PROJECT REPORT ON

## “Security camera”

**ABSTRACT**

The theme of this project is intelligent visual surveillance systems. In recent times, we used surveillance cameras for monitoring and recording moments, but manual surveillance and real-time monitoring is one of the most important and challenging branches of computer vision, which has been widely applied in peoples’ life, such as monitoring security. The presence of surveillance cameras and a warning sign indicating that the area is under monitoring can serve as a significant deterrent to criminals and thieves, as the recorded footage can be used to identify people and trace their activities. It can be more advanced with Wi-Fi, which is a local area network running in a local environment or in a distributed setting. Wi-Fi network protocol is one of the leading communication technologies used in the IoT world which supports low transmit power along with low cost. ESP32 is the second generation of Express if Corporation IoT solution and it includes Wi Fi. ESP32 reduces high network traffic and computing load.

Manual surveillance seems tedious and time consuming. Security can be defined in different terms in different contexts like theft identification, violence detection, chances of explosion etc. In crowded public places the term security covers almost all type of abnormal events. Among them violence detection is difficult to handle since it involves group activity. The anomalous or abnormal activity analysis in a crowd video scene is very difficult due to several real world constraints. The paper includes a deep rooted survey which starts from object recognition, action recognition, crowd analysis and finally violence detection in a crowd environment. Majority of the papers reviewed in this survey are based on deep learning technique. Various deep learning methods are compared in terms of their algorithms and models. The main focus of this survey is application of deep learning techniques in detecting the exact count, involved persons and the happened activity in a large crowd at all climate conditions. Paper discusses the underlying deep learning implementation technology involved in various crowd video analysis methods. Real time processing, an important issue which is yet to be explored more in this field is also considered. Not many methods are there in handling all these issues simultaneously. The issues recognized in existing methods are identified and summarized. Also future direction is given to reduce the obstacles identified.

## CONTENTS

[CERTIFICATE iii](#_2y3w247)

[DECLARATION iv](#_1d96cc0)

ACKNOWLEDGMENT v

[ABSTRACT vi](#_gjdgxs)

LIST OF TABLES vii

LIST OF FIGURES vii



**Chapter-1**

#### Internet of Things

Internet of Things

Since the coining of the term in 1999, the internet of things (IoT) has transformed from a mere vision to a palpable reality. This can be attributed to the extensive use of the Internet Protocol (IP), the rise of ubiquitous computing, and the continued advancement of data analytics, among other drivers of development. By 2020, it is estimated that there will be [20.4 billion](https://www.gartner.com/newsroom/id/3598917) devices connected to the IoT. Despite its continuing expansion, however, the IoT remains to some degree an obscure concept, something that’s often referred to in abstract terms even as it provides manifest benefits.

The IoT can be described as an extension of the internet and other network connections to different sensors and devices — or “things” — affording even simple objects, such as lightbulbs, locks, and vents, a higher degree of computing and analytical capabilities.

Interoperability is one of the key aspects of the IoT that contribute to its growing popularity. Connected or “smart” devices — as “things” in the IoT are often called — have the ability to gather and share data from their environments with other devices and networks. Through the analysis and processing of the data, devices can perform their functions with little or no need for human interaction.

Given the ever-increasing number of connected devices, the IoT continues its path of evolution, adding different layers to the data that is already being shared and processed, and giving rise to sophisticated algorithms that result in improved levels of automation. And because of the variety of “things” that can be connected to it, the IoT has enabled diverse applications for individual users and entire industries alike.

: How does the IoT work?

The “things” that make up the IoT can be anything from a wearable fitness tracker to an autonomous vehicle. No matter what function they serve for users, these devices must have the following components for them to properly operate as parts of their respective IoT systems.

**Sensors.** Data is first collected from the environment for the IoT system to begin processing. It is collected by sensors in devices that can measure observable occurrences or changes in the environment. The kind of data being measured by the device depends on its function: It can be a person’s pulse in the case of a fitness tracker or the distance of the nearest object in that of an autonomous vehicle.

**Connection and identification.** The data must be communicated from the device to the rest of the IoT system, be it to a computer or to another device. And for this communication to have any meaning, a device must have a unique identifiable presence on the internet, accomplished, and through its own IP address.

**Actuators.** Most IoT devices are capable of doing their primary functions without physical interaction with their users. IoT devices should be able to take action based on data from their sensors and the subsequent feedback from the network. A smart lightbulb, for example, can turn on upon the command of its user, even when the user is miles away. In the same manner, a valve in a smart factory can automatically open or close according to data gathered by its sensors along the production line.

Even though the devices are usually built with automation in mind, other technologies must be in place for IoT systems to work. Completing the links of how IoT systems process data are the following components.

**IoT gateway.** The IoT gateway acts as a bridge for the different devices’ data to reach the cloud. It also helps in translating the different protocols of the various IoT devices into just one standard protocol and in filtering out unnecessary data gathered by the devices.

**The cloud.** The cloud is where all the data from the different devices is gathered and where software can reach this data for processing. Because most of data processing happens in the cloud, it lessens the burden on individual devices.

**User interface.** The user interface communicates to the users the data gathered by the devices and allows the users to make the necessary commands to be executed by the devices.

The Internet Architecture Board released a [guiding document](http://www.rfc-editor.org/info/rfc7452) that outlines the four communication channels used by the IoT. The four models also demonstrate how the connectivity of IoT devices helps extend the value of each device and adds quality to the overall user experience:

**Device-to-Device.** This model represents how two or more devices connect and communicate directly with one another. Communication between devices is usually achieved through protocols such as Bluetooth, Z-Wave and Zigbee. This model is often found in in wearables and in home automation devices, where small packets of data are communicated from one device to another, as with a door lock to a lightbulb.

**Device-to-Cloud.** Many IoT devices connect to the cloud, often with the use of wired Ethernet or Wi-Fi. Connecting to the cloud allows users and related applications to access the devices, making it possible to course through commands remotely as well as push necessary updates to the device software. Through this connection, the devices can also collect user data for the improvement of their service providers.

**Device-to-Gateway.** Before connecting to the cloud, IoT devices can communicate first with an intermediary gateway device. The gateway can translate protocols and add an additional layer of security for the entire IoT system. In the case of a smart home, for example, all smart devices can be connected to a hub (the gateway) that helps the different devices to work together despite having different connection protocols.

**Back-End Data-Sharing.** An extension of the device-to-cloud model, this model allows users to gain access to and analyse a collection of data from different smart devices. A company, for instance, can use this model to access information from all of the devices working inside the company building as organized together in the cloud. This model also helps lessen issues with data portability.

: Working of IoT

Just as the internet at large affects a broad spectrum of users, so does the IoT. Depending on the scale of connectivity and the number of devices involved, the IoT can have significant and specific applications, be they for a single user or for an entire city. Common applications of the IoT include the following.

**People and homes.** People make direct use of IoT devices through technology that can be worn, such as smartwatches and fitness trackers, and devices that help make receiving and collecting information possible in real-time. Applied to households, IoT devices can be used for a more connected, energy-efficient, and conveniently run home. Different aspects of a connected home can also be remotely accessed and controlled by homeowners through a computer or a handheld smart device.

**Automobiles.** Sensors within a moving vehicle make it possible to collect real-time data about the vehicle and its surroundings. Autonomous vehicles use different sensors in combination with advanced control systems to assess their environments and consequently drive.

**Factories.** With the application of IoT in factories, manufacturers can automate repetitive tasks as well as access information on any part of the entire manufacturing process. Information provided by sensors on factory types of machinery can help in devising ways to make the entire production line more efficient and less accident-prone.

**Businesses.** On a larger scale, with the adoption of IoT technologies, businesses can be more cost-effective, efficient, and productive. For example, office buildings can be fitted with sensors that can monitor elevator traffic or overall energy consumption. Different industries naturally have different applications of the IoT: In the healthcare industry, IoT devices may be used to gain instant and accurate updates about the condition of patients, while in the retail industry, IoT devices may be deployed to help shoppers locate products and to monitor inventory.

**Cities.** The combined uses of different IoT devices can cover urban and public areas. IoT devices can gather data from and affect its environment to help manage the various aspects of city governance, such as traffic control, resource management, and public safety.

Standards and Regulations

While it broadens the scope of applications, the growing number of connected devices makes the standardization and [regulation](https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/securing-the-internet-of-things-through-effective-regulation) of the IoT a complicated and nettlesome affair. Standardization and regulation issues can range from technical problems to legal matters. Fragmentation, for example, is a technical problem faced by users because of the lack in IoT standards. Different smart devices may use various wireless communication protocols like Bluetooth, Wi-Fi, Zigbee, and 5G, hindering communication within IoT systems. On the other hand, lack of regulation highlights existing internet-related issues, as well as adding another layer of complexity to these issues. Determining accountability is one example: Should there be defects and breaches related to IoT device usage, lack of regulation leaves accountability difficult to determine. Standards and regulations affect the overall quality of services that IoT technologies render, and therefore concern all IoT stakeholders, be they individual users, device manufacturers, or organizations integrating the technologies into their processes.

Privacy

Privacy awareness has grown with the increase in diversity of shared personal information over the internet. The IoT further complicates this issue as it expands the types of data being recorded and shared over the internet. Since the IoT works better by getting as detailed a view of enviPronments as possible, it presents a trade-off between user privacy and quality ofservice. Determining the points where data collection should be limited, or even stopping the collection of data altogether on account of user privacy concerns, is also difficult to achieve, especially with the automated nature of most IoT systems.

