Addis Ababa University

College of Natural Sciences

Developing on-site solid waste segregation framework through image processing and Low performance IoT technologies

Melak Gezahegne Zegeye

A Thesis Submitted to the Department of Computer Science in

Partial Fulfillment for the Degree of Master of Science in

Computer Science

Addis Ababa, Ethiopia

May 25, 2020

Addis Ababa University

College of Natural Sciences

Melak Gezahegne Zegeye

Advisor: *Dagmawi Lemma Gobena(PHD)*

This is to certify that the thesis prepared by *Melak Gezahegne*, titled: *Developing on-site solid waste segregation framework through image processing and Low performance IoT technologies* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

	Name	. Signature	. Date
Advisor:	Dagmawi Lemma (PhD)		
Examiner:			
Examiner:			

Abstract

Internet of things (IoT) often uses devices that are resource scared having very limited processing power and storage capacity. However, some scenarios demand resource intensive technology such as camera technology that usually requires high computational resource due to the images captured by camera requires an image processing. One of the scenario is on-site solid waste segregation. To provide an alternative solution for such scenario, a framework is developed based on the technology of IoT, Image processing, and Deep Learning. The framework is presented based on four-layered IoT architecture and Faster-RCNN architecture. In the framework, the camera (used to capture real-time visual information) and actuator (to perform physical actuation) are placed at the perception layer. The captured video transmitted to through wired or wireless via transport layer to processing layer. In the processing layer, the real-time object detection facilitated based on Faster-RCNN objected detection architecture and represent the detected object to the lightweight information(title) after successful detection. The title which actually represent the object is communicated through serial communication to the Actuation service provider placed at the application layer. The Actuation service provider running on the low-performance IoT device at application layer and delivers actuation service to physical actuation (perception layer) based on the title it receives from processing layer. The Physical actuation makes a physical action using Actuator IoT device. For the scenario of on-site waste segregation, the actuator is attached to the dustbin hatch and performs opening and closing of the dustbin to realize the segregation based on the command it receives form Actuation service provider. The implementation of the thesis based on Tensor Flow object detection API framework (to develop real-time object detection) and the Arduino Uno IDE(C++) to develop the actuation service provider embedded system. In addition, the IoT device hardware circuit was designed and simulated using Proteus Simulator. During the evaluation, the concept is simulated and experimentally tested based on the scenario of on-site solid waste segregation. We have trained the object detection model to detect the class of plastic and glass objects. The result shows that real-time actuation can be successfully accomplished if the object detection model successfully trained to detect the corresponding object (visual information).

Keywords: Internet of things, image processing, visual information, framework

Acknowledgments

First and foremost, praises and thanks to the God, the Almighty, for His showers of blessings throughout my research work to complete the research successfully.

Next, I would like to express my sincere gratitude to my Advisor Dr. Dagmawi Lemma, for his support, guidance, constructive supervision, insightful comments and suggestions during this research work and from whom I have learned enormously. His constant encouragement throughout the period of this study has been the greatest inducement for me. He has taught me the methodology to carry out the research and to present the research works as clearly as possible. It was a great privilege and honor to work and study under his guidance. I am extremely grateful for what he has offered me. I would also like to thank him for his patience, friendship, empathy, and great sense of humor.

Lastly, I am very much thankful to my brother Awoke Gezahegne and his wife Muluken Animaw sacrifices for educating and preparing me for my future.

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

IoT	Internet of Things
CNN	Convolutional Neural Network
RCNN	Regional Convolutional Neural Network
ROI	Region of Interest
RPN	Region Proposal Network
RISC	Reduced Instruction Set Computer
ARM	Advanced RISC Machines
IDE	Integrated Development Environment
LED	Light Emitting Diodes
PWM	Pulse Width Modulation
TWI	Two Wire Interface
SPI	Serial Peripheral Interface
PWM	Pulse Width Modulation
IP	Internet Protocol
RFID	Radio Frequency Identification
MQTT	Message Queuing Telemetry Transport
NFC	Near-field communication
DL	Deep Learning

Chapter 1 : Introduction

1.1 Background

Today the dramatic growth of technology is making people's life easier than ever. One of the innovative and recent technological trends is Internet of Things (IoT), which is the network of the "things" around us, such as vehicles, home appliances, and other items embedded with electronics, software, sensors, actuators, and connectivity enabling these things to compute and communicate to one another as well as, collect and exchange data [1]. As a result, the IoT has become main enable to deliver smart service by interconnecting the physical and virtual world.

Image processing (visual information processing) is another technology that becomes among the rapidly growing and widely used technologies with its applications in various aspects of a business. Image processing is mainly concerned with performing some operations on an image and get an enhanced image or extract some useful information from the given image/video. There are several techniques of image processing according to the requirement of the application. The techniques may go from traditional image processing to deep learning algorithms. The traditional image processing involves image acquisition, preprocessing, segmentation, feature extraction, classification and recognition [2]. Image processing can also include the use of deep learning (DL), which is based on training DL algorithms on images dataset instead of executing all steps in a scattered way. Yet, the current algorithms and techniques demand a resourceful environment considerable memory size and processing capacity – unlike the resource available in IoT.

So, in building IoT along with image processing, one challenge is the low-performance nature of IoT devices which is unable to proceed with big computation like image processing since it requires high computational resources. To smooth such an issue of IoT, cloud computing comes is often used to perform such high computational operations on the cloud and sends the result back to IoT devices.

However, cloud computing faces the challenge of bandwidth latency and storage costs. Later, technology such as fog and edge computing were developed to facilitate the task of preprocessing before sending that raw data to the cloud. In fact, both technologies used as

cloud support services [3, 4]. However, it's unfair to push every high computational need to the cloud regardless of the issue of the scenario. Besides, when cloud services are contracted, one way or the other additional features (e.g., security and management services) would be associated; compelling to have a thicker service environment.

So, use cases that do not afford a thicker service environment should have an intermediate solution that can run locally. For instance, to facilitate on-site solid waste segregation using IoT and image processing it can be accomplished locally without cloud neither fog computing.

1.2 Motivation

The initiation of this thesis is started from the current problems associated with waste management that aims to handle the waste generated that is rising and has become a major issue and global environmental concern. World Bank in 2016 disclosed that the worlds' cities generated 2.01 billion tons of solid waste, amounting to 0.74 kilograms per person per day. With rapid population growth and urbanization, annual waste generation is expected to increase by 70% from 2016 levels to 3.40 billion tones in 2050 [5]. Since the rate of urbanization is becoming high; the amount of waste that is produced per day is unexpectedly increased. Hence it is pertinent to put in place an efficient waste management scheme.

These days, most waste management strategies focus on the three Rs, reduce, reuse and recycle. But as waste is collected in a mixed way reusing and recycling could be challenging if not assisted with efficient segregation methods.

Since segregation is an important phase of waste processing (to recycle or reuse), municipal and/or waste management companies lost extra time, cost and resource on the separation of waste as recyclable and non-recyclable ones.

Segregation would be efficient and cost-effective if it is done on-site instead of deploying a complex segregation system at the landfill on the waste processing site. Labeling and coding dustbins are one means to support on-site segregation.

Yet, the effectiveness of such an approach depends on societal docility. However, most people either don't perceive the signage or ignore them for various reasons. This was the motivation of this work and we are motivated to apply the technology of IoT along with

image processing. And we believe that the solution can be applied for solving similar problems hence motivated to develop a framework.

1.3 Statement of the problem

IoT involves deployment of the smart, connected device and allows physical devices to collect data from their environment and transfer it over a network to another device, or to a central location for processing [6]. It consists of sensors, actuators, and communication protocol. Sensors used to collect physically measured data and generate a considerate output in the form of electrical signals. Actuators used to manipulate the IoT environment, as it can cause physical or logical actuation after getting some input from the sensor and transforms the input into tangible action. Communication protocol facilitates data communication.

In building IoT solutions, data collection usually facilitated through sensors. A sensor is used to measure a physical phenomenon (like temperature, pressure, and so on), which is analog types of information. Most of the data collected via a sensor can be processed at low-performance IoT devices. Because the analog types of information are relatively requiring small storage and processing cost. The sensor's data collection is based on the condition in the environment when some conditions meet the required analog value is captured and processed in IoT devices. Apart from analog information, there are activities in a certain environment that needs to collect visual information. The visual information is an image/video information collected by the camera from the environment. To build IoT solutions with visual information, real-time object detection is required. However, real-time object detection is computationally intensive. It can’t be done with usual IoT trends because of the low-performance nature of IoT devices and the resource requirement of real-time object detection is out of the capacity of IoT devices. To facilitate computationally intensive tasks with IoT, technology such as cloud computing is used.

So, for scenarios that require visual information processing, one of the solutions were sending the visual information to cloud computing. The computation results later sent back to IoT devices. This trend faces another challenge in which sending every high computation to the cloud was not satisfactory because of bandwidth latency, cost of computation and storage. Later cloud supporting technology produced such as fog and edge computing. The aim of edge or fog computing was to perform just like preprocessing computation instead of

sending every row data to the cloud [7]. In fact, both are not cloud-free. To use the cloud, fog or edge computing with IoT, it's better to depend on the scenarios some scenarios can be facilitated even below edge computing especially those scenarios that don't require the security and management services from the cloud.

For instance, activities like on-site waste segregation demand visual information as it is expected to facilitate by manually labeling, coding, and symbolling on the dustbin as metallic, glass, and paper. The aim was to segregate the recycle from non-recycle at the beginning of the collection. Truthfully, people do not respect the instructions on the dustbin knowingly or unknowingly. This scenario should have an intermediate solution without a cloud, fog or edge computing.

To resolve such issues, scenarios that don't require cloud services like security and management should have an intermediate framework that can run locally. The intermediate framework should run below edge computing along with IoT. So, to realize this the camera will collect visual information. The visual information captured through the camera can be further identified by image processing then real-time decision or actuation can be made based on image processing result. For instance, to accomplish on-site segregation, the camera visualizes the wastes at the time of collection and identifies the waste type in real-time through image processing. Based on the image processing result the actuation can be controlled. For example, a dustbin flap should integrate with the actuator, and the actuator plays the role of opening and closing the bin-hatch based on the image processing result. In this regard, on-site waste segregation can be technologically resolved. To execute this, we would intend to develop a framework.

We raise the following research questions to realize the proposed framework: -

- What are the required components to develop the framework?
- How to develop the framework where the IoT device is low performance working with resource scared processor along with image processing where high-performance computing is important and requires more resources in a framework?

1.4 Objective

General Objective

The main objective of this work is to develop a framework that enables IoT environment to locally run resource-intensive systems (e.g., image processing); and to show how low-performance IoT device consumes image processing to make real-time actuation upon visual information.

Specific Objectives

To meet the general objective, the thesis will include the following specific objectives: -

- Review works of literature to understand and refine the problem
- Design component-based framework for the problem identified
- Develop a prototype of the proposed framework
- Evaluate and test the developed framework.

1.5 Methods

In order to achieve the general and specific objectives of this thesis, the initial task is to understand and analyze the research domain. This is performed by conducting a comprehensive review of the works of literature relevant to the research topic. Though, relevant literature such as books, journal articles, conference papers and resources from the Internet that is related to IoT and, image processing is reviewed. To be more specific to our problem, comprehensive investigations published papers directly related to our work revised. So, the gaps and the strength of existing work are reviewed. This helps us, in order to assess what others have done in the area and to understand the problem better. Furthermore, methods such as data collection, tools, and algorithms are investigated to develop the prototype then we make an experiment to evaluate the performance of the proposed solution.

Tools: There are two categories of tools for the development of the proposed framework, one is for the development of visual information processing, the other is for the development of IoT hardware's simulation. For visual information processing, we used the Python programming language. Python provides advanced capabilities, such as array and matrix

manipulation, digital signal processing, and visualization [8]. As well, python has several libraries that easy to install and configure for processing visual information (image data). Python has several development environments, but we used anaconda because it has an advantage of several pre-installed default python's packages, and libraries and which is resource ready python development environment. Anaconda has several code editors, but we used, the anaconda - Jupyter notebook [9] development environment which benefits the following: -

- Supports to produces client-server application
- Allows you to edit and run the notebooks via a web browser
- The application can be executed on a PC without Internet access, or it can be installed on a remote server, where you can access it through the Internet

On the other hand, the IoT hardware simulation is designed in Proteus 8 Professional circuit design tool. The proteus helps to design the IoT hardware's as circuit-level which provides the function of real-world IoT hardware services. We used hardware such as Arduino Uno R3, Servo Motor, Wires, and COMPIM to simulate the study. In addition, we used the Arduino IDE tool for the development of an embedded system to power up the design circuit for the further reality of the simulation.

Algorithms and models: To develop visual information processing, the framework is found to be important. In this regard, we chose the TensorFlow object detection API framework, which is an open-source framework created by Google to develop different DL processes [10]. It is built on top of TensorFlow helps our real-time object detection component easy to construct, train and deploy object detection models. On the other hand, the portability, flexibility, and performance of TensorFlow provide advantageously for the implementation of real-time object detection [11]. TensorFlow runs on GPUs, CPUs, desktops, servers, and mobile computing platforms. So, it serves as a true portability feature. As well it is a highly flexible system that provides multiple models or multiple versions of the same model that can be served simultaneously. The architecture of TensorFlow is highly modular, which means you can use some parts individually or can use all the parts together. Such flexibility facilitates non-automatic migration to new models/versions. TensorFlow allows you to make the most of your available hardware with its advanced support for threads, asynchronous

computation, and queues. Just assign compute elements of your TensorFlow graph to different devices and let it manage the copies itself. It also facilitates you with the language options to execute your computational graph. TensorFlow iPython notebook (Jupiter notebook) helps in keeping codes, notes, and visualization in a logically grouped and interactive style. In general, in order to develop the proposed framework, Faster-RCNN Deep Learning(DL) algorithms and TensorFlow Object detection API framework were used.

Data collection: To train the DL algorithm relevant data is gathered and we collect image datasets. The data is important because the Faster-RCNN would be trained to have a knowledge of classes. Accordingly, real-time object detection is developed based on the TensorFlow object detection API framework.

1.6 Scope and limitations

The focus of this thesis is to develop a framework upon real-time object detection with low-performance IoT technology to realize the real-time actuation upon visual information locally. The study doesn't consider shielded visual information means the object should clearly be visualized to be run on the proposed framework. The framework would be demonstrated with the scenario of onsite-solid waste segregation. The framework realization for on-site solid waste segregation can be implemented for public sites such as airlines, hotels in which people's waste disposal trend is not usually guided based on manual disposal.

1.7 Application of results

The output of this study may applicable activities that require visual inspection and a real-time decision like on-site waste segregation. In addition, the study will be an input for further research.

1.8 Organization of the rest of the thesis

This thesis contains six chapters. *Chapter 1* describes an introduction to the research, which covers the research background, Motivation, statement of the problem, outline of the research objectives and concludes with an outline of the structure of the thesis. *Chapter 2* presents the state of the art of IoT including IoT overview, basic requirements in building IoT solution, challenges of using image processing in IoT solution will be presented in

detail. *Chapter 3* discusses related works that have significant relation with this thesis. Here several studies discussed and their strengths, weaknesses will be reviewed. The chapter will be concluded by a summary of those studies by pointing out our research work going to fill the gap of those studies. *Chapter 4* presents the design of the proposed solution, the framework will be designed and discussed in detail. *Chapter 5* here the proposed solution is implemented to realize the proof of concept presented in chapter 4. Evaluation of the work, and experimental analysis accomplished here. The last chapter, *Chapter 6*, summarizes the contributions made in the thesis and concludes based on the results obtained from the thesis work. Furthermore, new issues that have been surfacing while working on the thesis are suggested as future work.

Chapter 2 : Literature Review

This chapter discusses the state of the art of IoT and presents the main technological motorists, concepts, and challenges of IoT in different environments with different types of information. In addition, knowledge of image processing will be discussed in the perspective of visual information collecting, processing in building IoT.

2.1 Overview Internet of things

IoT could be grasped from the terms that drew-up of itself, “Internet” and “things” [12]. The “internet” implies real-world things that shall have Internet Protocol and connected for supply and accessing information to and from the environment. The “things” refers to generic objects. The generic object shall uniquely identify through Internet Protocol(IP). In another way, “things belonging to the internet” to supply and access real-world information in the environment [13]. The generic objects may release heterogenous and duplicate information to the environment. This is a semantic perspective of IoT, which is defined as a “worldwide network of interconnected objects uniquely addressable, based on standard communication protocols” [12]. Generally, IoT vision categorized into internet-oriented vision, things-oriented vision and semantic-oriented vision as shown in Figure 2.1.

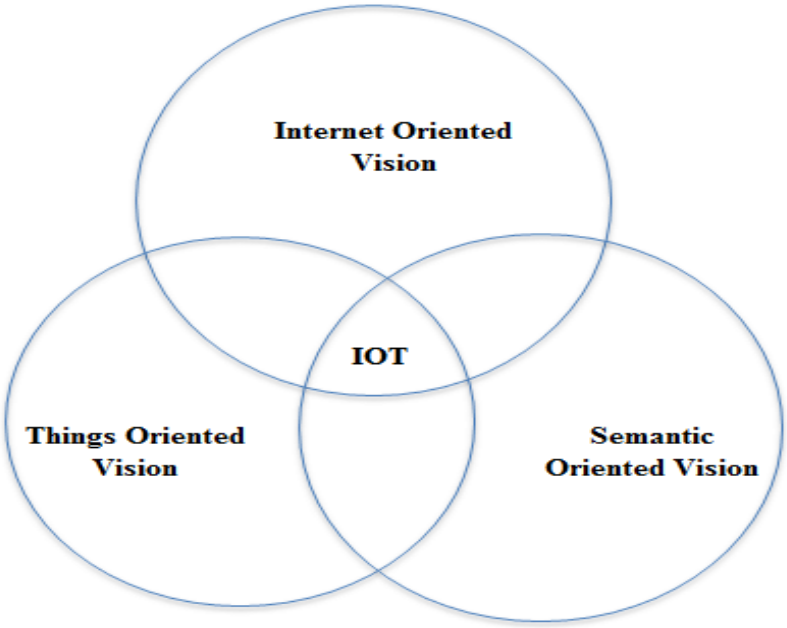

Figure 2.1 Vision of IoT [12]

In the internet-oriented vision, the physical object converted into a smart object. The objects would be sensor-enabled, and IP is assigned to uniquely identify. In the things-oriented vision, it is all about tracking and monitoring of objects using sensors and the ubiquitous technologies. The other is the semantic vision of IoT, which aims the number of objects connected over the Internet to become extremely high and possibly generate massive data. The data probably redundant and in the form of homogeneous and heterogeneous. So, the semantic perspective of IoT helps to process the data meaningfully and take the necessary action or decision appropriately. Semantic technologies will play a key role and can exploit appropriate modeling solutions for things description, reasoning over data generated by IoT [12]. IoT requires several empowering technologies for the reality of its vision. Some of the requirements are communication protocol, network technology, hardware and software technology, architecture, etc.

