CHAPTER 1
INTRODUCTION

1.1 GENERAL

The Embedded Technology is now in its prime and the wealth of knowledge available is mind blowing. Embedded System is a combination of hardware and Software. Embedded technology plays a major role in integrating the various functions associated with it. This needs to tie up the various sources of the Department in a closed loop system. This proposal greatly reduces the man power, saves time and operates efficiently without human interference. This project puts forth the first step in achieving the desired target. With the advent in technology, the existing systems are developed to have in built intelligence.

1.2 OBJECTIVE

Prevent unauthorized access and restrict access to sensitive areas of your organization, while collecting and analyzing traffic intelligence data with our Access Control systems. We afford you enterprise-wide safety and security, via **Intelligent Access Control System** and a safe environment through **Video Monitoring** for employees, visitors, staff, amenities and assets. This includes Safety Control Access for employee, as well as management of critical information and staff deployment in emergency situations.

Industrial safety refers to the protection of workers from the danger of industrial accidents. Our Intelligent Access Control systems. Protecting your employees and business against unsafely.

1.3 EXISTING SYSTEM

In the existing system, now a day there employee can enter inside the industry or factory with taking the primary precautions.

1.4 DISADVANTAGE OF EXISTING SYSTEM

The disadvantage of the existing system are , there is no proper restrictions to enter the industry.

1.5 PROPOSED SYSTEM

Here the proposed system, we have designed the system that can monitor the employee with safety and necessary equipment can enter inside the factory/Industry. So that employee can be safety refers to the protection of workers from the danger main priority.

1.6 ADVANTAGES OF PROPOSED SYSTEM

The Advantage of proposed system are

- Simplicity, mobility and low price.
- This system has the employee can be safety refers to the protection of workers from the danger main priority.

CHAPTER 2
PROJECT DISCRIPTION

2.1 Introduction to IOT

Internet of Things(IoT) comprises things that have unique identities and are connected to the Internet. While many existing devices, such as networked computers or 4G-enabled mobile phones, already have some form of unique identities and are also connected to the Internet, the focus on IoT is in the configuration, control and networking via the Internet of devices or "things" that are traditionally not associated with the internet. These include devices such as thermostats, utility meters, a Bluetooth connected headset, irrigation pumps and sensors, or control-circuits for an electric car's engine. Internet of Things is anew revolution in the capabilities of the end points that are connected to the Internet, and is driven by the advancement in capabilities(in combination with lower costs) in sensor networks, mobile devices, wireless communications, networking and cloud technologies. Experts forecast that by the year 2020 there will be a total of 50 billion devices/ things connected to the Internet. Therefore, the major industry players are excited by the prospects of new markets for their products. The products include hardware and software components for IoT endpoints, hubs, or control centers of the IoT universe.

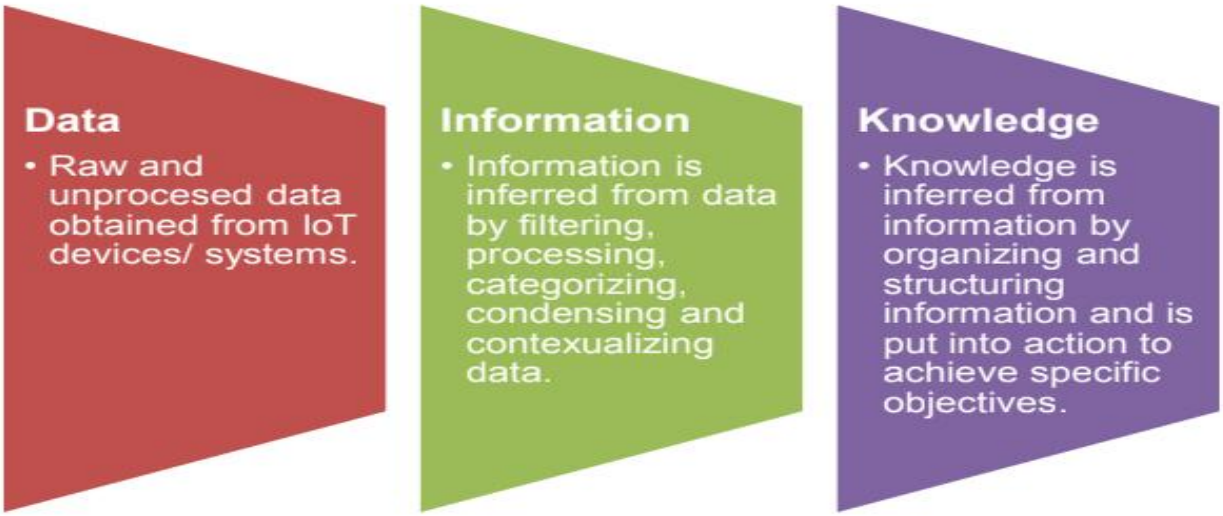

Figure 1: Inferring information and knowledge from data

The scope of IoT is not limited to just connecting things (devices, appliances, machines) to the Internet. IoT allows these things to communicate and exchange data (Control& information, that could include data associated with users) while executing meaningful application towards a common user or machine goal. Data itself does not have a meaning until it is contextualized processed into useful information. Applications on IoT networks extract and create information from lower level data by filtering, processing, categorizing, condensing and contextualizing the

data. This information obtained is then organized and structured to infer knowledge about the system and/or its users, its environment, and its operations and progress towards its objectives, allowing a smarter performance, as shown in Figure 1.1.For example, consider a series of raw sensor measurements ((72,45):(84,56)) generated by a weather monitoring station, which by themselves do not have any meaning or context. To give meaning to the data .a context is added . Which in this example can be that each tuple in data represents the temperature and humidity measured every minute .With this context added we know the meaning(or information)of the measured data tuples. Further information is obtained by categorizing .considering or processing this data .For example the average temperature and humidity readings for last five minutes is obtained by averaging the last five data tuples .The next step is to organize the information and understand the relationships between pieces of information to infer knowledge which can be put into action .For example, an alert is raised if the average temperature in last five minutes exceeds 120F,and this alert may be conditioned on the user's geographical
position as well.

The applications of Internet of Things span a wide range of domains including (but not limited to) homes, cities, environment, energy, systems, retail, logistics, industry, agriculture and health as listed .For homes ,IOT has several applications such as smart lighting that adapt the lighting to suit the ambient conditions .smart appliances that can be remotely monitored and controlled, intrusion detection systems ,smart smoke detectors ,etc. For cities, IOT has applications such as smart parking systems that provide status updates on available slots ,smart lighting that helps in saving energy ,smart roads that provide information on driving conditions and structural health monitoring systems .For environment ,IOT has applications such as weather monitoring , air and noise pollution , forest fire detection and river flood detection systems . For energy systems, IOT has applications such as including smart grids , grid integration of renewable energy sources and prognostic health management systems . For retail domain, IOT has applications such as inventory management , smart payments and smart vending machines. For agriculture domain, IOT has applications such as smart irrigation systems that help in saving water while enhancing productivity and green house control systems . Industrial applications such as smart irrigation systems. Industrial applications of IOT include machine diagnostics and prognosis systems. For health and lifestyle, IOT has applications such as health and fitness monitoring systems and wearable electronics.

2.1.1 Definition & Characteristics of IOT

Definition: A dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual " thing" have identities, physical attributes, and virtual personalities and use intelligent interfaces ,and are seamlessly integrated into the information network , often communicate data associated with users and their environments.

Let us examine this definition of IOT further to put some of the terms into perspective. Dynamic & Self-Adapting: IOT devices and systems may have the capability to

dynamically adapt with the changing contexts and take actions based on their operating conditions . user's context ,or sensed environment. For example, consider surveillance system compromising of a number of surveillance cameras. The surveillance cameras day or night. Cameras could switch from lower resolution modes when any motion is detected and alert nearby cameras to do the same. In this example, the surveillance system is adapting itself based on the context and changing(e.g., dynamic)conditions. Self-Configuring: IOT devices may have self-configuring capability, allowing a large number of devices to work together to provide certain functionality(such as weather monitoring). These devices have the ability configure themselves(in association with the IOT infrastructure), setup the networking, and fetch latest software upgrades with minimal manual or user intervention. Interoperable Communication Protocols: IOT devices may support a number of interoperable communication protocols and communicate with other devices and also with the infrastructure. We describe some of the commonly used communication protocols and models in later sections.

Unique Identity: Each IOT devices has a unique identity and a unique identifier (Such as an IP address or a URI). IOT systems ,may have intelligent interfaces which adapt based on the context, allow communicating with users and the environmental contexts. IOT device interfaces allow users to query the devices, monitor their status, and control them remotely, in association with the control, configuration and management infrastructure. Integrated into Information Network: IOT devices are usually integrated into the information network that allows them to communicate and exchange data with other devices and systems. IOT devices can be dynamically discovered in the network, by other devices and/or the network, and have the capability discovered in the network, and have the capability to describe themselves(and their devices or user applications. For example, a weather monitoring node can describe its monitoring capabilities to another connected node so that they can describe its monitoring capabilities to another connected node so that they can communicate and exchange data. Integration in to the information network helps in making IOT systems" smarter" due to the collective intelligence of the individual devices in collaboration with the infrastructure. thus , the data from a large number of connected weather monitoring IOT nodes can be aggregated and *analyzed to predict the wealth.

2.2 Physical Design of IOT

2.2.1 Things in IOT

The "Things" in IOT usually refers to IOT devices which have unique identities and can perform remote sensing, actuating and monitoring capabilities. IOT devices can exchange data with other connected devices and applications (directly or indirectly), or collect data from other devices and process the data either locally or send the data to centralized servers or cloud-based

application back-ends for processing the data, or perform some tasks locally and other tasks within the IOT infrastructure, based on temporal and space constraints (i.e., memory, processing capabilities, communication latencies and speeds, and deadlines). Figure1.3 shows a block diagram of a typical IOT device. An IOT device may consist of several interfaces for connections to other devices, both wired and wireless. these include (i) I/O interfaces for sensors, (ii) interfaces for internet connectivity, (iii) memory and storage interfaces and (iv) audio/video interfaces. An IOT device can collect various types of data from the on-board or attached sensors, such as temperature, humidity, light intensity. The sensed data can be connected actuators that allow them to interact with other physical entities (including non-IOT devices and systems) in the vicinity of the device. For example, a relay switch connected to an IOT device can turn an appliance on/off based on the commands sent to the IOT device over the Internet. IOT device can also be varied types, for instance, wearable sensors, smart watches, LED lights, auto mobiles and industrial machines, Almost all IOT devices generate data in some form or the other which when processed by data analytics systems leads to useful information to guide further actions locally or remotely. For instance, sensor data generated by a soil moisture monitoring device in a garden, when processed can help in determining the optimum watering schedules.

2.2.2 IOT Protocols :

Link Layer Link Layer protocols determine how the data is physically sent over the network's physical layer or medium(e. g., copper wire, coaxial a cable, or a radio wave). The scope of the link layer is the network connection to which host is attached. Hosts on the same link exchange data packets over the link layer using link layer protocols. Link layer determines how the packets are coded and signaled by the hardware device over the medium to which the host is attached (such as a coaxial cable). Let us now look at some link layer protocols which are relevant in the context of IOT.

• 802.3-ETHERNET: IEEE 802.3 is a collection of wired Ethernet standards for the link layer. For example, 802.3 is the standard for 10BASIES Ethernet that uses coaxial cable as a shared medium, 802.3.i is the standard for 10BASE-T Ethernet over copper twisted-pair connections, 802.3ae is the standard for 10Gbit/Ethernet over fiber, and so on. optic connections, These standards provide data rates from 10Mbs to40Gbs and higher. The shared medium in Ethernet can be a coaxial cable, twisted-pair wire or an optical fiber. the shared medium(i. e., broadcast medium)carries the communication for all the devices propagation conditions and transceiver capabilities. The specifications of the 802.3 standards are available on the IEEE802.3 working group website

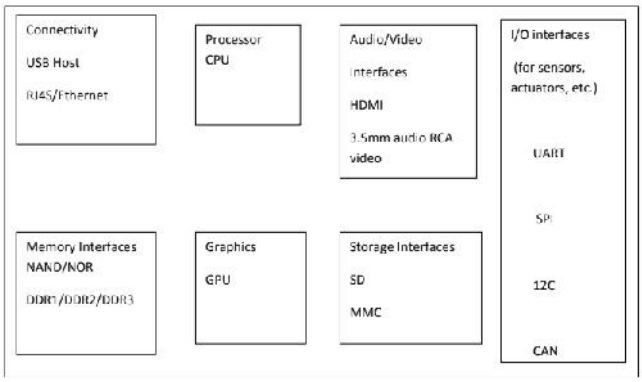

Figure 2:Generic block diagram of IOT Device

2.3 Logical Design of IoT

Logical design of an IoT system refers to an abstract representation of the entities and processes without going into the low-leve; specifies of the implementation. In this section we describe the functional blocks of an IoT system and the communication APIs that are used for the examples in this book. The steps in logical design are described in additional detail in Chapter-5.

2.3.1 IoT Functional Blocks

An IoT system comprises of a number of functional blocks that provide the system the capabilities for identification, sensing, actuation, and management as shown in Figure 1.6.
These functional blocks are described as follows:
• Device: An IoT system comprises of devices that provide sensing, actuation, monitoring and control functions. You learned about IoT devices in section 1.2.
• Communication: The communication block handles the communication for the Iot system. You learned about various protocols used for communication by IoT systems in section 1.2.
• Services: An IoT system uses various types of IoT services such as services for device monitoring, device control services, data publishing services and services for device discovery.
• Management: Management functional block provides various functions to govern the IoT system.
• Security: Security functional block secures the IoT system and by providing functions such as authentication, authorization, message and content integrity, and data security.
• Application: IoT applications provide an interface that the users can use to control and monitor various aspects of the IoT system. Applications also allow users to view the system status and view or analyze the processed data.

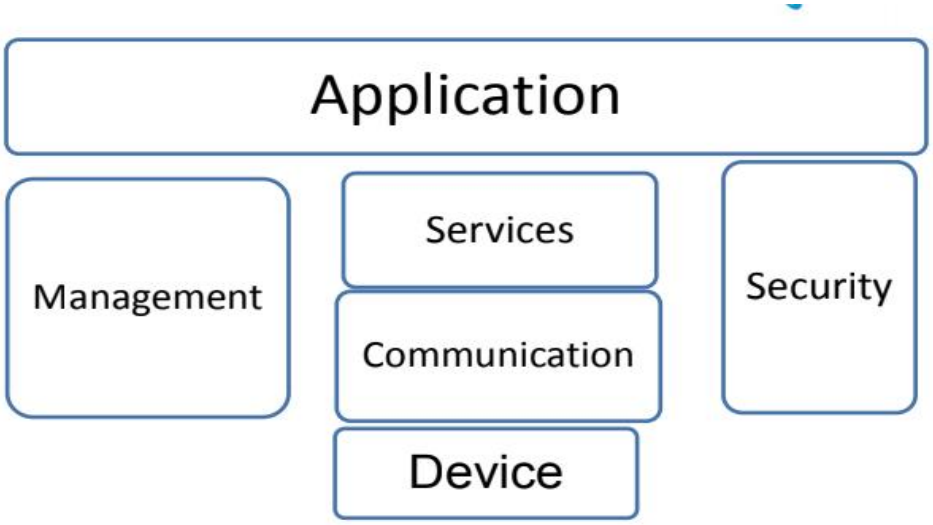

Figure 3: Functional Blocks of IoT

Request-Response: Request-Response is a communication model in which the client sends request to the server and the server responds to the requests. When the server receives a request, it decided how to respond, fetches the data, retrieves resource representations, prepares the response, and then sends the response to the client. Request- Response model is a stateless communication model and each request-response pair is independent of others. Figure1.7 shows the client-server interactions in the request-response model.

