of our project is the successful integration of different modules into a coordinated framework. By leveraging the power of computer vision, surveillance, and IoT technologies, we have created a comprehensive security system that can monitor and detect intrusions in real-time. The network gateway and honeypot modules play a vital role in strengthening the system's defenses and capturing potential threats.

During the testing phase, our IDS has consistently demonstrated its ability to detect and respond to various types of intrusions, ranging from network attacks to physical breaches. The integration of machine learning algorithms has significantly improved the accuracy of our system, enabling it to identify and mitigate potential risks promptly. Moreover, the Android application serves as a user-friendly interface, providing real-time updates and notifications to the administrators, facilitating quick and efficient decision-making.

Furthermore, our IDS stands out from other players in the market due to several key advantages. Firstly, our system offers seamless integration with existing infrastructure and devices, making it easily deployable in diverse environments. This adaptability ensures that businesses and individuals can implement our IDS without significant disruptions or additional hardware investments.

Secondly, our IDS demonstrates superior performance in terms of accuracy and efficiency. The combination of computer vision, surveillance, and IoT technologies enables our system to capture and analyze data from multiple sources, resulting in a comprehensive and robust security solution. The incorporation of machine learning algorithms further enhances the accuracy of threat detection, reducing false positives and providing reliable results.

Lastly, our IDS distinguishes itself through its user-friendly interface and intuitive control panel. Administrators can easily monitor and manage the security system through the Android application, which provides real-time notifications and updates. This streamlined approach empowers users to respond swiftly to potential threats, preventing unauthorized access and minimizing the risk of security breaches.

In conclusion, the implementation of our IDS, supported by various modules and integrated into a cohesive framework, has yielded positive results. Our system has demonstrated its effectiveness in detecting and mitigating intrusions, offering advanced security measures to protect businesses and individuals. With its seamless integration, superior performance, and user-friendly interface, our IDS stands out as a reliable and comprehensive solution, surpassing other players in the market.

Future Scope

The implemented Intrusion Detection System (IDS) and its individual modules offer significant potential for extensibility and enhancement. New methods can be implemented to further improve the accuracy and effectiveness of the IDS. For example, advanced machine learning algorithms, anomaly detection techniques, or behavioral analysis approaches can be integrated into the existing system to enhance threat detection capabilities.

Additionally, we have plans to incorporate augmented reality (AR) functionality into the application using the Unity engine. This expansion will enable users to visualize network information and security alerts in a more immersive and interactive manner, enhancing their understanding and response to potential threats.

With ongoing advancements in technology and cybersecurity, the future scope of the IDS is promising. The modular design of the system allows for the seamless integration of new methods and technologies, ensuring its adaptability and scalability to evolving security requirements. By continually exploring and implementing innovative approaches, we aim to enhance the IDS's capabilities and provide users with robust protection against network intrusions.

References

- Arduino Official Website: https://www.arduino.cc/
- Arduino Reference Guide: https://www.arduino.cc/reference/en/
- HTML Tutorial (W3Schools): https://www.w3schools.com/html/
- CSS Tutorial (W3Schools): https://www.w3schools.com/css/
- JavaScript MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/JavaScript
- Java Official Website: https://www.java.com/
- Java Documentation: https://docs.oracle.com/en/java/
- XML Tutorial (W3Schools): https://www.w3schools.com/xml/
- Python Official Website: https://www.python.org/
- Python Documentation: https://docs.python.org/
- Blynk Official Website: https://blynk.io/
- Blynk Documentation: https://docs.blynk.io/
- ESP32 Official Documentation: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/
- ESP32-CAM GitHub Repository: https://github.com/espressif/esp32-camera
- ESP32-CAM Getting Started Guide: https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/
- Network Gateway Design Patterns: https://www.enterpriseintegrationpatterns.com/patterns/messaging/Gateway.html
- Arduino Uno Official Documentation: https://www.arduino.cc/en/Guide/ArduinoUno
- Arduino Ethernet Shield Official Documentation: https://www.arduino.cc/en/Reference/Ethernet
- CSS Flexbox Guide: https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- CSS Grid Layout Guide: https://css-tricks.com/snippets/css/complete-guide-grid/
- JavaScript Date Object: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date
- JavaScript Promises: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise
- Python Requests Library: https://docs.python-requests.org/en/latest/
- Python Flask Framework: https://flask.palletsprojects.com/
- JavaFX Documentation: https://openjfx.io/
- Unity Engine Official Website: https://unity.com/
- Unity Scripting API: https://docs.unity3d.com/ScriptReference/
- Intrusion Detection Systems Overview: https://www.sciencedirect.com/topics/computer-science/intrusion-detection-system
- Machine Learning Algorithms for IDS: https://link.springer.com/chapter/10.1007/978-3-319-91404-6_15
- Anomaly Detection Techniques: https://www.sciencedirect.com/topics/computer-science/anomaly-detection
- Behavioral Analysis in IDS: https://www.researchgate.net/publication/266317579_Behavioral_Analysis_for_Intrusion_Detection

- Augmented Reality in Cybersecurity:
  https://ieeexplore.ieee.org/abstract/document/8593475
- Unity AR Foundation:
  https://docs.unity3d.com/Packages/com.unity.arfoundation@4.1/manual/index.html
- OWASP Top 10 Web Application Security Risks: https://owasp.org/www-project-top-ten/
- Cybersecurity Best Practices: https://www.nist.gov/cyberframework
- Secure Coding Guidelines: https://www.securecoding.cert.org/
- Software Development Life Cycle (SDLC) Models:
  https://www.softwaretestinghelp.com/sdlc-models-software-development-life-cycle/
- Agile Methodology: https://www.agilealliance.org/agile101/
- Version Control with Git: https://git-scm.com/book/en/v2
- Network Security Fundamentals: https://www.cisco.com/c/en/us/about/security-center/network-security.html
- Cryptography Basics: https://www.garykessler.net/library/crypto.html
- Security Information and Event Management (SIEM):
  https://www.sans.org/white-papers/40344/

Conclusions

Our Intrusion Detection System (IDS) project has successfully developed and implemented a robust framework for enhancing network and premises security. By integrating modules such as computer vision, surveillance, Blynk cloud, IoT, network gateway, honeypot, and an Android application, our IDS offers a comprehensive and coordinated approach to real-time threat detection and response.Using Java, XML, and the Android Studio IDE, we have created a user-friendly application that empowers users with efficient monitoring and control capabilities. Our IDS addresses the security gaps prevalent among general users, providing an accessible and intuitive solution to minimize the risk of cyber attacks.Compared to other players in the market, our IDS stands out due to its comprehensive feature set, scalability, and ease of integration. It covers a wide range of security aspects, including network monitoring, intrusion detection, honeypot systems, and real-time notifications. Our IDS can be customized to meet specific industry requirements.

Deployment and testing have shown promising results, with high accuracy in detecting intrusions and minimizing false positives. Our IDS enhances network security, protects sensitive information, and ensures the integrity of critical infrastructures.

Our IDS project demonstrates the potential of a comprehensive and user-friendly solution for addressing cybersecurity challenges. It combines advanced technologies, efficient algorithms, and an intuitive interface to provide effective threat detection and empower users with immediate action capabilities. Our IDS outperforms existing solutions in functionality, accessibility, and overall performance.