2.2 Basic Requirements of IoT

Building IoT is the creation of connected devices and systems which may be comprised of machines interacting devices (mote) and communicating with other machines, environments, objects, and infrastructures [14]. To realize this useful deployment of multiple technologies that covers in the domain of hardware, software and extremely robust applications around each domain of industries and operating sectors are essential. So, the basic requirements in building IoT applications are discussed next:-

2.2.1 Communication Protocols

Communication technology is one of the basic requirements in building IoT based application, however, the requirements of different IoT applications might vary across scenarios [15]. IoT requires new flexible protocols for heterogeneous and constrained devices, unlike common communication protocols. More specifically, constrained devices are insufficient to meet the high demands of HTTP communication. To reduce the high requirements of complex protocols used for constrained devices, the Internet Engineering Task Force (IETF) standardizes protocols that enable lightweight communication [16]. Some of the communication technology usually used in building IoT discussed below: -

Bluetooth

Bluetooth is a standard for short-range wireless communication typically up to 10 meters depending on the class of device and power [17]. The recently introduced Bluetooth Low-Energy (BLE) protocol, afford the range of conventional Bluetooth in combined with lower power consumption supremacy. However, BLE is not designed for transferring large files and will go perfectly with the small portions of data. Bluetooth does not require a line of sight between the transmitter and receiver.

Radio Frequency Identification (RFID)

RFID is a wireless communication protocol with RFID Tag and RFID Reader. It helps automatic identification of tagged objects while the objects within readers surrounding [18]. The RFID tag may be Passive Tag or Active Tag. Passive RFID tags are not battery powered and they use the power of the readers while identification. The tags carry data and send them in radio waves to an RFID reader.

ZigBee

It is an IEEE 802.15.4 standard for IoT security protocols that deliver secure, low-power, low cost, reliable and scalable solutions along with high node counts [19]. It usually operates at a frequency of 2.4GHz. Its signal transmission is two-way communication according to the model of client and server. ZigBee gateway configured with ZigBee coordinator acts as server and ZigBee sub-device acts as a client. Usually, ZigBee gateway sends a message of query and control to the ZigBee sub-device and ZigBee sub-device returns message of status to ZigBee gateway. ZigBee has a range of around 100 meters and a bandwidth of 250 kbps and the topologies that it works are a star, cluster tree, and mesh [20]. It is extensively used in home automation, industrial controls.

Wi-Fi

Wi-Fi is the most commonly used technology in devices as well as service delivery in infrastructure because it has quick data transfer rates along with the aptitude to control a large quantity of data [21]. The Wi-Fi technology is introduced under standards of IEEE 802.11. Due to this reason most devices such as laptops, PCs, printers, cellphones, and VoIP

phones have been combined with wireless technology. The wireless technology is running through LAN with Access Point (AP) configuration. The devices connected to AP to access the Wi-Fi and the coverage area is a radius of 100 meters.

LoRA

LoRA is a long-range low power wireless communication platform highly advantageous for battery-operated embedded devices. Technologies such as Zig-Bee, Wi-Fi, Bluetooth commonly used for a short distance, consumes high power and not suitable for battery-operated systems [22]. LoRA is preferred technology for IoT embedded system, because of its long-range, long battery life, efficient network, and interference immunity.

MQTT (Message Queuing Telemetry Transport)

MQTT is a lightweight Machine to Machine(M2M) communication protocol for constrained devices and unreliable networks [16]. It is designed as a publisher/subscriber model using TCP as a transport layer protocol. The aim of this protocol is to function on devices such as limited processing and memory capabilities [23]. The importance of using MQTT is its security and privacy because the message is encrypted by SSL/TLS. The Publish/Subscribe mechanism has capabilities such as one-to-one, many-to-many or one-to-one. Also, this mechanism provides bi-directional communication. In addition, it utilizes simple methods for communication and the communications among nodes are asynchronous. Moreover, messages can publish/subscribe anytime

2.2.2 Network technology

The IoT deployment requires developments of suitable network technology for implementing the vision of IoT to reach out to objects in the physical world and to bring them into the Internet [12]. In fact, the networks which can connect things identified in the physical world throughout various networks with different protocols [24]. IoT technology is deployed in many ways so no single network solution is right. It depends on the situation and where the devices are located. In addition, IoT can be applied in various networks such as personal area networks, Local area networks, and wide-area networks.

IoT in Personal Area Network (IoT-PAN)

IoT-PAN is the technology for “things” networked, communicated in a transparent, secure way around individual works space [25]. The devices and nodes create a human-centered environment both intra and inter-PAN communication capabilities. The communication may be wired, or wireless is running through standards of IEEE 802.15.4 developed for low rate personal area networks. The focus is on low power consumption and low transmission rate(256kbps) because the device is assumed to be operated in the battery.

IoT in Local Area Network (IoT-LAN)

IoT uses a communication function to connect to the Internet, the communication may via wired or wireless network [26]. The IEEE 802.3 Ethernet standard helps IoT devices connect to Ethernet when the devices are fixed in the facility. The wired networks are a mature technology and it is easy to get plugged into if you already have phone lines, power lines, and coaxial cable lines. Even in the case of wireless networks, those networks are usually connected to a wired network at some point; hence the most used network is a hybrid of both wired and wireless network connectivity. Some of the advantages of using LAN are its reliability, speed, and security whereas the limitation is cost, mobility, and scalability.

IoT in Wide Area Network (IoT-WAN)

IoT may use WAN to provide services for end IoT devices. So, long-range communication protocol could be applied to make IoT reality in WAN networks like 3GPP (Third Generation Partnership Project), GSMA/eSIM (embedded SIM), and LPWAN (Low-Power, Wide-Area Networks).

The 3GPP is for cellular networks system that works on high-speed communication, likewise, the recent standards LTE and 5G are for high-speed communication and low-speed communication for low power consumption. In addition, GSMA/eSIM used for provisioning and management for the machine to machine communications(M2M). The cellular device requires physical SIM used to connect operator network, now the trend is shifted to an embedded SIM. The GSMA/eSIM projects “over the air” provisioning of an initial operator subscription and change subscription from one operator to another. The other is Low

Power(LPWAN) which is a new wireless communication technology that supports long-distance communication, low data rate, and low power consumption.

2.2.3 Hardware and software technology

There are several hardware requirements in building IoT based system. IoT requires hardware that is miniaturization, ultra-low power, low cost and increased functionality in the design of systems [20]. The most used hardware devices in developing IoT based system Microcontrollers, sensors, and actuators.

A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system [27]. A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip

In other way, sensors are electronic devices composed of sensitive cells that can measure physical parameters like the light fluctuation, the temperature, to detect flames, sounds, movements, or any other fluctuation in the environment [28]. Thus, sensors are specific physical elements that allow us to measure a concrete physical parameter or detect something of the sensor's immediate environment.

Actuators are the other basic component of IoT, which allow actions over themselves or over other devices, and actions which a specific object allow to perform [29]. The actuators may be physical actuation. For instance, the servo motor is an actuator that makes physical action based given input.

Software technologies are also the basic requirements in building IoT applications, in which the embedded system or operating system should be applied in the hardware technology. However, the challenges are how to design a common underlying software fabric for different environments and how to build a coherent application out of a large collection of diverse software modules.

Currently, studies focused on service-oriented computing for developing distributed and federated applications to support interoperable machine-to-machine and thing-to-thing interaction over a network [12]. This is based on Internet Protocols, and on top of that, it defines new protocols to describe and address the service instances. service-oriented computing loosely organizes web services and makes it a virtual network.

2.2.4 Services and power storage technologies

Business logic is one of the key requirements for building IoT. Things in IoT interconnect to use services. The services may be from enterprise data systems, or PCs and mobile devices [30]. The resources in smart objects, include internal and external services, which may register themselves into the system with various roles, such as data producer, aggregator or interaction enablers.

The autonomous things operating in the IoT applications and performing either sensing or monitoring of the events need power and energy to perform the required job [12]. Since the environments have wide variations depending on where and how the thing is used, the power collection methods may vary, e.g. RF, solar, sound, vibration, heat, etc. Power and energy storage technologies are enablers for the deployment of IoT applications. These technologies must provide high power-density energy generation and harvesting solutions which, when used with today's low power nano-electronics, will enable us to design a self-powered intelligent sensor-based wireless identifiable device.

2.2.5 IoT Architecture

IoT architecture is one of the basic requirements of building IoT applications, consists of numerous elements such as sensors, protocols, actuators, cloud services, and layers. The most basic architecture in building IoT was three-layer architecture consists of perception, network, and application layer as shown in Figure 2.9 [31, 6]. In, perception layer principally performed capturing information from the environment through various IoT technology like camera, sensor, RFID tags and reader [32]. The network layer is responsible for connecting smart things, network devices and servers as well as transmitting, and processing information obtained from the perception layer. The application layer responsible for delivering application services to the user. It defines various applications in which IoT can be deployed, for instance, smart health, smart cities, and smart homes.

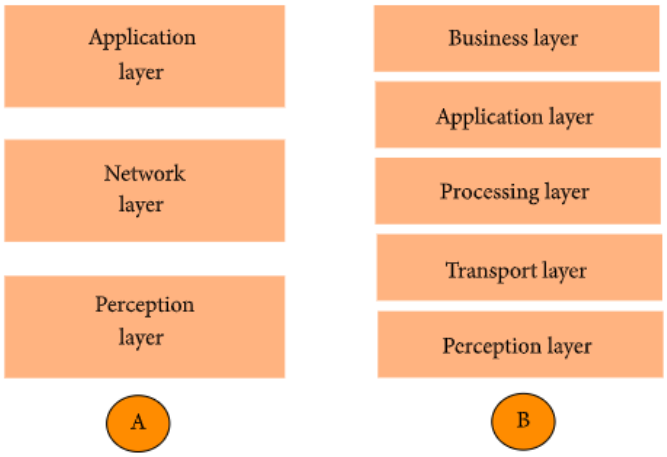

Figure 2.2 Architecture of IoT (A: three layers) (B: five layers) [6]

Later three-layer architecture is not enough for research on IoT because research often focuses on finer aspects of the IoT. Then five-layer architecture proposed, mainly increases the processing layer and business layers such as perception, transport, processing, application and business layer as shown in figure 2.9 [33, 32]. The role of the perception and application layers is the same as the architecture with three layers. The transport layer transfers the sensor data from the perception layer to the processing layer and vice versa through networks such as wireless, 3G, LAN, Bluetooth, RFID, and NFC. The processing layer is also known as the middleware layer which analyzes and processes huge amounts of data that come from the transport layer. It can manage and provide a diverse set of services to the lower layers. It employs many technologies such as databases, cloud computing, and big data processing modules. The business layer manages the whole IoT system, including applications, business and profit models, and users' privacy.

2.3 IoT Environment

As the minimum requirement is full filled in building IoT, the final deployment is in the environment. Environment means where "things" exist and the place the type of information collected and exchanged. It is defined in the perspectives of the physical environment, human environment, and ICT or virtual environment [34].

Human Environment

The human environment focuses on human-computer interaction related to the user or personal context [34]. The interaction is usefully constrained by users, in terms of identity, preferences, task requirements, social context and other activities like user experience and prior knowledge.

ICT of Virtual Environment

The concept here is more of a computer to computer interaction within the environment [34]. A component in a distributed system is aware of the services that are available internally and externally, locally and remotely, in the distributed system.

Physical Environment

This is the context of the computer to physical world interaction [34]. Mostly Sensors, controllers, and computers are embedded in the physical environment. So, the interaction there is run by data collection and communication between the computer and the environment.

In an IoT environment, the different applications may require one or more types of Information. Most IoT application uses sensors however sensors collect only analog types of information from the environment. For instance, some applications may require digital or visual information.

2.4 IoT and information collection

IoT bases, information collecting, exchanging or communicating and transferring environmental information between “things”. There are different types of information when things connect and communicate such as analog information, digital information, and visual information [35].

Analog information

Analog information is the continuous physically measured data in the environment, like measuring weather conditions such as temperature, humidity, and pressure [36]. Sensors are

the man equipment or electronic device used to measure analog information from the environment.

Digital information

Digital data that represent machine language systems that can be interpreted by various technologies. This machine language system is a binary representation, which simply stores audio, video and text information in a series of binary characters. Digital data seeks to capture elements of the physical world converting them to digital form for technological usage.

Visual Information

Visual information is another information available in the IoT environment and always collected through the camera as image/video data. The camera is a piece of equipment that captures an image and records it. However, the camera specification should be carefully selected for the successful building of IoT.

Currently, there are different types of cameras in the market for different purposes, in [37] more than 73 types of cameras are mentioned. The most used in building IoT are IP Camera, CCTV Camera, Serial JPEG, and Web Camera. However, the IP camera is more recommended than others because of the following reason [38]: -

- Can be monitored via any web browser remotely from any location
- Supports two-way communication
- The ability to send alerts in object detection
- Progressive scanning for more quality images and adjustable frame rates, for better resolution

To build IoT that consumes visual information, real-time object detection is required. IoT uses the processed result of visual information. The real-time object detection has its own steps, procedures, and algorithms to extract meaningful information from the image. In fact, those steps can be facilitated using traditional image processing or using DL. The traditional image processing steps are discussed next.

2.5 Image processing

The main reason that image processing usually requires more resources and challenges to be processed in low-performance IoT devices is that images always represented in the digital domain by a matrix/array of intensity values, and video sequences are represented by a series of matrices. These matrices are often large and require many storage spaces and/or transmission bandwidth [39].

The visual information collected as image data from the environment is further processed in digital image processing to extract useful information from the image. The general steps are image acquisition, pre-processing, segmentation, feature extraction, classification and recognition as shown in Figure 2 [2].

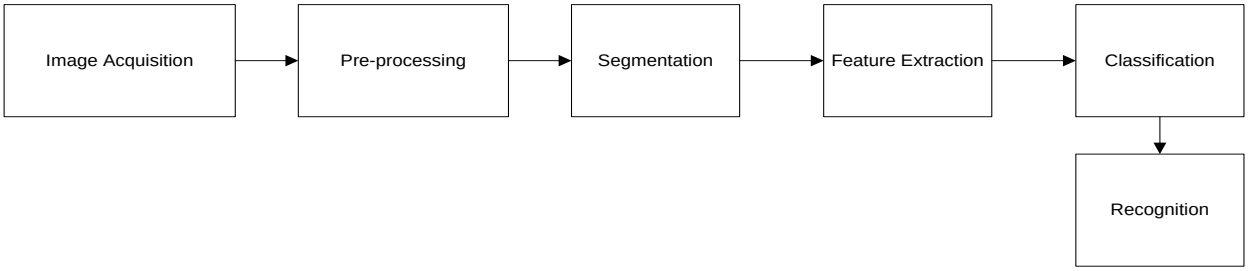

Figure 2.3 Fundamental steps of image processing [2]

In image Acquisition, visual information is converted into digitally encoded representation, usually, the camera performs tasks of image acquisition. In pre-processing, an improvement of the image by suppressing unnecessary distortions or enhancing major image features. Segmentation performs partitioning of an image into its constituent parts, it highly affects ahead of steps like feature extraction, classification, and recognition. The more accurate the segmentation, the more likely feature extraction, classification, and recognition. Feature extraction is the transformation of original data to a data set with a reduced number of variables, which contains the most discriminatory information [28].

Image classification refers to the labeling of images into one of several predefined categories and processed with predefined images stored in a database [40]. Then, image recognition proceeds and to identify objects, places, and people in an image. In another way, Image recognition is a machine vision technology in combination with a camera and software

technologies [2]. Machines require algorithms and models to recognize objects starting from the time of capturing by the camera. The accurate classification of images results in accurate recognition. The image recognition is one of the important technologies in building IoT based application that requires visual information processing. For instance, an IoT application which is developed on real-time object detection. The object detection performed after image recognition because it's the image of the object to be recognized then detected.

In another way, using DL-based image classification techniques can also be other ways of facilitating visual information processing. the techniques are supervised, unsupervised, and semi-supervised image classification.

2.6 Image classification techniques

Image classification could be smoothed through supervised, unsupervised and semi-supervised classification. Those classification methodologies have their own property to select one over others. For instance, the type of application which would be developed, that means the application may require small data or lots of data. So, the application which wants small data may use supervised learning whereas lots of data shall be developed on unsupervised or semi-supervised.

Supervised Classification

Supervised classification is the process of "using samples of known informational classes (training sets) to classify pixels of unknown identity" [40]. To achieve accurate categorization of an image in supervised classification pre-labeled and training of data is required [41]. The training of data is time consume however achieving an accurate result is high.

Unsupervised Classification

Unsupervised classification is the process of clustering of data based on the properties of data and performed without labeling and training data rather takes random data [41]. In other words, segmentation and learning about features in the data through algorithms. The large number of unknown pixels and divides into several classes based on natural groupings

present in the image values [40]. So, in unsupervised classification, no extensive prior knowledge is required.

Semi-supervised Classification

Semi-supervised classification takes advantage of both supervised and unsupervised [41]. This method is used to deal with the non-labeled samples to assist with the supervised classification. The semi-supervised method does its classification in three steps. Firstly, it selects the labeled or unlabeled data points then creates an initial classifier. And the last step is to be clustering the data points to find classifiers. In semi-supervised learning, an algorithm learns from a dataset that includes both labeled and unlabeled data, usually small labeled with a large amount of unlabeled data.

As mention above image classification could be performed through supervised, unsupervised or semi-supervised. The goal of classification techniques is to recognize or detect an object through its image. The recognition is accomplished by training the algorithm to have knowledge of the classified images. Currently, many algorithms are available for image recognition.

The images of real-world objects were the visual information captured from the environment, subsequently, one image may have one or more images to be recognized and detected. Such types of scenarios usually facilitated through machine learning algorithms. Among machine learning algorithms, Convolutional Neural Network(CNN) is specifically applied for computer vision applications that involve image classification and object recognition [42] because it's the most popular techniques for improving the accuracy of image classification, requires less memory and easier and better training [43, 44, 45]. The benefit of using CNNs is their ability to develop an internal representation of a two-dimensional image. This allows the model to learn the position and scale-invariant structures in the data, which is important when working with images.