Security

Security concerns will always be present when handling of data and information is involved. The IoT adds its own security challenges with its access to a wide variety of personal information and its close integration into individual and organizational activities. These characteristics of the IoT make the technology a viable target for cybercriminals. In addition, any breach, attack, and vulnerability with a single IoT device or system [weakens the overall](https://blog.trendmicro.com/trendlabs-security-intelligence/home-routers-mitigating-attacks-that-turn-them-to-zombies/) [security](https://blog.trendmicro.com/trendlabs-security-intelligence/home-routers-mitigating-attacks-that-turn-them-to-zombies/) of the networks concerned.

Other security threats related to IoT technologies include the following:

* The homogeneity of mass-produced smart devices means the proliferation of the same possible vulnerabilities.
* The automation of IoT systems makes it more difficult to detect vulnerabilities and breaches because of the reduced need for human interference.
* Environments in which IoT devices are deployed make these devices vulnerable to unforeseen physical threats where attackers may tamper with devices directly.
* The interconnectivity of IoT systems makes every part of the system an avenue for data breaches and cyberattacks, which can spread to the rest of the affected networks.

**How can the use of the IoT be secured?**

Different security practices may apply to the different types of IoT devices and systems. However, securing the IoT while also maintaining its relevance is the shared responsibility of its key players — from IoT manufacturers to end-users

Strong security features can be integrated from the design phase by manufacturers, while service providers can make sure that security is sustained by pushing updates and patches when necessary. Users like organizations that apply smart devices in their businesses can continually monitor all their devices, not depending completely on IoT automation. [Adequate cybersecurity solutions](https://www.trendmicro.com/us/iot-security/Solutions) can add multiple layers of defense against unforeseen risks for all the stakeholders.

The security responsibilities of each of those involved in the IoT do not exist in a vacuum. Taking a collaborative view on the safety of the IoT does not only protect things like personal and company assets but also has an extended effect of making the connected world more protected.

: Applications of IoT

### Smart Home and Office

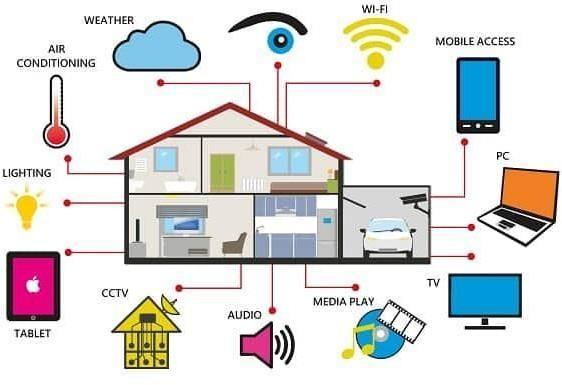


Fig. 1.1

Smart home applications with the use of smart sensors are becoming popular now. Any smart device can be configured and connected to the internet and control using simple mobile application.

Smart Door access control system

Smart locks and door access systems are one of the most popular and cost-effective solutions of Internet of Things. Smart locks are easy to implement and control using a web interface or Smartphone application.

Integration with RDIF tags and smart door access systems can be securely implemented. Users can grant access to the doors using a mobile app and lock them again once the person leaves the premises.

For example, if a person wants to enter your house while you are not around, you will be able to open the door for that person using the Smartphone application.

Smart lighting for home and office

Smart lighting is one the attractive smart home application using internet of things. In addition to energy saving, it also enables us to manage effectively. Light ambience can be changed using smart hub devices or smartphone apps

Smart lighting can be configured to respond to voice commands and motion detectors/proximity sensors. These sensors will activate the light when someone enters the room or leaves the room. Moreover, it can be configured to turn on when the ambient light is below a certain threshold (turn on during sunlight is low).

Automated Gate and garage

Using smart sensor technology and the internet of things, gates and garages can be controlled (operated) conveniently. Once you are about to enter the house or after leaving the premises, you may open or close the gate using your mobile devices.

Smart thermostats and humidity controllers

Smart thermostats are cost-effective and convenient smart home solutions that can be controlled using an internet connection and smart hub device (or using a Smartphone app).

Common sensors for home/office automation:

* Motion/proximity sensors
* Voice-controlled sensor
* Light sensor
* Temperature and humidity sensors
* Smoke/fire sensor
* Precipitation sensor

Traffic Management

Analyzing traffic over a period of time gives an insight into possible trends and patterns that could occur during peak hours. It will help to inform commuters to take alternative routes to avoid congestion and delay.

Smart lighting on streets

Smart lighting is an effective solution to save energy in the cities. Smart sensors can detect presents of people or vehicles in the proximity and increase light intensity when someone pass by.

Once the person or vehicle is away from that area, smart light will automatically reduce light intensity to save energy. During emergency situations, maximum light intensity will be activated to support recovery activities.

Since the smart lighting systems are connected to control and monitoring network, any faulty light units will be automatically reported and necessary maintenance will be initiated.

Pollution monitoring and reporting

Increasing air pollution is one of the challenges we are facing in every growing cities. In order to solve this issue, smart sensors are deployed across the cities to continuously monitor any changes.

Some of the common sensors are temperature, air quality (like CO2 level, haze, and smoke), moisture etc. Interconnected smart sensors collects data, sends these data to the monitoring stations and initiates warning messages during bad air quality detection.

Smart Parking Solutions

Smart sensors installed on parking area are collecting information about availability of parking slots and updating it to the database real time. Once the spot is occupied, it will be updated without any delay.

Service providers and customers can plan and manage parking issues with the use of smart parking solutions.

Water / waste management

Populations in cities are increasing every year, based on statistics this trend will grow in coming years. Increase in population contributes to increase in wastes as well.

Many cities are adapting recycling of water using water treatment units. With IoT system, the amount of waste water, consumption in a geographical area and trend of waste produced can be analyzed effectively.

IoT and smart sensor technology enables us to manage this issue efficiently. With smart waste management system, authorities will be able to predict the amount of waste produced in a particular location, how to process properly, trigger clearance of waste and analyse data for future planning etc.

Example: smart sensors implemented on trash bins can send alerts to the waste management system once the bin is full (or reached the threshold limit). If the waste quantity in the bin is low, it will not be emptied.

With analytics solutions, an overview of waste generated in every part of the city, and how much waste are generated in duration can be easily assessed. This information will be used to plan for the city expansion and upgrading projects.

Fleets for waste collection and treatment can be managed and any changing trends can be predicted via smart analytics solutions.

### Wearable Devices

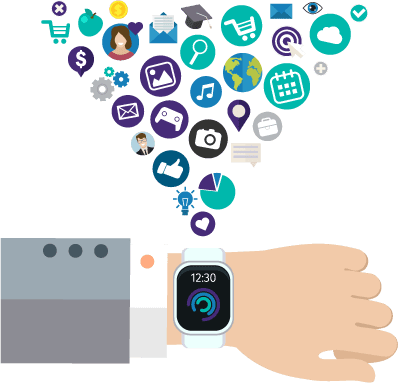


Fig. 1.2

Wearable smart devices introduced as smart watches around a decade ago and many more functions were added since then. Now our smart watches and wearable are capable of reading text messages, showing notifications of other apps, tracking location, monitor workout status, remind schedules and continuously monitoring health conditions.

With the Internet of Things, wearable technology can be used beyond these functions. Major smart wearable manufacturers are developing special operating systems and applications dedicated for smart wearable devices.

Many people have shared their stories how a smart watch saved their life during an accident and medical emergencies. Lifesaving applications make smart wearable one of the most favourite devices among other IoT devices.

Parents can track their child’s location; caretakers will get a notification if the patient’s vitals are low or blood sugar levels are changing. Doctors and medical professionals can continuously monitor their patient’s body conditions in real-time using wearable technology.

Future smart devices like smartwatches and fitness bands will be optimized to perform more functions and connect with other smart IoT devices in smart homes and other applications. Pairing with Smartphone applications will enable these smart wearables to initiate more tasks and get notified promptly.

Future of Wearable technology

Future wearable technology will be capable of early detection of diseases and trigger for treatment during early stages. Sensitive nano-sensors will have the capability to detect components in our body fluids (sweat, tear and saliva) and notify certain physical conditions that could trigger more severe disease in future.

Surgical implanted nano-sensors will give indication of possible medical conditions (like cancer) that could develop in our body before it become severe. Finding a medical condition in early stages has more effectiveness in treatment.

For example: if we are able to find out chances of developing diabetes before it affect our body, we can change our diet and seek medical advice to avoid or delay the decease as much as possible.

Future healthcare procedure will include more wearable devices for convenient, accurate detection of diseases and monitoring of many medical conditions.

### Healthcare



Fig. 1.3

Healthcare industry has been utilizing the possibilities of Internet of Things for life saving applications. Starting from collecting vital data from bed side devices, real-time diagnosing process, accessing medical records and patient information across multiple departments, the entire system of patient care can be improved with IoT implementation.

IoT will offer convenience for medical practitioners, improve accuracy in the information (helps to reduce error in the data), increase overall efficiency and saves time for each procedure.

Doctors can monitor patient’s status remotely and suggest necessary procedures when required.

Data loss and mistakes will be reduced to a lower level with IoT devices. Most of the modern medical devices can be connected to the network and data can be accessed securely (In future, all devices will have the capability to connect the network).

Round the clock patient monitoring is possible with smart IoT devices. Immediate change in the vitals of a patient will automatically notify responsible medical practitioners real-time.

Doctors can prescribe medicine after assessing patient remotely with the help of smart IoT devices. In many cases, hospital visits many not be required.

Example: Many hospitals are offering telemedicine facility. Patients can follow-up treatment via video conferencing.

Apart from the efficiency and cost effectiveness of the healthcare systems, IoT also offers better patient satisfaction. Overall hospital experience will be improved with implementation of IoT in healthcare.

