AUTOMATIC BIOMEDICAL WASTE SEGREGATION BIN

A PROJECT REPORT

Submitted by

ASHNA MS

LBT16EC020

NEETHA J

SCT16EC086

PARVATHY AJITH

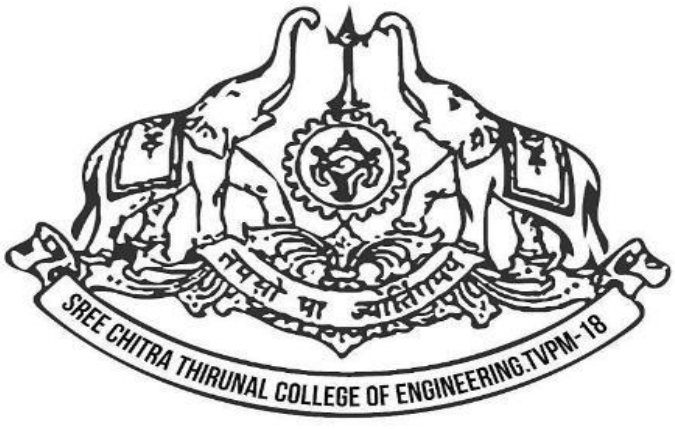
SCT16EC094

VYSHAK SP

SCT16EC122

to

the APJ Abdul Kalam Technological University in partial fulfilment
of the requirements for the award of the Degree
of
Bachelor of Technology
In
Electronics and Communication Engineering



Department of Electronics & Communication Engineering

Sree Chitra Thirunal College of Engineering,

Pappanamcode, Thiruvananthapuram, Kerala - 695018

May 2020

DECLARATION

We undersigned hereby declare that the project report "Automatic Biomedical Waste Segregation Bin", submitted for partial fulfilment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of Asst. Prof. Akhilaraj D**.** This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Thiruvanthapuram

10.07.2020

Ashna MS

Neetha J

Parvathy Ajith

Vyshak SP

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SREE CHITRA THIRUNAL COLLEGE OF ENGINEERING

THIRUVANANTHAPURAM-695 018

CERTIFICATE

This is to certify that the report entitled "**AUTOMATIC BIOMEDICAL WASTE SEGREGATION BIN**" submitted by **ASHNA MS, NEETHA J, PARVATHY AJITH, VYSHAK SP** to the APJ Abdul Kalam Technological University in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in (Electronics & Communication Engineering) is a bonafide record of the project work carried out by them under our guidance and supervision.. This report in any form has not been submitted to any other University or Institute for any purpose.

Prof. SAJITH SETHU P
Associate Professor

Prof. JISU ELSA JACOB
Assistant Professor

Prof. AKHILARAJ D
Assistant Professor

Prof. Dr. LIBISH TM
HEAD OF THE DEPARTMENT

ACKNOWLEDGEMNT

We take this opportunity to express our heartfelt gratitude to **Dr K Prabhakaran Nair**, Principal, Sree Chitra Thirunal College of Engineering, Trivandrum, for providing us with the necessary infrastructure for the successful completion of the implementation phase of our project.

We extend our profound gratitude to **Dr. Libish T M**, Professor and Head of Department, Electronics and Communication for his constant support and encouragement.

We are extremely grateful to our project co-ordinators, **Prof. Sajith Sethu P** and **Prof. Jisu Elsa Jacob** for their valuable suggestions and guidance which has been a major contribution to our understanding of the work.

We are deeply indebted to **Prof. Akhilaraj D**, Assistant Professor, Department of Electronics and Communication, for her guidance and immense support throughout the course of our project.

We express our profound gratitude to **Kerala Development and Innovation Strategic Council** (**KDISC)** for providing an opportunity to exhibit our idea at a state level completion and selecting our project as one of the best innovative ideas of **Young Innovators Program** (2019-22)

We also thank our family, friends and everyone who has supported us either directly or indirectly, without whom this undertaking might not have been possible.