2.7 Convolutional Neural Network

CNN is widely used in pattern and image-recognition problems that have a convolution layer at the beginning [45]. So, instead of feeding the entire image as an array of numbers, the image is broken up into several tiles, the machine then tries to predict what each tile is, as shown in Figure 2.4. Finally, the algorithm tries to predict what's in the picture based on the

prediction of all the tiles. This allows the computer to parallelize the operations and detect the object regardless of where it is in the image.

More generally, CNN works well with data that has a spatial relationship. The CNN input is traditionally two-dimensional, a field or matrix, but can also be changed to be one-dimensional, allowing it to develop an internal representation of a one-dimensional sequence. This allows CNN to be used more generally on other types of data that has a spatial relationship. As shown in Figure 2.4, CNN layers are the Convolution layer, pooling layer and fully connected layer [42]. Convolution layer (Conv Layer) applies filters to learn features from the input image and detect the lowest level features such as corners and edges in the image. In the Conv layer, each neuron in this layer will only connect to a smaller region in the input volume. The pooling layer compresses the incoming volume along the spatial dimensions. Finally, the fully connected layer predicts the final class or label of the input image.

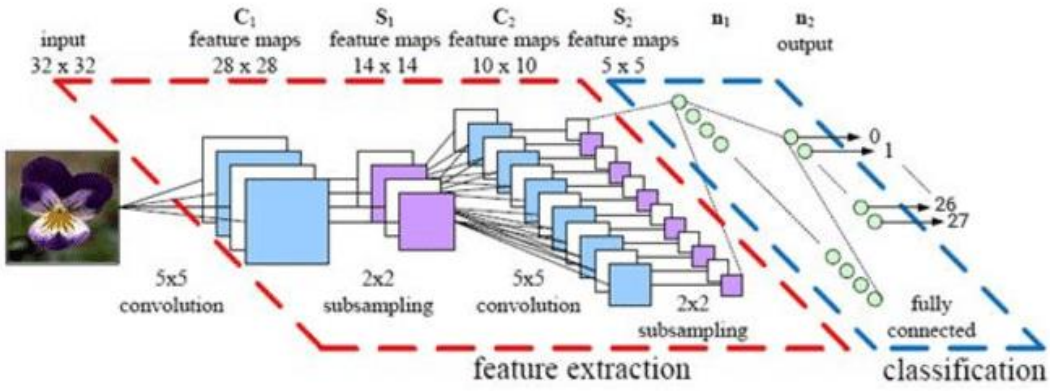

Figure 2.4 Convolutional Neural Network [44]

However, there are different CNN architecture, because the standard convolutional network to classify the presence of the object within that region. However, there could be several objects in the image, such problems produce difficulty to proceed CNN along with a fully connected layer because of the length of the output layer is variable and the number of occurrences of the objects of interest is not fixed in the image. For this reason, select a huge number of regions and this could computationally blow up. Detection algorithms can be

worked to detect multiple objects in the image, to resolve such challenge there are different types of CNN architecture such as RCNN, Fast-RCNN, and Faster-RCNN [46].

R-CNN

RCNN is based on a selective search algorithm first it proposes 2k candidates in the image [47] as shown in Figure 2.5. RCNN first takes an input image, second extracts around 2k bottom-up region proposals, third computes feature for each proposal using a large CNN, and finally extracted features feed into Support Vector Machine(SVM) to classify the presence of the object within that candidate region proposal.

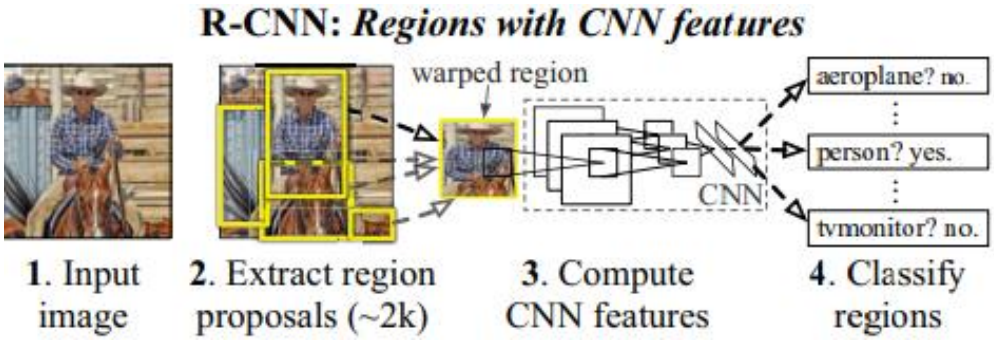

Figure 2.5 Object Detection With RCNN [47]

The challenges in using RCNN are a huge amount of time to train the network as you would have to classify 2k region proposals per image and the fixed nature of the selective search algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals. In addition, RCNN is computationally expensive.

Fast-RCNN

In Fast-RCNN, instead of feeding the region proposals to the CNN, input image should feed into the CNN to generate a convolutional feature map. As shown in Figure 2.6, From the feature map it is possible to identify region proposals and wrap them into squares. Then RoI layer reshapes into fixed size so that it can be easily fed into a fully connected layer. The SoftMax layer is used to predicts the proposed region from the RoI feature vector. The advantage in using Fast-RCNN is there are no feeding 2000 region proposals to the convolutional neural network every time

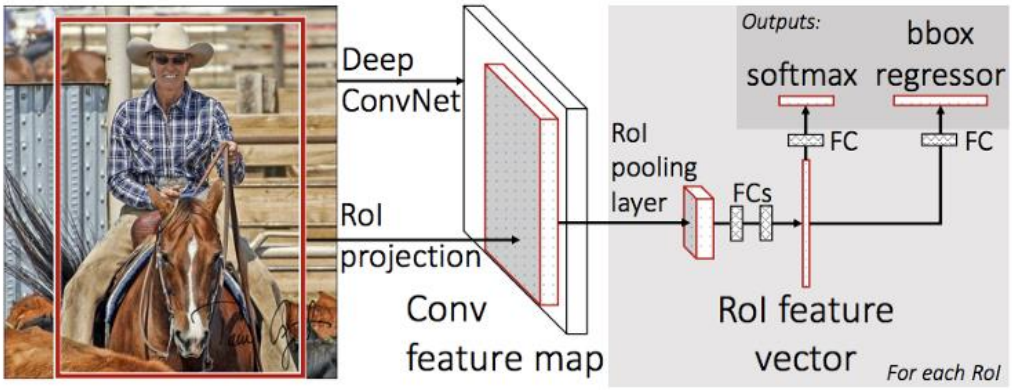

Figure 2.6 Object Detection with Fast-RCNN

The challenge in using fast-RCNN is for real-time object detection is still it uses a Selective search algorithm which is a slow and time-consuming process that affects the performance of the network.

Faster-RCNN

In Faster-RCNN, the image input to the Convolutional network and provides a convolutional feature map as shown in Figure 2.7. However, Faster-RCNN doesn't use a selective search algorithm, it uses a separate network to predict a region proposal called Region Proposal Networks(RPN) [48]. The purpose of RPN is to propose multiple objects that are identified within an image. The predicted region reshaped with the RoI pooling layer. After the shape is fixed size the classification proceeds and predicts the offset values for the bounding boxes.

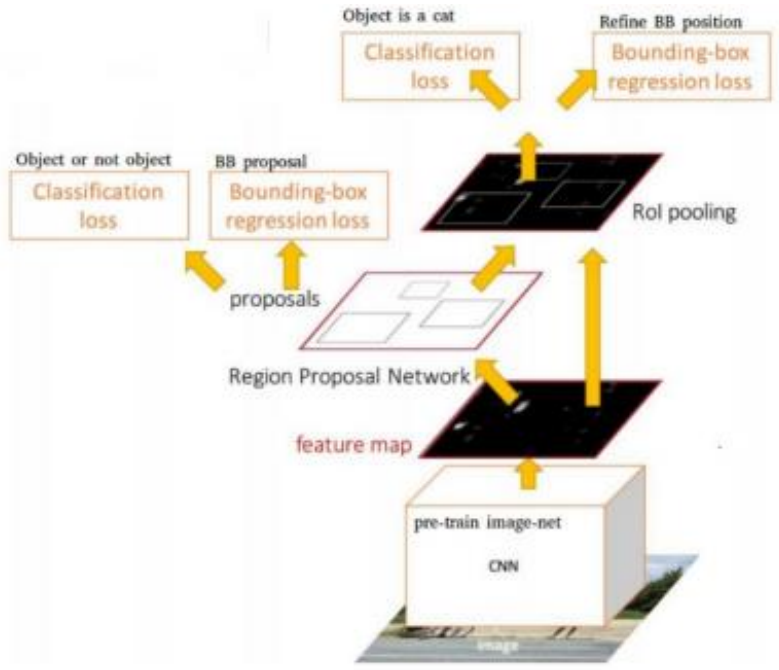

Figure 2.7 Faster-RCNN Architecture [49]

2.8 Image processing in IoT Based Applications

There is the various application of IoT along with image processing. The reason for combining IoT with the techniques of image processing to achieve accurate results [50]. In addition, the nature of some activities shall visually inspect in building IoT. Some of the applications of IoT along with image processing are discussed in this section.

Image Processing in IoT Based Agricultural Solutions

Technology is rising to automate various sectors; Agriculture is one such sector. Nowadays agriculture collaborates the technology such as the IoT combined with image processing which results in cheap yet effective methods of smart agriculture and gives rise to higher quality produce [50]. The aim is to have higher quality products and thus to reduce crop failure and constantly follow the health of the plant and deliver information for farmers [51, 52]. The role of IoT is to monitor the plant and to collect the environmental factors such as humidity and temperature whereas Image processing capable of identifying plants by using the images of their leaves and with the help of the image processing and use of pesticides can be controlled. In this regard, the agricultural domain was automated.

Image Processing in IoT Home Security Solutions

Home security automation plays a crucial role in all workplaces and living homes. The need for security systems for home is considered as one of the important aspects of our modern life. Home security describes security measures that are designed to deny unauthorized access to facilities, equipment, and resources and to protect personnel and property from damage or harm. IoT conceptualizes the idea of remotely connecting and monitoring real-world objects (things) through the Internet. Home security consists of hardware of a home includes doors, locks, alarm systems, lighting, motion detectors, security camera systems, etc. that are installed on a property. The security system mainly consists of two steps [53, 54]. One is sensor-based systems which are used as motion sensors to trigger the security camera. and the other is motion-based systems such as security cameras to capture the image for image analysis. The image analysis with IoT is used to make a decision by comparing the captured image to the image set to the database if the application finds the image is not house owner alert message sent to the owner of the house. Even the capture image could be sent to the house owner through a mobile application. In this manner, the home could be controlled anywhere.

Image Processing in IoT Parking Solutions

A smart parking system applied in several industries to make the park and traffic congestion more manageable and effective [55]. Initially, the smart parking system was to the integrated form of hardware-software application to help drivers to find the empty spot in the parking lot more easily with less time using the IoT. Now smart parking comes more advanced in which the technology of image processing along with IoT is developed. So, the users of a certain park can be monitored their car with plate number. The camera captures the palate number and image processing such as OCR could identify the authorization of the user. For instance, the valid and invalid employee in reserved parking can be known by extracting the plate number from the car using OCR (open character recognition) [56] and also it is possible to determine the occupancy of the parking space based on the information collected from multiple cameras placed in the parking zone.

Image Processing in IoT Based Monitoring Solutions

There are various monitoring applications implemented using image processing and IoT. Industrial monitoring is the one that has a vital role in an industrial area to monitor and control industrial applications or equipment. Industrial monitoring is used to know the dynamic condition of industrial devices or machines. IoT was the most favorable technique for industrial process monitoring [57]. IoT is a combination of embedded system and communication system in which industrial equipment are connected to the Internet with the help of wireless sensor network and devices or industrial application can monitor and control through mobiles and laptops. Now the monitoring comes more advanced in which IoT and image processing integrated and results in more effective and efficient industrial monitoring [58]. The products in the industry constantly captured using camera and stream to image processing and the quality of the product that the industry manufactured could be checked in real-time.

2.9 Challenges in developing IoT

The IoT applications, concepts, and future direction provide technologies for smart everything with different in any environment with different information. However, the challenges of IoT are resource-constrained nature of devices, power supply, and storage constraints.

Resource Constraint: The embedded computing devices deployed within the IoT are expected to be resource-constrained. Mostly, smart objects run with limited system resources such as processing power, memory, non-volatile storage, and transmission capacity.

Power Supply: Most of the devices in the IoT are having a considerably small size and are movable. Due to their size and frequently changing location property devices are not able to access the power all the time. So, low power consumption is the universal constraint of the IoT. Either they use battery technologies, or they can use some techniques for taking power from their environment using other devices like self-sufficient energy sources [18]. However, yet there is no effective solution and IoT devices are running with battery life, it results in difficulty to deliver continuous service.

Storage Constraint: In IoT huge amount of data need to be collected and transmitted leads to challenges in terms of framing the data appropriately [59]. Therefore, IoT data computations become complex and tedious. For instance, when IoT uses real-time object detection and transmission the images are challenging with the growing number of images sizes, real-time interaction with compressed images, and the variety of bandwidths on which transmission needs to be supported.

Chapter 3 : Related Work

3.1 Overview

Building IoT along real-time object detection is the technique of using visual information in IoT technology. The aim was to visually inspect objects and to make some physical or logical actuation on objects. The visual information is captured or monitored by camera technology. The captured visual information further processed to extract useful information form the images/video. The place of Real-time object detection is usually either cloud or cloud supporting technology such as fog or edge computing. The reason is the low-performance nature of IoT devices and unable to perform big computations such as image processing.

A lot of work has been done on building IoT along with visual information processing. Most of the study is developed cloud-based, fog and edge computing platform. So, in this chapter, we present the efforts that have been made on previous studies in building IoT with Real-time object detection when building IoT. The application area with the techniques their implementation will be presented.

3.2 Image processing in IoT and cloud computing

Yuzhong Yan and Lei Huang present image processing cloud architecture, and big data processing engine based on Hadoop [60]. As per the study, big data refers to images and video which demands high data storage and computation power. So, the image processing cloud was proposed to deliver Platform as a service(PaaS) for image processing researchers and developers. It should be able to store a large number of images and videos, as well as be able to process them and meet the performance requirements. Then the user can develop image processing applications on the cloud along IoT with their familiar programming language. The reason for delivering image processing PaaS on the cloud is the high-computation power nature of image processing which is impossible to implement locally along with low-performance IoT devices. So, IoT device usually consumes the image processing result from the cloud.

Similarly, the thesis in [61] presents a solution of cloud-based medical image processing along IoT. The aim of the study is to resolve the lack of image processing and analysis tools

for handling large image datasets. The study reviews medical image processing and its challenges, states cloud computing and cloud computing benefits due to medical image processing and proposes to migrate the image processing part to the cloud due to the low-performance nature of IoT devices.

3.3 Image processing in IoT and Fog computing

Beatrice Dorothy *et al.* [53], proposes an IoT based home security framework upon fog computing as shown in Figure 3.1. The framework which consists of components such as sensor, digital camera, database in the fog and the mobile phone. Sensors are placed in the front of the door which alerts the camera, to capture an image when moving body avail front of the door, then sends the image to the database or dataset that is stored in the fog. Then, image analysis is performed to detect and recognize and match the image with the stored dataset of the authenticated people or pets. They use template matching image analysis technique, which is a technique that identifies the segment on an image that matches a predefined template. If the image captured does not match with the dataset then an alert message is sent to the owner of the house.

Figure 3.1 Fog computing based Home Security Framework [53]

Similarly, Pengfei Hu *et al.* [62], deals a fog computing-based face identification and resolution scheme. A face identifier is firstly generated by the identification system model to identify an individual. Then, a fog computing-based resolution framework was proposed to

efficiently resolve the individual's identity. The study also aims to offload some computing from cloud to network edge devices in order to improve processing efficiency and reduce network transmission. The thesis clearly shows the scheme can effectively save bandwidth and improve the efficiency of face identification and resolution. As shown in Figure 3.2, the framework consists of a client, fog node and cloud components. In the client, the camera performs the tasks of image acquisition and sent to fog computing. The fog computing proceeds the tasks of facial image face detection, facial image preprocessing, feature extraction and face identifier generation which can make full use of the computing power of network edge devices rather than all the resolution processes that are executed in the cloud. It only transmits the extracted face identifier in the network, rather than the raw facial image data. In this way, the burdens of cloud computing were reduced results improving computation and storage capability, but also reduces the amount of network transmission remarkably and saves the bandwidth

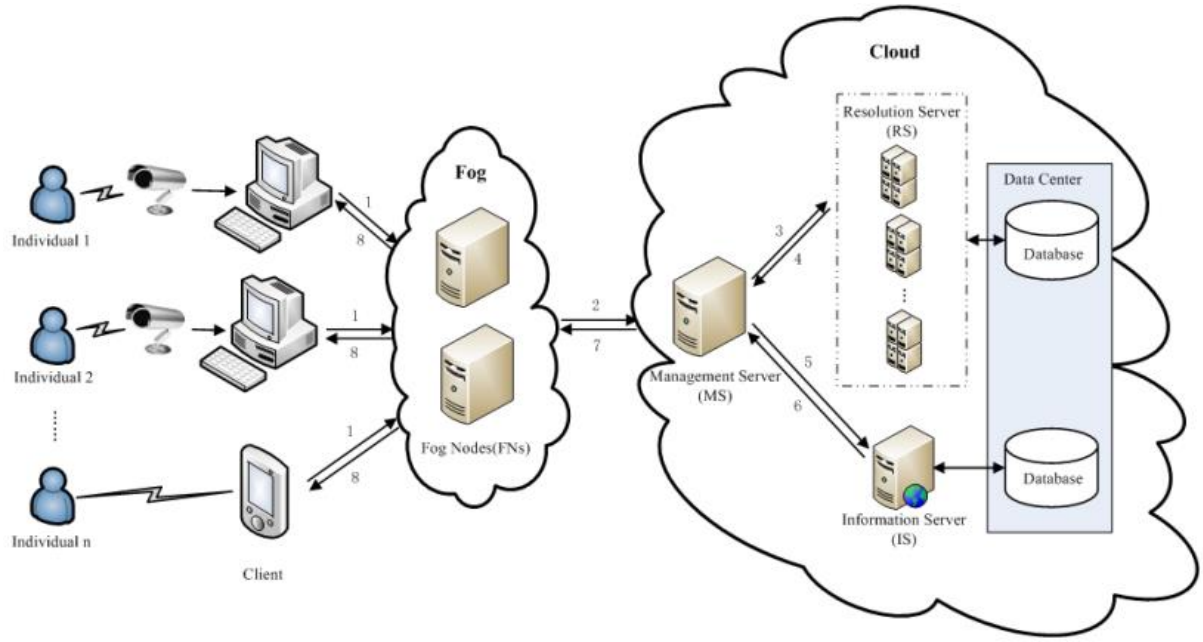

Figure 3.2 The fog computing-based face resolution framework [62]

3.4 Image processing in IoT and Edge computing

An edge computing framework along with IoT for video processing was developed in [63]. The aim is to perform video processing on the edge node instead of sending the entire video to the cloud. The general framework is shown in Figure 3.3 consists of three main nodes, such as a camera node, edge node, and the server node. The framework takes a human

detection scenario. So, the camera node can be a static camera device fixed at the top of a street lamp, which invokes video tasks, divides them into smaller sub-tasks (video chunks), compresses video chunks and finally transmits them to edge nodes within the scope of a certain distance via Device to Device communications. Then, a mobile device was used as an edge node with enough computational ability and storage capacity, helping process video sub-tasks, e.g., image feature detection and extraction. The edge nodes process the received video sub-tasks and upload the computing results via a Long-Term Evolution (LTE) network to a cloud-based IoT server for further video analysis, such as human detection.