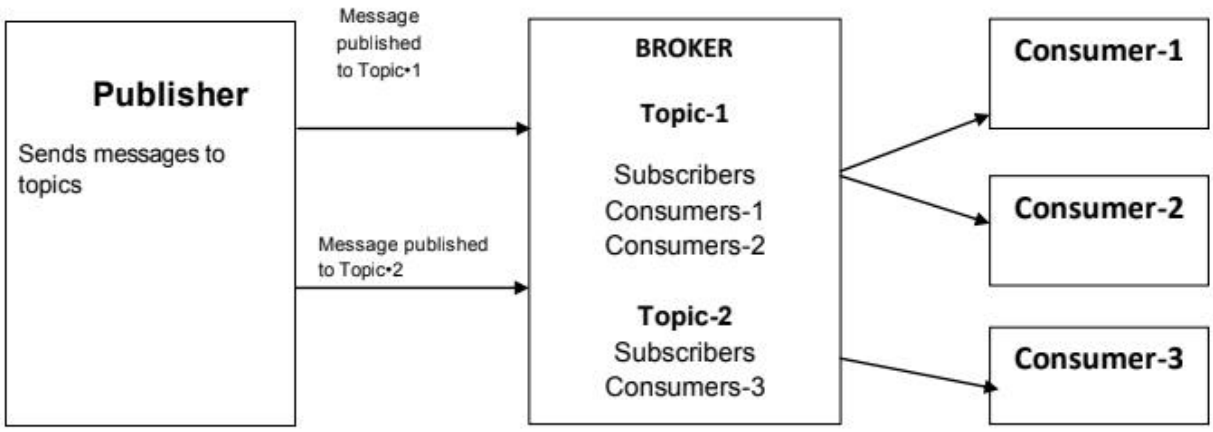

Figure 4 : Publish-Subscribe communication model

Publish-Subscribe : Publish-Subscribe is a communication model that involves publishers, brokers and consumers. Publishers are the source of data. Publishers send the data to the topics which are managed by the broker. Publishers are not aware of the consumers. Consumers subscribe to the topics which are managed by the broker. When the broker receives data for a topic from the publisher, it sends the data to all the subscribed consumers. Figure 1.8 shows the publisher-broker-consumer interactions in the publish-subscribe model.

• **Push-Pull :** Push-Pull is a communication model in which the data producers push the data to queues and the consumers pull the data from the queues. Producers do not need to be aware of the consumers. Queues help in decoupling the messaging between the producers and consumers. Queues also act as a buffer which helps in situations when there is a mismatch between the rate at which the producers push data and the rate at which the consumers pull data. Figure 1.9 shows the publisher- queue-consumer interactions in the push-pull model.

• **Exclusive Pair:** Exclusive Pair is a bi-directional, fully duplex communication model that uses a persistent connection between the client and server. Once the connection is setup it remains open until the client sends a request to close the connection. Client and server can send messages to each other after connection setup. Exclusive pair is a stateful communication model and the server is aware of all the open connections. Figure 1.10 shows the client -server interactions in the exclusive pair model.

2.4 IoT Enabling Technologies:

IOT is enabled by several technologies including wireless sensor networks, cloud computing, Big Data analytics, embedded systems, security protocols and architectures, communication protocols, web services, mobile Internet and semantic search engines. This section provides an overview of some of these technologies which play a key role in IoT Wireless sensor networks: A wireless Sensor Network (WSN) comprises of distributed devices with Sensors which are used to monitor the environmental and physical conditions. a WSN consist of a number of end-nodes and routers and a coordinator. The coordinator collects the data from all the nodes.

Coordinator also acts as a gateway that connects the WSN to the Internet. Some examples are WSNs used in IoT systems are described as follows:

• Weather monitoring systems use WSNs in which the nodes collect temperature, humidity and other data, which is aggregated and analyzed
• Indoor air quality monitoring systems use WSNs to collect data on the indoor air quality and concentration of various gaseous
• Soil moisture monitoring systems use WSNs to monitor soil moisture at various locations
• Surveillance systems use WSNs for collecting surveillance data(such as motion detection data)
• Smart grids use WSNs for monitoring the grid at various point
• Structural health monitoring systems use WSNs to monitor the health of structures(buildings, bridges)by collecting vibration data from sensor nodes deployed at various points at the structure
WSNs are enabled by wireless communication protocols such as IEEE 802.15.4.ZigBee is one of the most popular wireless technologies used by WSNs. ZigBee specifications are based on IEE 802.15.4.ZigBee operates at 2.4GHZ frequency and offer data rates up to 250KB/s and range from 10 to 100 meters depending on the power output and environmental conditions. The power of WSNs lies in their ability to deploy large number of low cost and low power- sensing nodes for continuous monitoring of environmental and physical conditions . WSNs are self organizing networks. Since WSNs have large number of nodes, manual configuration of each node is not

possible. The self organizing capability of WSN makes the network robust. In the event of failure of some node or addition of new nodes to the network, the network can reconfigure itself

2.4.1 Cloud Computing:

Cloud computing is transformative computing paradigm that involves delivering applications and services over the Internet. Cloud computing involves provisioning of computing, networking and storage resources on demand and providing these resources as metered services to the users, in a "Pay As You Go" model. Cloud computing resources can be provisioned on demand by the users, without requiring interactions with the cloud service provider. The process of provisioning resources is automated. Cloud computing resources can be accessed over the network using standard access mechanisms that provide platform=Independent access through the use of heterogeneous client platforms such as work stations Laptops tablets and Smart phones. The computing and storage resources provided by cloud service providers are pooled to serve multiple users using multi-tenancy. Multi- tenant aspects of the cloud allow multiple users to be served by the same physical hardware. Users are assigned virtual resources that run on top of the physical resources. Cloud computing services are offered to users in different forms.

• **Infrastructure as a service (IASS):** IASS provides the users the ability to provision computing and storage resources .these resources are provided to the users as virtual machine instances and virtual storage. Users can start, stop configure and manage the virtual machine instances and virtual storage. Users can deploy operating systems and applications of their choice on the virtual resources provisioned in the cloud. The cloud service provider manages the underlined infrastructure. Virtual resources provisioned by the users are build based on pay-per-use paradigm.

• **Platform as a service(PAAS):** PASS provides the users the ability to develop and deploy applications in cloud using the development tools, application programming interfaces (API's), software libraries and services provided by the cloud service including servers networks operating systems and storage/ the users themselves are responsible for developing, deploying, configuring, managing applications on the cloud infrastructure.

• **Software as a Service (SAAS):** SAAS provides the users a complete software application or the user interface to the application itself. The cloud service provider manages the underlying cloud infrastructure including servers, networks, operating systems, storage and application software, and the user is unaware of underlying architecture of the cloud. Applications are provided to the user through a client interface. SAAS applications are platform independent and can be accessed by various client devices such as workstations, laptops, tablets and Smart phones, running different operating system. since the cloud service provider manages both the application and data, the users are able to access the applications from anywhere.

2.4.2 Big Data Analytics:

Big Data is defined as collection of data whose volume, velocity, variety is so large that it is difficult to store, manage, process and analyze the data using traditional database and data processing tools. Big data analytics involves several steps starting from data cleansing, data managing, data processing and visualization. Some examples of Big data generated by Iot system are described as follows:

• Sensor data generated by IoT system such as weather monitoring system

• Machine sensor data collected from sensors embedded in industrial and energy systems for monitoring their health and detecting failures

• Health and fitness data generated by IoT device such as wearable fitness bands.

• Data generated by retail inventory monitory monitoring systems. The underlying characteristics of big Data include:

• Volume: Though there is no fixed threshold for the volume of data to be considered as a big data, however, typically, the term big data is used for massive scale data that is difficult to store, manage and process using traditional database and data processing architectures. The volume of data generated by modern IT, industrial and healthcare systems, for example, is growing exponentially driven by the lowering costs of data storage and processing architectures and the need to extract valuable insights from the data to improve business processes, efficiency and service to consumers. • Velocity: velocity is another important characteristic of Big data analytics and the primary reason for exponential growth of data. Velocity of data refers to hoe fast the data is generated and how frequently it varies. Modern IT, Industrial and other systems are generating data at increasingly higher speeds.

• Variety: variety refers to the forms of the data. Big data comes in different forms such as structured or unstructured data, including text data, image, audio, video and sensor data.

2.4.3 Communication Protocols:

Communication protocols form the backbone of IoT systems and enable network connectivity and coupling to applications. Communication protocols allow devices to exchange data over the network. These protocols define the data exchange formats, data encoding, addressing schemes for devices and routing of packets from source to destination. Other functions of the protocols include sequence control, flow control and retransmission of lost packets.

2.4.4 Embedded Systems:

An embedded system is a computer system that has computer hardware and software embedded to perform specific tasks. In contrast to general purpose computers or personal computers which can perform various tasks, embedded systems are designed to perform a specific set of tasks. Key component of an embedded system include microprocessor, micro controller , memory, networking unit, input/output unit and storage. Some embedded systems have specialized processors such as digital signal processors, graphics processors and application specific processors. Embedded systems rum embedded running operating systems such as real time operating systems(RTOS). Embedded systems range from low cost miniaturized devices such as

digital watches to devices such as digital cameras, point of sale terminals, vending machines, appliances, etc.

2.5 IoT Levels & Deploying Templates:

An IoT system comprises of following components:

• **Device:** An IoT device allows identification, remote sensing, actuating and remote monitoring capabilities
• **Resources:** resources re software components on IoT devices for accessing, processing and storing sensor information, or controlling actuators connected to the data device. Resources also include the software components that enable network access for the device
• **Control Service:** it is a native service that runs on the device and interacts with the web services. Controller service sends data from the device to the web service and receives commands from the application(via web services) for controlling device.
• **Web Service:** web services serve as a link between an IoT device, application, database and analysis components. Web services can be either implemented using HTTP and REST principles or using WebSocket Protocol.

Comparison of REST and WebSocket:

Stateless/stateful: REST services are stateless in nature, each request contains all the information needed to process it. Requests are independent of each other. WebSocket on the other hand is stateful in nature where the server maintains the state and is aware of all the open connections

Uni-directional/Bi-directional: Rest services operate over HTTP and are uni- directional. Request ois always sent by a client and the server responds to the requests. On the other hand WebSocket is a bi-directional protocol and allows both client and server to send messages to each other.

TCP connections: For REST services each HTTP requests involves setting up a new TCP connection. WebSocket on the other hand involves a single TCP connection over which the client and server communicate in full duplex mode.

Header Overheader: Rest services operate over HTTP, and each request is independent of others. Thus each request carries HTTP headers which are an Overhead. Due the overhead of HTTP headers, REST is not suitable for real-time applications .WebSockets on the other hand does not involve overhead of header. After the initial Handshake (that happens over HTTP), the client and server exchange messages with minimal frame information. Thus WebSocket is suitable for real time applications.

Scalability: scalability is easier in the case of REST services as requests ate independent and no state information needs to be maintained by the server. Thus both horizontal (scaling out) and vertical scaling (scaling up) solutions are possible for REST services. For WebSocktes, horizontal scaling can be cumbersome due to the stateful nature of the communication. Since the server maintains the state of connection, vertical scaling is easier for WebSockets than horizontal scaling

2.5.1 Analysis component:
The analysis component is responsible for analyzing the IoT data and generates results in a form which are easy for the user to understand. Analysis of IoT data can be performed either locally or in the cloud. Analyzed results are stored in the local or cloud databases

Application: IoT application provides an interface that the users can use to control and monitor various aspects of the IoT system. Application also allows users to view the system status and view the processed data. IoT level 1:
A level 1 IoT system has a single node or device that performs sensing and /actuation, stores data performs analysis and hosts the application as shown In figure 1.1. Level 1 IoT systems are suitable for modeling low cost and low complexity solutions where the data involved is not big and the analysis requirements are not computationally intensive. Lets us now consider an example of level 1 IoT system for home automation. The system consists of a single node that allows controlling the lights and appliances in a home remotely. The device used in this system interfaces with the lights and appliances uses electronic relay switches. The status information of each light or appliance or light is maintained in a local data base. REST services deployed locally allow retrieving and updating the state of each light or appliance in the status database. The controller service continuously monitors the state of each light or appliance and triggers the relay switches accordingly. The application which is deployed locally ha a user interface for controlling the light or appliances. Since the device is connected to the internet the application can be accessed remotely as well.

2.6 Introduction to Raspberry pi:

The Raspberry Pi is Series Of Small Single-Board computers Developed the United Kingdom by the Foundation to promote teaching of basic computer science in schools and in developing countries. The original model became far more popular than anticipated, selling outside its target market for uses such as robotics. It does not have Peripherals (Such as Keyboards and Mice) and cases. However there some Accessories been included in official and unofficial bundles

The organization behind the Raspberry Pi consists of two arms. The first two models were developed by the Raspberry Pi Foundation. After the Pi Model B was released, the Foundation set up Raspberry Pi Trading, with Eben Upton as CEO, to develop the third model, the B+. Raspberry Pi Trading is responsible for developing the technology while the Foundation is an educational charity to promote the teaching of basic computer science in schools and in developing countries.

2.7 Generation of released models:

Several generations of Raspberry Pi's have been released. All models feature a Broadcom system on a chip (SoC) with an integrated ARM-compatible central processing unit (CPU) and on-chip graphics processing unit (GPU).

Processor speed ranges from 700 MHz to 1.4 GHz for the Pi 3 Model B+; on-board memory ranges from 256 MB to 1 GB RAM. Secure (SD) cards are used to store the operating system and program memory in either SDHC (early Raspberry Pi's) or MicroSDHC (Later Raspberry Pi's) sizes. The boards have one to four USB ports. For video output, HDMI and composite video are supported, with a standard 3.5 mm tip-ring-sleeve jack for audio output. Lower-level output is provided by a number of GPIO pins, which support common protocols like I²C. The B-models have an 8P8C Ethernet port and the Pi 3 and Pi Zero W have on-board Wi-Fi 802.11n and Bluetooth. Prices range from US$5 to $35.

2.7.1 First generation:

The first generation (Raspberry Pi 1 Model B) was released in February 2012, followed by the simpler and cheaper Model A. In 2014, the Foundation released a board with an improved design, Raspberry Pi 1 Model B+. These boards are approximately credit-card sized and represent the standard *mainline* form-factor. Improved A+ and B+ models were released a year later.