ASHNA MS

NEETHA J

PARVATHY AJITH

VYSHAK SP

ABSTRACT

The existing system of hospital waste disposal requires hospital staff to manually categorize the waste and dispose them into the appropriate coloured bin. This method is susceptible to human errors. Accidentally disposing the waste into a wrong bin may lead to grave consequences as retrieval of the same is impractical and unhygienic. Thus, automating the system only leads to ensuring the avoidance of such human errors. Through this project, we aim to replace the currently existing system of hospital waste disposal, which employs colour coded bins for different types of wastes, by a single unit which is capable of automatically identifying the type of waste and segregating it into one of the four bins provided inside. The valuable time of the hospital staff which could be used to save a life is also saved in the process. The automatic identification is accomplished with the help of object detection techniques using DNN. The identified waste is disposed into one of the four bins incorporated in the unit. These bins are placed on a rotating platform which, as soon as the category of the waste is identified, rotates in order to bring the appropriate bin under the platform containing the waste which then opens to allow the waste to fall into the bin.

TABLE OF CONTENTS

ACKNOWLEDGEMNT i

ABSTRACT ii

LIST OF TABLES vi

LIST OF FIGURES vii

LIST OF ABBREVIATIONS viii

CHAPTER 1 1

INTRODUCTION 1

1.1 OVERVIEW OF BIOMEDICAL WASTE SEGREGATION 1

1.2 MOTIVATION 3

1.3 PROBLEM DEFINITION 4

1.4 SCOPE OF THE WORK 4

CHAPTER 2 5

LITERATURE SURVEY 5

2.1 DEEP LEARNING 5

2.2 DEEP LEARNING PLATFORMS 6

2.2.1 Deep Learning Hardware 6

2.2.2 Software Frameworks 7

2.2.3 Cloud Services 7

2.3 REAL TIME OBJECT DETECTION MODELS 8

CHAPTER 3 11

METHODOLOGY 11

3.1 BLOCK DIAGRAM 11

3.2 FLOWCHART 12

3.3 DATASET CREATION 13

3.4 TRANSFER LEARNING 14

3.5 OBJECT DETECTION AND LOCALIZATION 14

CHAPTER 4 17
COMPONENTS AND SOFTWARES USED 17
4.1 LIST OF COMPONENTS AND SOFTWARES 17
4.2 THEORY 19
4.2.1 Stepper motor 19
4.2.2 Servo Motor 20
4.2.3 Infrared (IR) Sensor 21
4.2.4 Singleshot Multibox Detector (SSD) 22
4.2.5 Arduino UNO 23
4.2.6 Raspberry pi Model 3B 26
4.2.7 OpenCV 27
4.2.8 TensorFlow 1.13.0 27
4.2.9 Creo Parametric 28
4.2.10 Arduino IDE 28
CHAPTER 5 30
IMPLEMENTATION 30
5.1 OBJECT DETECTION AND ARDUINO PROGRAMMING 30
5.2 HARDWARE 35
CHAPTER 6 37
RESULTS AND DISCUSSION 37
6.1 OBJECT DETECTION MODEL 37
6.1.1 Comparing non-quantized and quantized SSD-MobileNet models 37
6.1.2 Training of model and integration onto Raspberry pi 40
6.1.3 Arduino Programming 41
6.2 HARDWARE 43
CHAPTER 7 44
CONCLUSION 44

CHAPTER 8 45

FUTURE SCOPE 45

CHAPTER 9 46

FUNDING RECEIVED 46

REFERENCES 47

APPENDIX 48

LIST OF TABLES

Table 2. 1 Popular open source deep learning frameworks 7
Table 2. 2 Selection of commercial cloud services offering deep learning capabilities 8
Table 4. 1 Pin Description 25