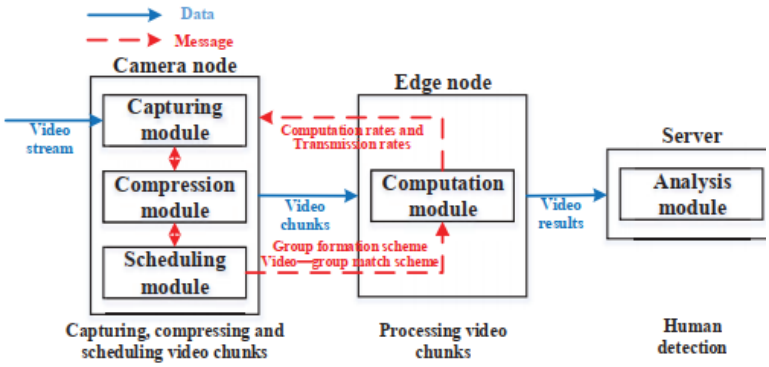

Figure 3.3 Architecture of the edge computing framework [63]

Similarly, Dirk Jacobsen, and Petter Ott [64], develops cloud computing with image processing for the case of surface quality control of metal sheet for the door of car production. A smart camera with a small signal processor is used to inspect fabrication process images directly at the camera system in real-time. The output data of the smart camera is used to classify the quality of the produced part. However, the capacity of the smart camera is limited and can't be changed without changing the hardware. So, to overcome this a sensor-cloud architecture as a platform for image processing was developed.

3.5 Summary

In general, the works in this chapter provided the different ways of using IoT and Image processing to resolve several problems in various applications. In most of the studies, the image processing part is pushed to the cloud. In some studies, fog and edge computing were proposed and play as supporting technology of the cloud. The aim was, before sending the captured data from the environment, minimal computation should be performed on edge or fog. So, the cloud shouldn't receive the raw data rather processed data assuming to save

computation cost, and communication bandwidth. The fact is those use cases are one- or other-way cloud-dependent. However, for the scenarios that don't require cloud mandatory services like security and management should have an intermediate framework below edge computing. This type of issue going to overcome on the proposed framework.

Chapter 4 : Design of on-site solid waste segregation framework

4.1 Introduction

In this chapter we are going to design a framework for real-time object detection and physical actuation on an object, specifically for facilitating the scenario of on-site solid waste segregation. The on-site solid waste segregation performed by development of real-time object detection and IoT which makes real-time physical actuation based up on detected object. The object refers the solid waste which is going to be detected at real-time while camera visualized it and then segregation processed.

The way of representing an object (visual information) which is image data to lightweight information that low performance IoT devices could easily consume will be researched and how physical actuation proceeded. The development of the on-site solid waste segregation would pass through the development of object detection, title-based communication, communication protocol, actuation service provider, and actuation. In this chapter the term architecture and framework would be used interchangeable, but to refer the framework that would be developed for on-site solid waste segregation.

4.2 Architecture of the system

The architecture of the system is shown in the Figure 4.1. The architecture is based on IoT architecture [6] and Faster-RCNN object detection architecture [49]. Initially, the visual information is real-time visualized in perception layer via a camera, and then transported to processing layer for real-time object detection. The real-time object detection is facilitated using Faster-RCNN DL algorithm. The consequence of detection result is parallelly represented as title and communicated to application layer for further actuation service providing that realize the physical actuation which is on-site solid waste segregation.

We are presented the design of on-site solid waste segregation in four layers IoT architecture because is the scenario itself can be executed in four layers, without additional layer such as business layer available in five-layer IoT architecture. The business layer is more of managing users, service, or security. So, we consider the on-site solid waste segregation scenarios can be easily facilitated without management and security aspect. Due to this

reason, the proposed architecture shall be presented in four-layer having perception, transport, processing, and application layer.

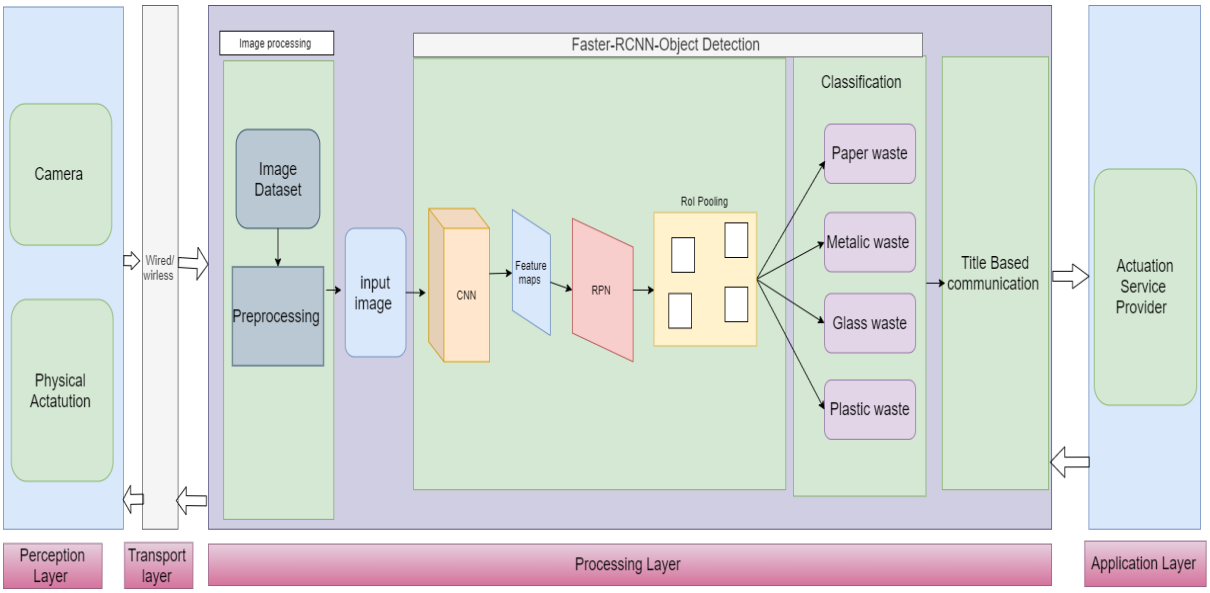

Figure 4.1 Architecture of the system

In the perception layer the task of visual information (solid waste) streaming through camera, and physical actuation is performed. The visual information which will be collected from the environment and shall be inspected for further action. To realize this, the camera component begins collecting visual information and sends it to the upper layer through the transport layer. On the other hand, the actuation also placed in the perception layer which makes physical action based on the actuation service (actuation command) receives from the Actuation service provider (application layer). The physical action is like denying or giving permission to access something. This realizes the physical actuation in the framework.

The visual information collected by the camera is transferred to the next layer through the transport layer via a wired or wireless network. so, the transport layer is to transfer the information collected in the perception layer to the processing layer. In the processing layer, three main tasks are performed, such as image processing, object detection and title-based communication. The image preprocessing performed on image dataset before object detection proceed. The object detection performs the task of classification visual information (solid waste). The title-based communication creates the medium for sending the titles to the upper layer which represents the inspected visual information. In the application layer, the

actuation service provider is placed which facilitates the task of delivering actuation command to the physical actuation (perception layer).

4.3Image processing

The image processing is performed on image dataset for further object detection model training. The image processing mainly performs the preprocessing tasks of the image dataset. The preprocessed images will be going to be an input to the Faster-RCNN detection. The Faster-RCNN DL algorithm wants input of images the same size for the training of its network. The preprocessing flowchart algorithm is shown in the Figure 4.3. The algorithm mainly performs the task of making all image dataset to the same size before training the model, like resizing every image to 800X600 height and width. In addition, the types of images going to be preprocessed may also be cleared out, because the image dataset may consist of icons, GIF and others image type which is not worthful for further object detection model training.

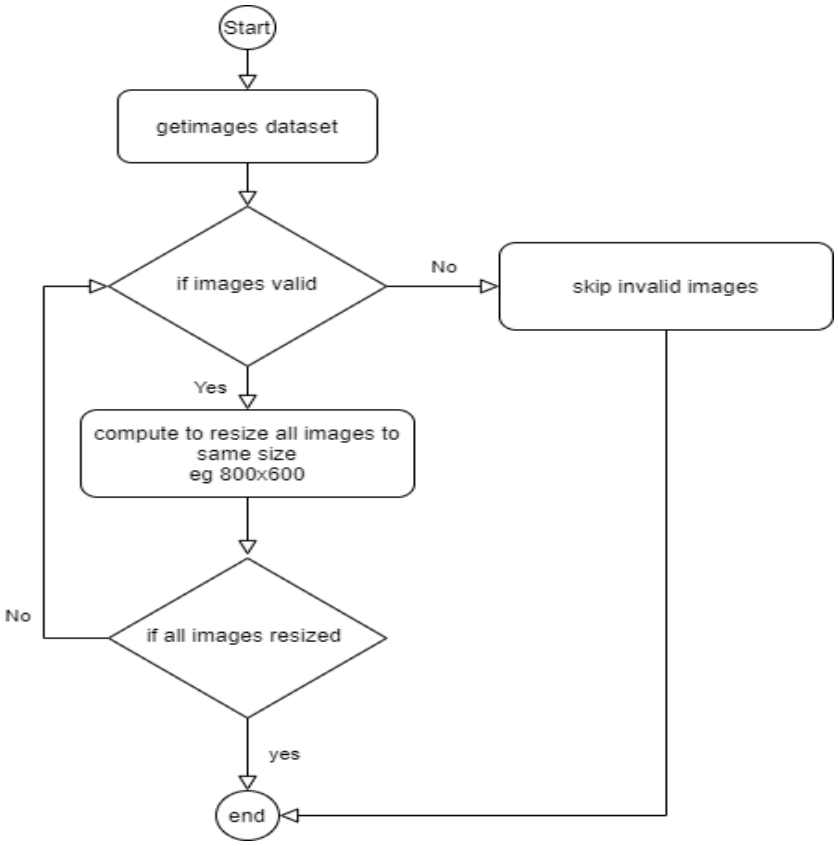

Figure 4.2 Image preprocessing algorithm Flowchart

4.4 Object detection

The object detection performs processing of visual information in real-time, detect and display the detected object with the accuracy and name, further represent the detected object as light weight information which is important to execute in IoT devices for later physical actuation. The inspection result is the lightweight information which is going to communicate to the external world (actuation service provider). To realize the inspection, we develop the object detection using Faster-RCNN object detection architecture. we choose the Faster RCNN algorithm because it has capable to learn and predict with a more sophisticated way to classify any unknown phenomenon from given datasets and has high accuracy of visualizing objects is much more achieved in DL with continuous improvement of accuracy and efficiency, which means as the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster. The advantage of using Faster-RCNN is much faster to train, high accuracy in detection and tracking objects. Faster-RCNN uses RPN that makes it faster than other CNN inspired algorithms. So, Faster-RCNN would be trained and used as an object detection model. On the other hand, we choose the techniques supervised visual information classification technique because we have small volume of data, since for small amount of data supervised technique is more recommended results all data could be labeled results in more accurate.

To accomplish the real-time solid waste segregation, we take an input image and pass it to the CNN(ConvNet) which returns feature maps for the image, then we apply RPN on these feature maps and get object proposals. Then, the ROI pooling layer bring down all the proposals to the same size. Finally, the proposals passed to a fully connected layer to classify any predict the bounding boxes for the image which produces the classification of the image. In overall, we develop the Faster-RCNN algorithm is to perform the scenario of on-site solid waste detection based on the four classes of solid object we have. We create a model upon this architecture by training Faster-RCNN network up on the collected image dataset and to classify solid wastes as plastic, paper glass, and metal waste. The algorithm general execution procedure is shown in the Figure 4.4.

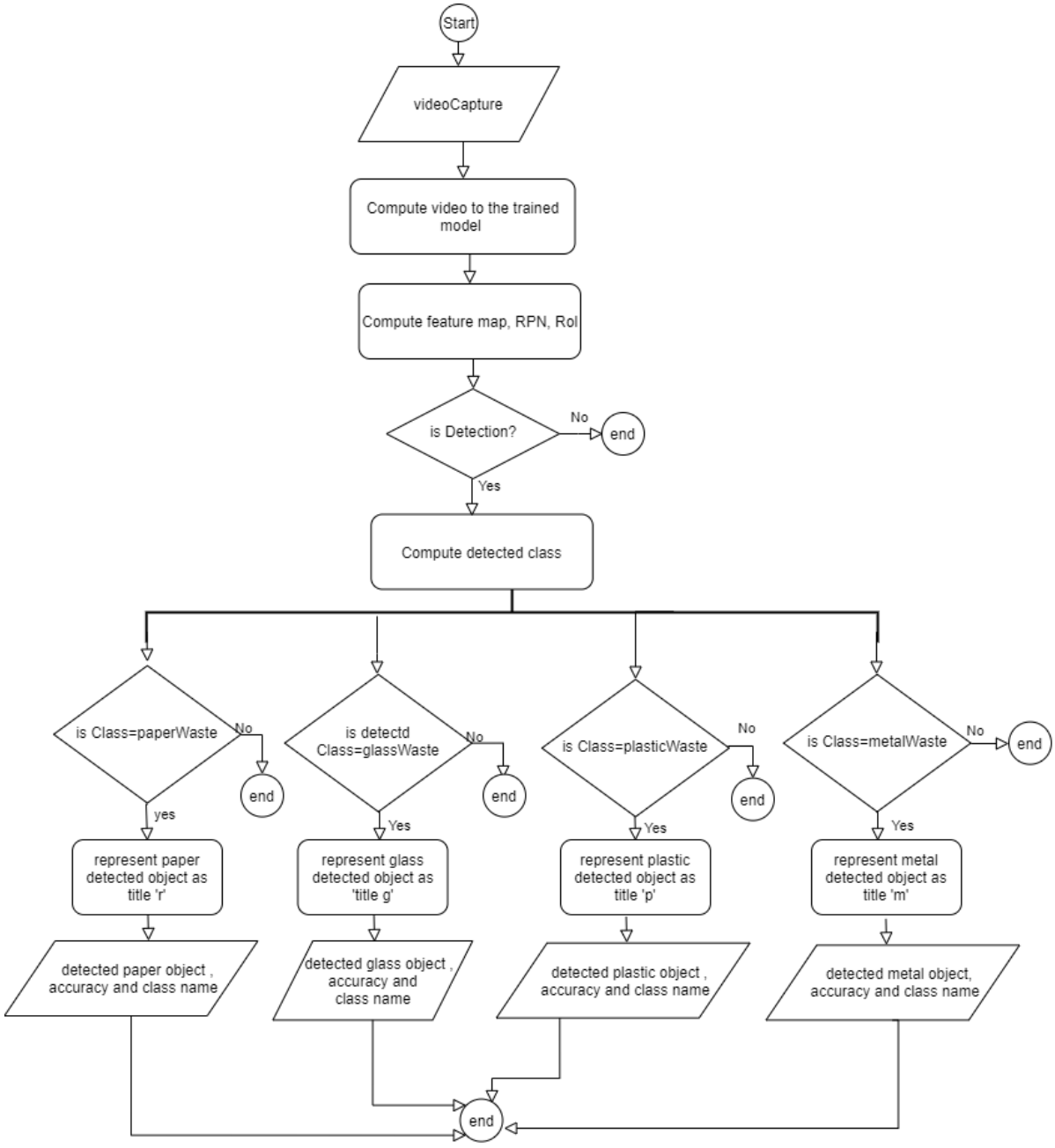

Figure 4.3 Object detection Flowchart algorithm

The algorithm defined in Figure 4.4, through flow chart algorithm describes the algorithm running behind the real-time object detection. Initially, the algorithm waits the real-time visual information(video) streaming, while the algorithm gets visual information, it goes to compute the newly video to the trained model. Then, the detection elements computation proceeds as of Faster-RCNN algorithm such as feature map, RPN and RoI. Comparing to the newly video to the model and if any detection comparing to the trained model, represent

the detected object as light weigh information(title) and it goes to display the specific detected image including its class name and accuracy of detection.

4.5 Title based communication

The overall result of the real-time object detection is inspecting visual information and represent the inspected object with corresponding lightweight information which is a title. It is the tittle to be communicated as inspected visual information to the IoT device for further actuation service providing.

The purpose of the title-based communication is to deliver light weigh information(title) to the external world (IoT devices). The title represents the visual information as a single character (8 bits) which is lightweight information which makes it easy to deliver services based on the title for IoT devices that act as actuation service providing which realizes actuation based on visual information. The following algorithm is based on python, to realize the title-based communication is the following: -

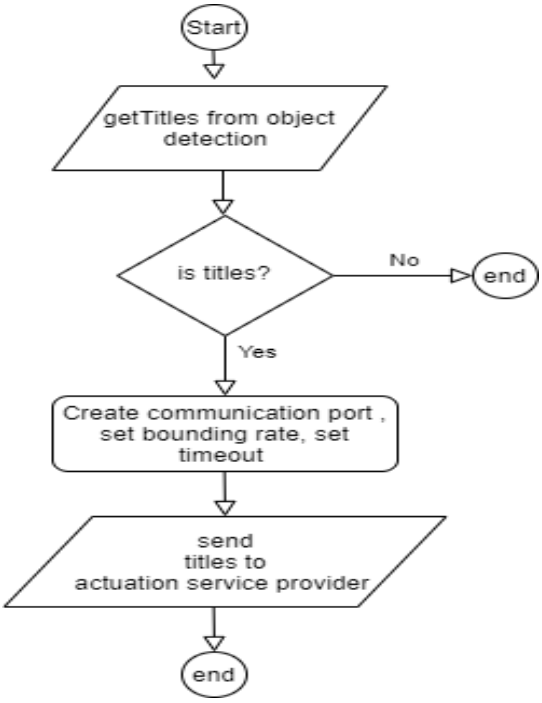

Figure 4.4 Title based communication Flowchart algorithm

The tile-based algorithm performed while the object detection gets detected object and represent title for the detected object. It facilitates the communication between object

detection and actuation service provider. First, it reads the represented title, then it creates the communication bounding rate, port, finally the title which represent the detected object will be sent to actuation service provider. The reason behind sending the title instead of sending the actual detected object is that IoT devices are low-performance and unable to process big computation like image processing and the title that represent the detected object shall be sent, accordingly the actuation service provider which running on low performance IoT device can react based on the title it receives which indirectly represents the detected object.