2.7.2 Raspberry pi Zero:

A Raspberry Pi Zero with smaller size and reduced input/output (I/O) and general-purpose input/output (GPIO) capabilities was released in November 2015 for US$5. By 2017, it became the newest mainline Raspberry Pi. On 28 February 2017, the Raspberry Pi Zero W was launched, a version of the Zero with Wi-Fi and Bluetooth capabilities, for US$10. On 12 January 2018, the Raspberry Pi Zero WH was launched, the same version as the Zero W with pre-soldered GPIO headers

2.7.3 Raspberry pi 3 Model B:

Raspberry Pi 3 Model B was released in February 2016 with a 64-bit quad core processor, on-board WiFi, Bluetooth and USB boot capabilities On Pi Day 2018 model 3B+appeared with a faster 1.4 GHz processor and a three times faster network based on gigabit Ethernet (throughput limited to ca. 300 Mbit/s by the internal USB 2.0 connection) or 2.4 / 5 GHz dual-band Wi-Fi (100 Mbit / s Other options are: Power over Ethernet (PoE), USB boot and network boot (an SD card is no longer required).

2.8. Hardware:

The Raspberry Pi hardware has evolved through several versions that feature variations in memory capacity and peripheral-device support.

This block diagram describes Model B and B+; Model A, A+, and the Pi Zero are similar, but lack the Ethernet and USB hub components. The Ethernet adapter is internally connected to an additional USB port. In Model A, A+, and the Pi Zero, the USB port is connected directly to the system on a chip (SoC). On the Pi 1 Model B+ and later models the USB/Ethernet chip contains a five-port USB hub, of which four ports are available, while the Pi 1 Model B only provides two. On the Pi Zero, the USB port is also connected directly to the SoC, but it uses a micro USB (OTG) port.

2.8.1 Processor:

The Broadcom BCM2835 SoC used in the first generation Raspberry Pi includes a 700 MHz ARM11 76JZF-S processor, Video Core IV graphics processing unit (GPU) and RAM. It has a level 1 (L1) cache of 16 KB and a level 2 (L2) cache of 128 KB. The level 2 cache is used primarily by the GPU. The SoC is stacked underneath the RAM chip, so only its edge is visible. The 1176JZ(F)-S is the same CPU used in the original iPhone, although at a higher clock rate, and mated with a much faster GPU.

The earlier V1.1 model of the Raspberry Pi 2 used a Broadcom BCM2836 SoC with a 900 MHz 32-bit, quad-core ARM Cortex-A7 processor, with 256 KB shared L2 cache. The Raspberry Pi 2 V1.2 was upgraded to a Broadcom BCM2837 SoC with a 1.2 GHz 64-bit quad-core ARM Cortex-A53 processor, the same SoC which is used on the Raspberry Pi 3, but under clocked (by default) to the same 900 MHz CPU clock speed as the V1.1. The BCM2836 SoC is no longer in production as of late 2016.

The Raspberry Pi 3+ uses a Broadcom BCM2837B0 SoC with a 1.4 GHz 64-bit quad-core ARM Cortex-A53 processor, with 512 KB shared L2 cache.

2.8.2 Performance:

While operating at 700 MHz by default, the first generation Raspberry Pi provided a real-world performance roughly equivalent to 0.041 GFLOPS On the CPU level the performance is similar to a 300 MHz Pentium II of 1997–99. The GPU provides 1 Gpixel/s or 1.5 Gtexel/s of graphics processing or 24 GFLOPS of general purpose computing performance. The graphical capabilities of the Raspberry Pi are roughly equivalent to the performance of the Xbox of 2001.

Raspberry Pi 2 V1.1 included a quad-core Cortex-A7 CPU running at 900 MHz and 1 GB RAM. It was described as 4–6 times more powerful than its predecessor. The GPU was identical to the original. In parallelized benchmarks, the Raspberry Pi 2 V1.1 could be up to 14 times faster than a Raspberry Pi 1 Model B+.

The Raspberry Pi 3, with a quad-core ARM Cortex-A53 processor, is described as having ten times the performance of a Raspberry Pi 1. This was suggested to be highly dependent upon task threading and instruction set use. Benchmarks showed the Raspberry Pi 3 to be approximately 80% faster than the Raspberry Pi 2 in parallelised tasks

2.8.3 Overclocking:

Most Raspberry Pi systems-on-chip could be over clocked to 800 MHz, and some to 1000 MHz's There are reports the Raspberry Pi 2 can be similarly over clocked, in extreme cases, even to 1500 MHz (discarding all safety features and over-voltage limitations). In the Raspbian Linux distort the overclocking options on boot can be done by a software command running "sudoraspi-config" without voiding the warranty.[30] In those cases the Pi automatically shuts the overclocking down if the chip temperature reaches 85 °C (185 °F), but it is possible to override automatic over-voltage and overclocking settings (voiding the warranty); an appropriately sized heat sink is needed to protect the chip from serious overheating.

Newer versions of the firmware contain the option to choose between five overclock ("turbo") presets that when used, attempt to maximize the performance of the SoC without impairing the lifetime of the board. This is done by monitoring the core temperature of the chip and the CPU load, and dynamically adjusting clock speeds and the core voltage. When the demand is low on the CPU or it is running too hot the performance is throttled, but if the CPU has much to do and the chip's temperature is acceptable, performance is temporarily increased with clock speeds of up to 1 GHz, depending on the individual board and on which of the turbo settings is used.

The seven overclock presets are:

- none; 700 MHz ARM, 250 MHz core, 400 MHz SDRAM, 0 overvaulting
- modest; 800 MHz ARM, 250 MHz core, 400 MHz SDRAM, 0 overvaulting,
- medium; 900 MHz ARM, 250 MHz core, 450 MHz SDRAM, 2 overvaulting,
- high; 950 MHz ARM, 250 MHz core, 450 MHz SDRAM, 6 overvaulting,
- turbo; 1000 MHz ARM, 500 MHz core, 600 MHz SDRAM, 6 overvaulting,
- Pi 2; 1000 MHz ARM, 500 MHz core, 500 MHz SDRAM, 2 overvaulting,
- Pi 3; 1100 MHz ARM, 550 MHz core, 500 MHz SDRAM, 6 overvaulting. In system information the CPU speed will appear as 1200 MHz's When idling, speed lowers to 600 MHz's

In the highest (turbo) preset the SDRAM clock was originally 500 MHz, but this was later changed to 600 MHz because 500 MHz sometimes causes SD card corruption. Simultaneously in high mode the core clock speed was lowered from 450 to 250 MHz, and in medium mode from 333 to 250 MHz's

The Raspberry Pi Zero runs at 1 GHz.

The CPU on the first and second generation Raspberry Pi board did not require cooling, such as a heat sink or fan, even when overclocked, but the Raspberry Pi 3 may generate more heat when overclocked.

2.9 RAM

On the older beta Model B boards, 128 MB was allocated by default to the GPU, leaving 128 MB for the CPU. On the first 256 MB release Model B (and Model A), three different splits were possible. The default split was 192 MB (RAM for CPU), which should be sufficient for standalone 1080p video decoding, or for simple 3D, but probably not for both together. 224 MB was for Linux only, with only a 1080p frame buffer, and was likely to fail for any video or 3D. 128 MB was for heavy 3D, possibly also with video decoding (e.g. XBMC). Comparatively the Nokia 701 uses 128 MB for the Broadcom Video Core IV.

For the later Model B with 512 MB RAM new standard memory split files (arm256_start.elf, arm384_start.elf, arm496_start.elf) were initially released for 256 MB, 384 MB and 496 MB CPU RAM (and 256 MB, 128 MB and 16 MB video RAM) respectively. But a week or so later the RPF released a new version of start. Elf that could read a new entry in config.txt (gpu_mem=*xx*) and could dynamically assign an amount of RAM (from 16 to 256 MB in 8 MB steps) to the GPU, so the older method of memory splits became obsolete, and a single start. Elf worked the same for 256 MB and 512 MB Raspberry Pis.

The Raspberry Pi 2 and the Raspberry Pi 3 have 1 GB of RAM. The Raspberry Pi Zero and Zero W have 512 MB of RAM.

2.10 Networking:

The Model A, A+ and Pi Zero have no Ethernet circuitry and are commonly connected to a network using an external user-supplied USB Ethernet or Wi-Fi adapter. On the Model B and B+ the Ethernet port is provided by a built-in USB Ethernet adapter using the SMSC LAN9514 chip. The Raspberry Pi 3 and Pi Zero W (wireless) are equipped with 2.4 GHz Wi-Fi 802.11n (150 Mbit/s) and Bluetooth 4.1 (24 Mbit/s) based on the Broadcom BCM43438 FullMAC chip with no official support for monitor mode but implemented through unofficial firmware patching, and the Pi 3 also has a 10/100 Mbit/s Ethernet port. The Raspberry Pi 3B+ features dual-band IEEE 802.11b/g/n/ac Wi-Fi, Bluetooth 4.2, and Gigabit Ethernet (limited to approximately 300 Mbit/s by the USB 2.0 bus between it and the SoC).

2.11 Special-Purpose Features:

The Pi Zero can be used as a USB device or "USB gadget", plugged into another computer via a USB port on each machine. It can be configured in multiple ways, for example to show up as a serial device or an Ethernet device. Although originally requiring software patches, this was added into the mainline Raspbian distribution in May 2016.

The Pi 3 can boot from USB, such as from a flash drive. Because of firmware limitations in other models, the Pi 3 is the only board that can do this.

2.12 Peripherals:

The Raspberry Pi may be operated with any generic USB computer keyboard and mouse. It may also be used with USB storage, USB to MIDI converters, and virtually any other device/component with USB capabilities.

Other peripherals can be attached through the various pins and connectors on the surface of the Raspberry Pi.

2.13 Video:

The video controller can generate standard modern TV resolutions, such as HD and Full HD, and higher or lower monitor resolutions as well as older NTSC or PAL standard CRT TV resolutions. As shipped (i.e., without custom overclocking) it can support the following resolutions: 640×350 EGA; 640×480 VGA; 800×600 SVGA; 1024×768 XGA; 1280×720 720p HDTV; 1280×768 WXGA variant; 1280×800 WXGA variant; 1280×1024 SXGA; 1366×768 WXGA variant; 1400×1050 SXGA+; 1600×1200 UXGA; 1680×1050 WXGA+; 1920×1080 1080p HDTV; 1920×1200 WUXGA.

Higher resolutions, up to 2048×1152, may work or even 3840×2160 at 15 Hz (too low a frame rate for convincing video). Note also that allowing the highest resolutions does not imply that the GPU can decode video formats at these resolutions; in fact, the Pis are known to not work reliably for H.265 (at those high resolutions), commonly used for very high resolutions (however, most common formats up to Full HD do work).

Although the Raspberry Pi 3 does not have H.265 decoding hardware, the CPU is more powerful than its predecessors, potentially fast enough to allow the decoding of H.265-encoded videos in software. The GPU in the Raspberry Pi 3 runs at higher clock frequencies of 300 MHz or 400 MHz, compared to previous versions which ran at 250 MHz.

The Raspberry Pis can also generate 576i and 480i composite video signals, as used on old-style (CRT) TV screens and less-expensive monitors through standard connectors – either RCA or 3.5 mm phono connector depending on models. The television signal standards supported are PAL-BGHID, PAL-M, PAL-N, NTSC and NTSC-J

2.14 Real-time clock:

None of the current Raspberry Pi models have a built-in real-time clock, so they are unable to keep track of the time of day independently. As a workaround, a program running on the Pi can retrieve the time from a network time server or from user input at boot time, thus knowing the time while powered on. To provide consistency of time for the file system, the Pi automatically saves the current system time on shutdown, and re-loads that time at boot.

A real-time hardware clock with battery backup, such as the DS1307, may be added (often via the I²C interface). Note however that this conflicts with the camera's CSI interface, effectively disabling the camera.

2.15 Specifications

Model A

Generation	**1**	**1+**	**3+**
Release Date	February 2013	November 2014	November 2018
Instruction set	ARMv6Z (32 bit)	ARMv6Z (32 bit)	ARMv8(64 bit)
SOC	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2837
CPU	1× ARM1176JZF-S 700 MHz	1× ARM1176JZF-S 700 MHz	4× Cortex-A53 1.4 GHz
Memory (SDRAM)	256MB	512 MB	512 Mb
USB Ports	1 (direct from BCM 2835 chip)	1 (direct from BCM 2835 chip)	1 (direct from BCM 2837 chip)
Video Input	15 Pin Camera	15 Pin Camera	15 Pin Camera
Video Output	HDMI	HDMI	HDMI
On Board Storage	SD,SDIO card	Micro SDHC slot	Micro SDHC slot
Power rating	300mA	200mA	-
Size	85.60mmX56.5mm	65mmx56.5mm	65mm x 56.5

Table 1 Raspberrypi Model A Specifications

Model B:

Generation	**1**	**1+**	**2**	**2 ver1.2**	**3**	**3+**
Release Date	April-June 2012	July 2014	Feb-2015	Oct 2016	February 2016	14th march 2108
Instruction set	ARMv6Z (32-bit)	ARMv6Z (32-bit)	ARMv7-A (32-bit)	ARMv8-A (64/32-bit)	ARMv8-A (64/32-bit)	ARMv8-A (64/32-bit)
SOC	BCM2835	BCM2835	BCM2836	BCM2837	BCM2837	BCM2837

CPU	1× ARM1176JZF-S 700 MHz	1× ARM1176JZF-S 700 MHz	4× Cortex-A7900 MH	4× Cortex-A53900 MHz	4× Cortex-A53 1.2 GHz	4× Cortex-A531.4 GHz
Memory (SDRAM)	1GB	1GB	1GB	1GB	1GB	1GB
USB Ports	4	4	4	4	4	4
Video Input	15 Pin MIPI	15 Pin MIPI	15 Pin MIPI	15 Pin MIPI	15 Pin MIPI	15 Pin MIPI
Video Output	HDMI,MIPI Display	HDMI,MIPI Display	HDMI,MIPI Display	HDMI,MIPI Display	HDMI,MIPI Display	HDMI,MIPI Display
On Board Storage	Micro SDHC Slot	Micro SDHC Slot	Micro SDHC Slot	Micro SDHC Slot & USB	Micro SDHC Slot & USB	Micro SDHC Slot & USB
Power rating	200mA	220mA	220mA	300mA	459mA	459mA
Size	86.60mm x 56.5mm	86.60mm x 56.5mm	86.60mm x 56.5mm	85.60 mm x 56.5 mm	85.60 mm x 56.5 mm	85.60 mm x 56.5 mm

Table 2 Raspberrypi Model B Specifications

2.16 General purpose input-output (GPIO) connector:

Raspberry Pi 1 Models A+ and B+, Pi 2 Model B, Pi 3 Models A+, B and B+, and Pi Zero and Zero W GPIO J8 have a 40-pin pin out.Raspberry Pi 1 Models A and B have only the first 26 pins.