LIST OF FIGURES

Fig 2. 1Basic CNN layers – convolution (left) and pooling (right) 6
Fig 2. 2Nvidia JetsonTX 7
Fig 3. 1 Flowchart 12
Fig 3. 2 An example of Data Augmentation 13
Fig 4. 1 Arduino Uno 17
Fig 4. 2 Raspberry Pi 17
Fig 4. 3 Stepper Motor 18
Fig 4. 4 A4988 Stepper motor driver module 18
Fig 4. 5 Capacitor 18
Fig 4. 6 Servo Motor 19
Fig 4. 7 Raspberry pi Camera Module 19
Fig 4. 8 Working principle of stepper motor 20
Fig 4. 9 Timing diagram of Servo motor 21
Fig 4. 10 Basic working of IR senor 22
Fig 4. 11 Pin structure of Arduino Uno 23
Fig 4. 12 Pin structure of Raspberry Pi Model 3B 27
Fig 4. 13 Creo Parametric 28
Fig 4. 14 Arduino IDE 29
Fig 5. 1 Dataset 31
Fig 5. 2 Label Map 32
Fig 5. 3 3D Model 35
Fig 6. 1 Figure showing the accuracy of cotton (58%) and needle (82%) during detection 38
Fig 6. 2 Figure showing the accuracy of cotton (82%) during detection 38
Fig 6. 3 Figure showing the accuracy of insulin cartridge (87%) during detection 39
Fig 6. 4 Figure showing the accuracy of insulin cartridge (98%) and needle (98%) during detection 39
Fig 6. 5 Training the model in Google Colab 40
Fig 6. 6 Visualization of Total Loss Graph in TensorBoard 41
Fig 6. 7 Arduino IDE 42
Fig 6. 8 Stepper motor and servo motor rotating in response to the output from Raspberry Pi and IR Sensor 42
Fig 6. 9 Prototype 43

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Networks
DNN	Deep Neural Networks
FCN	Fully Connected Network
BMW	Biomedical Waste Segregation
HIV	Human Immunodeficiency Virus
HBV	Hepatitis B Virus
HCU	Healthcare Unit
SVM	Support Vector Machine
YOLO	You Only Look Once
FPS	Frames Per Second
SSD	Singleshot Multibox Detector
RNN	Recurrent Neural Network
IR	Infrared Sensor
GPU	Graphical Processing unit
TPU	Tensor processing unit
OS	Operating system
API	Application programming interface
ROI	Regions of Interest
UART	Universal Asynchronous Reception and Transmission

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF BIOMEDICAL WASTE SEGREGATION

Until fairly recently, medical waste management was not generally considered an issue. In the 1980s and 1990s, concerns about exposure to human immunodeficiency virus (HIV) and hepatitis B virus (HBV) led to questions about potential risks inherent in medical waste. Thus, hospital waste generation has become a prime concern due to its multidimensional ramifications as a risk factor to the health of patients, hospital staff and extending beyond the boundaries of the medical establishment to the general population.

Hospital waste refers to all waste, biologic or non-biologic that is discarded and not intended for further use. Biomedical waste is a subset of hospital waste, which refers to the material generated as a result of diagnosis, treatment or immunization of patients and associated biomedical research. Biomedical waste (BMW) is generated in hospitals, research institutions, health care teaching institutes, clinics, laboratories, blood banks and veterinary institutes. Hospital waste management has been brought into focus in India recently, particularly with the notification of the BMW (Management and Handling) Rules, 1998. The rule makes it mandatory for the health care establishments to segregate, disinfect and dispose their waste in an eco-friendly manner.

Biomedical waste consists of:

1. Human anatomical waste like tissues, organs and body parts

2. Animal wastes generated during research from veterinary hospitals

3. Microbiology and biotechnology wastes

4. Waste sharps like hypodermic needles, syringes, scalpels and broken glass

5. Discarded medicines and cytotoxic drugs

6. Soiled waste such as dressing, bandages, plaster casts, material contaminated with blood, tubes and catheters

7. Liquid waste from any of the infected areas

8. Incineration ash and other chemical wastes

The process of biomedical waste management is laborious, requiring meticulous handling and scientific attention. The multiple phases involved can be sorted into the following processes:

1. Segregation

2. Collection

3. Transportation

4. Scientific treatment

5. Disposal

Segregation refers to the basic separation of different categories of waste generated at source and thereby reducing the risks as well as cost of handling and disposal. Segregation is the most crucial step in bio-medical waste management. Effective segregation alone can ensure effective bio-medical waste management, and therefore our focus is on the segregation stage.

According to the BMW Rules 2016, biomedical waste is classified into four categories, and the segregated waste should be disposed into the appropriate colour coded bins/bags. The category-wise segregation is given below:

Yellow - Human anatomical, animal, soiled expired/discarded medicines, liquid waste

Red - Recyclable plastic such as tubing, bottles, catheters, syringes, gloves

White - Waste sharps including metals

Blue – Glassware

Further, the treatment or disposal options are suggested based on the category/nature of the waste.