### Autonomous Driving

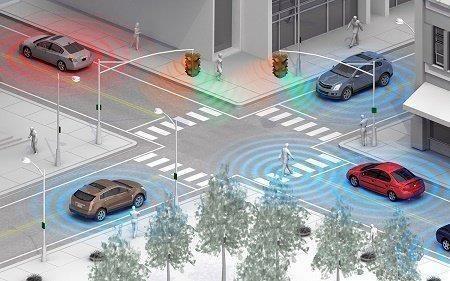


Fig. 1.4

Autonomous driving has been evolving with the use of artificial intelligence and smart sensor technology in Internet of Things. Earlier generation of autonomous vehicle (partial automation) will assists drivers to drive safely, avoid collisions and warn about the conditions of the road and vehicle.

Example: cruise control assistance, parking assistance, line changing assistance and efficient fuel /energy management etc.

As we collect huge amount of data from thousands of vehicles (using millions of sensors and camera units), AI can predict certain scenarios on the road and help to implement in the future generation of vehicles for better safety and efficiency.

Self-driving cars and connected car concept will offer a much safer road experience in future with the use of Internet of Things and artificial intelligence (AI). One of the significant components of IoT in automobiles is smart sensors which are continually collecting information about the vehicle, road condition, about other vehicles, objects on the road and road conditions.

The system consists of camera units, proximity sensors, RADARs, RF antenna arrays to collect information and help the vehicle to make decision based on the sudden changes on the road. Vehicles and smart objects can share information each other using RF technology.

Example scenarios: ice fall on road, vehicle breakdown / accident on one line and heavy traffic in a particular direction on highway etc.…

Accurate information is significant in making a split-second decision while driving. There would be a bigger impact if the data is not accurate or delayed and it could even lead to fatal accidents.

5G technology offers faster data rate with low latency network which is curtail for autonomous driving technology.

Sophisticated algorithms are being developed to learn different scenarios from various conditions on the road. This powerful software with continuously learning AI, manufacturers can enhance the safety of the self-driving (full automation) vehicles in future.

### Agriculture and Smart farming



Fig. 1.5

There are lot of challenges in the agriculture and farming industry to produce more crops and vegetable to feed increasing human population. Internet of Things can assist farmers and researchers in this area to find more optimized and cost-effective ways to increase production.

Internet of Things is one of the promising solutions to make entire agriculture and farming industry more efficient with a smaller number of workers. Smart sensor technology will help improve each stage of agriculture and automation helps to reduce manual labor.

Smart irrigation

Smart irrigation is a method of efficient use of water for agriculture using Internet of Things. Smart sensors are deployed into soil which constantly monitoring and sending information about soil conditions to the control station.

Once the soil starts to dry or reaches a threshold value defined by farmer, control system initiates the water flow and it will be stopped after a set time. Wastage or water and manual labor can be reduced by implementing automated irrigation system into agriculture.

Smart Greenhouse using sensors

Greenhouse farming is one of the successful agriculture methods to artificially controlling the environment for increased production of vegetables and fruits. Inside the green house, the essential parameters like **CO2 level**, **temperature** and **moisture** level are monitored

round the clock and automatic precipitation, light and moisture control will be activated using IoT system.

This smart monitoring system control is much more efficient and cost effective than the same task performed by workers. The data collected using various sensors deployed in the greenhouse will be sent to the cloud, it helps to easily access the data for further analysis. Predication farming is a method of applying useful information collected over duration of time for improved quantity and quality of agriculture products. Experts will analyze when would be the best time/season for farming, what should be best parameters for maximum productivity, suitable fertilizers and how to plan a particular product ready for harvest etc.

Smart Farming

Internet of Things offers many solutions for convenient tracking of animals with the use of smart RFID tags. Farmers can easily record data of each animal with implementation of IoT and smart tags.

For example: movement (cow, sheep) from a particular location, age and weight of individual and vaccination details can be stored in database and easily accessed by just scanning the smart tag.

### Industrial IoT for manufacturing



Fig. 1.6

Manufacturing industry is one of the early adopters of Internet of Things which entirely changed several stages of a product development cycle. Industrial IoT will help optimize various stages of product manufacturing such as:

* + Monitoring of supply chain and inventory management
  + Optimization in product development
  + Automate mass production processes
  + Quality testing and product improvement
  + Improves packaging and management
  + Process optimization using data collected from huge number of sensor networks Cost effective solution for overall management of factories

### Disaster management



Fig. 1.7

Internet of Things with wide range of smart sensors allow engineers to build a more effective emergency response system for factories, schools, hospitals, airports and any other public gathering places. Any emergency situations like fire outbreak or flooding will be

automatically detected using sensors and this information is shared to responsible work groups in real time.

Disaster management team can respond effectively within seconds to start recovery operations. With better preparation plan, disaster management team can work safely and assist each individual to evacuate safely during an emergency situation.

During an emergency, fire department, emergency response volunteers, police force, ambulance units and nearby hospitals will receive an alert about the scenario. Automated warning system improves the preparedness and allows authorities to plan and handle any kind of situations immediately.

Some of the common sensors: smoke detector, temperature sensor, humidity sensor, CO2 monitoring sensors and precipitation detector.

Sensor deployed in many locations (where higher possibility of wildfire could happen) is continuously monitoring CO2 emission levels, fire and smoke. These smart sensors are connected to a network where any changes in the data will be immediately identified and alert warnings will be sent during a wildfire.

### Logistic and fleet management



Fig. 1.8

Smart logistics is a complex task since the goods must be handled with greater care and efficiency. Apart from moving from one location to another location, service providers have to make sure perfect condition is maintained during transportation.

Smart sensors capable of connecting to IoT network continuously monitoring the GPS location, temperature, humidity, shock and tilt angle of the container used for transpiration. Data collected from these sensors are processed and analyzed in a central cloud system.

Logistics team can access this information from anywhere using an internet connection. Movement of fleet can be monitored real-time and updated to customers about the progress of delivery. Any delay during transportation will be notified to the responsible members of the team.

### Smart Grids and energy management

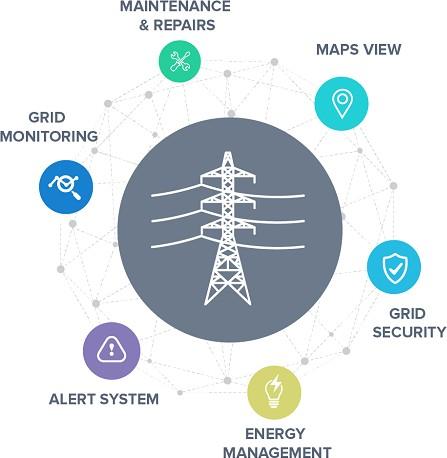


Fig. 1.9

[Smart grid](https://www.webnms.com/iot/smart-power-grid.html) concept is an enhancement of existing power grids with sensors deployed on the transmission lines and individual customer outlets. Theses sensors helps to notify any failure, abnormality in the line, understand the nature of usage and behavior pattern over time.

These data can be used to find out areas of improvement, lossy nodes during transmission, and peak time usage statistics with the use of smart meters and sensors. Energy companies

can use this information to improve existing grids and implement new changes during upgrade and thus reduce carbon emission.

In case one of the transmission line is down, smart sensors will automatically trigger to switch to another grid to provide uninterrupted supply. Manual action from a worker could take longer response time and this could cause long power outages and losses.

Customers also will benefit from implementation of IoT for energy management. With optimized use and energy saving, overall efficiency and energy wastage can be significantly reduced.

### BIG DATA ANALYTICS

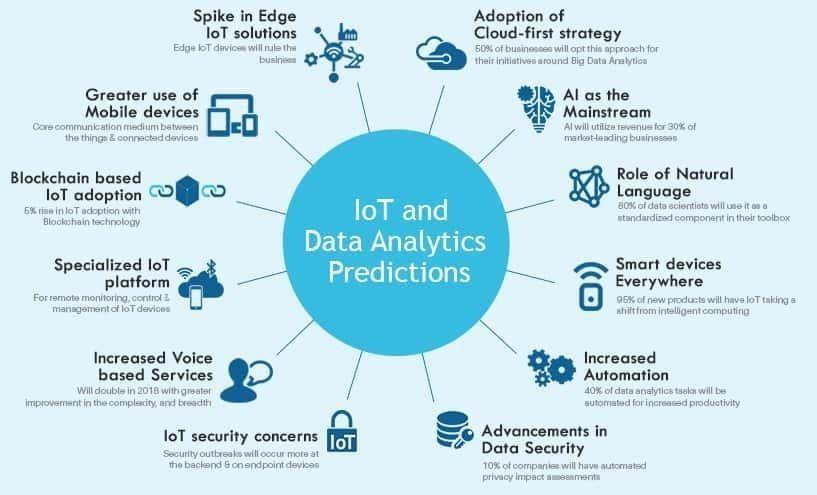


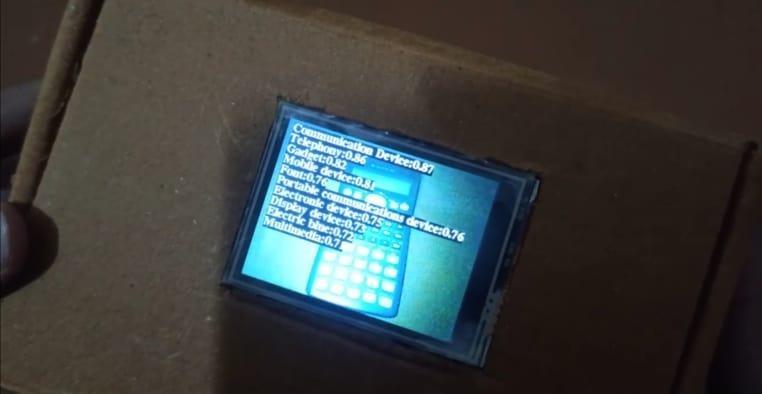
Fig. 1.10

One of the basic components of big data analytics is the data itself; many organizations consider data as most valuable asset to grow their business strategies. The source of data could be from anywhere like machines, environment, plants, peoples or even animals.