GPIO#	2nd func.	Pin#		Pin#	2nd func.	GPIO#
	+3.3 V	1		2	+5 V	
2	SDA1 (I²C)	3		4	+5 V	

3	SCL1 (I²C)	5	6	GND	
4	GCLK	7	8	TXD0 (UART)	14
	GND	9	10	RXD0 (UART)	15
17	GEN0	11	12	GEN1	18
27	GEN2	13	14	GND	
22	GEN3	15	16	GEN4	23
	+3.3 V	17	18	GEN5	24
10	MOSI (SPI)	19	20	GND	
9	MISO (SPI)	21	22	GEN6	25
11	SCLK (SPI)	23	24	CE0_N (SPI)	8
	GND	25	26	CE1_N (SPI)	7
(Pi 1 Models A and B stop here)					
EEPROM	ID_SD	27	28	ID_SC	EEPROM
5	N/A	29	30	GND	

6	N/A	31	32		12
13	N/A	33	34	GND	
19	N/A	35	36	N/A	16
26	N/A	37	38	Digital IN	20
	GND	39	40	Digital OUT	21

Table 3 Raspberrypi Pinout

Models A and B provide GPIO access to the ACT status LED using GPIO 16. Models A+ and B+ provide GPIO access to the ACT status LED using GPIO 47, and the power status LED using GPIO 35

2.17 Accessories:

- Garboard – A Raspberry Pi Foundation sanctioned device, designed for educational purposes, that expands the Raspberry Pi's GPIO pins to allow interface with and control of LEDs, switches, analogue signals, sensors and other devices. It also includes an optional Arduino compatible controller to interface with the Pi.
- Camera – On 14 May 2013, the foundation and the distributors RS Components & Premier Farrell/Element 14 launched the Raspberry Pi camera board alongside a firmware update to accommodate it. The camera board is shipped with a flexible flat cable that plugs into the CSI connector which is located between the Ethernet and HDMI ports. In Raspbian, the user must enable the use of the camera board by running Raspi-config and selecting the camera option. The camera module costs €20 in Europe (9 September 2013). It can produce 1080p, 720p and 640x480p video. The dimensions are 25 mm × 20 mm × 9 mm. In May 2016, v2 of the camera came out, and is an 8 megapixel camera.
- Infrared Camera – In October 2013, the foundation announced that they would begin producing a camera module without an infrared filter, called the Pi NoIR.
- Official Display – On 8 September 2015, The foundation and the distributors RS Components & Premier Farnell/Element 14 launched the Raspberry Pi Touch Display.
- HAT (Hardware Attached on Top) expansion boards – Together with the Model B+, inspired by the Arduino shield boards, the interface for HAT boards was devised by the Raspberry Pi Foundation. Each HAT board carries a small EEPROM (typically a CAT24C32WI-GT3) containing the relevant details of the board, so that the Raspberry Pi's OS is informed of the HAT, and the technical details of it, relevant to the OS using the HAT. Mechanical

details of a HAT board, which uses the four mounting holes in their rectangular formation, are available online.

2.18 Software:

2.18.1 Operating System:

The Raspberry Pi Foundation provides Raspbian, a Debi an-based Linux distribution for download, as well as third-party Ubuntu, Windows 10 IoT Core, RISC OS, and specialized media centre distributions. It promotes Python and Scratch as the main programming languages, with support for many other languages. The default firmware is closed source, while an unofficial open source is available. Many other operating systems can also run on the Raspberry Pi, including the formally verified microkernel, seL4. Other third-party operating systems available via the official website include Ubuntu MATE, Windows 10 IoT Core, RISC OS and specialized distributions for the Kodi media center and classroom management.

Other operating systems (not Linux-based)

- Broadcom VCOS – Proprietary operating system which includes an abstraction layer designed to integrate with existing kernels, such as Thread (which is used on the VideoCore4 processor), providing drivers and middleware for application development. In case of Raspberry Pi this includes an application to start the ARM processor(s) and provide the publicly documented API over a mailbox interface, serving as its firmware. An incomplete source of a Linux port of VCOS is available as part of the reference graphics driver published by Broadcom.
- RISC OS Pi (a special cut down version RISC OS Pico, for 16 MB cards and larger for all models of Pi 1 & 2, has also been made available.)
- Free BSD
- Net BSD
- Open BSD (only on 64-bit platforms, such as Raspberry Pi 3)
- Plan 9 from Bell Labs and Inferno (in beta)
- Windows 10 IoT Core – a no-cost edition of Windows 10 offered by Microsoft that runs natively on the Raspberry Pi 2.
- Haiku – an open source BeOS clone that has been compiled for the Raspberry Pi and several other ARM boards. Work on Pi 1 began in 2011, but only the Pi 2 will be supported.
- Helens – a portable microkernel-based multiserver operating system; has basic Raspberry Pi support since version 0.6.0

2.19 Driver API:

Raspberry Pi can use a Video Core IV GPU via a binary blob, which is loaded into the GPU at boot time from the SD-card, and additional software, that initially was closed source. This part of the driver code was later released. However, much of the actual driver work is done

using the closed source GPU code. Application software makes calls to closed source run-time libraries (OpenMax, OpenGL ES or OpenVG), which in turn call an open source driver inside the Linux kernel, which then calls the closed source Video Core IV GPU driver code. The API of the kernel driver is specific for these closed libraries. Video applications use OpenMAX, 3D applications use OpenGL ES and 2D applications use OpenVG, which both in turn use EGL. OpenMAX and EGL use the open source kernel driver in turn.

2.20 Firmware:

The official firmware is a freely redistributable binary blob, that is closed-source. A minimal proof-of-concept open source firmware is also available, mainly aimed at initializing and starting the ARM cores as well as performing minimal startup that is required on the ARM side. It is also capable of booting a very minimal Linux kernel, with patches to remove the dependency on the mailbox interface being responsive. It is known to work on Raspberry Pi 1, 2 and 3, as well as some variants of Raspberry Pi Zero. While it is in a working state, it is not actively developed, with last significant commits made around mid-2017.

2.21 Third party application software

- C/C++ Interpreter Ch – Released 3 January 2017, C/C++ interpreter Ch and Embedded Ch are released free for non-commercial use for Raspberry Pi, Chide is also included for the beginners to learn C/C++.
- Mathematical & the Wolfram Language – Since 21 November 2013, Raspbian includes a full installation of this proprietary software for free. As of 24 August 2015, the version is Mathematical 10.2. Programs can be run either from a command line interface or from a Notebook interface. There are Wolfram Language functions for accessing connected devices. There is also a Wolfram Language desktop development kit allowing development for Raspberry Pi in Mathematical from desktop machines, including features from the loaded Mathematical version such as image processing and machine learning.
- Mine craft – Released 11 February 2013, a modified version that allows players to directly alter the world with computer code.
- Real NC – Since 28 September 2016, Raspbian includes RealVNC's remote access server and viewer software. This includes a new capture technology which allows directly-rendered content (e.g. Mine craft, camera preview and omxplayer) as well as non-X11 applications to be viewed and controlled remotely.
- User Gate Web Filter – On 20 September 2013, Florida-based security vendor Entensys announced porting User Gate Web Filter to Raspberry Pi platform.
- Steam Link – On 13 December 2018, Valve released official Steam Link game streaming client for the Raspberry Pi 3 and 3 B+.

2.22 Software development tools

- Arduino IDE – for programming an Arduino.
- Algoid – for teaching programming to children and beginners.
- BlueJ – for teaching Java to beginners.
- Green foot – Green foot teaches object orientation with Java. Create 'actors' which live in 'worlds' to build games, simulations, and other graphical programs.
- Julia – an interactive and cross-platform programming language/environment, that runs on the Pi 1 and later.IDEs for Julia, such as Juno, are available. See also Pi-specific Github repository JuliaBerry.
- Lazarus – a Free Pascal RAD IDE resembling Delphi
- Live Code – an educational RAD IDE descended from HyperCard using English-like language to write event-handlers for WYSIWYG widgets runnable on desktop, mobile and Raspberry Pi platforms.
- Ninja-IDE – a cross-platform integrated development environment (IDE) for Python.
- Object Pascal– an object oriented variant (the one used in Delphi and Lazarus) of Nicklaus Wirth's original Pascal language. Free Pascal is the compiler in Lazarus
- Processing – an IDE built for the electronic arts, new media art, and visual design communities with the purpose of teaching the fundamentals of computer programming in a visual context.
- Scratch – a cross platform teaching IDE using visual blocks that stack like Lego, originally developed by MIT's Life Long Kindergarten group. The Pi version is very heavily optimize for the limited compute resources available and is implemented in the Squeak Smalltalk system. The latest version compatible with The 2 B is 1.6.
- Squeak Smalltalk – a full scale open Smalltalk.
- Tensor Flow – an artificial intelligence framework developed by Google. The Raspberry Pi Foundation worked with Google to simplify the installation process through pre-built binaries.

2.23 Community:

The Raspberry Pi community was described by Jamie Ayre of FLOSS software company Adore as one of the most exciting parts of the project. Community blogger Russell Davis said that the community strength allows the Foundation to concentrate on documentation and teaching. The community developed a fanzine around the platform called which in 2015, was handed over to the Raspberry Pi Foundation by its volunteers to be continued in-house. A series of community *Raspberry Jam* events have been held across the UK and around the world

2.24 Use in education

As of January 2012, enquiries about the board in the United Kingdom have been received from schools in both the state and private sectors, with around five times as much interest from the latter. It is hoped that businesses will sponsor purchases for less advantaged schools. The

CEO of Premier Farnell said that the government of a country in the Middle East has expressed interest in providing a board to every schoolgirl, to enhance her employment prospects

In 2014, the Raspberry Pi Foundation hired a number of its community members including ex-teachers and software developers to launch a set of free learning resources for its website. The Foundation also started a teacher training course called Pica demy with the aim of helping teachers prepare for teaching the new computing curriculum using the Raspberry Pi in the classroom.

In 2018, NASA launched the *JPL Open Source Rover Project*, which is a scaled down of Curiosity rover and uses a Raspberry Pi as the control module, to encourage students and hobbyists to get involved in mechanical, software, electronics, and robotics engineering

2.25 Use in home automation:

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

CHAPTER 3

CLARIFAI

3.1 Installation Guide

Note: Generally, the python client works with Python 2.x and Python 3.x. But it is only tested against 2.7 and 3.5. Feel free to report BUGs if you encounter on other versions.

Install the package You can install the latest stable clarifai using pip (which is the canonical way to install Python packages). To install using pip run

```
pip install clarifai –upgrade
```

You can also install from git using latest source:

```
pip install git+git://github.com/Clarifai/clarifai-python.git
```

Configuration The client uses CLARIFAI_API_KEY for authentication. Each application you create uses its own unique ID and secret to authenticate requests. The client will use the authentication information passed to it by one of three methods with the following precedence order:

1. Passed in to the constructor through the api_key parameter.

2. Set as the CLARIFAI_API_KEY environment variable.

3. Placed in the .clarifai/config file using the command below.

You can get these values from https://developer.clarifai.com/account/applications and then run:

```
$ clarifai config
CLARIFAI_API_KEY: []: **********************************YQEd
```

If you do not see any error message after this, you are all set and can proceed with using the client.

3.1.1Windows Users

For Windows users, you may fail running the clarifai config when you try to configure the runtime environ-ment. This is because Windows uses the file extension to determine executables and by default file clarifai without file extension is nonexecutable. In order to run the command, you may want to launch it with the python interpreter.

```
C:\Python27>python.exe Scripts\clarifai config
CLARIFAI_API_KEY:[]: **********************************YQEd
```

3.1.2 AWS Lambda Users

For AWS Lambda users, in order to use the library correctly, you are recommended to set two environmental variables CLARIFAI_API_KEY in the lambda function configuration, or hardcode the APP_ID and APP_SECRET in the API instantiation.

3.2 Tutorial

Each of the examples below is a small independent code snippet within 10 lines that could work by copy and paste to a python source code file. By playing with them, you should be getting started with Clarifai API. For more information about the API, check the API Reference.

3.2.1 Predict Tutorial

Predict with Public Models

For more information on any of the public models, visit https://developer.clarifai.com/models

```
from clarifai.rest import ClarifaiApp
app = ClarifaiApp()
#General model
model = app.models.get('general-v1.3')
response = model.predict_by_url(url='https://samples.clarifai.com/metro-north.jpg')
#Travel model
model = app.models.get('travel-v1.0')
response = model.predict_by_url(url='https://samples.clarifai.com/travel.jpg')
#Food model
model = app.models.get('food-items-v1.0')
response = model.predict_by_url(url='https://samples.clarifai.com/food.jpg')
#NSFW model
model = app.models.get('nsfw-v1.0')
response = model.predict_by_url(url='https://samples.clarifai.com/nsfw.jpg')
#Apparel model
model = app.models.get('apparel')
response = model.predict_by_url(url='https://samples.clarifai.com/apparel.jpg')
#Celebrity model
model = app.models.get('celeb-v1.3')
response = model.predict_by_url(url='https://samples.clarifai.com/celebrity.jpg')
#Demographics model
model = app.models.get('demographics')
response = model.predict_by_url(url='https://samples.clarifai.com/demographics.jpg')
#Face Detection model
model = app.models.get('face-v1.3')
response = model.predict_by_url(url='https://developer.clarifai.com/static/images/ ‹→model-
samples/face-001.jpg')
#Focus Detection model
model = app.models.get('focus')
```

```
response = model.predict_by_url(url='https://samples.clarifai.com/focus.jpg')
#General Embedding model
model = app.models.get('general-v1.3', model_type='embed')
response = model.predict_by_url(url='https://samples.clarifai.com/metro-north.jpg')
#Logo model
model = app.models.get('logo')
response = model.predict_by_url(url='https://samples.clarifai.com/logo.jpg')
#Color model
model = app.models.get('color', model_type='color')
response = model.predict_by_url(url='https://samples.clarifai.com/wedding.jpg')
```

3.2.2 Feedback Tutorial

Concept model prediction

```
from clarifai.rest import ClarifaiApp
from clarifai.rest import FeedbackInfo
app = ClarifaiApp()
spositive feedback: this is a dog m = app.models.get('general-v1.3')
m.send_concept_feedback(input_id='id1', url='https://samples.clarifai.com/dog2.jpeg',
concepts=['dog', 'animal'],
feedback_info=FeedbackInfo(output_id='OID',
session_id='SID',
end_user_id='UID',
event_type='annotation'))
negative feedback: this is not a cat m = app.models.get('general-v1.3')
m.send_concept_feedback(input_id='id1', url='https://samples.clarifai.com/dog2.jpeg',
not_concepts=['cat', 'kitty'],
feedback_info=FeedbackInfo(output_id='OID',
session_id='SID',
end_user_id='UID',
event_type='annotation'))
all together: this is a dog but not a cat m = app.models.get('general-v1.3')
m.send_concept_feedback(input_id='id1', url='https://samples.clarifai.com/dog2.jpeg',
concepts=['dog'], not_concepts=['cat', 'kitty'], feedback_info=FeedbackInfo(output_id='OID',
session_id='SID',
end_user_id='UID',
event_type='annotation'))
```

3.2.3 Detection model prediction

```
from clarifai.rest import ClarifaiApp
from clarifai.rest import FeedbackInfo
from clarifai.rest import Region, RegionInfo, BoundingBox, Concept
app = ClarifaiApp()
m.send_region_feedback(input_id='id2', url='https://developer.clarifai.com/static/
‹ → images/model-samples/celeb-001.jpg',
regions=[Region(region_info=RegionInfo(bbox=BoundingBox(top_
‹ → row=0.1,
left_
‹ → col=0.2,
bottom_
‹ → row=0.5,
right_
‹ → col=0.5)),
concepts=[Concept(concept_id='people',
‹ → value=True),
Concept(concept_id='portrait',
‹ → value=True)])],
feedback_info=FeedbackInfo(output_id='OID',
session_id='SID',
end_user_id='UID',
event_type='annotation'))
```