4.6 Actuation service provider

The Actuation service providing delivers services based on the title it receives from the title-based communication. The main function of the actuation service is to deliver actuation command to realize real-time actuation on visual information. The actuation service provider has embedded system to deliver real-time actuation service to physical actuation component back to perception layer. The actuation service algorithm is developed by C++ and it will be function as embedded system running on low performance IoT devices such as Arduino Uno. The Actuation service (actuation command) should be sent in real-time while a title is received from real-time object detection through serial communication to the physical actuation. The code definition generates machine readable file that can be encoded to IoT devices to facilitate the actuation service. The actuation service providing defined as of the following using C++ and the algorithm is shown in the Figure 4.6.

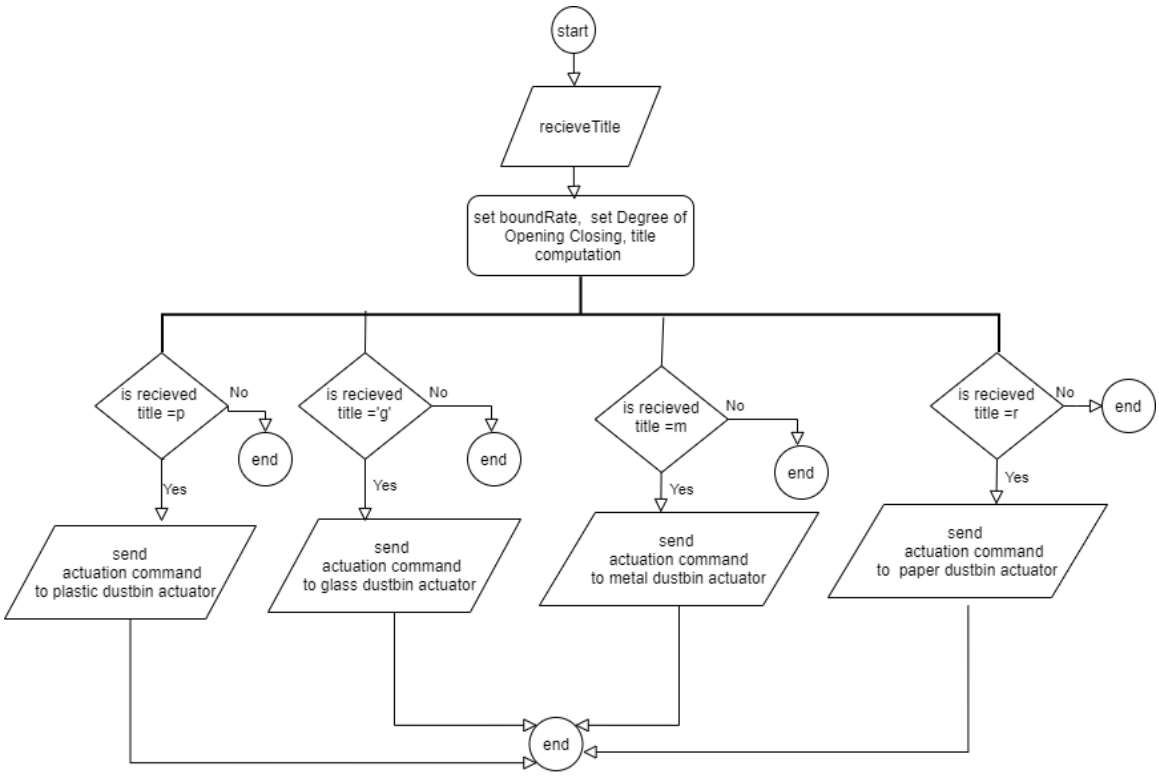

Figure 4.5 Actuation service providing algorithm Flowchart

The algorithm shown in Figure 4.6, gets title from the title-based communication and goes to deliver actuation service for physical actuation. The actuation service will be delivered according into the title received. For instance, if the title received is 'p', actuation command will be sent to the required actuator attached on dustbin to open and close for waste disposal. The actuation command is analog value which is communicated through wired or wireless communication that commands the actuator to open and close with some degree like 90 or 180 degree opening and closing.

4.7 Physical actuation

The physical actuation is just act based upon the command it receives from the actuation service providing. The types of command are analog command which will be communicated through wired communication. The command is open and close command. In the case of on-site solid waste segregation, the physical actuation is facilitated by attaching IoT technology on dustbin flap that receives open and close command received from actuation service provider. The type of technology which would be attached to the dustbin flap is actuator. So, the actuator plays a role of opening and closing the solid waste dustbin based on the types of

solid waste. For instance, if plastic solid waste is visualized and detected, the plastic dustbin automatically opened and ready for waste disposal.

The command communication is between actuation service providing from application layer and the actual physical actuation in perception layer. The command enforces the dustbin flap to open 180 degrees and back to 0-degree closing. The actual command comes from the embedded system of actuation service providing though wired or wireless communication. The actuator which enables the dustbin flab should have power and shall be activated to receive the command. In our case four types of solid waste would be segregated such as plastic, glass, paper, and metal solid waste. So, four dustbins having attached four actuators on the flap is required.

The four actuators would be attached for the four dustbin such as plastic dustbin, paper dustbin, glass dustbin, and metal dustbin. If one of the solid waste out four solid waste types such as plastic, paper, glass and metal detected one of the actuator will be react to the event For instance, if the plastic solid waste streamed via the camera and detected, the plastic dustbin would be open in which the actuator received the command to open the dustbin flap.

The physical actuation is placed in perception layer which performs on-site solid waste segregation based on the command it receives from the application layer. The physical actuation is facilitated via dustbins having attached actuators to play the role of closing and opening of the bin hatch based on the command it receives from the application layer.

Chapter 5 : Experiment

5.1 Introduction

In this chapter, we present the detail prototype implementation of the framework with their description. The development of the prototype includes demonstrating and validating some of the concepts in the designed framework. The main goal of the prototype implementation is to prove those concepts. The implementation may comprise software and hardware implementation. The software implementation includes system development, simulation tool to show how much the framework could be applicable in a real-world environment. The hardware implementation is referring to the hardware components we have used in framework development. Most of the components in the framework have been implemented as per the theoretical specifications formulated for each component. The development of the prototype includes demonstrating and validating some of the concepts we have proposed during the design of the framework architecture.

5.2 Implementation

The implementation and demonstration of the framework would be presented in the perspective of on-site solid waste segregation. On-site solid waste segregation requires a such IoT solution in which real-time object detection is required. Usually, solid waste segregation facilitated through manual dustbin which has labeled instructions like plastic, metal, and paper on dustbins. Off course users not quite respect the instructions. So, the proof of concepts defined in the chapter, step by step realization would present. Initially the camera is implemented using a built-in web camera available on the laptop. We used built-in web camera one is for data collecting and for streaming of visual information on demonstration of the thesis. The communication is implemented using wired communication which serial communication.

Real-time object detection implementation

The real time object detection implementation is facilitated in an anaconda- jupyter development environment with python 3.7.3 using the TensorFlow object detection API framework. The anaconda based Jupyter notebook development environment shown in Figure 5.1.

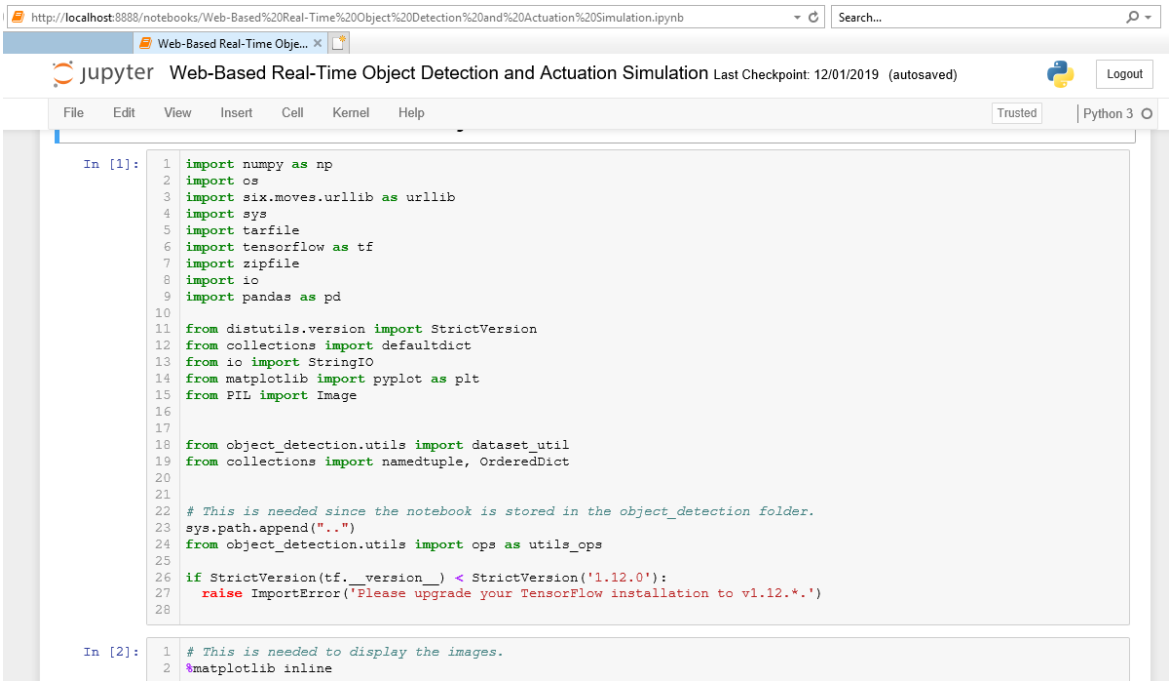

Figure 5.1 real-time object detection implementation environment

The real-time object detection was developed and running in window 10 Enterprise version operating system with specification Intel(R)Core(TM) i7-8550U CPU@1.80GHz. The machine does not have a GPU.

The real-time object detection implementation begins by training an Object Detection Model. The Object detection model would be developed to detect the class of plastic, glass, paper, and glass objects in real-time. Those objects are part of solid wastes and the framework would be simulated on such real-time solid object detection and actuation. The Object detection model training will be accomplished by Faster-RCNN and TensorFlow object detection API framework. The TensorFlow model is freely available at https://github.com/tensorflow/models. The TensorFlow object detection API framework is available in TensorFlow Models with directory models/research/object_detection. The model has lots of frameworks and libraries in it. The directory of the TensorFlow object detection API is shown in Figure 5.2. The bulletin is to indicate there are files and folders in the same directory. The object_detection directory contains many files and folders that make the TensorFlow object detection API framework functional for the object detection related

research and development. So, to develop in our own real-time object detection model we may add or modify codes on the existing file directory. That would be clearly indicated.

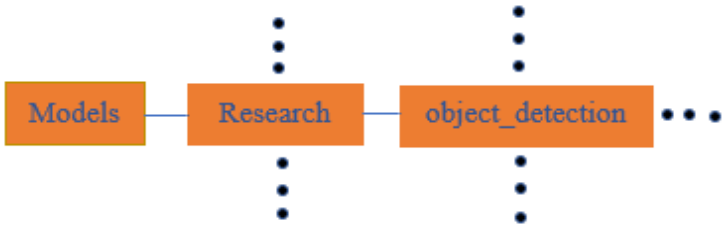

Figure 5.2 TensorFlow object detection directory structure

The real-time object detection development passes through the following steps:

I. Gathering data

II. Labeling the data

III. Training model

I. Gathering data

The total data collected for the processing is shown in the Table 5.1. This image dataset would be preprocessed before training of the Faster-RCNN object detection model.

Solid waste image type	No of image
Plastic	50
Glass	50
Paper	128
Metal	120
Total	348

Table 5.1 Total image dataset

The collected data set is placed in model/research/object_detection/images, by defining the images folder. We placed 62 images (32 plastic and 27 glass) in *models/research/object_detection/images/train* for training by defining train folder. The image preprocessing also performed here to make the collected image to the same size, we

used Annex 3 script to accomplish. Once the data is collected, preprocessed, and placed in the right place, the next action is labeling the data.

II. Labelling Data

We used LabelIMG version 1.8.0 available *https://tzutalin.github.io/labelImg/windows_v1.8.0* tool for labeling the images. There may have several labeling tools, but we choose the LabelIMG tool because it is free and easy to use. We labeled all the data as plastic and glass in which supervised image classification. While labeling, the tool auto-generates the XML properties of the data as shown in Figure 5.3. This property of the image further used to train the model.

```
1  <annotation>
2      <folder>train</folder>
3      <filename>WIN_20190615_11_41_53_Pro.jpg</filename>
4      <path>D:\models\research\object_detection\images\train\WIN_20190615_11_41_53_Pro.jpg</path>
5      <source>
6              <database>Unknown</database>
7      </source>
8      <size>
9              <width>800</width>
10             <height>600</height>
11             <depth>3</depth>
12     </size>
13     <segmented>0</segmented>
14     <object>
15             <name>plastic</name>
16             <pose>Unspecified</pose>
17             <truncated>0</truncated>
18             <difficult>0</difficult>
19             <bndbox>
20                     <xmin>214</xmin>
21                     <ymin>13</ymin>
22                     <xmax>558</xmax>
23                     <ymax>599</ymax>
24             </bndbox>
25     </object>
26 </annotation>
```

Figure 5.3 Autogenerated XML based image property

As shown in Figure 5.3 on the image annotation script, the class of object is plastic mentioned in Line number 15. Similarly, all the labeled images have auto-generated

corresponding XML annotations. Once, the image is successfully labeled and annotation generated, the next is training or learns the model to have the knowledge of classes of objects.

To begin the training process, we transform the autogenerated XML-based property of an object to CSV format because the training runs by reading CSV property of an object. To make this we use scripts on Annex 4 file named xml_to_csv.py which is added under the object_detection folder. The XML to CSV transforming could be performed using command *python xml_to_csv.py* running on Anaconda prompt on the path of *models/research/object_detection*. After those commands are successfully executed, *train_labels.csv and test_labels.csv* files are created. The sample contents of the file are shown in Table 5.1.

Table 5.2 CSV based image property

Filename	Width	Height	Class	xmin	Ymin	xmax	Ymax
WIN_20190615_11_41_53_Pro.jpg	800	600	Plastic	214	13	558	599
WIN_20190615_11_41_54_Pro.jpg	800	600	Plastic	7	182	358	533
WIN_20190615_11_41_55_Pro (2).jpg	800	600	Plastic	325	82	571	328

III. Creating TensorFlow Records (TFrecord)

TFrecord is a binary file format storage of data that has a significant impact on the performance of their training time of your model. The TFRecords would serve as input data for the training of the model. To generate TFRecords binary files we used generate_tfrecord.py scripts available in https://github.com/datitran/raccoon_dataset/blob/master/generate_tfrecord.py and placed under the object_detection folder. In the script, we define the class of objects like plastic and glass class from line numbers 21 to 27 as shown in Figure 5.4.

```
21  def class_text_to_int(row_label):
22          if row_label == 'plastic':
23              return 1
24          elif row_label == 'glass':
25              return 2
26          else
27  None
```

Figure 5.4 Image class definition

Once, generate_tfrecord.py modified according to our class, the CSV file generated using Annex 4 script would generate as tfrecords binary format file. We will generate a file named train_labels.csv. The train_labels.csv would be generated as train.record using command running on object_detection path: python generate_tfrecord.py --csv_input=images/train_labels.csv --image_dir=images/train --output_path=train.record

Now, TensorFlow file named train.record are generated and placed under the object_detection folder path, which would be used as input data for training the model.

IV. Training model

We used the pre-trained faster_rcnn_inception_v2_coco_2018_01_28 model. We choose a pre-trained model because it saves time from developing the model from scratch. Now we are going to train our own classes of data set to the pre-trained model. To begin the training two important things should be successfully achieved, one is defining label-map and the other is defining training configuration files. Both files should be saved in object_detection/training, in which the training folder should be created. The label map is an id to a class name definition and could be written with any text editor and should be saved extension file .pbtxt. In addition, one should be careful while defining a label map because the number of items in the label map must be matched to the class labels defining generate_tfrecord.py script. The content of the label map is shown in Figure 5.5 and saved as labelmap.pbtxt in the object_detection/training folder.

```
item{
    id:1
    name:'plastic'

}
item{
    id:2
    name:'glass'
}
item{
    id:3
    name:'paper'

}
item{
    id:4
    name:'metal'
}
```

Figure 5.5 Label map content

Once label mapping is completed, we configure the training configuration file.The configuration is modified according into the scenario of on-site solid waste segregation. The pre-trained model has their own configuration file(faster_rcnn_inception_v2_coco_2018_01_28) and we place under object_detection/training. In the configuration, the number of classes, the pre-trained model, and TensorFlow files such as train.record should be rereferred. The number of classes set to 4, since we have plastic ,paper, metal and glass class of a dataset as: -

num_classes: 4.

The pre-trained model should be referred at line number 106 on fine_tune_checkpoint and set to: -

fine_tune_checkpoint:"D:/models/research /object_detection/faster_rcnn_inception_v2_coco_2018_01_28/model.ckpt".

The train.record is referred as:-

input_path: "D: /models/research/object_detection/train.record"

Finally, we should refer our label map as: -

label_map_path:"D:/models/research/object_detection/training/labelmap.pbtxt"

After defining training configuration successfully, we can start the training process by the following command on anaconda prompt at the path object_detection folder.

python model_main.py --logtostderr --train_dir=training/ --pipeline_config_path=training/faster_rcnn_inception_v2_pets.config , here the model_main.py is used to train the model available in the object_detection path.