Internet of Things uses hundreds of types of sensors designed to collect data from wide range of applications. Huge amount of data from millions of smart sensors will help big data analytics to improve its decision-making algorithm using artificial intelligence and machine learning. For example: autonomous driving technology needs lot of data from many sensors embedded in to the vehicles. These smart sensors are collecting behavior of engine, field data, maps, camera feeds to improve self-driving algorithm to handle any situation that could occur while driving.

# Overview: DIY AI Camera with Google Vision & ESP32 CAM Module

In this project we will develop an **AI Camera** using **Google Vision API** & **ESP32 CAM Module**. This is basically detailed testing of **Google Vision API** with ESP32 Camera for the applications of **Artificial Intelligence** and **Machine Learning**. The developed **AI camera** can detect objects in the frame captured and displays the frame as well as detected labels on the TFT LCD Screen.

The [**Google Vision API**](https://cloud.google.com/vision) allows developers to easily integrate **vision detection** features within applications, including image labeling, face, and landmark detection, optical character recognition (OCR), and tagging of explicit content. We will be implementing the same Google Vision functionalities with the **ESP32 Camera Module**. We selected ESP32 CAM module because it is an ideal solution for image processing **IoT applications**.

The project requires some time and patience as a lot of steps are involved in it. We will write the **Arduino Code** for ESP32 CAM Module and add some libraries like **TFT Library**, **JSON Library** & **Decoder Library**. The next process involves setting up Google Vision API & **NodeJS installation** with some settings required for **GCP**. All the hardware setup along with **Arduino** & **NodeJS code** is fully explained in this article.

Thus developing a homemade **AI Camera** using Google Vision & ESP32 CAM Module would be easy.



# Components Required

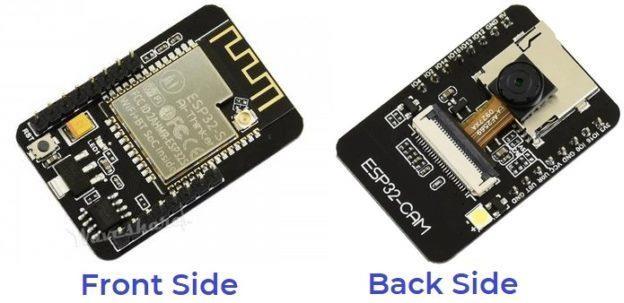
The following is the list Component required **with Google Vision & ESP32 CAM**. TheESP32 CAM when combined with other **hardware** & **firmware** Take a Picture & Google Vision scans for details.You can purchase all these components from Amazon.

| **S.N.** | **COMPONENTS** |
| --- | --- |
| 1 | ESP32-CAM Board Ai-Thinker |
| 2 | TFT LCD Display ILI9341 2.8" |
| 3 | Push Button Switch |
| 4 | FTDI Module |
| 5 | USB Cable |
| 6 | Jumper Wires |



# ESP32 CAM Module

The ESP32 Based Camera Module developed by **AI-Thinker**. The controller is based on a **32-bit CPU** & has a combined **Wi-Fi** + **Bluetooth/BLE** Chip. It has a built-in **520 KB SRAM** with an external **4M PSRAM**. Its GPIO Pins have support like **UART, SPI, I2C, PWM, ADC,** and **DAC**.



The module combines with the **OV2640 Camera Module** which has the highest Camera

Resolution up to **1600 × 1200**. The camera connects to the ESP32 CAM Board using a 24 pins gold plated connector. The board supports an **SD Card** of up to **4GB**. The SD Card

stores capture images.

To learn in detail about the ESP32 Camera Module you can refer to our previous [**Getting**](https://how2electronics.com/getting-started-with-esp32-cam-board-video-streaming-over-wifi/)

[**Started Tutorial**](https://how2electronics.com/getting-started-with-esp32-cam-board-video-streaming-over-wifi/).



# ESP32-CAM ardino Connection

The board doesn’t have a **programmer chip**. So In order to program this board, you can use any type of **USB-to-TTL Module**. There are so many **ardino Module** available based on [**CP2102**](https://www.silabs.com/documents/public/data-sheets/CP2102-9.pdf) or [**CP2104**](https://www.silabs.com/documents/public/data-sheets/cp2104.pdf) Chip or any other chip.

Make a following **connection between ardino Module and ESP32 CAM** module.

| **ESP32-CAM** | **Ardino Programmer** |
| --- | --- |
| GND | GND |
| 5V | VCC |
| U0R | TX |
| U0T | RX |
| GPIO0 | GND |

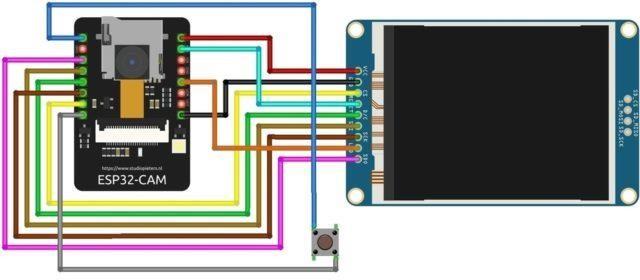
Connect the **5V** & **GND** Pin of ESP32 to 5V & GND of FTDI Module. Similarly, connect the **Rx** to **UOT** and **Tx** to **UOR** Pin. And the most important thing, you need to short

the **IO0** and **GND** Pin together. This is to put the device in **programming mode**. Once programming is done you can remove it.



# Project Schematic Design

In order to program the **ESP32 CAM Module**, you can use the above **schematic**. But the schematic for the project Google Vision API with ESP32 Camera is a little different. We have used the **ILI9341 2.8″ TFT LCD Display** so that the display will be used for displaying the captured image. Here is the **connection diagram** for the project.



| **S.N.** | **2.8" SPI LCD DISPLAY** | **ESP32 CAM** |
| --- | --- | --- |
| 1 | VCC | 3.3V |
| 2 | GND | GND |
| 3 | CS | IO2 |
| 4 | RESET | IO16 |
| 5 | D/C | IO15 |
| 6 | SDI | IO13 |
| 7 | SCK | IO14 |
| 8 | LED | VCC |
| 9 | SDO | IO12 |

The connection between LCD Display and ESP32 CAM are as follows.

A **push button** is also used in this project, which is used for capturing images. The push- button is connected to ESP32 CAM **I04 pins** and the other end is held high with **VCC**. When the button is pressed the high logic level is enabled and the image is captured.

Here is my homemade setup made using card board. All the components are easily placed inside the box.

The top side of the box only has TFT LCD Display and a Push Button Switch. The TFT LCD is used for displaying captured images with AI detection.

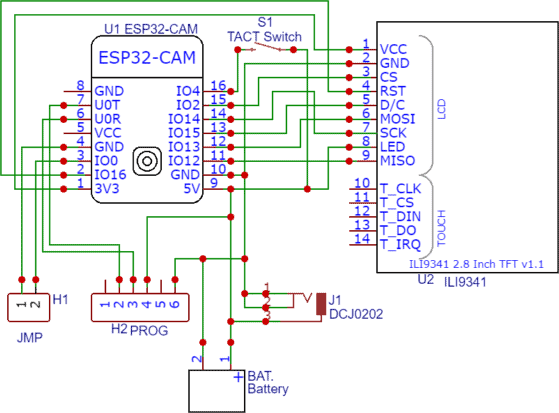
The bottom side of the box only has a camera outlet which is used for the camera function.



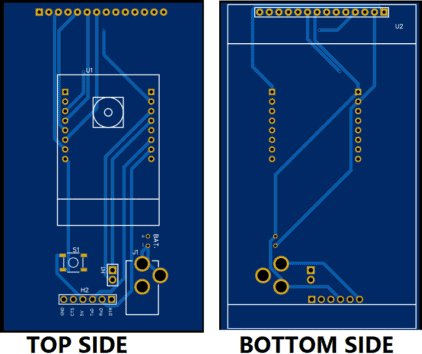
# PCB Design + Gerber Files + PCB Ordering Online

If you don’t want to assemble the circuit on a breadboard and you want PCB for the

project, then here is the PCB for you. The PCB Board for ESP32 CAM AI Board is designed using [**EasyEDA**](https://easyeda.com/) online Circuit Schematics & PCB designing tool. The schematic looks something like below.



The schematic is then converted to PCB. The top view and bottom view of the PCB are given below.



The Gerber File for the PCB is given below. You can simply download the Gerber File and order the PCB from [**ALLPCB**](https://www.allpcb.com/activity/prototype2023.html?code=tw11) at **1$** only.

Download Gerber File: [**ESP32-CAM AI Camera PCB**](https://drive.google.com/file/d/1SgBOri4rbxbC79ul3lz0e5ruf-psBgUO/view?usp=sharing)

You can use this Gerber file to order high quality PCB for this project. To do that visit the **ALLPCB** official website by clicki

# Flow of Data

Here we have explained the whole workflow, from how the object is detected to displaying labels on the screen. We are having our **ESP32 CAM module** which captures the **image** of the environment or the object and then sends it to the **TFT screen** using **SPI protocol** so that the image is displayed on the screen.

Now the same image is sent to the **NodeJS server** which is having the **Authentication ID**.

Here the engine that detects the Object or creates labels for the object(s) in the image frame is [Google Cloud Vision API](https://cloud.google.com/vision/).