3.2.4 Face detection model prediction

send feedback for celebrity model

```
from clarifai.rest import ClarifaiApp
from clarifai.rest import FeedbackInfo
from clarifai.rest import Region, RegionInfo, BoundingBox, Concept from clarifai.rest
import Face, FaceIdentity
from clarifai.rest import FaceAgeAppearance,
FaceGenderAppearance,‹ → FaceMulticulturalAppearance
app = ClarifaiApp()
#send feedback for celebrity model
m.send_region_feedback(input_id='id2', url='https://developer.clarifai.com/static/
‹ → images/model-samples/celeb-001.jpg',
regions=[Region(region_info=RegionInfo(bbox=BoundingBox(top_
‹ → row=0.1,
left_
‹ → col=0.2,
bottom_
‹ → row=0.5,
right_
```

```
‹→col=0.5)),
‹→face=Face(identity=FaceIdentity([Concept(concept_id='celeb1',value=True)]))
)
],
feedback_info=FeedbackInfo(output_id='OID',

session_id='SID',
end_user_id='UID',
event_type='annotation'))
#send feedback for age, gender, multicultural appearance
m.send_region_feedback(input_id='id2', url='https://developer.clarifai.com/static/
‹→images/model-samples/celeb-001.jpg',
regions=[Region(region_info=RegionInfo(bbox=BoundingBox(top_
‹→row=0.1,
left_
‹→col=0.2,
bottom_
‹→row=0.5,
right_
face=Face(age_
‹→appearance=FaceAgeAppearance([Concept(concept_id='20',      value=True),
‹→Concept(concept_id='30',  value=False)
]),
gender_
‹→appearance=FaceGenderAppearance([Concept(concept_id='male',value=True)]),
multicultural_
‹→appearance=FaceMulticulturalAppearance([Concept(concept_id='asian',          value=True)])
)
)
],
feedback_info=FeedbackInfo(output_id='OID',
session_id='SID',
end_user_id='UID',
event_type='annotation'))
```

3.2.5 Upload Images

```
from clarifai.rest import ClarifaiApp
app = ClarifaiApp()
app.inputs.create_image_from_url(url='https://samples.clarifai.com/puppy.jpeg',
‹→concepts=['my       puppy'])
app.inputs.create_image_from_url(url='https://samples.clarifai.com/wedding.jpg', not_
‹→concepts=[’my puppy’])
```

Create a Model

Note: This assumes you follow through the tutorial and finished the “Upload Images” Otherwise you may not be able to create the model.

```
model = app.models.create(model_id="puppy", concepts=["my puppy"])
```

Train the Model

Note: This assumes you follow through the tutorial and finished the “Upload Images” and “Create a Model” to create a model. Otherwise you may not be able to train the model.

```
model.train()
```

Predict with Model

Note: This assumes you follow through the tutorial and finished the “Upload Images”, “Create a Model”, and “Train the Model”. Otherwise you may not be able to make predictions with the model.

```
from clarifai.rest import ClarifaiApp
app = ClarifaiApp()
model = app.models.get('puppy')
model.predict_by_url('https://samples.clarifai.com/metro-north.jpg')
```

Instantiate an Image

```
from clarifai.rest import Image as ClImage
# make an image with an url
img = ClImage(url='https://samples.clarifai.com/dog1.jpeg')
# make an image with a filename
img = ClImage(filename='/tmp/user/dog.jpg')
# allow duplicate url
img = ClImage(url='https://samples.clarifai.com/dog1.jpeg', allow_dup_url=True)
# make an image with concepts
img = ClImage(url='https://samples.clarifai.com/dog1.jpeg', \
        concepts=['cat', 'animal'])

img = ClImage(url='https://samples.clarifai.com/dog1.jpeg', \
        concepts=['cat', 'animal'], \

        metadata={'id':123,

        'city':'New York'

        })
```

Bulk Import Images

If you have a large amount of images, you may not want to upload them one by one by calling app.inputs.create_image_from_url('https://samples.clarifai.com/dog1.jpeg')

Instead you may want to use the bulk import API.

Note: The max number images per batch is 128. If you have more than 128 images to upload, you may want to chunk them into 128 or less, and bulk import them batch by batch.

In order to use this, you have to instantiate Image() objects from various sources.

```
from clarifai.rest import ClarifaiApp
from clarifai.rest import Image as ClImage
# assume there are 100 urls in the list
5images = []
for url in urls:
img = ClImage(url=url)
images.append(img)
app.inputs.bulk_create_images(images)
```

Search the Image

Note: This assumes you follow through the tutorial and finished the "Upload Images" Otherwise you may not be able to search

```
from clarifai.rest import ClarifaiApp
app = ClarifaiApp()
app.inputs.search_by_annotated_concepts(concept='my puppy')
app.inputs.search_by_predicted_concepts(concept='dog')
app.inputs.search_by_image(url='https://samples.clarifai.com/dog1.jpeg')
app.inputs.search_by_metadata(metadata={'key':'value'})
```

3.3 Basic Concepts:

This page lists a few basic concepts used in the Clarifai API.

Image

Image is straightforward. It represents a picture in digital format.

In Clarifai API, an image could be represented by an url, a local filename, an opened io stream, raw bytes of the image, or bytes encoded in base64.

There is aImage() class to construct an Image object in Clarifai API. Along with the image bytes, an image could also be associated with an unique ID, as well as one or more concepts indicating what it is and what is it not.Image object could be used in uploading, prediction, as well as search.

Input

Input is a generalized image, because input could be image, or video in the near future. In the current API, we use Image and Input interchangeably because Image is the only Input type the API supports.

Similarly, input could be associated with ID and concepts. And Input is used in uploading, prediction, and search.

Concept

Concept represents a class to associate with images. It could be a concrete concept like “dog” or “cat”, or a abstract concept like “love”. It is the output of the concept Models. Clarifai Concept has an unique ID, and a name. Sometimes the concept name is also referred to label or tag for the predictions. They are interchangeably used in some places.

Model

Model is a machine learning algorithm that takes input, such as image or video, and outputs some results for prediction. For custom training, the models trained are all concept models. The output of a concept model is a list of concepts in prediction, associated with probabilities for confidence of the prediction.

Image Crop

Image crop is the definition of the crop box within an image. We use this in visual search so user does not have to crop an image before the search.

We use percentage coordinates instead of pixel coordinates to specify the crop box.

A four-element-tuple represents a crop box, in (top_y, left_x, bottom_y, right_x) order.

So a (0.3, 0.2, 0.6, 0.4) represents a box horizontally spanning from 20%-40% and vertically spanning 30%-60% of the image, measured from the top left corner.

Installation Guide Install Clarifai python library

Tutorial Getting started with Clarifai API using python client

Basic Concepts Getting familiar with basic concepts used in Clarifai API

3.4 API Reference

This is the API Reference documentation extracted from the source code.

Application

class clarifai.rest.client.**ClarifaiApp**(app_id=None,app_secret=None, base_url=None, api_key=None, quiet=True, log_level=None)

Clarifai Application Object

This is the entry point of the Clarifai Client API. With authentication to an application, you can access all the models, concepts, and inputs in this application through the attributes of this class.

To access the models: use app.models

To access the inputs: use app.inputs

To access the concepts: use app.concepts

check_upgrade()

Check for a client upgrade. If the client has been installed for more than one week, the check will be trig-gered. If the newer version is available, a prompt message will pop up as a warning message in STDERR. The API call will not be paused or interrupted.

tag_files(files,model_name='general-v1.3',model_id=None)

tag files on disk with user specified models by default tagged by 'general-v1.3' model

Parameters

files – a list of local file names for tagging. The max length of the list is 128, which isthe max batch size

model – the model name to tag with. The default model is general model for generaltagging purpose

Returns the JSON string from the predict call

Examples

```
files = ['/tmp/metro-north.jpg',
'/tmp/dog2.jpeg']
app.tag_urls(files)
```

tag_urls(urls,model_name='general-v1.3',model_id=None)

tag urls with user specified models by default tagged by 'general-v1.3' model

Parameters

urls– a list of URLs for tagging. The max length of the list is 128, which is the maxbatch size.

model – the model name to tag with. The default model is general model for generaltagging purpose

Returns the JSON string from the predict call

Examples

```
urls = ['https://samples.clarifai.com/metro-north.jpg',
'https://samples.clarifai.com/dog2.jpeg']
app.tag_urls(urls)
```

wait_until_inputs_delete_finish()

Block until a current inputs deletion operation finishes

The criteria for unblocking is 0 inputs returned from GET /inputs

Parameters **void** –

Returns None

wait_until_inputs_upload_finish(max_wait=666666)

Block until the inputs upload finishes

The criteria for unblocking is 0 “to_process” inputs from GET /inputs/status

Parameters **void** –

Returns None

wait_until_models_delete_finish()

Block until the inputs deletion finishes

The criteria for unblocking is 0 models returned from GET /models

Parameters **void** –

Returns None

Concepts

class clarifai.rest.client.**Concepts**(api)

bulk_create(concept_ids,concept_names=None)

Bulk create concepts

When the concept name is not set, it will be set as the same as concept ID.

Parameters

concept_ids– a list of concept IDs, in sequence

concept_names– a list of corresponding concept names, in the same sequenc

Returns A list of Concept objects

Examples

```
app.concepts.bulk_create(['id1', 'id2'], ['cute cat', 'cute dog'])
```

bulk_update(concept_ids,concept_names,action='overwrite')

Patch multiple concepts

Parameters

concept_ids– a list of concept IDs, in sequence

concept_names– a list of corresponding concept names, in the same sequence

Returns the new Concept object

Examples

```
app.concepts.bulk_update(concept_ids=['myid1', 'myid2'], concept_names=[
‹→'name2', 'name3'])
```

create(concept_id,concept_name=None)

Create a new concept

Parameters

concept_id– concept ID, the unique identifier of the concept

concept_name– name of the concept. If name is not specified, it will be set to the sameas the concept ID

Returns the new Concept object

get(concept_id)

Get a concept by id

Parameters **concept_id** – concept ID, the unique identifier of the concept

Returns If found, return the Concept object. Otherwise, return None

Examples

```
app.concepts.get('id')
```

get_all()

Get all concepts associated with the application

Parameters **void** –

Returns all concepts in a generator function

get_by_page(page=1,per_page=20)

get concept with pagination

Parameters

page – page number

per_page– number of concepts to retrieve per page

Returns a list of Concept objects

Examples

```
for concept in app.concepts.get_by_page(2,10):
print concept.concept_id
```

search(term,lang=None)

search concepts by concept name with wildcards

Parameters

term – search term with wildcards allowed

lang– language to search, if none is specified the default for the application will be used

Returns a generator function with all concepts pertaining to the search term

Examples

```
app.concepts.search('cat')
# search for Chinese label name
app.concepts.search(u'*', lang='zh')
```

update(concept_id,concept_name,action='overwrite')

Patch concept

Parameters

• **concept_id** – id of the concept

• **concept_name** – the new name for the concept

Examples

```
app.inputs.create_image(Image(url='https://samples.clarifai.com/metro-‹ → north.jpg'))
```

create_image_from_base64(base64_bytes,image_id=None, concepts=None, not_concepts=None, crop=None, metadata=None, geo=None, allow_duplicate_url=False)

create an image by base64 bytes

Parameters

base64_bytes – base64 encoded image bytes

image_id– ID of the image

concepts – a list of concept names this image is associated with

not_concepts– a list of concept names this image is not associated with

crop – crop information, with four corner coordinates

metadata – meta data with a dictionary

geo – geo info with a dictionary

allow_duplicate_url– True or False, the flag to allow a duplicate url to be im-ported

Returns the Image object that just got created and uploaded

Examples

```
app.inputs.create_image_bytes(base64_bytes="base64 encoded image bytes...
‹→")
```

create_image_from_bytes(img_bytes,image_id=None,concepts=None,not_concepts=None,

crop=None, metadata=None, geo=None, al-low_duplicate_url=False)

create an image by image bytes

Parameters

img_bytes– raw bytes of an image

image_id– ID of the image

concepts – a list of concept names this image is associated with

not_concepts– a list of concept names this image is not associated with

crop – crop information, with four corner coordinates

metadata – meta data with a dictionary

geo – geo info with a dictionary

allow_duplicate_url– True or False, the flag to allow a duplicate url to be im-ported

Returns the Image object that just got created and uploaded

Examples

```
app.inputs.create_image_bytes(img_bytes="raw image bytes...")
```

create_image_from_filename(filename,image_id=None,concepts=None,not_concepts=None, crop=None, metadata=None, geo=None, al-low_duplicate_url=False)

create an image by local filename

Parameters

filename – local filename

image_id– ID of the image

concepts – a list of concept names this image is associated with

not_concepts– a list of concept names this image is not associated with

app.inputs.create_image_filename(filename="a.jpeg")

crop – crop information, with four corner coordinates

metadata – meta data with a dictionary

geo – geo info with a dictionary

allow_duplicate_url– True or False, the flag to allow a duplicate url to be im-ported

Returns the Image object that just got created and uploaded

Examples

```
app.inputs.create_image_filename(filename="a.jpeg")
```

create_image_from_url(url,image_id=None,concepts=None,not_concepts=None,crop=None, metadata=None, geo=None, allow_duplicate_url=False)

create an image from Image url

Parameters

url– image url

image_id– ID of the image

concepts – a list of concept names this image is associated with

not_concepts– a list of concept names this image is not associated with

crop – crop information, with four corner coordinates

metadata – meta data with a dictionary

geo – geo info with a dictionary

allow_duplicate_url– True or False, the flag to allow a duplicate url to be im-ported

Returns the Image object that just got created and uploaded

Examples

```
app.inputs.merge_concepts('id', ['cat', 'kitty'], ['dog'])
```

merge_metadata(input_id,metadata)

merge metadata for the image

This is to merge/update the metadata of the given image

Parameters

input_id– the unique ID of the input

metadata – the metadata dictionary

Examples

```
# merge the metadata
# metadata will be appended to the existing key/value pairs
app.inputs.merge_metadata('id', {'key1':'value1', 'key2':'value2'})
```

merge_outputs_concepts(input_id,concept_ids)

Merge new concepts into the outputs predictions. The concept ids must be present in your app

Parameters

input_id– the unique identifier of the input

concept_ids– the list of concept ids to be merged

Returns the patched input in JSON object

remove_outputs_concepts(input_id,concept_ids)

Remove concepts from the outputs predictions. The concept ids must be present in your app

Parameters

input_id– the unique identifier of the input

concept_ids– the list of concept ids to be removed

Returns the patched input in JSON object

search(qb,page=1,per_page=20,raw=False)

search with a clarifai image query builder

WARNING: this is the advanced search function. You will need to build a query builder in order to use this.