At the time of training, the model would be generated under *object_detection/training* path with file name model.ckpt-XXXX, the XXXX are auto-generated numbers. There may have several autogenerated models with postfix numbers. For instance, in our case during training the generated model starting from model.ckpt-808, 908, 1007,1109, and model.ckpt-1207. The number indicates the accuracy of the model. The latest generated model the more the accuracy. Once, the model is successfully trained and generated, we need to export the latest models to object_detection/inference_graph. The path of the inference graph is directly called for real-time object detection. To generate such files, one must run the following command in anaconda prompt on the object_detection path.

python export_inference_graph.py --input_type image_tensor --pipeline_config_path training/faster_rcnn_inception_v2_pets.config --trained_checkpoint_prefix training/model.ckpt-1207 --output_directory inference_graph

The trained model exported to the inference_graph folder in which the real-time object detection referred to as Object Detection Model. Finally, the real-time object detection can be function as shown in the Figure 5.6 by running the code available in the Annex 5.

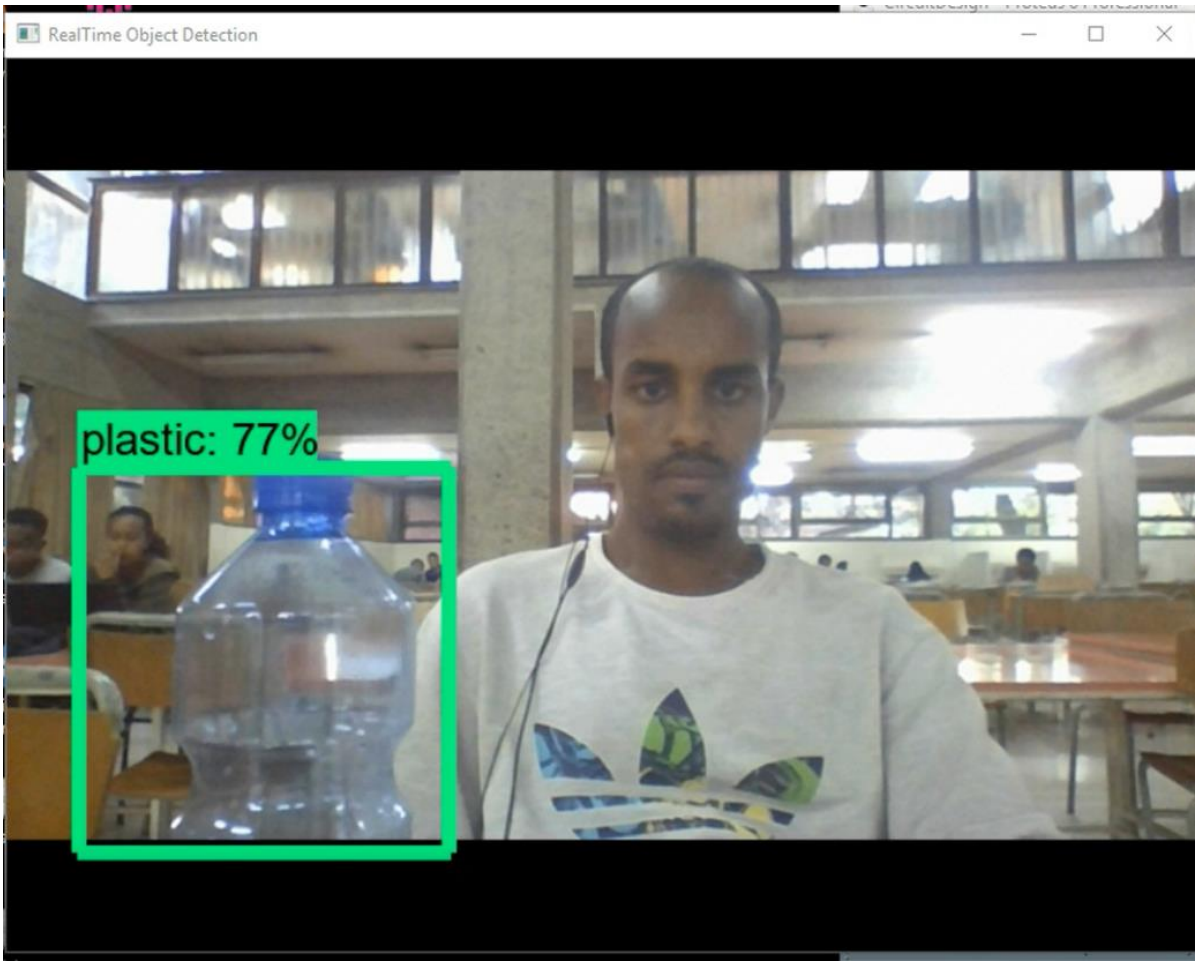

Figure 5.6 Real-time plastic waste detection

After the real-time object detection is successfully implemented the title-based communication implementation is required to send the title to actuation service provider. To, implement the Title-Based Communication, initially, the PySerial library should be installed. The advantage of PySerial Library is to communicate Python-related applications to the external world. Through PySerial library, we can define COM ports and bound raters which used as a tunnel for external communication. The real-time object detection is to identify the visual information at real-time whereas the Title-Based Communication component facilitates the title-based communication to service provider. For instance, if the running state of real-time object detection, detects plastic object, the titled based communication component communicates a message ‘p’ to actuation service providing. The full implementation of the title-based communication is available in Annex 2

Actuation service provider implementation

The Service Provider component delivers services based on the title it receives from the real-time object detection. Actuation service providing can be realized by Microcontroller which is usually used as central processing in building IoT based applications. There are different IoT hardware simulation devices and we compare as of the following in Table 4.2.

Table 5.3IoT -breadboard comparison

	Arduino	**Raspberry Pi**	**Beagle board**
Processor	ARM11	ATMEL AVR	ARM CORTEX
GPU	None	Broadcom VideoCore IV	Power VR SGX530
Pins	14(6 of providing PWM output)	40 Pin	2x 46 pin headers
RAM	2KB (ATmega328)	512MB SDRAM	512 MB DDR3
EEPROM	1KB (ATmega328)	SD	2GB
Operating system	None	Linux	Linux, Android, Cloud9 IDE on Node.js
Compatibility platform	Microsoft, Linux and Mac OS	Linux/window10 Pi 2B and above	Linux
Clock Speed	16 MHz	700 MHz to 1.4 GHz	1.5 GHz
USB	1	2	1
Ethernet	No	Yes	Yes
Hardware opensource	Full opensource	Partially opensource	Full opensource
I/O Connectivity	SPI I2C UART GPIO	SPI DSI UART SDIO CSI GPIO	UART, SPI, I2C, CAN bus

Based on the above comparison of different breadboards, Arduino is selected. Because the device is a completely opensource hardware platform that is programmed using an opensource programming environment which is Arduino IDE (Integrated Development Environment) either C++ or C. In windows platform of Arduino IDE generates the .hex files at the time of compilation and could be easily uploaded to through USB and act as an embeds system of the Arduino Uno microcontroller. It simple for electronics projects and prototyping such as you can easily connect some LED sensors, actuators into the board directly.

The Actuation service providing implementation is simulated as hardware and software implementation. The hardware implementation is designed using Proteus 8 professional Electronic Circuit design tool which is a proprietary software tool suite used primarily for electronic design automation [65]. This tool mainly used by electronic design engineers and technicians to create schematics and electronic prints for manufacturing printed circuit boards. The hardware implementation is using Arduino Uno R3 microcontroller consists of functions shown in Figure 5.7.

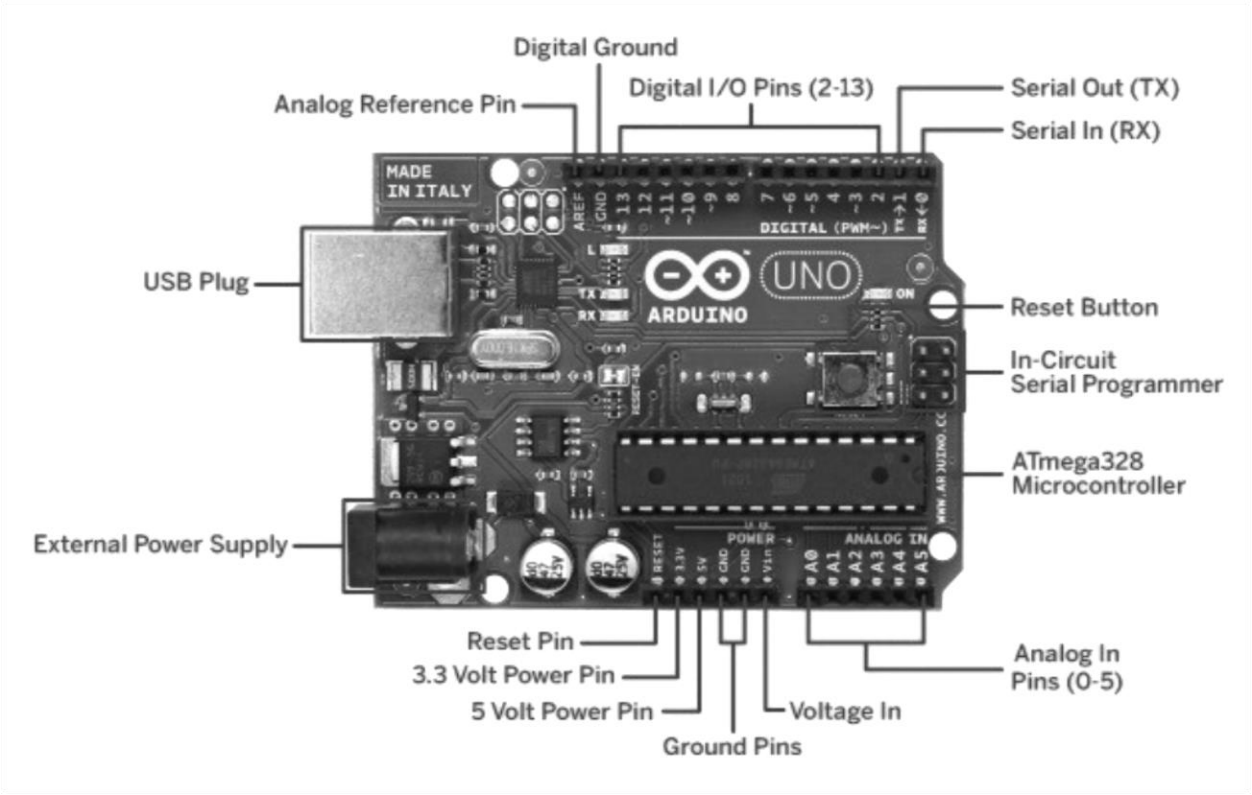

Figure 5.7 Arduino pin functions

Arduino Power Pins

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or another regulated power

source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3-volt supply generated by the on-board regulator. The maximum current draw is 50 mA.
- **GND.** Ground pins.

Arduino Input/output Pins

Each of the 14 digital pins shown on Figure 4.6 on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions.

They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **SERIAL**: 0(RX) and 1(TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- External Interrupts 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- **PWM**: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
- **SPI**: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- **LED**: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

- **TWI**: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

- **AREF**. Reference voltage for the analog inputs. Used with analogReference().

- **RESET**. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.
- The Arduino Uno has 6 analog inputs, labeled A0 through A5, each of which provides 10 bits of resolution (i.e. 1024 different values) and they measure from ground to 5 volts.

For the purpose of simulation, the Actuation service provider and physical actuation is designed and simulated as one circuit implementation. The general hardware circuit design is shown in Figure 5.8, consists of Arduino Uno R3, PWM-Servo motor, wires, COMPIM, VCC, and GND. As shown in Figure 5.8 four servo motors are connected to Arduino Uno R3.

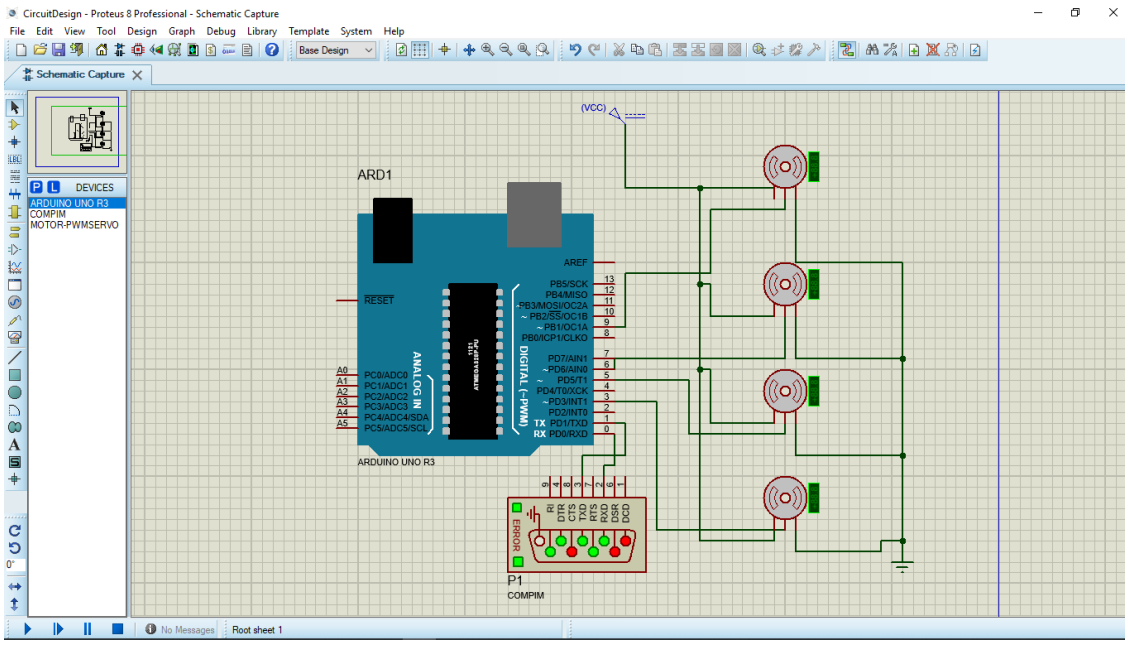

Figure 5.8 IoT Service Provider -Actuation circuit Design

To realize the function Actuation service provider, Arduino Uno R3 has an embedded system that is used to power up the Arduino Uno hardware implementation. So, to an embedded system was implemented using Arduino IDE and the full code is available in Annex 1. The embedded system is developed using the C++ programming language and uploaded in the Arduino Uno microcontroller designed in the previous circuit designed. As per circuit design since Actuation service provider (Arduino Uno R3) and Physical actuation (Servo Motor) designed together.

The embedded system running on Arduino Uno R3 is depend on the title it receives from the Real-time object detection component. All the write() function is the actuation service, which sends actuation command to the respective servo motor(Actuation component). The actuator implementation also designed in the Proteus tool with a wired connection to the Arduino Uno R3. The servo motor is to open and close based on the service(actuation command) received from the Arduino Uno R3 component. We define the servo motor to rotate 180 degrees while receiving an actuation command and back to 0 degrees.

As shown in the circuit design and code, four servo motors attached to Arduino Uno R3 pins of 3, 5, 6, 9 which is used to provide an 8-bit PWM output with the analogWrite() function. The other is wires which are used to connect components and facilitates the communication between components. The servo motors are powered up 5v using VCC and connected to GND as shown in Figure 5.8.

In addition, the COMPIM component is attached to Arduino Uno R3. This component is used to receive titles from the Real-time object detection component by defining COM port and bound rate for Arduino Uno R3 which is connected to the TX and RX pin. So, through COMPIM, it is incoming serial data is buffered and presented to the Arduino Uno R3 as a digital signal. The definition of the COMPIM is shown in Figure 5.9, the physical port is defined as COM1 and the bounding rate as 9600.

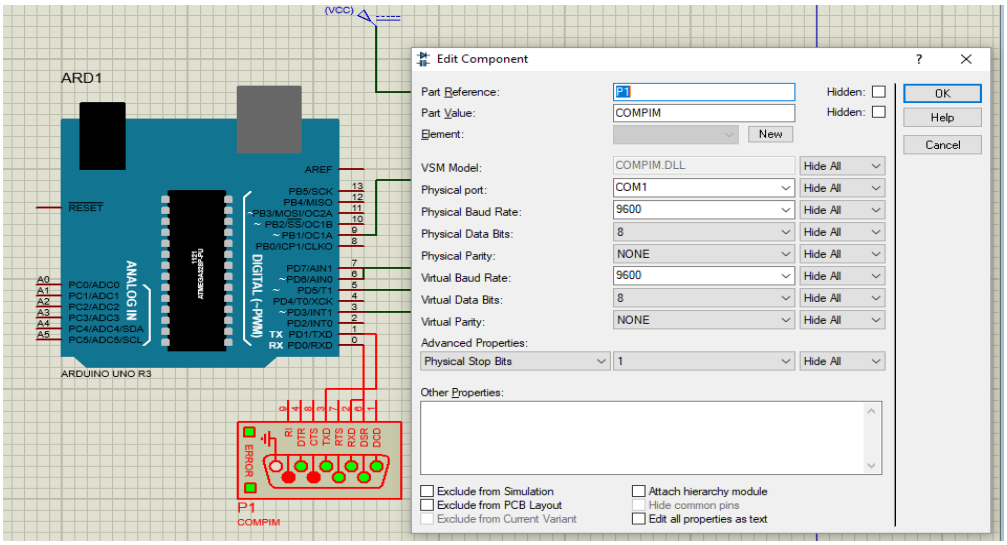

Figure 5.9 COMPIM Definition

To simulate the communication between the real-time object detection component and the Arduino Uno microcontroller component, we used a VSPE tool. This tool used to create COM ports, and support to communicate two or more application port wise even in the single machine as shown in Figure 5.10. two ports were created COM1 and COM2. COM1 is onside of the Service Provider Component whereas COM2 is onside of the Real-time object detection component.