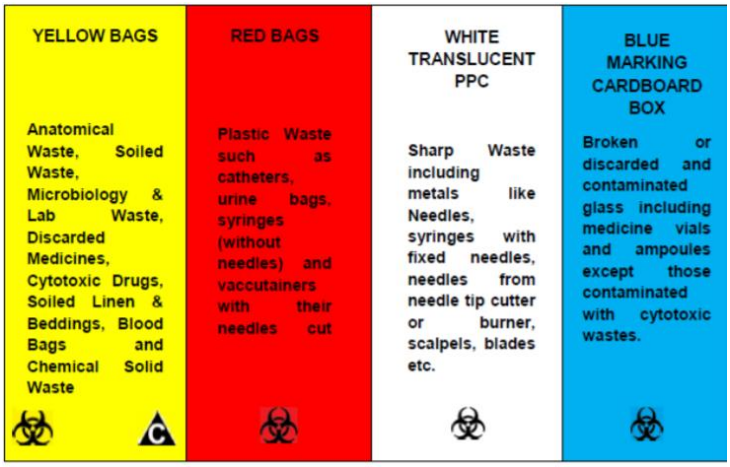


Fig 1. 1 Colour coding scheme for BMW segregation

All the BMW generated at HCUs should be labelled properly as waste type, site of generation and date of generation before transportation. All the health workers handling the waste should wear adequate personal protective equipment (PPE). Each patient care area/wards should be facilitated with the coloured containers to ensure proper disposal of waste. It is required to maintain a waste receipt book to record the quantity of waste generated or number of bags handed over to the treatment facility.

1.2 MOTIVATION

Improperly contained contaminated sharps pose great infectious risk associated with hospital waste. There is also health risk to medical waste handlers from the pathogens that may be that is associated with certain medical waste management or treatment practices. Physical injury and health hazards are also associated with the high operating temperatures of incinerators and steam sterilizers and with toxic gases vented into the atmosphere after waste treatment.

Public impacts are confined to degradation of the environment from careless disposal and the environmental impact of improperly operated incinerators or other medical waste treatment equipment. There may be increased risk of infections in patients due to poor waste management. Improper waste management can lead to change in microbial ecology and spread of antibiotic resistance.

1.3 PROBLEM DEFINITION

Even though biomedical waste management is an issue of major concern in the country, it has not received much attention and is still at a nascent stage. Stringent legislations laid by authorities have compelled few institutions of high profile to implement good practices. However, there are many HCUs which have substandard practices. This is due to the busy schedule of the HCU staff who do not find time to segregate the waste before dumping them, and also due to their lack of awareness about public safety norms and regulations.

Our project aims to automate the biomedical waste segregation process in order to reduce the time taken for segregation and training, so that the colour coded segregation scheme would be implemented in more HCUs.

1.4 SCOPE OF THE WORK

The management of health-care waste requires increased attention and diligence to avoid adverse health outcomes. "Automatic biomedical segregation bin" ensures automatic, error-free, safe, time-saving waste segregation. This system aims at eliminating human errors which may lead to disposal of waste in the wrong bin and eliminating the momentary confusions that may arise while manually categorizing the different types of waste which would in turn result in wastage of valuable time. Implementing this product in hospitals would lead to the improvement of the overall efficiency of the hospitals

CHAPTER 2

LITERATURE SURVEY

For the literature survey, a number of papers were referred. The topics of interest for this project mainly included Deep Learning, Convolutional Neural Networks, Deep Learning Platforms and various Object Detection models. The papers have shed light on the fact that training networks with a large number of layers changes an important paradigm: Features are now learned from raw data rather than being engineered by humans. This approach is successful wherever structure needs to be discovered instead of merely approximating functional dependencies. Reviewing the same has also helped in comparing the various object detection models available and to come to a conclusion as to which would most suit the purpose of the project keeping well within the resource constraints.

2.1 DEEP LEARNING

Deep learning is the machine learning method that changed the field of artificial intelligence in the last five years. In the view of industrial research, this technology is disruptive: It considerably pushes the border of tasks that can be automated, changes the way applications are developed, and is available to virtually everyone. Subjects of deep learning are artificial neural networks with a large number of layers, which motivates the name. Compared to the earlier approaches with a single layer, this allows training black-box models directly on raw data with a minimum of engineering work, called end-to-end learning [6].