The **NodeJS server** sends the image to the **Vision AI API**. But to interact with the API it needs some authentication which is done using the **Authentication ID**. Once the frame is sent, the API returns the labels to the server, and from the server, these labels are sent to the **ESP-CAM** and from there, labels are displayed on **TFT-Screen**.

#### Arduino IDE

: Introduction to Arduino IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with themThe open-source Arduino Software (IDE) makes it easy to write code and upload it to the board.

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Active development of the Arduino software is hosted by GitHub.

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output bythe Arduino Software (IDE), including complete error messages and other information.The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

: Steps in Arduino IDE

STEPS:



1. *New* Creates a new sketch.



1. *Open* Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

*Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the File* ***|*** *Sketchbook menu instead.*



1. *Save* Saves your sketch.



1. *Serial Monitor* Opens the [serial monitor](https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics#serialmonitor).

Additional commands are found within the five menus: File**,** Edit**,** Sketch**,** Tools**,** Help.The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

: Sketch

Verify/Compile Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.

Upload Compiles and loads the binary file onto the configured board through the configured Port.

Upload Using Programmer This will overwrite the bootloader on the board; you will need to use Tools > Burn Bootloader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash memory for your sketch. Please note that this command will NOT burn the fuses. To do so a Tools -> Burn Bootloader command must be executed.

Export Compiled Binary Saves a .hex file that may be kept as archive or sent to the board using other tools.

Show Sketch Folder Opens the current sketch folder.

Include Library Adds a library to your sketch by inserting #include statements at the start of your code. For more details, see [libraries](https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics#libraries) below. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.

Add File... Adds a supplemental file to the sketch (it will be copied from its current location).

The file is saved to the data

subfolder of the sketch, which is intended for assets such as documentation. The contents of the data folder are not compiled, so they do not become part of the sketch program.

: Tools

*Auto Format* This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.

*Archive Sketch* Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.

*Fix Encoding & Reload* Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.

*Serial Monitor* Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected Port. This usually resets the board, if the board supports Reset over serial port opening.

Board Select the board that you're using. See below for [descriptions of the various boards](https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics#boards).

Port This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.

Programmer For selecting a hardware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're [burning a bootloader](https://docs.arduino.cc/en/Tutorial/BuiltInExamples/ArduinoISP) to a new microcontroller, you will use this.

Burn Bootloader The items in this menu allow you to burn a [bootloader](https://docs.arduino.cc/hacking/software/Bootloader) onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino board but is useful if you purchase a new ATmega microcontroller (which normally come without a bootloader). Ensure that you've selected the correct board from the Boards menu before burning the bootloader on the target board. This command also set the right fuses.

: Libraries

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with yoursketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its **#**include statements from the top of your code.

There is a [list of libraries](https://docs.arduino.cc/software/ide-v1/tutorials/arduino.cc/en/Reference/Libraries) in the reference. Some libraries are included with the Arduino software.

: Uploading

Before uploading your sketch, you need to select the correct items from the **Tools >Board** and **Tools > Port** menus. The [boards](https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics#boards) are described below. On the Mac, the serial port is probably something like **/dev/tty.usbmodem241** (for an UNO or Mega2560 or Leonardo) or **/dev/tty.usbserial-1B1** (for a Duemilanove or earlier USB board), or

**/dev/tty.USA19QW1b1P1.1** (for a serial board connected with a Keyspan USB-to- Serial adapter). On Windows, it's probably **COM1** or **COM2** (for a serial board) or **COM4**, **COM5**, **COM7**, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be

**/dev/ttyACMx** , **/dev/ttyUSBx** or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the **Upload** item fromthe **Sketch** menu.

The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems suchas Windows, Mac OS X, and Linux. It supports the programming languages C and C++. Here, IDE stands for Integrated Development Environment.

The Arduino IDE will appear as:

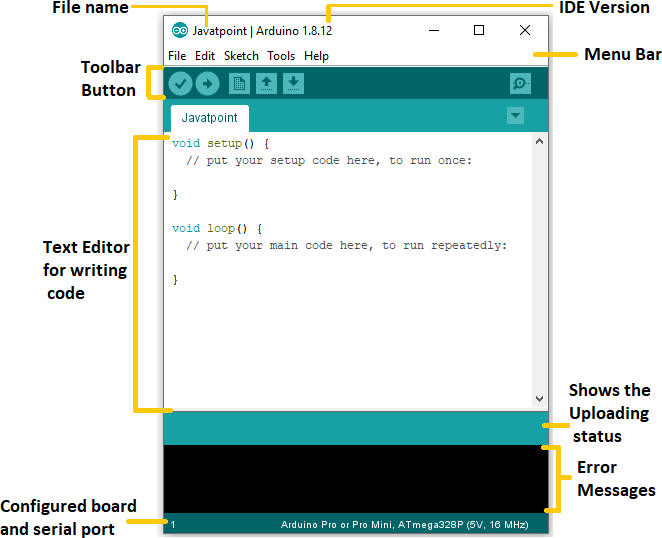


Fig. 5.1

: Toolbar Button

The icons displayed on the toolbar are New, Open, Save, Upload, and Verify. It is shown below:

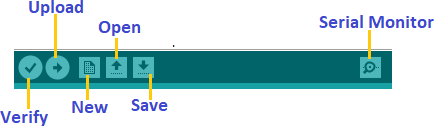


Fig. 5.2

Upload

The Upload button compiles and runs our code written on the screen. It further uploads the code to the connected board. Before uploading the sketch, we need to make sure that the correct board and ports are selected.

We also need a USB connection to connect the board and the computer. Once all the above measures are done, click on the Upload button present on the toolbar.

The latest Arduino boards can be reset automatically before beginning with Upload. In the older boards, we need to press the Reset button present on it. As soon as the uploading is done successfully, we can notice the blink of the Tx and Rx LED.

If the uploading is failed, it will display the message in the error window. We do not require any additional hardware to upload our sketch using the Arduino Bootloader. A Bootloader is defined as a small program, which is loaded in the microcontroller present on the board. The LED will blink on PIN 13.

Open

The Open button is used to open the already created file. The selected file will be opened in the current window.

Save

New

The save button is used to save the current sketch or code.

It is used to create a new sketch or opens a new window.

Verify

The Verify button is used to check the compilation error of the sketch or the written code.

: Serial Monitor

The serial monitor button is present on the right corner of the toolbar. It opens the serial monitor.

It is shown below:

Arduino IDE

When we connect the serial monitor, the board will reset on the operating system Windows, Linux, and Mac OS X. If we want to process the control characters in our sketch, we need to use an external terminal program. The terminal program should be connected to the COM port, which will be assigned when we connect the board to the computer.

Menu Bar

File

When we click on the File button on the Menu bar, a drop-down list will appear. It is shown below:

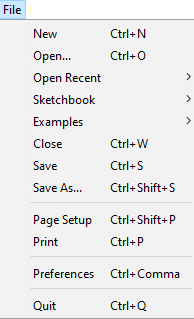


Fig. 5.3

Let's discuss each option in detail.

New

The New button opens the new window. It does not remove the sketch which is already present.

Open

It allows opening the sketch, which can be browsed from the folders and computer drivers.

Open Recent

The Open Recent button contains the list of the recent sketches.

Sketchbook

It stores the current sketches created in the Arduino IDE software. It opens the selected sketch or code in a new editor at an instance.

Examples

It shows the different examples of small projects for a better understanding of the IDE and the board. The IDE provides examples of self-practice.

Close

The Close button closes the window from which the button is clicked.

Save

The save button is used to save the current sketch. It also saves the changes made to the current sketch. If we have not specified the name of the file, it will open the '**SaveAs...'** window.

Save As...

We can save the sketch with a different name using the '**Save As...'** button. We can also change the name accordingly.

Page Setup

It allows setting the page margins, orientation, and size for printing. The '**Page Setup**' window will appear as:

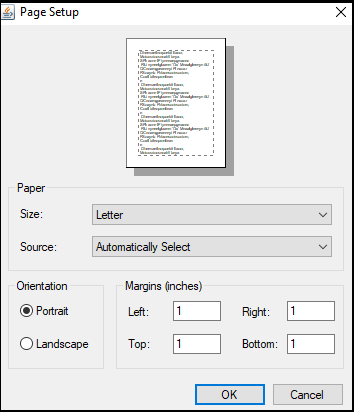


Fig. 5.4

Print

According to the settings specified in the 'Page Setup', it prepares the current sketch for printing.

Preferences

It allows the customization settings of the Arduino IDE.

Quit

The Quit button is used to close all the IDE windows. The same closed sketch will be reopened when we will open the Arduino IDE.

Edit

When we click on the Edit button on the Menu bar, a drop-down list appears. It is shown below:

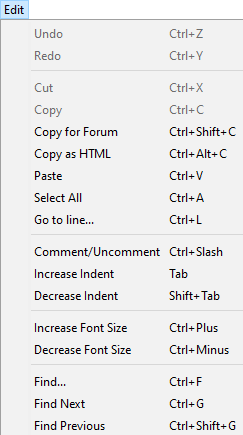


Fig. 5.5

Let's discuss each option in detail.

Undo

The Undo button is used to reverse the last modification done to the sketch while editing.

Redo

The Redo button is used to repeat the last modification done to the sketch while editing.

Cut

It allows us to remove the selected text from the written code. The text is further placed to the clipboard. We can also paste that text anywhere in our sketch.

Copy

It creates a duplicate copy of the selected text. The text is further placed on the clipboard.

Copy for Forum

The 'Copy for Forum' button is used to copy the selected text to the clipboard, which is also suitable for posting to the forum.