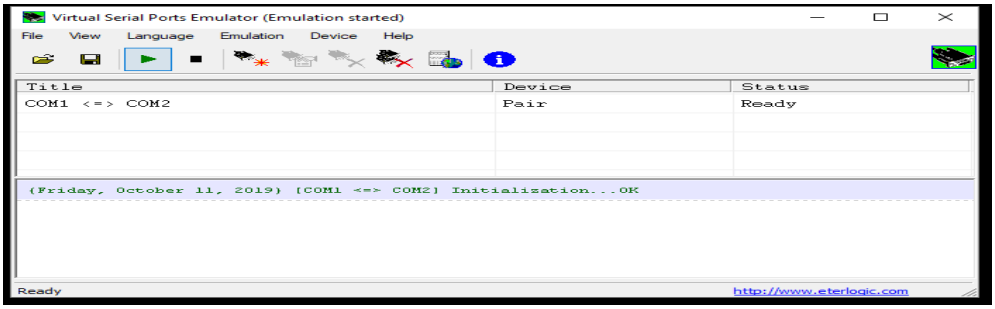

Figure 5.10 VSPE Port

5.3 Prototype demonstration

As mention earlier, the framework would be demonstrated by the application of real-time solid waste segregation. Consider airlines, hotels, and any public area. One can see here and there solid wastes thrown like plastic, glass or paper or it is available manual dustbin which is labeled as plastic, glass, or paper. The institution usually avails manual dustbin because to automate this it asks great investment, may requires renting cloud service, or may be processed on edge computing. The reason is that solid wastes such as plastic, glass, or paper should be visually inspected. So, real-time object detection can't be processed in low-performance IoT devices. Usually, after the camera, the captured images of solid waste sent to the cloud or edge server for further identification. Now, we demonstrate our proposed minimal solution on real-time solid waste segregation which can be implemented below edge computing.

The demonstration of the proposed framework is facilitated by running the real-time object detection, Designed Circuit, and VSPE tool. If all components are at running state, we can check whether the Real-time object detection component inspects the visual information and sends the title through Title-Based Communication to Actuation service provider. For instance, if the Real-time object detection detects plastic material (plastic waste), a title will

be sent to the Actuation service provider (Arduino Uno) microcontroller through serial communication. Then, the Arduino Uno microcontroller automatically sends an actuation command to the respective servo motor. Finally, the servo motor rotates 180 degrees anticlockwise and backs to 180 degrees clockwise to facilitate the opening and closing actuation.

For the simulation of the framework we had been trained in our Object Detection Model for only two objects such as plastic and glass, so the Real-time object detection component only inspects two objects results in two Title-Based communication. These further results the Service Provider component deliver only two actuation service and two servo motor will be used for simulation purposes.

Plastic- visual information and Actuation simulation

As shown in the Figure 5.11, there are two windows, the first is Real-time object detection component running window which it detects plastic object at Accuracy of 88%, the second window is the hardware circuit of Service Provider Component and Actuation which is at running state, now bottom servo motor is rotating right since it receives actuation command from Arduino Uno.

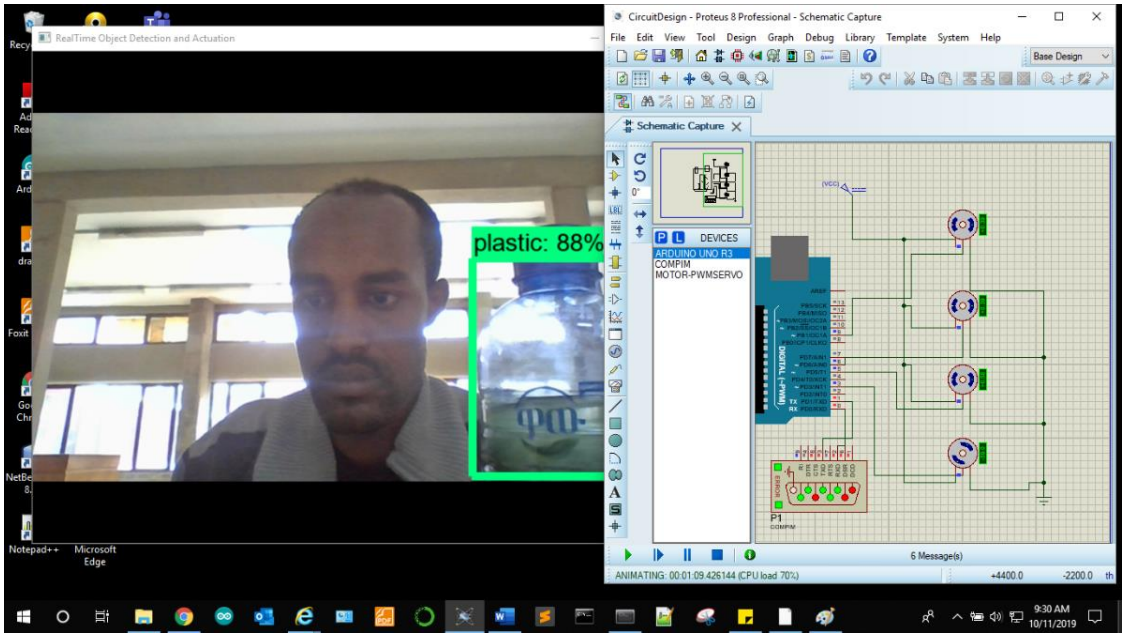

Figure 5.11 Plastic Object Detection and Actuation simulation

Glass-visual information and Actuation simulation

This is another Real-time object detection simulation for the reality of the proposed framework. The glass objects are detected at the accuracy of 70% and another servo motor is rotated based on the commands it receives from the Real-time object detection component as shown in Figure 5.12.

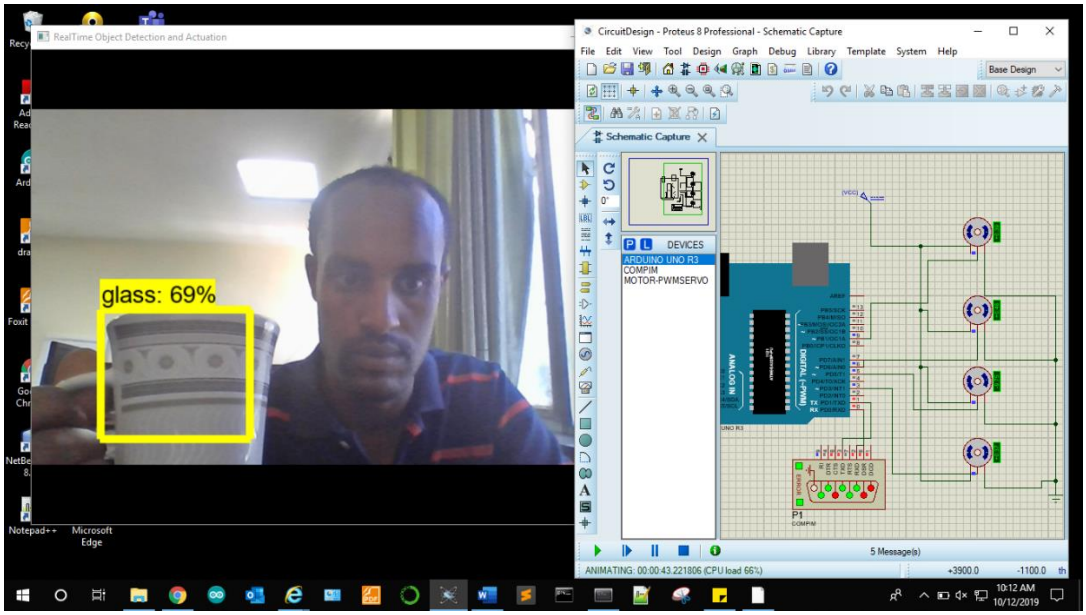

Figure 5.12 Glass-Object Detection and Actuation

Glass and Plastic-visual information and Actuation simulation

Now, what if two objects detected simultaneously, as shown in Figure 5.13 the two servo motors attached to pin 3 and 5 are rotated whereas the other two servo motors remain 0 degrees.

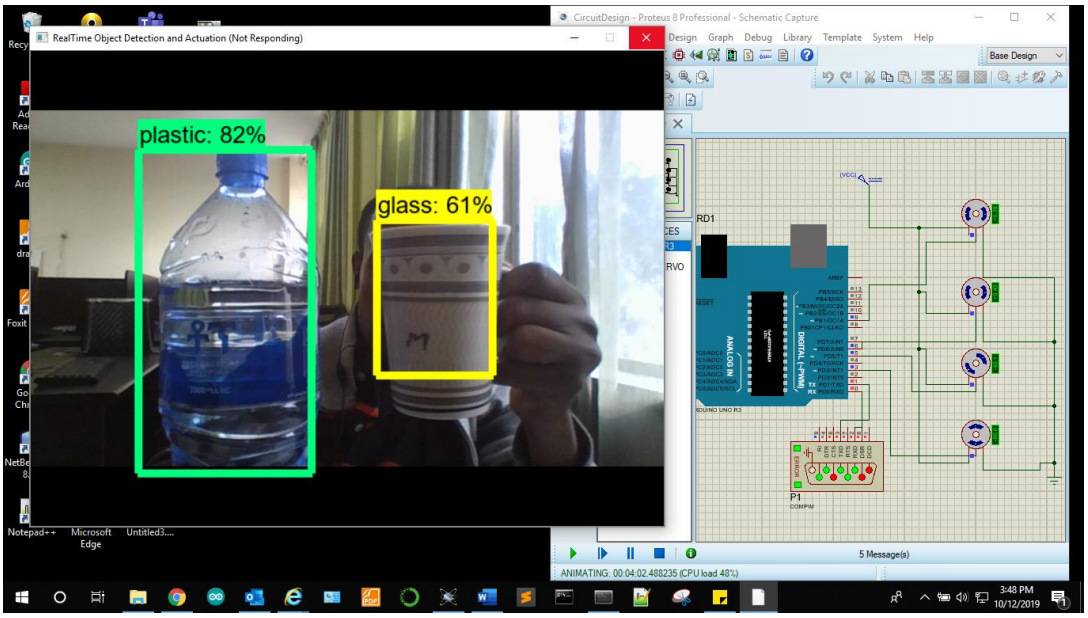

Figure 5.13 Glass and Plastic-Object Detection and Actuation

5.4 Experiments

The importance of experiments is showing the perfection and gap on the proposed solution and to indicated important tips for future development on the solution based on the current experimental outputs. To experiment with the proposed framework, we would facilitate in the way of how much the action is exactly taken based on the detected object. The accuracy, object type, and the action would be clearly indicated. Table 5.2 is the detail of the experiments.

Table 5.4 Experiments of the proposed solution

No	Visualized Image type	Trained/Untrained image	Detected Class	Accuracy	Actuator action
1	Plastic	Trained	Plastic	95	Action taken
2	Glass	Trained	Glass	56	Action taken
3	Paper	Untrained	-	-	No action
4	Metal	Untrained	-	-	No action
5	Plastic	Trained	Plastic	95	Action taken
6	Glass	Trained	Glass	56	Action taken
7	Plastic	Trained	Plastic	80	Action taken

8	Glass	Trained	No detection	-	No Action
9	Plastic	Trained	Plastic	86	Action taken
10	Glass	Trained	Glass	53	Action taken
11	Plastic	Trained	Plastic	70	Action taken
12	Glass	Trained	No detection	-	No action

In table 5.2 heading, visualize image type is the object type visualize by the camera while real-time object detection is running. The trained/untrained column is to indicate whether the image is a category of trained in the DL process or not. The detected class is to indicate the class of detected objects from the model. The accuracy is the measurement of how much image is correctly recognized according to the trained model. The more training the model the more accuracy. The actuator action refers to whether an action is taken or not based on the detected object. As per implementation, the action of the actuator is to rotate 180-degree opening and back to 0-degree closing if actuate command is received.

So, the experiments in Table 5.1 there are twelve experimental results. We check ten trained categories of the image while real-time object detection is running state. For the experiment, eight times are correctly detected and correctly the action taken by the actuator. However, two times not detected as well no action taken. One thing noted here is if the visualized image is not detected no action is taken, this implies there wouldn't have the wrong action. But, if Real-time object detection wrongly detects as the plastic class of image the actuator acts, which is wrong.

In addition, during the experiment, we test two untrained images i.e. paper and metal which are not trained during Deep learning. Her visual information does not detect so no action taken, this meets the target of the framework. The actuator should block for those unrecognized objects.

In general, the accuracy of the actuator depends on the accuracy of the DL model classifier. 'no action' refers to two scenarios. One the image which is not completely trained to be detected and the other is even though it trained the model mayn't detect the object. So, the error rate of the actuator is highly depending on the errors made in the DL model detection error.

Chapter 6 : Conclusions and Future Work

6.1 Conclusion

In this thesis, we have developed a framework for the scenarios that require moderate processing capacity in IoT and further requires real-time actuation that fixes the challenges of low-performance IoT devices to act on visual information like the on-site solid waste segregation use cases. Due to the low-performance nature of IoT devices and the high computational resource requirement of visual information processing, building IoT on real-time object detection is usually cloud service dependent. So, we have been developed a framework that can be run locally which further solution for several scenarios especially for those scenarios that does not require cloud mandatory services like security and management. The framework was presented in a four-layer approach having perception, transport, processing, and application layer. Because, in three layers, the storage, processing, and transportation collapsed in one layer(network layer) which doesn't have opportunity to detach the real-time object detection from low-performance IoT devices, and five-layer includes the security and management in its last layer(business layer) which is not applicable for our framework target solution. In this regard, four-layer were preferred.

Through four-layer, the real-time object detection component can be detached even locally and adding a communication component as part of real-time object detection component to communicate the processed result to service provider IoT devices, and the low-performance nature of IoT device which is unable to process on high computation were resolved and some scenarios like on-site waste segregation can be facilitated locally in four-layer approach. To realize the framework, we have been different approaches and different methodologies. We used the latest DL algorithm for real-time object detection which is the Faster-RCNN algorithm to develop object detection model and TensorFlow Object Detection API framework which is one of the most known Google DL process development frameworks. We design and simulate the IoT components of the framework on the proteus circuit design tool, tested the framework to act at real-time upon visual information while based on the real-time object detection result. In general, this thesis contribution high in which lots of scenarios can be resolved through the framework and the study may highly contribute to playing as groundwork for further advancement of the thesis.

6.2 Contribution

The thesis contributes as an alternative solution for the scenarios that requires real-time object detection, inspection, and physical actuation like an on-site solid waste segregation. For on-site solid waste segregation, it can play as a supportive solution at the time of collection. If one has going to use for similar scenario, the model should be retrained by collecting new types of image dataset for new types of scenario. So, the design architecture can be applicable for any types of scenarios which wants real-time object detection and actuation. The Faster-RCNN architecture is adapted to detect solid waste at real-time while the types of solid are visualized by any standard camera. The embedded code which would be run on IoT devices like Arduino uno would deliver an actuation service for real-time physical actuation, this can be another contribution of this thesis.

In addition, the real-time on-site solid waste segregation is developed which can be supportive solution to minimize the mixed solid waste collection. So, at the time of waste collection this solution can play a role of segregation waste as early before the waste disposed in mixed way. This helps the recycling process of final waste management.

6.3 Future work

Regarding future work, the achievement of this thesis is the framework that can be implemented for those scenarios that require visual information which further requires actuation. However, as shown in the experiment, while the real time object detection is running and detects, the trained object actuation is automatically accomplished. It is clearly visible that if one implements the framework for certain scenarios, while the camera visualizes the trained object, the framework will proceed with normal functioning. It only depends on the camera vision. For instance, if the framework is implemented on on-site solid waste segregation, while the camera visualizes the trained object like plastic the dustbin will perform opening and closing, however, this may happen if one has passed through having plastic bottles of water around dustbin. To resolve such issue, the concept of context-awareness may important and one can further improve the framework by adding context awareness component. In addition, this study wants further enhancement regards the communication protocol. Serial communication(wired) were used, but on further study other types of communication protocol should be considered, like MQTT.

References

[1] Andrew Whitmore, Anurag Agarwal, Li Da Xu, "The Internet of Things—A survey of topics and trends," *Springer*, 2014.

[2] Rafael C.Gonzalez, Richar E. Woods, Digital Image Processing, Biblioteca Central: Prentice Hall, 2002.

[3] Hany F. Atlam, Gary Wills and Robert John Walters, "Fog Computing and the Internet of Things: A Review," *researchgate*, 2018.

[4] Soumya Kanti Datta ; Christian Bonnet, "An edge computing architecture integrating virtual IoT devices," in *IEEE*, Nagoya, Japan, 2017.

[5] Silpa Kaza, Lisa Yao, Perinaz Bahda-Tata, and Frank Van Woerden, "A Global Snapshot of Solid Waste Management to 2050," *WORLD BANK GROUP*, 20 10 2018.

[6] Pallavi Sethi and Smruti R. Sarangi, "Internet of Things: Architectures, Protocols, and Applications," *Journal of Electrical and Computer Engineering*, vol. 2017, p. 25, 2017.

[7] Zach Leidall,Abhishek Chandra, Jon Weissman, "An Edge-based Framework for Cooperation in Internet of Things Applications," *The advanced computing systems association* , 2018.

[8] By Erik W. Anderson, Gilbert A. Preston, and Claudio T. Silva, "Using Python for signal Processing andVisUalization," *IEEE*, vol. 12, no. 4, pp. 90 - 95, 2010.

[9] Bernadette M. Randles, Milena S. Golshan, Irene V. Pasquetto Christine L. Borgman, "Using the Jupyter Notebook as a Tool for Open Science: An Empirical Study," University of California, Los Angeles USA, 2017.

[10] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, "TensorFlow: A System for Large-Scale Machine Learning," in *USENIX*, Savannah, GA, USA, 2016 .

[11] Christopher Olston, Noah Fiedel, Kiril Gorovoy, Jeremiah Harmsen, Li Lao, Fangwei Li,Vinu Rajashekhar, Sukriti Ramesh and Jordan Soyke, "TensorFlow-Serving: Flexible, High-Performance ML Serving," *arxiv*, 2017.

[12] Debasis Bandyopadhyay ,Jaydip Sen, "Internet of Things - Applications and Challenges in Technology and Standardization," *arxiv*, 2011.

[13] Dhananjay Singh, Gaurav Tripathi , and Antonio J. Jara, "A survey of Internet-of-Things: Future Vision, Architecture, Challenges and Services," *IEEE*, 2014.

[14] Vandana Sharma, Ravi Tiwari, "A review paper on “IOT” & It‟s Smart Applications,"

International Journal of Science, Engineering and Technology Research (IJSETR), vol. 5, no. 2, 2016.

[15] Sreeraj S, Suresh Kumar N,G Santhosh Kumar, "A Framework for predicting the performance of IoT protocols, a Use Case based approach," in *IEEE*, 2017.

[16] Burak H. Çorak,Feyza Y. Okay, Metehan Güzel, Şahin Murt, Suat Ozdemir, "Comparative Analysis of IoT Communication Protocols," *IEEE*, pp. 1-6, 2018.

[17] Abhishek R. Chandan ; Vaishali D. Khairnar, "Bluetooth Low Energy (BLE) Crackdown Using IoT," in *IEEE*, Coimbatore, India, 2018.

[18] Dr. M L Sharma, Sachin Kumar, Nipun Mehta, "INTERNET OF THINGS APPLICATION, CHALLENGES AND FUTURE SCOPE," *International Research Journal of Engineering and Technology (IRJET)*, vol. 05, no. 02, 2018.