Deep convolutional networks (CNNs) are characterized by a sequence of convolutional and pooling layers, followed by one or two fully connected layers. A convolutional layer is a particular type of layer for image and signal processing. It is built symbolic to the view that neurons in the visual cortex have only small receptive fields. This is reflected by connecting each neuron only to a small neighbourhood of image pixels and sharing the weights among all neurons of a particular feature map, which is equivalent to convoluting the image with a small filter kernel. It shrinks the number of weights to the filter size and makes them independent of the position in the image. The second construction element of CNNs are pooling layers, which perform subsampling of the feature maps by simple operations like maximum or mean. The importance of pooling is twofold: First, it reduces the input size for the next layer and allows learning more mappings instead. And second, it helps combining the output from the previous layer on a coarser scale [6].

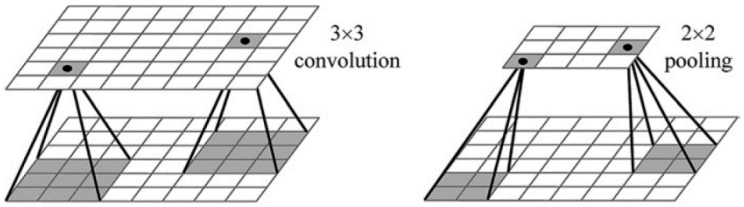


Fig 2. 1Basic CNN layers – convolution (left) and pooling (right)

The deep part of the CNN can be seen as a feature extractor: The first layer detects, e.g., edges and corners at pixel level; the next layers identify textures, object parts, and so on. By construction, the spatial relationship is propagated through all convolutional layers; the fully connected layers finally learn the associated classes and predict probabilities for the object contained in the image. All weights of this network are trained simultaneously. While hand-crafted features used to be the clue to the art of object recognition over decades, they are now learned from the raw image data.

2.2 DEEP LEARNING PLATFORMS

Besides algorithms and data, the third component that enables deep learning is the availability of fast GPU hardware and software to easily use it for neural network training. This is often regarded as the most important element.

2.2.1 Deep Learning Hardware

Evaluating a CNN consists mainly of convolutions and large matrix multiplications. Both operations are perfectly suited for GPUs with their large number of computational cores and high memory bandwidth. For gradient computation during training, the mini-batch sizes are usually adjusted so that all data fits into the GPU's memory, which easily leads to factors of 10 to 100 in speed compared to CPU implementations. The market is dominated by NVIDIA hardware together with CUDA and cuDNN libraries.

• For network training, many vendors offer hardware with multiple GPUs, either cheap consumer cards or more specific processors like NVIDIA Tesla.

• For deploying neural networks, embedded systems are available in addition to standard GPU cards. NVIDIA Jetson TX is a credit-card sized module for general applications; NVIDIA Drive PX in diverse configurations particularly serves the needs of vehicle automation.

Fig 2. 2Nvidia JetsonTX

2.2.2 Software Frameworks

During the last years, a variety of software libraries have been developed for configuring, manipulating and training neural networks. They are often accessible on a high abstraction level through script languages like Python or graphical user interfaces. This allows building deep learning applications without much detailed knowledge. The frameworks support training on multiple GPUs and deploying trained models to PCs or embedded systems, thereby enabling quick development cycles.

Framework	API	Maintainer	Started
Caffe	Python, C++, Matlab	UC Berkeley	2013
CNTK	Python, C++	Microsoft, Inc.	2016
TensorFlow	Python, C++	Google, Inc.	2015
Theano	Python	Univ. of Montréal	2010
Torch	C++, Lua	R. Collobert et al.	2002

Table 2. 1Popular open source deep learning frameworks

2.2.3 Cloud Services

Table 4 lists a number of vendors who have started to include deep learning capabilities into their cloud computing products. The offers start with services for well-defined use-cases like object detection or text translation. They are usually accessed by HTTP/S requests implementing the REST paradigm. Data (images, text) is sent to the service's URL, which returns the result in JSON or XML formats. Also training networks on own data is possible.