Copy as HTML

The 'Copy for Forum' button is used to copy the selected text as HTML to the clipboard. It is desirable for embedding in web pages.

Paste

The Paste button is used to paste the selected text of the clipboard to the specified position of the cursor.

Select All

It selects all the text of the sketch.

Go to line...

It moves the cursor to the specified line number. The window will appear as:

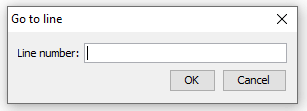


Fig. 5.6

Comment/Decomment

The Comment/ Decomment button is used to put or remove the comment mark (**//**) at the beginning of the specified line.

Increase Indent

It is used to add the space at the starting of the specified line. The spacing moves the text towards the right.

Decrease Indent

It is used to subtract or remove the space at the starting of the specified line. The spacing moves the text towards the left.

Increase Font Size

It increases the font size of the written text.

Decrease Font Size

It decreases the font size of the written text.

Find...

It is used to find the specified text. We can also replace the text. It highlights the text in the sketch.

The window will appear as:

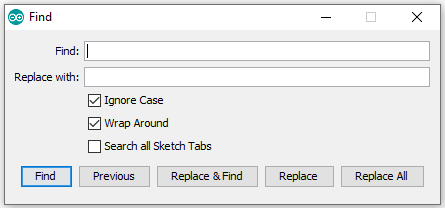


Fig. 5.7

Find Next

It highlights the next word, which has specified in the '**Find...'** window. If there is no such word, it will not show any highlighted text.

Find Previous

It highlights the previous word, which has specified in the '**Find...'** window. If there is no such word, it will not show any highlighted text.

Sketch

When we click on the Sketch button on the Menu bar, a drop-down list appears. It is shown below:

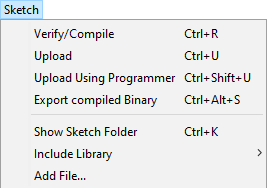


Fig. 5.8

Let's discuss each option in detail.

Verify/Compile

It will check for the errors in the code while compiling. The memory in the console area is also reported by the IDE.

Upload

The Upload button is used to configure the code to the specified board through the port.

Upload Using Programmer

It is used to override the Bootloader that is present on the board. We can utilize the full capacity of the Flash memory using the '**Upload Using Programmer**' option. To implement this, we need to restore the Bootloader using the **Tools**-> **Burn Bootloader** option to upload it to the USB serial port.

Export compiled Binary

It allows saving a .**hex** file and can be kept archived. Using other tools, .hex file can also be sent to the board.

Show Sketch Folder

It opens the folder of the current code written or sketch.

Include Library

Include Library includes various Arduino libraries. The libraries are inserted into our code at the beginning of the code starting with the #. We can also import the libraries from .zip file.

Add File...

The Add File... button is used to add the created file in a new tab on the existing file. For example, let's add '**Blink**' file to the '**JavatPoint**' file. The tab will now appear as:

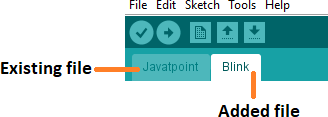


Fig. 5.9

We can also delete the corresponding file from the tab by clicking on the **small triangle** -

> **Delete** option.

: Tools

When we click on the Tools button on the Menu bar, a drop-down list appears. It is shown below:

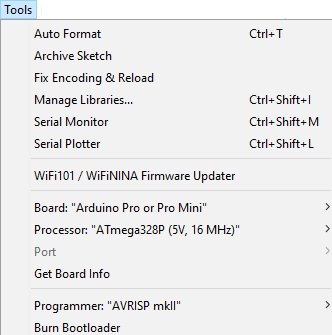


Fig. 5.10

Let's discuss each option in detail.

Auto Format

The Auto Format button is used to format the written code. For example, lining the open and closed curly brackets in the code.

Archive Sketch

The copy of the current sketch or code is archived in the .zip format. The directory of the archived is same as the sketch.

Fix Encoding and Reload

This button is used to fix the inconsistency between the operating system char maps and editor char map encoding.

Manage Libraries...

It shows the updated list of all the installed libraries. We can also use this option to install a new library into the Arduino IDE.

Serial Monitor

It allows the exchange of data with the connected board on the port.

Serial Plotter

The Serial Plotter button is used to display the serial data in a plot. It comes preinstalled in the Arduino IDE.

WiFi101/WiFiNINA Firmware Updater

It is used to check and update the Wi-Fi Firmware of the connected board.

Board

We are required to select the board from the list of boards. The selected board must be similar to the board connected to the computer.

Processor

It displays the processor according to the selected board. It refreshes every time during the selection of the board.

Port

It consists of the virtual and real serial devices present on our machine.

Programmer

We need to select the hardware programmer while programming the board. It is required when we are not using the onboard USB serial connection. It is also required during the burning of the Bootloader.

Burn Bootloader

The Bootloader is present on the board onto the microcontroller. The option is useful when we have purchased the microcontroller without the bootloader. Before burning the bootloader, we need to make sure about the correct selected board and port.

5:6: Help

When we click on the Help button on the Menu bar, a drop-down list will appear. It is shown below:

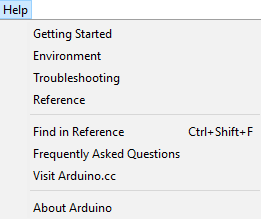


Fig. 5.11

The Help section includes several documents that are easy to access, which comes along with the Arduino IDE. It consists of the number of options such as Getting Started, Environment, Troubleshooting, Reference, etc. We can also consider the image shown above, which includes all the options under the Help section.

Some documents like Getting started, Reference, etc., can be accessed without the internet connection as well. It will directly link us to the official website of Arduino.



Fig.5

# Arduino Libraries Installation

Now in order to the use **TFT screen** and read the data from the server we require a few libraries which can be installed using the **Arduino library manager**. To open Library manager press *Ctrl+shift+I*, it might take a few seconds to open according to the system specifications. Now in the search bar type the name of libraries and **install** them.

1. **TFT\_eSPI** by Bodmer: <https://github.com/Bodmer/TFT_eSPI>
2. **TJpg\_Decoder** by Bodmer: <https://github.com/Bodmer/TJpg_Decoder>
3. **ArduinoJson** by Benoit Blanchon: <https://github.com/bblanchon/ArduinoJson>



# Code for TFT Display Test

Now we need to first display the frames captured by the camera on the TFT screen. To do so, upload the code below once the libraries are installed.

| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54 | #include "esp\_camera.h" #include <TJpg\_Decoder.h> #include <SPI.h>  #include <TFT\_eSPI.h>  #define PWDN\_GPIO\_NUM 32  #define RESET\_GPIO\_NUM -1  #define XCLK\_GPIO\_NUM 0  #define SIOD\_GPIO\_NUM 26  #define SIOC\_GPIO\_NUM 27  #define Y9\_GPIO\_NUM 35  #define Y8\_GPIO\_NUM 34  #define Y7\_GPIO\_NUM 39  #define Y6\_GPIO\_NUM 36  #define Y5\_GPIO\_NUM 21  #define Y4\_GPIO\_NUM 19  #define Y3\_GPIO\_NUM 18  #define Y2\_GPIO\_NUM 5  #define VSYNC\_GPIO\_NUM 25  #define HREF\_GPIO\_NUM 23  #define PCLK\_GPIO\_NUM 22  #define GFXFF 1  #define FSB9 &FreeSerifBold9pt7b TFT\_eSPI tft = TFT\_eSPI();  bool tft\_output(int16\_t x, int16\_t y, uint16\_t w, uint16\_t h, uint16\_t\* bitmap)  {  // Stop further decoding as image is running off bottom of screen if ( y >= tft.height() ) return 0;  // This function will clip the image block rendering automatically at the TFT boundaries tft.pushImage(x, y, w, h, bitmap);  // This might work instead if you adapt the sketch to use the Adafruit\_GFX library  // tft.drawRGBBitmap(x, y, bitmap, w, h);  // Return return 1;  }  void setup() { Serial.begin(115200); delay(1000);  Serial.println();  Serial.println("INIT DISPLAY"); tft.begin();  tft.setRotation(3); tft.setTextColor(0xFFFF, 0x0000); tft.fillScreen(TFT\_CYAN); |
| --- | --- |

| 55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115 | tft.setFreeFont(FSB9);  TJpgDec.setJpgScale(1); TJpgDec.setSwapBytes(true); TJpgDec.setCallback(tft\_output);  Serial.println("INIT CAMERA"); 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; 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 = 10000000; config.pixel\_format = PIXFORMAT\_JPEG;  //init with high specs to pre-allocate larger buffers if(psramFound()){  config.frame\_size = FRAMESIZE\_QVGA; // 320x240 config.jpeg\_quality = 10;  config.fb\_count = 2;  } else {  config.frame\_size = FRAMESIZE\_SVGA; config.jpeg\_quality = 12;  config.fb\_count = 1;  }  // 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;  }  }  camera\_fb\_t\* capture(){ camera\_fb\_t \*fb = NULL; esp\_err\_t res = ESP\_OK; fb = esp\_camera\_fb\_get(); return fb;  }  void showingImage(){ camera\_fb\_t \*fb = capture();  if(!fb || fb->format != PIXFORMAT\_JPEG){ Serial.println("Camera capture failed"); esp\_camera\_fb\_return(fb);  return; |
| --- | --- |

| 116  117  118  119  120  121  122  123  124 | }else{  TJpgDec.drawJpg(0,0,(const uint8\_t\*)fb->buf, fb->len); esp\_camera\_fb\_return(fb);  }  }  void loop() { showingImage();  } |
| --- | --- |

Make sure the IO0 pin is grounded during the uploading of the code. And if Connecting…. Text appears on the log screen followed by “……. …….\_ ” just press the Reset Button.