[19] Guanghui Pan, Jia He, "Automatic Stabilization of Zigbee Network," in *IEEE*, Chengdu, China, 2018.

[20] Mritunjay Kumar,Km Annoo,Raman Kumar Mandal, "The Internet of Things Applications for Challenges and Related Future Technologies & Development," *International Research Journal of Engineering and Technology (IRJET)*, vol. 05, no. 01., pp. 1-7, 2018.

[21] Shubham Saloni ,Achyut Hegde, "WiFi-Aware as a Connectivity Solution for IoT," in *Maharashtra Institute of Technology*, Pune, India , 2016.

[22] Shilpa Devalal , A.Karthikeyan , "LoRa technology-an overview," in *IEEE*, Vellore, India, 2018.

[23] Yuri F. Gomes, Danilo F. S. Santos, Hyggo O. Almeida and Angelo Perkusich, "Integrating MQTT and ISO/IEEE 11073 for Health Information Sharing in the Internet of Things," in *IEEE*, Federal University of Campina Grande, Brazil, 2015.

[24] Shang Guoqiang, Chen Yanming, Zuo Chao,Zhu Yanxu, "Design and Implementation of a Smart IoT Gateway," in *IEEE*, Beijing, China, 2013.

[25] Hao Chen,Xueqin Jia, Heng Lia, "A Brief Introduction to IoT Gate way," in *IEEE*, Bejjing ,China, 2011.

[26] Katushiro Naito, "A Surey of Internet of things:Standards , challenges, Future prospects," *Journal of information processing*, vol. 25, pp. 23-31, 2017.

[27] C Rajasekaran ; K Raguvaran, "Microcontroller Based Reconfigurable IoT Node," in *IEEE*, Poitiers, France, 2018.

[28] Yunze He, Ruizhen Yang,, "Magnetic Sensor Based Pulsed Eddy Current for Defect Detection and Characterization," *ScienceDirect*, 2017.

[29] Cristian González García, Daniel Meana-Llorián, B. Cristina Pelayo G-Bustelo, and Juan Manuel Cueva Lovelle, "A review about Smart Objects, Sensors, and Actuators," *Special Issue on Advances and Applications in the Internet of Things and Cloud Computing*, 2017.

[30] Bo Yuanab. Lu Liub, Nick Antonopoulosb, "Efficient service discovery in decentralized online social networks," *Future Generation Computer Systems*, vol. 86, pp. 775-791, 2018.

[31] D. T. I. R. M.R. Abdmeziem, "Architecting the Internet of Things: State of the Art," *Springer* , 2016.

[32] Sankar Sennan, Palaniyappan Srinivasan, "Internet of Things (IoT): A survey on empowering technologies, research opportunities and applications," *International Journal of Pharmacy and T echnology*, vol. 8, no. 4, 2016.

[33] Miao Wu, Ting-lie Lu, Fei-Yang Ling, ling Sun, Hui-Ying Du , "Research on the architecture of Internet of things," in *ICACTE*, Beijing, 2010.

[34] Posland S., Ubiquitous Computing:Smart Devices ,Environments and Interactions, Wiley, 2009.

[35] Keyur K Patel, P G Scholar, Sunil M Patel and Carlos Salazar, "Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges," *researchgate*, vol. 6, no. 5, 2016.

[36] Salha M. Alzahrani , "Sensing for the Internet of Things and Its Applications," in *IEEE*, Taif, Saudi Arabia, 2017.

[37] wikipedia, "wikipedia," wikipedia, 04 01 4 January 2019. [Online]. Available: https://en.wikipedia.org/wiki/List_of_camera_types. [Accessed 29 06 2019].

[38] Gradimirka Popovic, Nebojsa Arsic, Branimir Jaksic, Boris Gara, Mile Petrovic, "Overview, Characteristics and Advantages of IP Camera Video Surveillance Systems Compared to Systems with other Kinds of Camera," *IJESIT*, vol. 2, p. 5, 2013.

[39] Ya-Qin Zhang, Visual Information Representation, Communication, and Image Processing, New York: Marcel Dekker AG, 1999.

[40] Pooja Kamavisdar, Sonam Saluja, Sonu Agrawal, "A Survey on Image Classification Approaches and Techniques," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, no. 1, 2013.

[41] Maneela Jain, Pushpendra Singh Tomar, "Review of Image Classification Methods and Techniques," *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, no. 8, 2013.

[42] I.Gogul, V.Sathiesh Kumar, "Flower species recognition system using convolution neural networks and transfer learning," in *IEEE*, Chennai, India, 2016 .

[43] Shaukat Hayat, She Kun, Zuo Tengtao, Yue Yu, Tianyi Tu, Yantong Du , "A Deep Learning Framework Using Convolutional Neural Network for Multi-class Object Recognition," in *IEEE*, Chengdu, China , 2018.

[44] Muhammad Imran Razzak,Saeeda Naz, "Deep Learning for Medical Image Processing: Overview, Challenges and the Future," *ResearchGate*, pp. 6-25, 2018.

[45] Hijazi, Samer, Rishi Kumar, and Chris Rowen, "Using convolutional neural networks for image recognition," in *Cadence Design Systems Inc*, San Jose, CA, USA, 2015.

[46] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio, Guadarrama, Kevin Murphy, "Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors," in *IEEE*, Honolulu, HI, USA, 2017.

[47] Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE*, Columbus, OH, USA, 2014.

[48] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *arXiv*, 2015.

[49] Derry Alamsyah and Muhammad Fachrurroz, "Faster R-CNN with Inception V2 for Fingertip Detection in Homogenous Background Image," in *IOP Publishing* , 2019.

[50] Ayush Kapoor, Suchetha I Bhat, "IMPLEMENTATION OF loT (INTERNET OF THINGS) AND IMAGE PROCESSING IN SMART AGRICULTURE," in *International Conference on Computational Systems and Information Systems for Sustainable Solutions* , Sir MVIT, Bangalore , 2016.

[51] R.Divya, A.Hemalatha, M.Priyadarshini M.E., "A Iot-Based Farm Management System In Modern Cities Using Image Processing," *IOSR Journal of Engineering (IOSRJEN)* , pp. 37-38, 2018.

[52] Martin K. van Ittersuma,, Frank Ewert, "Integrated assessment of agricultural systems – A component-based framework for the European Union (SEAMLESS)," *ScienceDirect*, 2008.

[53] A. Beatrice Dorothy, Dr. S. Britto Ramesh Kumar, J. Jerlin Sharmila, "IoT based Home Security through Digital Image Processing Algorithms," *World congress on Computing and Communication Technologies(WCCCT)*, 2017.

[54] Hardik Asnani, Suhaib Khan, Suhaas Nandeesh, Prarthana T.V., "Securing an IoT based Home using Digital Image Processing and an Android Application," *International Research Journal of Engineering and Technology (IRJET)* , vol. 05, no. 08, pp. 1-6, 2016.

[55] Jaya Karthik K, Dr. Sarabpreet Kaur ,Naveen Reddy M, Uma Maheswara , "Smart Parking Using Image Processing," *IJRASET*, vol. 5, no. X, 2017.

[56] Sagar Rane ,Aman Dubey, Tejisman Parida, "Design of IoT Based Intelligent Parking System Using Image Processing Algorithms," in *ICCMC*, Pune, 2017.

[57] Komal S. Shinde ; Prachi H. Bhagat, "Industrial Process Monitoring Using IoT," in *IEEE*, Palladam, India, 2017.

[58] Rukmani P, Gunda Krishna Teja, M Sai Vinay, Bhanu Prakash Reddy K, "InIdustrial Monitoring Using Image Processing, IoT and Analyzing the Sensor Values Using Big Data," in *Science Direct*, Chennai , India, 2018.

[59] V. Krishna Sree, T. Divija, "Survey on Image Compression Techniques and IOT Challenges," *CIIT*, vol. 9, no. 3, 2017.

[60] Yuzhong Yan, Lei Huang, "Large-Scale Image Processing Research Cloud," in *IARIA*, Prairie View, TX, 2014.

[61] Ali Mirarab, Najmeh Ghasemi Fard and Mahboubeh Shamsi, "A cloud solution for medical image processing," *Journal of Engineering Research and Applications*, vol. 4, no. 7, 2014.

[62] Pengfei Hu,Huansheng Ning,Tie Qiu,Yanfei Zhang, and Xiong Luo, "Fog Computing-Based Face Identification and Resolution Scheme in Internet of Things," in *IEEE*, 2016.

[63] Changchun Long , Yang Cao, Tao Jiang and Qian Zhang , "Edge Computing Framework for Cooperative Video Processing in Multimedia IoT Systems," *IEEE*, vol. 20, no. 5, 2017.

[64] Dirk Jacobsen, and Petter Ott, "Cloud Architecture for industrial image processing," *IEEE*, 2017.

[65] Proteus, "Proteus," [Online]. Available: https://www.labcenter.com/. [Accessed 2019 05 20].

Annex

Annex 1: Actuation service provider embedded system

```
#include<Servo.h>

 Servo plastic;
 Servo glass;
 Servo metal;
 Servo paper;
 int servoPos = 1;
 char val;
 void setup() {
  Serial.begin(9600);
  plastic.attach(3);
  glass.attach(5);
  metal.attach(6);
  paper.attach(9);
 }
 void loop() {
  if (Serial.available() > 0) {
            val = Serial.read();
          //Servo motor rotates 180 degree(opening)
            for (servoPos = 0; servoPos < 180; servoPos++)
            {
             if (val == 'p')
                      plastic.write(servoPos);
             if (val == 'g')
                      glass.write(servoPos);
             if (val == 'm')
                      metal.write(servoPos);
```

```
                if (val == 'r')
                        paper.write(servoPos);
                delay(10);
                }
              // Servo motor rotates back to 0 degree (closing)
                for (servoPos = 180; servoPos > 0; servoPos--)
                {
                if (val == 'p')
                        plastic.write(servoPos);
                if (val == 'g')
                        glass.write(servoPos);
                if (val == 'm')
                        metal.write(servoPos);
                if (val == 'r')
                        paper.write(servoPos);
        delay(10);
                }
        }
}
```

Annex 2: Title based communication code

```
from object_detection.core import standard_fields as fields
from object_detection.utils import shape_utils
import serial
s=serial.Serial()
comport=2
s.port="COM{}".format(comport)
s.timeout=100
…
```

```
if classes[i] in category_index.keys():
 class_name = category_index[classes[i]]['name']
  if class_name=='plastic':
  s.open()
  s.write(str.encode('p'))
  s.readline()
  s.close()
  elif class_name=='glass':
  s.open()
  s.write(str.encode('g'))
  s.readline()
  s.close
  else:
  None
  else:
  class_name = 'N/A'
  display_str = str(class_name)
```

Annex 3: image preprocessing code

```
from PIL import Image

import os

import argparse

 def rescale_images(directory, size):

 for img in os.listdir(directory):

  im = Image.open(directory+img)

  im_resized = im.resize(size, Image.ANTIALIAS)

  im_resized.save(directory+img)

 if __name__ == '__main__':

 parser = argparse.ArgumentParser(description="Rescale images")

 parser.add_argument('-d', '--directory', type=str, required=True, help='Directory containing the images')

 parser.add_argument('-s', '--size', type=int, nargs=2, required=True, metavar=('width', 'height'), help='Image size')

 args = parser.parse_args()

 rescale_images(args.directory, args.size)
```

Annex 4: xml to csv transformer script

```
import os

import glob

import pandas as pd

import xml.etree.ElementTree as ET


def xml_to_csv(path):
```

```
        xml_list = []
        for xml_file in glob.glob(path + '/*.xml'):
            tree = ET.parse(xml_file)
            root = tree.getroot()
            for member in root.findall('object'):
                value = (root.find('filename').text,
                         int(root.find('size')[0].text),
                         int(root.find('size')[1].text),
                         member[0].text,
                         int(member[4][0].text),
                         int(member[4][1].text),
                         int(member[4][2].text),
                         int(member[4][3].text)
                         )
                xml_list.append(value)
        column_name = ['filename', 'width', 'height', 'class', 'xmin', 'ymin', 'xmax',
'ymax']
        xml_df = pd.DataFrame(xml_list, columns=column_name)
        return xml_df
    def main():
        for folder in ['train','test']:
            image_path = os.path.join(os.getcwd(), ('images/' + folder))
            xml_df = xml_to_csv(image_path)
```

```
        xml_df.to_csv(('images/' + folder + '_labels.csv'), index=None)
        print('Successfully converted xml to csv.')
main()
```

Annex 5: Real-time object detection full code

In [1]:

```
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile
import io
import pandas as pd

from distutils.version import StrictVersion
from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image

from object_detection.utils import dataset_util
from collections import namedtuple, OrderedDict

# This is needed since the notebook is stored in the object_detection folder.
sys.path.append("..")
from object_detection.utils import ops as utils_ops
```

```
if StrictVersion(tf.__version__) < StrictVersion('1.12.0'):

raise ImportError('Please upgrade your TensorFlow installation to v1.12.*.')
```

In [2]:

```
# This is needed to display the images.

%matplotlib inline
```

Importing Object detection packages¶

In [3]:

```
from utils import label_map_util



from utils import visualization_utils as vis_util
```

Load Model¶

In [4]:

```
MODEL_NAME='inference_graph'

PATH_TO_FROZEN_GRAPH = MODEL_NAME + '/frozen_inference_graph.pb'

PATH_TO_LABELS = 'training/labelmap.pbtxt'
```

Load a (frozen) Tensorflow model¶

In [5]:

```
detection_graph = tf.Graph()

with detection_graph.as_default():

 od_graph_def = tf.GraphDef()

 with tf.gfile.GFile(PATH_TO_FROZEN_GRAPH, 'rb') as fid:

  serialized_graph = fid.read()

  od_graph_def.ParseFromString(serialized_graph)

  tf.import_graph_def(od_graph_def, name='')
```

Loading label map¶

Label maps map indices to category names, so that when our Faster-RCNN predicts 1, we know that this corresponds to plastic.

In [6]:

```
category_index = label_map_util.create_category_index_from_labelmap(PATH_TO_LABELS, use_display_name=True)
```

In [7]:

```
def load_image_into_numpy_array(image):
 (im_width, im_height) = image.size
 return np.array(image.getdata()).reshape(
   (im_height, im_width, 3)).astype(np.uint8)
```

Detection¶

In [8]:

```
# If you want to test the code with your images, just add path to the images to the TEST_IMAGE_PATHS.
PATH_TO_TEST_IMAGES_DIR = 'D:/0-Thesis/Demonistration/models/research/object_detection/test_images'
TEST_IMAGE_PATHS = [ os.path.join(PATH_TO_TEST_IMAGES_DIR, 'image{}.jpg'.format(i)) for i in range(1, 3) ]


# Size, in inches, of the output images.
IMAGE_SIZE = (12, 8)
```

In [9]:

```
def run_inference_for_single_image(image, graph):
  if 'detection_masks' in tensor_dict:
    # The following processing is only for single image
    detection_boxes = tf.squeeze(tensor_dict['detection_boxes'], [0])
    detection_masks = tf.squeeze(tensor_dict['detection_masks'], [0])
    # Reframe is required to translate mask from box coordinates to image coordinates and fit the image size.
    real_num_detection = tf.cast(tensor_dict['num_detections'][0], tf.int32)
    detection_boxes = tf.slice(detection_boxes, [0, 0], [real_num_detection, -1])
    detection_masks = tf.slice(detection_masks, [0, 0, 0], [real_num_detection, -1, -1])
    detection_masks_reframed = utils_ops.reframe_box_masks_to_image_masks(
```

```
      detection_masks, detection_boxes, image.shape[0], image.shape[1])
    detection_masks_reframed = tf.cast(
      tf.greater(detection_masks_reframed, 0.5), tf.uint8)
    # Follow the convention by adding back the batch dimension
    tensor_dict['detection_masks'] = tf.expand_dims(
      detection_masks_reframed, 0)
  image_tensor = tf.get_default_graph().get_tensor_by_name('image_tensor:0')
  # Run inference
  output_dict = sess.run(tensor_dict,
                         feed_dict={image_tensor: np.expand_dims(image, 0)})


  # all outputs are float32 numpy arrays, so convert types as appropriate
  output_dict['num_detections'] = int(output_dict['num_detections'][0])
  output_dict['detection_classes'] = output_dict[
    'detection_classes'][0].astype(np.uint8)
  output_dict['detection_boxes'] = output_dict['detection_boxes'][0]
  output_dict['detection_scores'] = output_dict['detection_scores'][0]
  if 'detection_masks' in output_dict:
    output_dict['detection_masks'] = output_dict['detection_masks'][0]
  return output_dict
```

In [10]:

```
import cv2
#import serial
cap = cv2.VideoCapture(0)


'"s=serial.Serial()
comport=2
s.port="COM{}".format(comport)
```

```
s.timeout=3"'
try:
  with detection_graph.as_default():
    with tf.Session() as sess:
        # Get handles to input and output tensors
        ops = tf.get_default_graph().get_operations()
        all_tensor_names = {output.name for op in ops for output in op.outputs}
        tensor_dict = {}
        for key in [
         'num_detections', 'detection_boxes', 'detection_scores',
         'detection_classes', 'detection_masks'
        ]:
          tensor_name = key + ':0'
          if tensor_name in all_tensor_names:
            tensor_dict[key] = tf.get_default_graph().get_tensor_by_name(
           tensor_name)

        while True:
          ret, image_np = cap.read()
          # Expand dimensions since the model expects images to have shape: [1, None, None, 3]
          image_np_expanded = np.expand_dims(image_np, axis=0)
          # Actual detection.
          output_dict = run_inference_for_single_image(image_np, detection_graph)
          # Visualization of the results of a detection.
          vis_util.visualize_boxes_and_labels_on_image_array(
            image_np,
            output_dict['detection_boxes'],
            output_dict['detection_classes'],
            output_dict['detection_scores'],
```

```
            category_index,
            instance_masks=output_dict.get('detection_masks'),
            use_normalized_coordinates=True,
            line_thickness=8)
          cv2.imshow('RealTime Object Detection and Actuation', cv2.resize(image_np, (800, 600)))
          if cv2.waitKey(25) & 0xFF == ord('q'):
            cap.release()
            cv2.destroyAllWindows()
            break
except Exception as e:
  print(e)
  cap.release()
```

Signed Declaration Sheet

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all source of materials used for the thesis have been duly acknowledged.

Declared by:

Name: _____

Signature: _____

Date: _____

Confirmed by advisor:

Name: _____

Signature: _____

Date: _____