Vendor	Product	Main entry point
Amazon	Web Services	https://aws.amazon.com/
Google	Cloud Platform	https://cloud.google.com/
IBM	Bluemix/Watson	https://www.ibm.com/watson/
Microsoft	Azure	https://azure.microsoft.com/

Table 2. 2 Selection of commercial cloud services offering deep learning capabilities

2.3 REAL TIME OBJECT DETECTION MODELS

In R CNN, regions are formed, which are sent to convolutional neural network and this network tries to predict whether the object belongs to a particular class or not. A method known as selective search is employed in the input layer which proposes the regions by creating boxes. Once this is done, it is sent to the CNN. Here, forward propagation is done and the features are saved to the disc. Support Vector Machine (SVM) is applied on top of this. There is one SVM per class. We also have regression so that we add corrective features to our bounding box regression. We try to merge more and more pixels with the neighbouring pixels which results in the formation of blobs. As this process continues, we get well defined regions [10].

Disadvantages

1. It still takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image.

2. It cannot be implemented real time as it takes around 47 seconds for each test image.

3. The selective search algorithm is a fixed algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals

In Fast R CNN, instead of running all the proposed regions independently, we run the entire image in one go. So, the entire input image is taken and passed through the CNN in order to get feature maps [10]. These feature maps are used to create the regions of interest (RoI). After this, size of the bounding box is calculated. If the bounding box is bigger in the input image, it is reduced to the particular scale in the feature map. After this, we try to create certain warped regions or a particular fixed size input for the further layers that are fully connected. Once this is done, a SoftMax activation is calculated which tries to identify the classes to which the particular object belongs to. A linear loss is also calculated for the bounding box regression which tries to add a characteristic feature again. This will be a combined loss, called multitask loss which is the summation of soft max and linear bounding box regression loss. In the RoI

pooling layer, we try to reduce the size of the feature map to a fixed size vector, where there are different classifications which is sent to the network further to be classified as particular objects.

Disadvantage:

Considering the performance of Fast R-CNN during testing time, including region proposals slows down the algorithm significantly when compared to not using region proposals. Therefore, region proposals become bottlenecks in Fast R-CNN algorithm affecting its performance.

Faster R CNN replaces the selective search algorithm of its predecessors with a method known as Region Proposal Networks (RPN). On the feature map that is generated after the convolutional layers, a sliding window is slid, which creates k anchor boxes. These anchor boxes are the proposed boxes that are created by RPNs. The anchor boxes are of different sizes and different aspect ratios. Faster R CNN has a four-fold loss. These are two bounding box losses, object loss and classification loss. The object loss tells which class an object belongs to whereas the classification loss from the RPN tells whether the bounding box contains an object or not [4].

Disadvantage: When object detection is done from a video, there is some latency and the video is pretty choppy.

YOLO is a neural network that gets detection out of a full image in a single pass i.e., instead of looking at an image multiple times (in R CNN) you look only once. When an image is given as input, a grid is overlaid on top of the image. Each cell in the grid is responsible for predicting a few different parameters. Each cell predicts a number of bounding boxes and a confidence value for each of the bounding boxes. This is the probability of that cell containing an object. The cells that do not contain any objects predict less number of bounding boxes and so their confidence value will be low. When all these predicted bounding boxes are visualized, it basically results in a map of all the objects in the image and a bunch of boxes ranked by their confidence value. This gives information about where objects are located in an image but not about what those objects are. After this, each cell generates a class probability which looks like a coarse segmentation map of the image. The conditional probability that a cell contains a particular image is multiplied with the previously obtained confidence values to get the bounding boxes weighted by their actual probabilities for containing an object. This is followed

by thresholding the predictions and performing non-max suppression to get rid of duplicate detections. Thus, the actual images are detected with high speed and precision [5].

SSD is a single-shot detector for multiple categories that is faster than the previous state-of-the-art for single shot detectors (YOLO), and significantly more accurate, in fact as accurate as slower techniques that perform explicit region proposals and pooling (including Faster R-CNN). The core of SSD is predicting category scores and box offsets for a fixed set of default bounding boxes using small convolutional filters applied to feature maps. To achieve high detection accuracy, we produce predictions of different scales from feature maps of different scales, and explicitly separate predictions by aspect ratio. These design features lead to simple end-to-end training and high accuracy, even on low resolution input images, further improving the speed Vs. accuracy trade-off. Experiments include timing and accuracy analysis on models with varying input size evaluated on PASCAL VOC, COCO, and ILSVRC and are compared to a range of recent state-of-the-art approaches. SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. SSD with a 300 x 300 input size significantly outperforms its 448 x 448 YOLO counterpart in accuracy on VOC2007 test while also improving the speed [11].