Now the image on the screen must be visible.



# Google Vision API

**Vision API** offers powerful pre-trained machine learning models

through **REST** and **RPC APIs**. Assign labels to images and quickly classify them into millions of predefined categories. Detect objects and faces, read printed and handwritten text, and build valuable **metadata** into your **image catalog**.

It also allows developers to easily integrate **vision detection** features within applications, including image labeling, face, and landmark detection, optical character recognition (OCR), and tagging of explicit content.



# NodeJS Installation

Now in order to create a server we will be using **NodeJS**. To Install latest version of

NodeJS from [**nodejs.org**](https://nodejs.org/en/download/).

From here download the compatible version of the software, according to your system. Make sure that you have downloaded LTS Version.



# GCP and API Setup

Before the whole setup makes sure you have added your credit/debit card in GCP, although the project won’t cost even a single penny. Still, because we are using GCP features they require a billing address.

At first, go to Google Cloud Platform(GCP) through the link [here](https://console.cloud.google.com/home/dashboard). And Create a new

project.

* Here click on New Project
* Add a name to the project and organization can be kept as No Organization

Once the Billing is Enabled and Project is Created. Now we must enable the Vision API for your project. To do so go to this link [here](https://console.cloud.google.com/flows/enableapi). And enable the API.

Now we have to create a service account for authentication. Go to this [link](https://console.cloud.google.com/projectselector/iam-admin/serviceaccounts/create).

* Select the project created
* Add a name to service account as well as ID(we don’t have to provide the full ID just a name, although the service ID is automatically generated)
* Now click on Create and Continue
* Click the Select a role field.
* Under Quick access, click Basic, then click Owner.
* Click on continue
* Now click on Done

Now we create a service account key:

1. In the Cloud Console, click the email address for the service account that you created.
2. Click **Keys**.
3. Click **Add key**, then click **Create new key**.
4. Click **Create**. A JSON key file is downloaded to your computer.
5. Click **Close**.

Now open a Command prompt and run the command:

First, locate yourself to a directory where you wish to save the project and create a server. Now open the command prompt at that location.

* *npm install –save @google-cloud/vision*
* *set GOOGLE\_APPLICATION\_CREDENTIALS=KEY\_PATH*

For example:

set GOOGLE\_APPLICATION\_CREDENTIALS=”*/home/user/Downloads/service- account-file.json*”

* Now in the same directory create a new file with “.js” extension(say test.js)
* Also create a folder “resources” and in that add an image(that you wish to test for object detection) with the name “test.jpg”
* Now open test.js and paste the code below

| 1  2  3  4 | ‘use strict’;  function main() {  // [START vision\_quickstart] |
| --- | --- |

| 5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | async function quickstart() {  // Imports the Google Cloud client library  const vision = require(‘@google-cloud/vision’);  // Creates a client  const client = new vision.ImageAnnotatorClient();  // Performs label detection on the image file  const [result] = await client.labelDetection(‘./resources/test.jpg’); const labels = result.labelAnnotations;  console.log(‘Labels:’);  labels.forEach(label =>{ console.log(label)});  }  quickstart();  // [END vision\_quickstart]  }  process.on(‘unhandledRejection’, err => { console.error(err.message); process.exitCode = 1;  });  main(…process.argv.slice(2)); |
| --- | --- |

* Now run the above code by writing “node test.js” in the command prompt (make sure you are in the right directory)
* You will see the labels printed in the log that is detected in the test.jpg image Congratulations our major work is done, now we have to do the same using server as esp- cam
* Create a new folder with the name VisionServer, in that create another folder named resources and also a file named server.js
* Open server.js and paste the code below

| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | var fs = require('fs');  const http = require('http');  const server = http.createServer(); const filePath = './resources/test.jpeg';  server.on('request', (request, response)=>{  if(request.method == 'POST' && request.url === "/imageUpdate"){  var ImageFile = fs.createWriteStream(filePath, {encoding: 'utf8'}); request.on('data', function(data){  ImageFile.write(data);  });  request.on('end',async function(){ ImageFile.end();  const labels = await labelAPI(); |
| --- | --- |

| 17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48 | response.writeHead(200, {'Content-Type' : 'application/json'});  response.end(JSON.stringify(labels));  });  }else{  console.log("error");  response.writeHead(405, {'Content-Type' : 'text/plain'}); response.end();  }  });  async function labelAPI() { var o = [];  // Imports the Google Cloud client library const vision = require('@google-cloud/vision');  // Creates a client  const client = new vision.ImageAnnotatorClient();  // Performs label detection on the image file  const [result] = await client.labelDetection(filePath); const labels = result.labelAnnotations;  labels.forEach(label => {  o.push({description: label.description, score: label.score});  });  return o;  }  const port = 8888; server.listen(port) console.log(`Listening at ${port}`) |
| --- | --- |

* Save the code and run it by opening the command prompt and go to the right directory and write ”node server.js”

#### Jumper Wires

: Using Jumper Wires

A **jump wire** (also known as **jumper**, **jumper wire**, **DuPont wire**) is an [electrical wire,](https://en.wikipedia.org/wiki/Electrical_wire) or group of them in a cable, with a connector or pin at each end (or sometimes without them –

simply "tinned"), which is normally used to interconnect the components of a [breadboard](https://en.wikipedia.org/wiki/Breadboard) or other prototype or test circuit, internally or with other equipment or components, without soldering.[[1]](https://en.wikipedia.org/wiki/Jump_wire#cite_note-1)

Individual jump wires are fitted by inserting their "end connectors" into the slots provided in a breadboard, the [header connector](https://en.wikipedia.org/wiki/Pin_header#Header_connector) of a circuit board, or a piece of test equipment.

Jumper wires typically come in three versions: male-to-male, male-to-female and female-to- female. The difference between each is in the end point of the wire. Male ends have a pin protruding and can plug into things, while female ends do not and are used to plug things into

**How to use Jumper wires?**

Step 1

Follow all electrical safety precautions. You must take more care when you're dealing with high-voltage circuits than when you're dealing with low-voltage circuits (such as certain battery-powered electronic devices).

Step 2

Determine the appropriate type of connector for the jumper wire. In some cases (especially if you have steady hands), you can simply use the bare wire ends to connect the two points of a circuit. If you need your hands for other tasks, however, then you need to decide if the task requires alligator clips, soldering or some other type of connection for the jumper wire.

Step 3

Use the jumper wire to bypass a portion of the circuit or to otherwise establish a connection between two points. This use can be helpful in determining if part of the circuit is faulty or if the connection in the circuit is broken.

Step 4

Connect the jumper wire (or two jumper wires) to an ammeter or voltmeter (or multimeter) to take a numerical measurement of a circuit property. This allows you to measure the properties of the circuit between or across two points.

Step 5

Select the appropriate type of jumper wire if you're making a permanent modification to a circuit. The gauge and type of insulation are critical considerations.

Step 6

Test circuit modifications with a temporary jumper wire connection before making a permanent connection (with solder, for instance). Check the performance of the circuit with the temporary connection to ensure that it works as you expect.

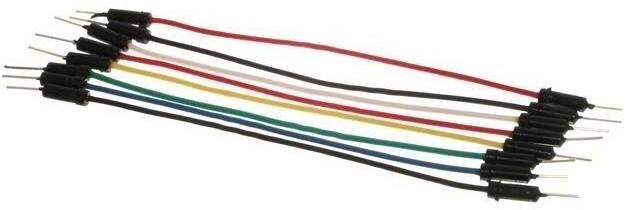
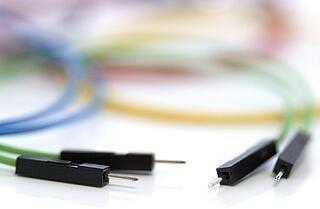
: Colors in Jumper Wires

Fig. 6.1

Though jumper wires come in a variety of colors, the colors don’t actually mean anything. This means that a red jumper wire is technically the same as a black one. But the colors can be used to your advantage in order to differentiate between types of connections, such as ground or power.

: Making Jumper Wires

While jumper wires are easy and inexpensive to purchase, it can also be a fun taskto [challenge](http://www.dummies.com/programming/electronics/how-to-make-jumper-wires/) [students to make their own.](http://www.dummies.com/programming/electronics/how-to-make-jumper-wires/) Doing so requires insulated wire and wire strippers.However, beware that it is important not to nick the wire when stripping off the insulation.

: Types of Jumper Wires

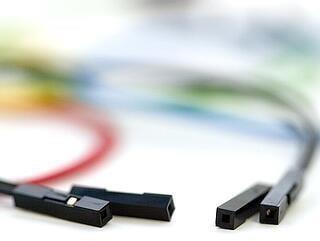


Fig. 6.2

Jumper wires typically come in three versions: male-to-male, male-to-female and female-to-female. The difference between each is in the end point of the wire. Male ends have a pin protruding and can plug into things, while female ends do not and are used to plug things into. Male-to-male jumper wires are the most common and what you likely will use most often. When connecting two ports on a breadboard, a male-to-male wire is what you’ll need.

CONCLUSION

The paper reviews intelligent surveillance video analysis techniques. Reviewed papers cover wide variety of applications. The techniques, tools and dataset identified were listed in form of tables. Survey begins with video surveillance analysis in general perspective, and then finally moves towards crowd analysis. Crowd analysis is difficult in such a way that crowd size is large and dynamic in real world scenarios. Identifying each entity and their behavior is a difficult task. Methods analyzing crowd behavior were discussed. The issues identified in existing methods were listed as future directions to provide efficient solution.