There are a few simple search functions: search_by_annotated_concepts()

search_by_predicted_concepts() search_by_image() search_by_metadata()

Parameters

qb– clarifai query builder

raw – raw result indicator

Returns a list of Input/Image object

search_by_annotated_concepts(concept=None,concepts=None,value=True,values=None,concept_id=None, concept_ids=None, page=1, per_page=20, raw=False)

search using the concepts the user has manually specified

Parameters

concept – concept name to search

concepts – a list of concept name to search

concept_id– concept IDs to search

concept_ids– a list of concept IDs to search

value – whether the concept should be a positive tag or negative

values – the list of values corresponding to the concepts

page – page number

per_page– number of images to return per page

raw – raw result indicator

Returns a list of Image objects

Examples

```
model = self.app.models.get('model_id')
model.update(model_name="new_model_name")
model.update(concepts_mutually_exclusive=False)
model.update(closed_environment=True)
model.update(concept_ids=["bird", "hurd"])
model.update(concepts_mutually_exclusive=True, concept_ids=["bird", "hurd ‹ → "])
```

Search Syntax

class clarifai.rest.client.**SearchTerm**

Clarifai search term interface. This is the base class for InputSearchTerm and OutputSearchTerm

It is used to build SearchQueryBuilder

class clarifai.rest.client.**InputSearchTerm**(url=None, input_id=None, concept=None, con-

cept_id=None,value=True, metadata=None,

geo=None)

Clarifai Input Search Term for an image search. For input search, you can specify search terms for url string match, input_id string match, concept string match, concept_id string match, and geographic information. Value indicates whether the concept search is a NOT search

Examples

```
# search for url, string match
InputSearchTerm(url='http://blabla')
# search for input ID, string match
InputSearchTerm(input_id='site1_bla')
# search for annotated concept
InputSearchTerm(concept='tag1')
# search for not the annotated concept
InputSearchTerm(concept='tag1', value=False)
# search for metadata
InputSearchTerm(metadata={'key':'value'})
# search for geo
InputSearchTerm(geo=Geo(geo_point=GeoPoint(-40, 30), geo_limit=GeoLimit(
‹ → 'withinMiles', 10)))
```

class clarifai.rest.client.**OutputSearchTerm**(url=None, base64=None, input_id=None,

concept=None, concept_id=None, value=True,

crop=None)

Clarifai Output Search Term for image search. For output search, you can specify search term for url, base64, and input_id for visual search, or specify concept and concept_id for string match. Value indicates whether the concept search is a NOT search

Examples

```
    # search for visual similarity from url
OutputSearchTerm(url='http://blabla')
# search for visual similarity from base64 encoded image
OutputSearchTerm(base64='sdfds')
```

```
# search for visual similarity from input id
OutputSearchTerm(input_id='site1_bla')
# search for predicted concept
OutputSearchTerm(concept='tag1')
# search for not the predicted concept
OutputSearchTerm(concept='tag1', value=False)
```

class clarifai.rest.client.**SearchQueryBuilder**(language=None)

Clarifai Image Search Query Builder

This builder is for advanced search use ONLY.

If you are looking for simple concept search, or simple image similarity search, you should use one of the existing functions search_by_annotated_concepts, search_by_predicted_concepts, search_by_image or search_by_metadata

Currently the query builder only supports a list of query terms with AND. InputSearchTerm and Output-SearchTerm are the only terms supported by the query builder

Examples

```
qb = SearchQueryBuilder()
qb.add_term(term1)
qb.add_term(term2)
app.inputs.search(qb)
# for search over translated output concepts
qb = SearchQueryBuilder(language='zh')
qb.add_term(term1)
qb.add_term(term2)
app.inputs.search(qb)
```

add_term(term)

add a search term to the query. This can search by input or by output. Construct the term argument with an InputSearchTerm or OutputSearchTerm object.

dict()

construct the raw query for the RESTful API

Authentication

class clarifai.rest.client.**Auth**(api)

Clarifai Authentication

This class is initialized as an attribute of the clarifai application object, accessed with app.auth

get_token()

get token string

Returns The token as a string

Exceptions

class clarifai.rest.client.**ApiError**(resource, params, method, response, api)

API Server error

class clarifai.rest.client.**ApiClientError**

API Client Error

class clarifai.rest.client.**UserError**

User Error.

3.5 Installation Guide:

As the Raspberry pi has the Input Ports which can be declared in two methods. One is has the GPIO ports and BCM Numbering. The Detailed Programming pins can be found in pinout.xyz .

As for this we are using python 2.Its indicates that the programming for this should be python 2.

To install python 2 in the raspberry pi in the we need to

```
./configure
make
sudo make install
```

use these command for installing of Pyhton 2.7.9 version. As we have many Python versions we are using python 2.

3.6 Webcamera Installation:

As we can are doing the Intelligence Employee Safety here we can are Providing the Safety to the employee. So while entering the employee into the factory or any Industry He / She have to be Provided Intelligence entering system.

So , We need to make smarter and have to Provide Safety to the employee It should capture the Image of the Person and make clarifier and it should have to open.

For that to Installing of camera to raspberry pi in Python 2 we use the command

Sudo apt-get install fswebcam

Here its capture the Image of the person who stands in front of the door and send its to the clarifier.

3.7 Motion Installation:

As the Installation of webcam it capture the image of the employee when each an every Person who enters the Factory/Industry. So that in openCV there is an another option that its record the Live video of the entrance of camera. So doing this here when the person came near the Door the camera capture the image of the Person and train it.

It is a continuously Process where it detect the Person and train. All the Process take step by step.

To do this type we have to install the motion command

Sudo apt-get install motion

Motion monitors an incoming camera stream and detects 'motion' by finding the pixel values that have changed from frame to frame. If a threshold (i.e. total pixels changed) is exceeded then Motion triggers a "motion detected" event and optionally creates a snapshot of the video stream. Motion includes an inbuilt webserver offering both a video stream. By Using the above command the user can install the Video Streaming motion in the raspberry pi.

The command window can use to install the above driver in the raspberry pi.

3.8 Open CV installation:

OpenCV was started at Intel in 1999 by **Gary Bradsky** and the first release came out in 2000. **Vadim Pisarevsky** joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle who won 2005 DARPA Grand Challenge. Later its active development continued under the support of Willow Garage, with Gary Bradsky and Vadim Pisarevsky leading the project. Right now, OpenCV supports a lot of algorithms related to Computer Vision and Machine Learning and it is expanding day-by-day.

Currently OpenCV supports a wide variety of programming languages like C++, Python, Java etc and is available on different platforms including Windows, Linux, OS X, Android, iOS etc. Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations.

OpenCV-Python is the Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language.

3.9 OpenCV Python:

Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation. And the support of Numpy makes the task more easier. **Numpy** is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV, which increases number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this.

Sudo apt-get install python opencv

3.10 Numpy Installation:

NumPy is a fundamental package that can be used to scientifically compute with Python. It is a matrix library for linear algebra. NumPy can also be used as an efficient multidimensional container of generic data. Arbitrary data types can be defined and used. NumPy is an extension of the Python programming language. It adds support for large multidimensional arrays and matrices, along with a large library of high-level mathematical functions that can be used to operate on these arrays. We will use NumPy arrays throughout this book to represent images and carry out complex mathematical operations. NumPy comes with many inbuilt functions for all of these operations. So, we do not have to worry about basic array operations. We can directly focus on the concepts and code for computer vision. All the OpenCV array structures are converted to and from NumPy arrays. So, whatever operations you perform in NumPy, you can combine NumPy with OpenCV.

Sudo pip install numpy

3.11 Jpg file:

As the camera is capturing the image of the person who enters the Industry. Where the capture pictures should have to save in the device. Because these images can also be a security purpose for the Industry where they can know weather who is entered into the Industry.

So, every Capture Image have to stored in the raspberry pi so that we are installing the .jpg file command.

Fswebcam –S 20 123.jpg

By installing the above command the Image which are capture by Camera are stored in the .jpgfile with the file names **123.jpg**

3.12 Text to Speech Installation:

pyttsx includes drivers for the following text-to-speech synthesizers. Only operating systems on which a driver is tested and known to work are listed. The drivers may work on other systems

In this Instance installation of Text to Speech here the contain of text is converted into Speech. So this gives the Announcement to the Person which he / She doesn't have the suitable requirements to enter into the Industry. .

Sudo pip install pyttsx

The above command is use to install python 2 text to speech version of the pyttsx.

3.13 Espeak Installation:

The python espeak is use to get the loud voice of the text to speech conversion. This can help to get the loud voice from the raspberrypi. It can be installed from the python text to speech service.

eSpeak has a lot of options to play with, but I'm perfectly happy with the defaults. You can read eSpeak's.

Sudo apt-get installespeak

These are some of the installation are used in this project for the requirements. By this we can conclude the project by implementing this at the industry for the employee safety.In some industries it is necessary for the workers to wear safety helmets and shoes while working. So to check weather workers are taking safety precautions or not we are proposing this system. We can train our classifier to identify helmet and safety shoes with clarifai API. There will be video streaming near the entry of the industries where we can detect if person is wearing a helmet and shoes. If he is wearing them then the door will be open ,if he is not wearing then we can restrict his entry and we can warn him to wear by giving desired warnings through speakers.

CHAPTER 4

SOFTWARE SPECTIFICATIONS

4.1 Python Software:

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

4.2 General Information:

Python 2.7 is the last major release in the 2.x series, as the Python maintainers have shifted the focus of their new feature development efforts to the Python 3.x series. This means that while Python 2 continues to receive bug fixes, and to be updated to build correctly on new hardware and versions of supported operated systems, there will be no new full feature releases for the language or standard library.

However, while there is a large common subset between Python 2.7 and Python 3, and many of the changes involved in migrating to that common subset, or directly to Python 3, can be safely automated, some other changes (notably those associated with Unicode handling) may require careful consideration, and preferably robust automated regression test suites, to migrate effectively.

Some key consequences of the long-term significance of 2.7 are:

As noted above, the 2.7 release has a much longer period of maintenance when compared to earlier 2.x versions. Python 2.7 is currently expected to remain supported by the core development team (receiving security updates and other bug fixes) until at least 2020 (10 years after its initial release, compared to the more typical support period of 18–24 months).

As the Python 2.7 standard library ages, making effective use of the Python Package Index (either directly or via a redistributors) becomes more important for Python 2 users. In addition to a wide variety of third party packages for various tasks, the available packages include backports of new modules and features from the Python 3 standard library that are compatible with Python 2, as well as various tools and libraries that can make it easier to migrate to Python 3. The Python Packaging User Guide provides guidance on downloading and installing software from the Python Package Index.

While the preferred approach to enhancing Python 2 is now the publication of new packages on the Python Package Index, this approach doesn't necessarily work in all cases, especially those

related to network security. In exceptional cases that cannot be handled adequately by publishing new or updated packages on PyPI, the Python Enhancement Proposal process may be used to make the case for adding new features directly to the Python 2 standard library. Any such additions, and the maintenance releases where they were added, will be noted in the New Features Added to Python 2.7 Maintenance Releases section below.

For projects wishing to migrate from Python 2 to Python 3, or for library and framework developers wishing to support users on both Python 2 and Python 3, there are a variety of tools and guides available to help decide on a suitable approach and manage some of the technical details involved. The recommended starting point is the Porting Python 2 Code to Python 3.

4.3 Python IDE Overview:

4.3.1 Introduction

We'll be using Python in the workshop, and it'll save lots of time if, before arriving at the workshop, everyone has this installed and is familiar how to open, edit, and run a script (which is just a text file) using Python.

Furthermore, we'll be using IDLE, Python's own IDE (Integrated Development Environment) — combined source code editor and Python interpreter GUI. The workshop does not rely on any of its specific features, but it makes working with Python on Windows much easier and provides a multi-platform basis for examples.

Install Python (and IDLE)

There's a new major release of Python out, version 3

4.3.2 Windows Installation

Python and IDLE are not installed by default.

- Browse to http://www.python.org/download.
- Look for the Windows downloads, choose the one appropriate for your architecture (32-bit or 64-bit). At the time of writing, the choices are:
  - **32-bit**: Python 2.7.3 Windows Installer
- Run the installer and click through the prompts. Default options are usually just fine. This installs IDLE, too, by default.
- Browse to your *WORKSHOP* directory.
- Right-click in the empty space, choose New -> Text Document, name it idle.bat (accept the warning about file extensions).
- Edit the file by right-clicking and choosing Edit.
- Enter the single line:

- C:\Python27\Lib\idlelib\idle.bat
- Close the file.

You should now be able to double-click idle.bat to open IDLE.

4.3.3 Try IDLE:

You should now have an IDLE session open that looks something like this:

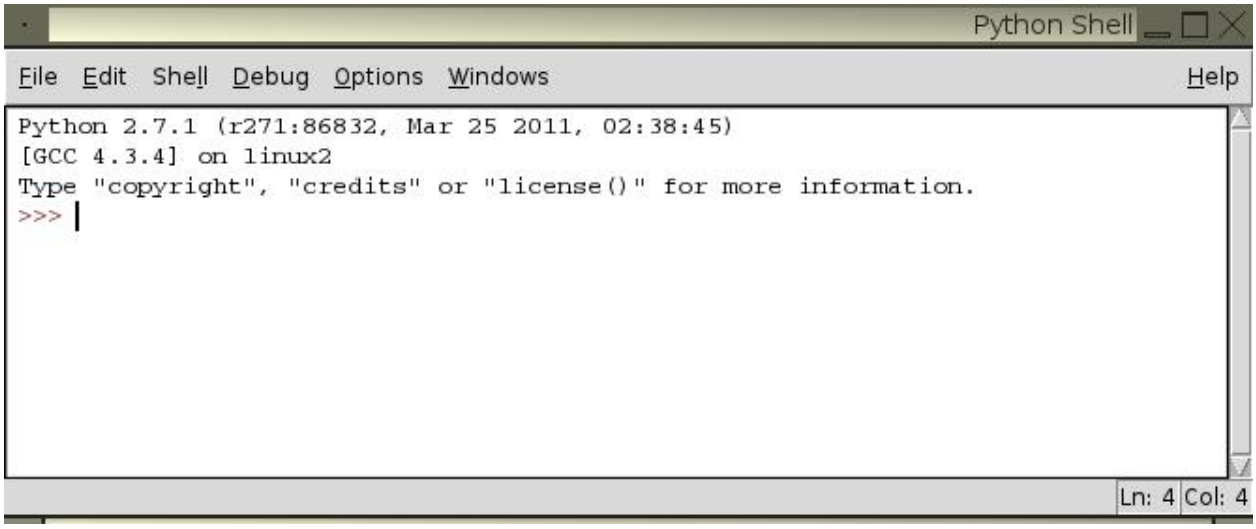

Figure 5: Python Shell

This is where we'll pick up the workshop. If you're feeling ambitious, try creating a Python script by clicking File -> New Window, which will open a text editor window, and enter the following line:

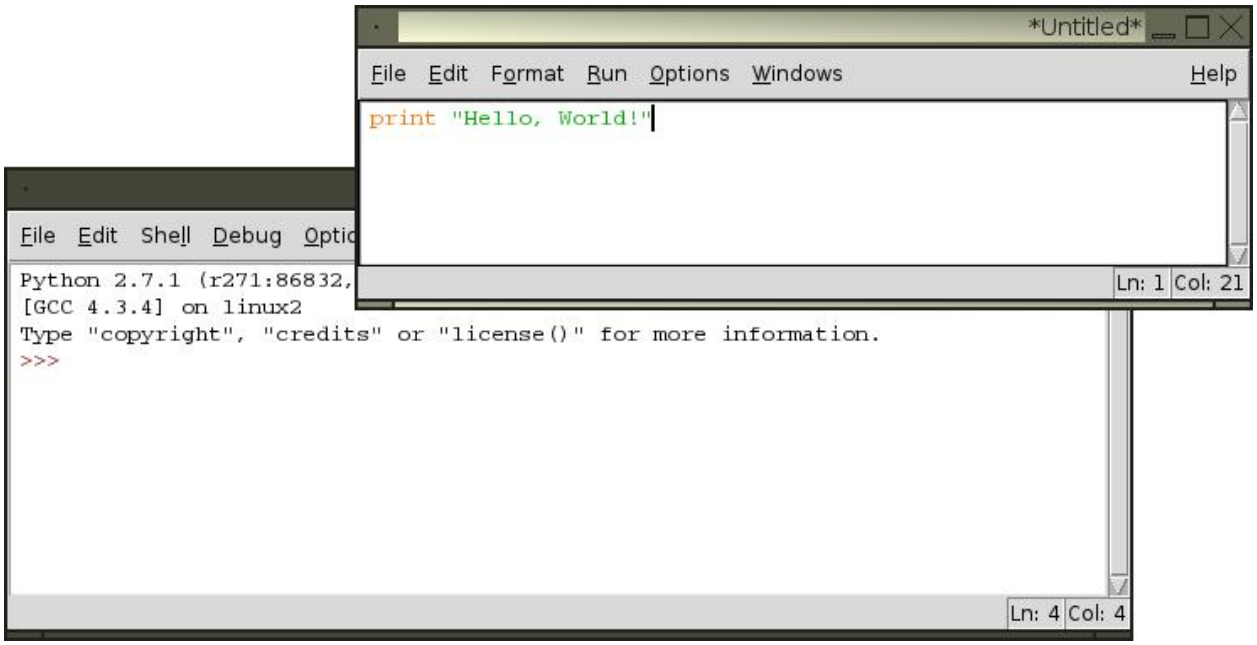

Figure 6: Python IDLE

Click File -> Save and enter hw.py for the filename. Then click Run -> Run Module to run the script:

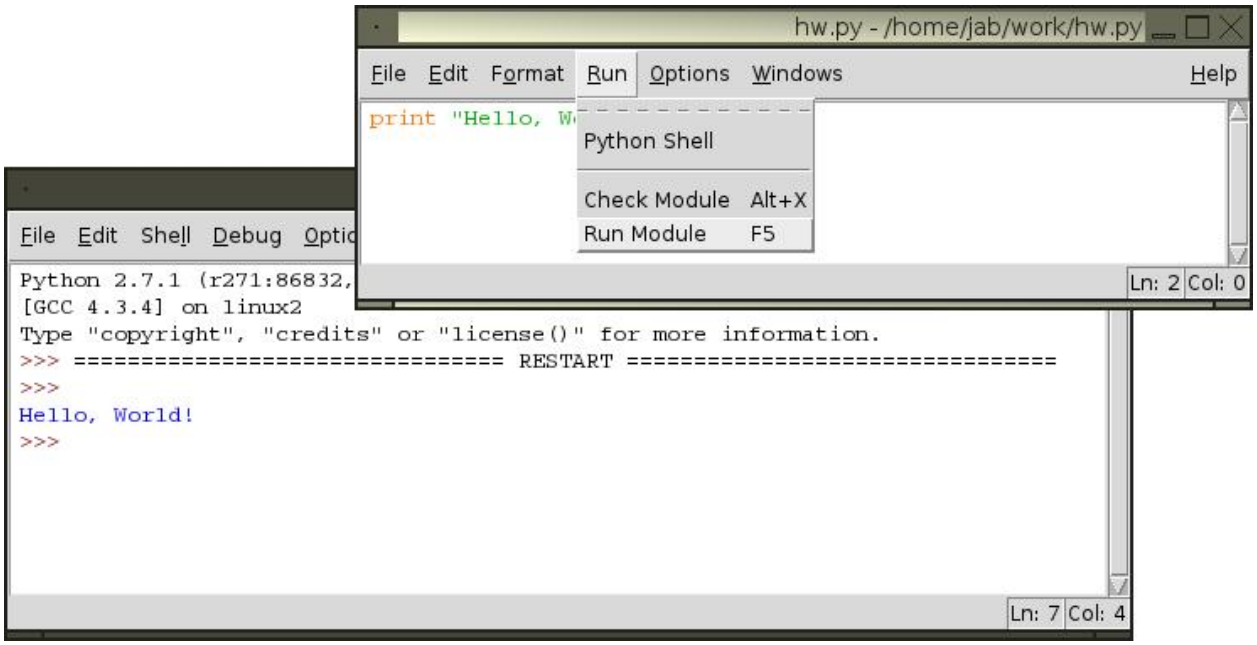

Figure 7: Python IDLE with Run Module

CHAPTER 5

IMPLEMENTATION

Intelligent Access Control System For Safety Critical Areas In Industries can be implemented by the Project with the help of the required block diagram.

5.1 Block Diagram:

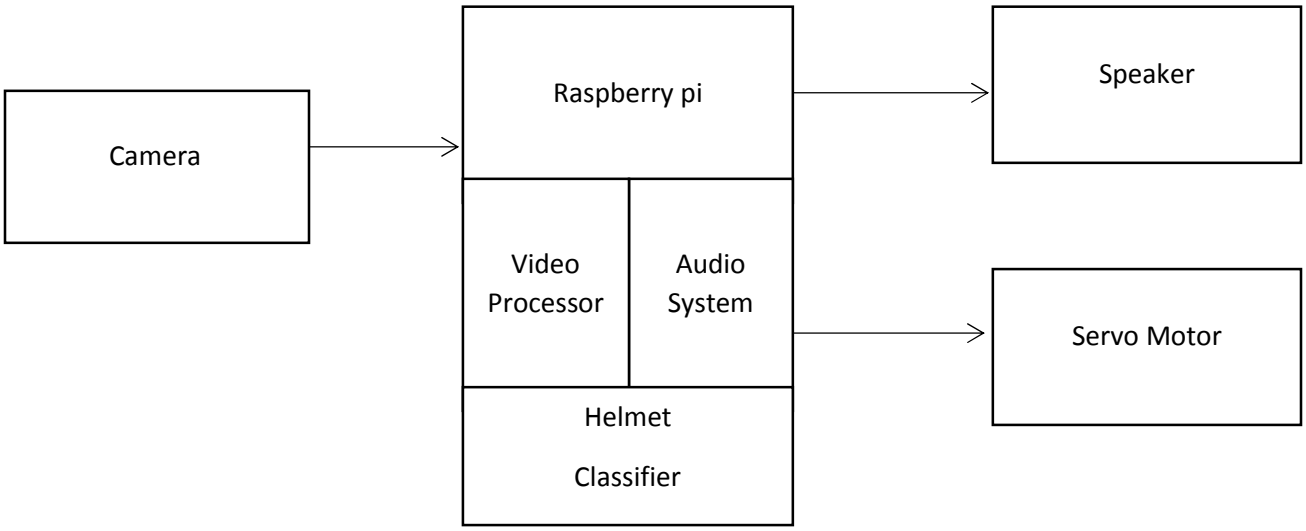

Figure 8: Implementation Block Diagram

This is the block diagram we can implement for Intelligent Access Control System for Safety.

5.2 Working Principle:

In this System, with the Use of Raspberry , we have designed the system that can check weather workers are taking safety precautions or not we are proposing this system. We can train our classifier to identify helmet and safety shoes with clarifai API. There will be video streaming near the entry of the industries where we can detect if person is wearing a helmet and shoes. If he is wearing them then the door will be open ,if he is not wearing then we can restrict is entry and we can warn him to wear by giving desired warnings through speakers..

CHAPTER 6

SIMULATION AND DESGIN

6.1 Steps connecting the Raspberrypi to Laptop:

This is the first step to connect the raspberry pi to the laptop. Firstly we need to provide the network the raspberry pi by sharing the wifi of the connected network.

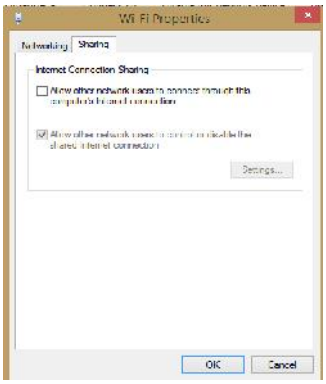

Figure 9: WIFI Sharing settings

Here we need to Allow the wifi shared network from the network shared Properties.

Then Open the command prompt from the Run and there type "**ping raspberrypi.mshome.net"**

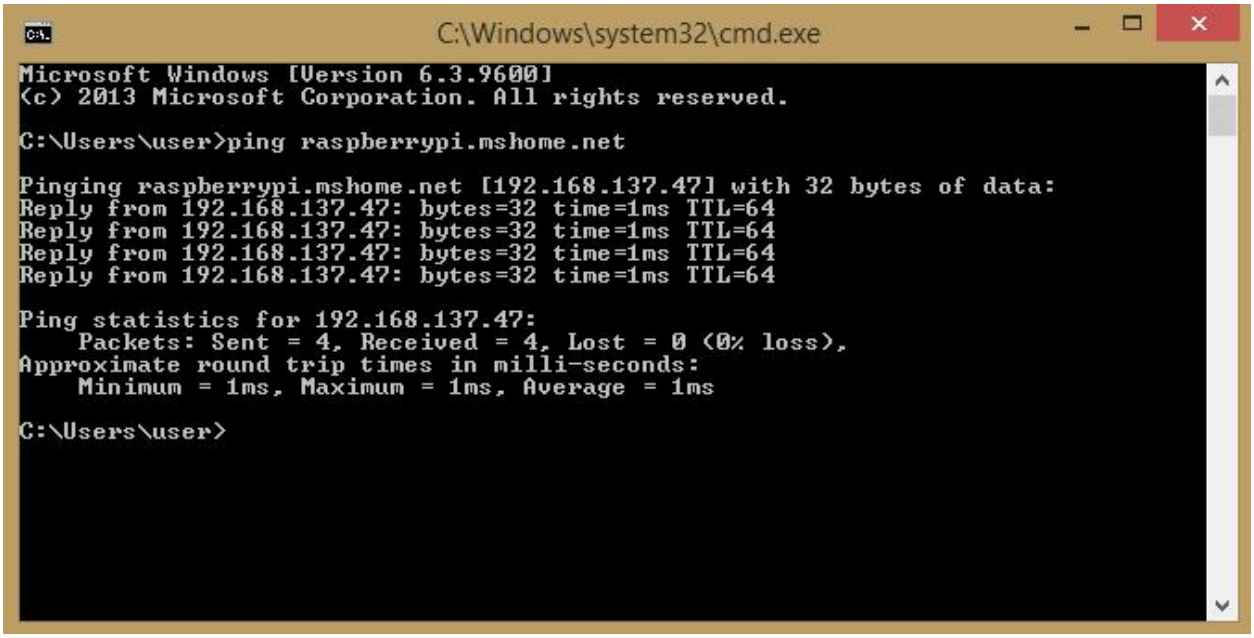

Figure 10: Command Prompt window

Here we have to get the 0% loss from the sharing . So that we can connect or raspberrypi to the laptop.

From here we can copy the ip address of ”ping statistics IP”

Firstly to connect the raspberry pi we need the third party server desktop. So we have the VNC Viewer to connect the raspberry.

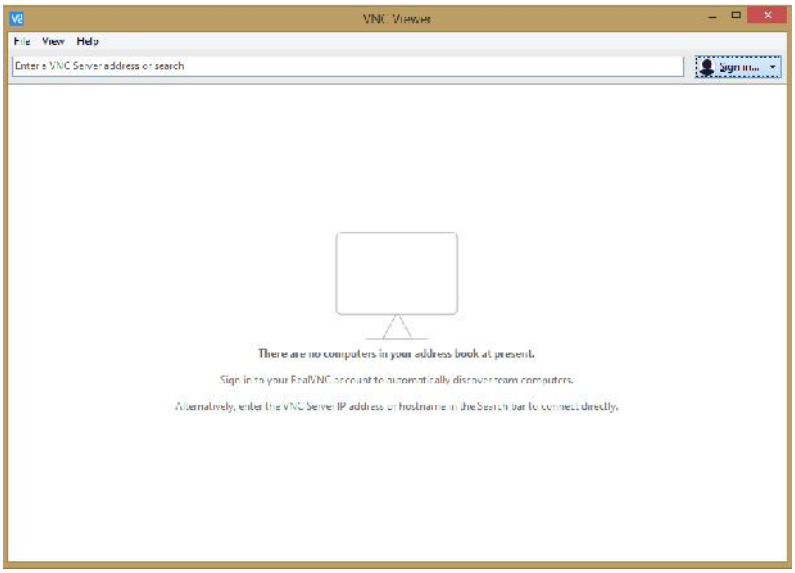

Figure 11: VNC Viewer Window

Here, Enter the IP address which you have copied from the command Prompt.

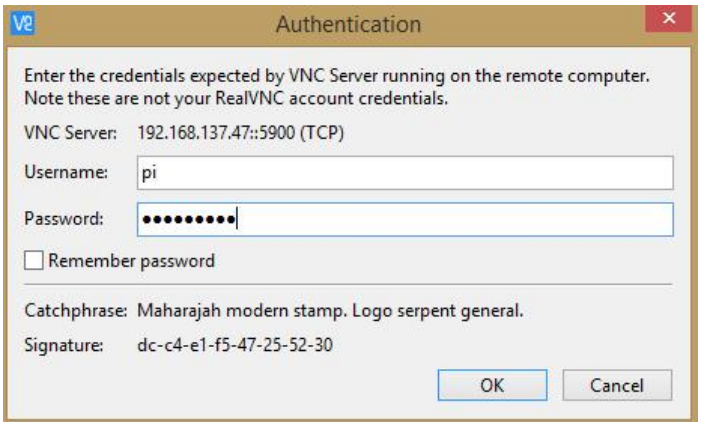

Figure 12: Raspberry Authentication

Then, we will get the Authentication details for connecting the raspberrypi. Here the default username and password is “**pi**” and “**raspberry”.** Then we will get into the raspberry pi desktop.

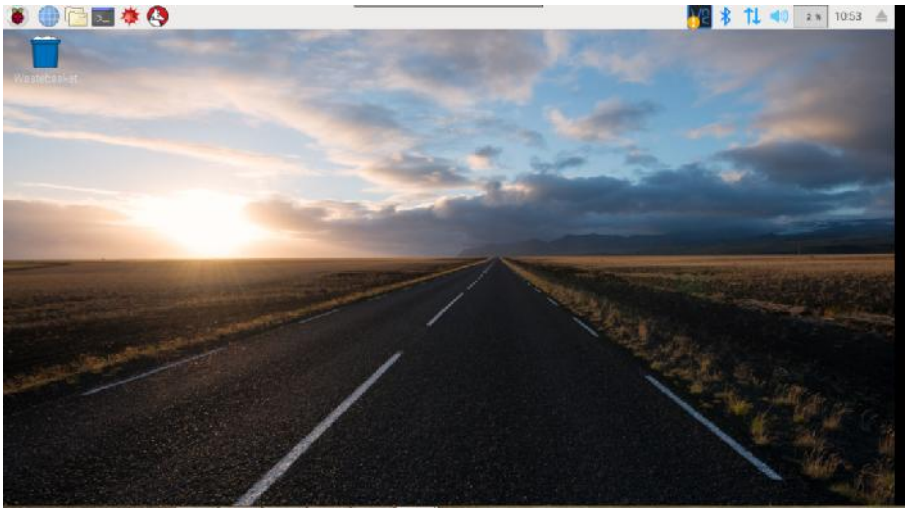

Figure 13: Raspberry welcome window

Here, we have the different command prompt. And we also we have the Bluetooth, Wi-Fi and Speaker option at the top right of the raspberry window. The user can use many of the inputs.

The symbol the raspbrian shows the applications of the raspberrypi.. and Programming options of the pi.

In the options of Preferences the user can gives the many chages from there as software and hardware. If any things wants to change the user input can change from the applied options.

6.2 Project Results:

As we are using the python 2 the raspbeery pi doesn't have the Idle to open directly. So to open the Python 2 IDLE go the command prompt and enter the idle so it is directly redirect to the workspace of python 2 window.

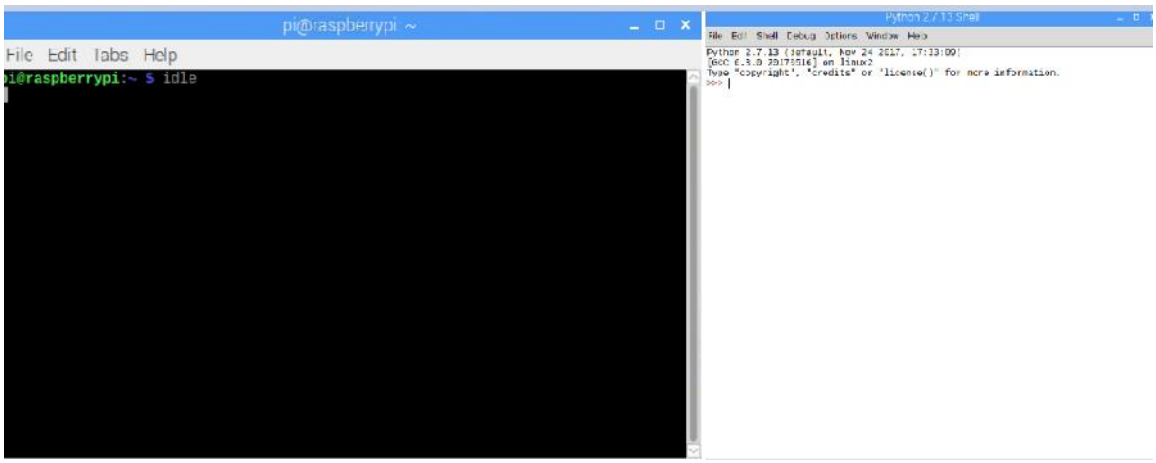

Figure 14: Python IDLE Shell

Then, Form here our programming part is stared. Here is the code in the below run in the pyhton2.

```
import numpy as np
import cv2
import time
from datetime import datetime
import pyttsx
from clarifai.rest import ClarifaiApp
from clarifai.rest import Image as ClImage
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(7, GPIO.OUT)
p = GPIO.PWM(7, 50)
p.start(7.5)


app1= ClarifaiApp(api_key='6cd50e41c156483cbfde2771293ffe10')
#app1= ClarifaiApp(api_key='a75a5d1ddf924167a62f700e800b31be')
model1 = app1.models.get('apparel')

face_cascade = cv2.CascadeClassifier('haar-face.xml')

def voice():
   engine = pyttsx.init()
engine.say(Voice)
engine.runAndWait()
time.sleep(2)

cap = cv2.VideoCapture(0)
print 'camera is initialized'

while True:
   Voice='please stand in the position...we are capturing your image'
voice()
time.sleep(5)
   ret, img = cap.read()
   if ret:
      gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

      faces = face_cascade.detectMultiScale(gray, 1.3, 5)

      for (x,y,w,h) in faces:
```

```
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
roi_gray = gray[y:y+h, x:x+w]
roi_color = img[y:y+h, x:x+w]
picname = datetime.now().strftime("%y-%m-%d-%H-%M")
picname = picname+'.jpg'
        cv2.imwrite(picname,img)
        print "Saving Photo"
        pic='/home/pi/Downloads/10.jpeg'
        print pic
        pic1='/home/pi/Downloads/'+picname
        print pic1
        image = ClImage(file_obj=open(pic1, 'rb'))
        response=model1.predict([image])
        data1 = response['outputs'][0]['data']['concepts']
        print data1
        for row in data1:
            if row['name'] == 'helmet':
                if row['value']>= 1.590712e-08:
                    x=1
                else:
                    print 'please wear the helmet'
time.sleep(2)
        t=model1.predict([image])
        data2 = t['outputs'][0]['data']['concepts']
        print data2
        d="Men"+"'"+"s"+" "+"Sandals"
        e="Men"+"'"+"s"+" "+"Boots"
        for row in data2:
            if row['name'] ==d:
                if row['value']>0.00001:
                    y=1
elif row['name'] ==e:
                if row['name']>0.00001:
                    y=1
            else:
                z=0
if(x==1 and y==1):
            print "you can enter inside"
p.ChangeDutyCycle(12.5) #180°
time.sleep(2)
p.ChangeDutyCycle(7.5)
            Voice = 'you   can   enter   inside'
voice()

        else:
```

```
            print "your     entry is restricted"
            Voice = 'please     wear     the     shoes  and   helmet  to  enter'
voice()
time.sleep(10)

   cv2.imshow('img',img)
time.sleep(0.1)
   k = cv2.waitKey(30) & 0xff
   if k == 27:
      break

cap.release()
cv2.destroyAllWindows()
```

Then run the code from the rum Module. Here the classifier API key have to enter in the Classifier key. Where we have trained our classifier in the classifier.

Where it can train the images using the camera of the connected raspberry pi.

Figure 15: Hardware components Connections

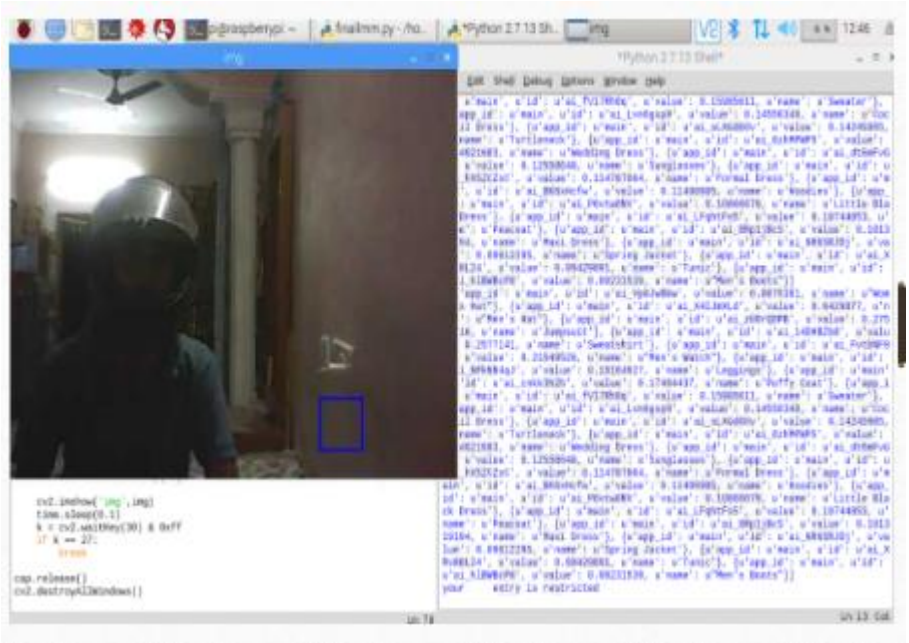

Figure 16: Output Observed

CHAPTER 7

CONCLUSION

In this paper, a novel methodology is proposed for programmed de-tection of protective cap utilizes for development security utilizing PC vision and machine learning procedures. The proposed framework has two noteworthy parts: one section joins recurrence area data of the picture with a famous human identification algorithm HOG for human (i.e., development laborer) location; the other part works for head protector use recognition consolidating shading data and Circle Hough Transform (CHT) As of now, our framework can identify head protectors made out of some specific hues, for example, yellow, blue, red, and white. As an expansion of this work, we mean to make the framework versatile to recognize head protectors with different hues. In future, the framework will be made well equipped for separating between typical top and protective cap, as the proposed framework demonstrates low execution for this situation. Additionally, we intend to apply some profound learning procedures for enhancing the general precision of the framework. Likewise, applying chest area looking calculation as opposed to distinguishing entire human as object of intrigue can enhance the protective cap recognition exactness.

REFERNCES:

[1]Y.-S. Ahn, J. F. Bena, and A. J. Bailer. Comparison of unintentional fatal occupational injuries in the Republic of Korea and the United States. In Injury Prevention, Vol. 10(4), 199-205, 2004.

Bureau of Labor Statistics. Safety and healthdangerous jobs. Compensa- tion and Working Conditions. In http://www.bls.gov/iif/oshwc/cfar0020

.pdf, 1997.

Can Stock Photo. Roofing stock photos and images. In http://www. canstockphoto.com/images-photos/roofing.html

Can Stock Photo. Roofing stock photos and images. In http://www. canstockphoto.com/images-photos/roofing.html#start:75

Can Stock Photo. Roofing stock photos and images. In http://www. canstockphoto.com/images-photos/roofing.html#rows:200

Center for Construction Research and Training. The Construction Chart Book: The U.S. Construction Industry and Its Workers. 5th edition ed. 2013, Silver Spring, MD: CPWR.

[7]N. Dalal, and B. Triggs. Histograms of oriented gradients for human detection. In CVPR, 2005.

[8]DeBruyne, N.F. and A. Leland. American War and Military Operations Casualties: Lists and Statistics 2015. In Congressional Research Service, 2015.

[9]L. Ding, and A. Goshtasby. On the Canny edge detector. In Pattern Recognition, Vol. 34(3), 721–725, 2001.

[10]S. Du, M. Shehata, and W. Badawy. Hard hat detection in video sequences based on face features, motion and color information. In 3rd International Conference on Computer Research and Development (ICCRD), Vol. 4, 2011.

[11]J. Gavin. 2015 Construction Outlook: An Economic Recovery Finds Its Footing, 2015.

[12]D. Gavrila. Pedestrian detection from a moving vehicle. In ECCV, Vol.

2:37-49, 2000.

Getty Images. Construction images. In http://www.gettyimages.com

/search/more-like-this/91988434?assettype=image&excludenudity=true& family=creative&license=rf&sort=best

[14]Getty Images. Construction images. In http://www.gettyimages.com

Getty Images. Roof construction pictures and images. In http://www. gettyimages.com/photos/achillestarace?excludenudity=true&mediatype= photography&page=1&phrase=roof%20construction&sort=mostpopular

[16]O. Hamdoun, F. Moutarde, B. Stanciulescu, and B. Steux. Person re- identification in multi-camera system by signature based on interest point descriptors collected on short video sequences. In ICDSC, 2008.

[17]Jaselskis, E.J., S.D. Anderson, and J.S. Russell. Strategies for achieving excellence in construction safety performance. In Journal of Construction Engineering and Management, 122(1): 6170, 1996.

[18]J. Han, M. Kamber, and J. Pei. Data Mining: Concepts and Techniques (3rd ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.

[19]R. L. Hsu, M. Abdel-Mottaleb, and A. K. Jain. Face detection in color images. In IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 24(5), 696-706, 2002.

[20]R. Lienhart, and J. Maydt. An extended set of Haar-like features for rapid object detection. In Proc. of IEEE International Conference of Image Processing, Vol. 1, 900-903, 2002.

[21]A. Micilotta, E. Ong, and R. Bowden. Detection and Tracking of Humans by Probabilistic Body Part Assembly. In Proceedings on British Machine Vision Conference, Vol. 1, 429-438, 2005.

[22]K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In ECCV, Vol. 1, 69-81, 2004.

[23]D. T. Nguyen, W. Li, and P. O. Ogunbona. Human detection from images and videos: A survey. In Pattern Recognition, Vol. 51, 148-175, 2016. [24]M. Nixon, and A. Aguado. Feature Extraction and Image Processing,

1st Edition. In Elsevier, Burlington, MA, USA, 2002. [25]Occupational Safety & Health Administration. Constructions Fatal Four.

In https://www.osha.gov/oshstats/commonstats.html, 2013.

[26]M.-W. Park, E. Palinginis, and I. Brilakis. Detection of construction workers in video frames for automatic initialization of vision trackers. In Construction Research Congress, 2012.

[27]Public Citizen. The Price of Inaction: A Comprehensive Look at the Costs of Injuries and Fatalities in Maryland's Construction Industry, 2012.

[28]D. Ramanan, D. Forsyth, and A. Zisserman. Strike a pose: tracking people by finding stylized poses. In CVPR, 1, 2005.

[29]T. Roberts, S. McKenna, and I. Ricketts. Human pose estimation using learnt probabilistic region similarities and partial configurations. In ECCV, Vol. 4, 291-303, 2004.

[30]M. D. Rodriguez, and M. Shah. Detecting and segmenting humans in crowded scenes. In Proceedings of the 15th international conference on Multimedia, 2007.

[31]R. Ronfard, C. Schmid, and B. Triggs. Learning to parse pictures of people. In ECCV, Vol. 4, 700-714, 2002.

[32]K. Sabottka, and I. Pitas. Segmentation and tracking of faces in color images. In Proc. of International Conference on Automatic Face and Gesture Recognition, 236-241, 1996. Shutter Stock. Roofing stock –

photos, illustrations, and vector rrt. In http://www.shutterstock.com/s/roofing/search.html?page=1&thumb size=mosaic

Shutter Stock. Roofing stock photos, illustrations, and vector rrt. In http://www.shutterstock.com/s/roofing/search.html?page=2&thumb size

=mosaic

[35]U.S. Bureau of Labor Statistics. National Census of Fatal Occupational Injuries in 2014 (Preliminary Results).

[36]P. Viola, M. Jones, and D. Snow. Detecting Pedestrians Using Patterns of Motion and Appearance. In IJCV, Vol. 63(2), 153-161, 2005.

[37]P. Viola, and M. Jones. Rapid object detection using a boosted cascade of simple features. In CVPR, Vol. 1, 511-518, 2001.

[38]C.-Y. Wen, S.-H. Chiu, J.-J. Liaw, and C.-P. Lu. The safety helmet detection for ATM's surveillance system via the modified Hough trans- form. In Proceedings of IEEE 37th Annual 2003 International Carnahan Conference on Security Technology, 2003.

[39]Y. Wu, and T. Yu. A Field Model for Human Detection and Tracking.

In CVPR, 28, 2006.

[40]H. K. Yuen, and J. Princen, and J. Illingworth, and J. Kittler. Compara- tive study of Hough transform methods for circle finding. In Image and vision computing Vol. 8(1), 71-77, 1990.

[41]L. Zhao, and C. Thorpe. Stereo-and neural network-based pedestrian detection. In IEEE Transactions on ITS, Vol. 1(3), 148-154, 2000.

[42]L. Zhao, and L. Davis. Closely Coupled Object Detection and Segmen- tation. In ICCV, Vol. 1, 2005.