The IoT has the potential to dramatically increase the availability of information, and is likely to transform companies and organizations in virtually every industry around the world. As such, finding ways to leverage the power of the IoT is expected to factorinto the strategic objectives of most technology companies, regardless of their industryfocus.

The number of different technologies required to support the deployment and further growth of the IoT places a premium on interoperability, and has resulted in widespreadefforts to develop standards and technical specifications that support seamless communication between IoT devices and components.

Collaboration between various standards development groups and consolidation of some current efforts will eventuallyresult in greater clarity for IoT technology companies.



# Final Arduino Code

Here is the final **Arduino Code** for **AI Camera with Google Vision & ESP32 CAM Module**. So, open Arduino IDE & paste the code below by doing a few changes.

First add an SSID and password of the network with which your laptop is connected.

| 1  2 | const char\* ssid = "\*\*\*\*\*\*\*\*\*\*";  const char\* password = "\*\*\*\*\*\*\*\*\*\*"; |
| --- | --- |

Now we have to update the IP address of the server, in this case, our computer is our server thus updating its IP address as in the following line of code.

| 1 | client.begin("http://192.168.116.56:8888/imageUpdate"); |
| --- | --- |

Here is the final code that you have to upload to the ESP32 CAM Board.

| #include "esp\_camera.h" |  |
| --- | --- |
| #include <TJpg\_Decoder.h> |
| #include <SPI.h> |
| #include <TFT\_eSPI.h> |
| #include <WiFi.h> |
| #include <HTTPClient.h> |
| #include <ArduinoJson.h> |
| #define PWDN\_GPIO\_NUM | 32 |
| #define RESET\_GPIO\_NUM | -1 |
| #define XCLK\_GPIO\_NUM | 0 |
| #define SIOD\_GPIO\_NUM | 26 |
| #define SIOC\_GPIO\_NUM | 27 |
| #define Y9\_GPIO\_NUM | 35 |
| #define Y8\_GPIO\_NUM | 34 |
| #define Y7\_GPIO\_NUM | 39 |
| #define Y6\_GPIO\_NUM | 36 |
| #define Y5\_GPIO\_NUM | 21 |
| #define Y4\_GPIO\_NUM | 19 |
| #define Y3\_GPIO\_NUM | 18 |
| #define Y2\_GPIO\_NUM | 5 |

#define VSYNC\_GPIO\_NUM 25

#define HREF\_GPIO\_NUM 23

#define PCLK\_GPIO\_NUM 22

#define GFXFF 1

#define FSB9 &FreeSerifBold9pt7b TFT\_eSPI tft = TFT\_eSPI();

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

const unsigned long timeout = 30000; // 30 seconds

int buttonState;

int lastButtonState = LOW;

unsigned long lastDebounceTime = 0; // the last time the output pin was toggled unsigned long debounceDelay = 50; // the debounce time; increase if the output flickers bool isNormalMode = true;

bool tft\_output(int16\_t x, int16\_t y, uint16\_t w, uint16\_t h, uint16\_t\* bitmap)

{

// Stop further decoding as image is running off bottom of screen if ( y >= tft.height() ) return 0;

// This function will clip the image block rendering automatically at the TFT boundaries tft.pushImage(x, y, w, h, bitmap);

// This might work instead if you adapt the sketch to use the Adafruit\_GFX library

// tft.drawRGBBitmap(x, y, bitmap, w, h);

// Return return 1;

}

void setup() { Serial.begin(115200); delay(1000);

Serial.println(); pinMode(buttonPin, INPUT);

Serial.println("INIT DISPLAY"); tft.begin();

tft.setRotation(3); tft.setTextColor(0xFFFF, 0x0000); tft.fillScreen(TFT\_YELLOW); tft.setFreeFont(FSB9);

TJpgDec.setJpgScale(1); TJpgDec.setSwapBytes(true); TJpgDec.setCallback(tft\_output);

Serial.println("INIT CAMERA"); 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; 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 = 10000000; config.pixel\_format = PIXFORMAT\_JPEG;

//init with high specs to pre-allocate larger buffers if(psramFound()){

config.jpeg\_quality = 10;

config.fb\_count = 2;

} else {

config.frame\_size = FRAMESIZE\_SVGA; config.jpeg\_quality = 12;

config.fb\_count = 1;

}

// 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;

}

}

bool wifiConnect(){

unsigned long startingTime = millis(); WiFi.begin(ssid, password);

while(WiFi.status() != WL\_CONNECTED){ delay(500);

if((millis() - startingTime) > timeout){ return false;

}

}

return true;

}

void buttonEvent(){

int reading = digitalRead(buttonPin); if (reading != lastButtonState) { lastDebounceTime = millis();

}

if ((millis() - lastDebounceTime) > debounceDelay) { if (reading != buttonState) {

buttonState = reading;

if (buttonState == HIGH) { isNormalMode = !isNormalMode;

//Additional Code if(!isNormalMode) sendingImage();

//

}

}

}

lastButtonState = reading;

}

camera\_fb\_t\* capture(){ camera\_fb\_t \*fb = NULL; esp\_err\_t res = ESP\_OK; fb = esp\_camera\_fb\_get(); return fb;

}

void showingImage(){ camera\_fb\_t \*fb = capture();

if(!fb || fb->format != PIXFORMAT\_JPEG){ Serial.println("Camera capture failed"); esp\_camera\_fb\_return(fb);

return;

}else{

[lxv

TJpgDec.drawJpg(0,0,(const uint8\_t\*)fb->buf, fb->len); esp\_camera\_fb\_return(fb);

}

}

void parsingResult(String response){ DynamicJsonDocument doc(1024); deserializeJson(doc, response); JsonArray array = doc.as<JsonArray>(); int yPos = 4;

//tft.setRotation(1); for(JsonVariant v : array){

JsonObject object = v.as<JsonObject>();

const char\* description = object["description"]; float score = object["score"];

String label = ""; label += description; label += ":";

label += score;

tft.drawString(label, 8, yPos, GFXFF); yPos += 16;

}

//tft.setRotation(3);

}

void postingImage(camera\_fb\_t \*fb){ HTTPClient client;

client.begin("http://192.168.116.56:8888/imageUpdate"); client.addHeader("Content-Type", "image/jpeg");

int httpResponseCode = client.POST(fb->buf, fb->len); if(httpResponseCode == 200){

String response = client.getString(); parsingResult(response);

}else{

//tft.setRotation(1);

//Error

tft.drawString("Check Your Server!!!", 8, 4, GFXFF);

//tft.setRotation(3);

}

client.end(); WiFi.disconnect();

}

void sendingImage(){ camera\_fb\_t \*fb = capture();

if(!fb || fb->format != PIXFORMAT\_JPEG){ Serial.println("Camera capture failed"); esp\_camera\_fb\_return(fb);

return;

}else{

TJpgDec.drawJpg(0,0,(const uint8\_t\*)fb->buf, fb->len);

//tft.setRotation(1);

tft.drawString("Wifi Connecting!", 8, 4, GFXFF);

//tft.setRotation(3); if(wifiConnect()){

//tft.drawString("Wifi Connected!", 8, 4, GFXFF); TJpgDec.drawJpg(0,0,(const uint8\_t\*)fb->buf, fb->len); postingImage(fb);

}else{

//tft.setRotation(1);

tft.drawString("Check Wifi credential!", 8, 4, GFXFF);

//tft.setRotation(3);

}

}

}

void loop() { buttonEvent();

if(isNormalMode) showingImage();

}

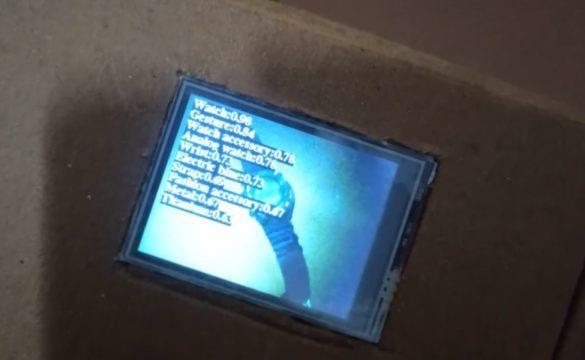


# Testing DIY AI Camera using Google Vision & ESP32 CAM

Now when you press the **button** for 1 second the flash will appear and the image is **clicked**.



And after a few seconds the labels appear, congratulations the **AI camera** started working. Here are the samples of other images taken with this DIY AI Camera.





CONCLUSION

The paper reviews intelligent surveillance video analysis techniques. Reviewed papers cover wide variety of applications. The techniques, tools and dataset identified were listed in form of tables. Survey begins with video surveillance analysis in general perspective, and then finally moves towards crowd analysis. Crowd analysis is difficult in such a way that crowd size is large and dynamic in real world scenarios. Identifying each entity and their behavior is a difficult task. Methods analyzing crowd behavior were discussed. The issues identified in existing methods were listed as future directions to provide efficient solution.

The IoT has the potential to dramatically increase the availability of information, and is likely to transform companies and organizations in virtually every industry around the world. As such, finding ways to leverage the power of the IoT is expected to factorinto the strategic objectives of most technology companies, regardless of their industryfocus.

The number of different technologies required to support the deployment and further growth of the IoT places a premium on interoperability, and has resulted in widespreadefforts to develop standards and technical specifications that support seamless communication between IoT devices and components.

Collaboration between various standards development groups and consolidation of some current efforts will eventuallyresult in greater clarity for IoT technology companies.

**REFERENCES**

<https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics>

<https://www.javatpoint.com/arduino-ide> <https://www.hunker.com/13414381/how-to-use-electrical-jumper-wires>

https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition- arduino-ide/