The key difference between training SSD and training a typical detector that uses region proposals, is that ground truth information needs to be assigned to specific outputs in the fixed set of detector outputs. Some version of this is also required for training in YOLO and for the region proposal stage of Faster R-CNN and MultiBox. Once this assignment is determined, the loss function and back propagation are applied end to end. Training also involves choosing the set of default boxes and scales for detection as well as the hard-negative mining and data augmentation strategies. For the deployment of neural networks in low computational devices and object detection is done in real time, SSD gives the best results.

CHAPTER 3

METHODOLOGY

The project involves development of an automatic biomedical waste segregation bin which automates the segregation of biomedical wastes at hospitals thereby eliminating human errors that may occur during the disposal and also helps save valuable time.

3.1 BLOCK DIAGRAM

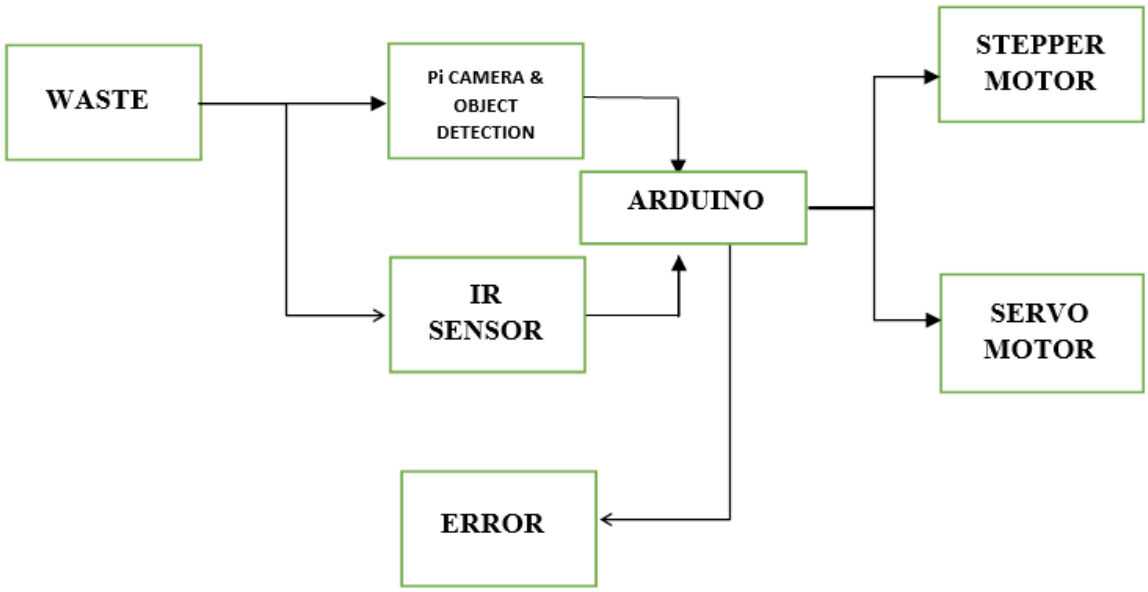


Fig 3.1 Block Diagram

Fig 3.1.1 shows the overall working principle behind the Automatic Biomedical Waste Segregation Bin. As soon as the waste is deposited into the unit through the opening provided, it falls into a chamber and onto a platform. The chamber is incorporated with an IR sensor and a pi camera module connected to the Raspberry Pi. The platform is controlled by means of a servo motor, which in turn is controlled by an Arduino. The Arduino also controls a stepper motor. Four bins, one for each category of waste, is mounted on a rotating platform which in turn is mounted on the stepper motor. The video input of waste is sent to the Raspberry Pi which detects the object that has fallen onto the platform. The IR sensor detects the presence of an object on the platform. The information about the type of object along with the output of the IR sensor is sent to the Arduino. If either the input from Raspberry Pi or IR sensor is low,

an error message is displayed. If both the inputs are high, depending on the category of waste that has been deposited, the Arduino will prompt the stepper motor to rotate such that the corresponding bin will be below the platform containing the waste. Then the servo motor will be rotated, allowing the waste to fall into the bin.

3.2 FLOWCHART

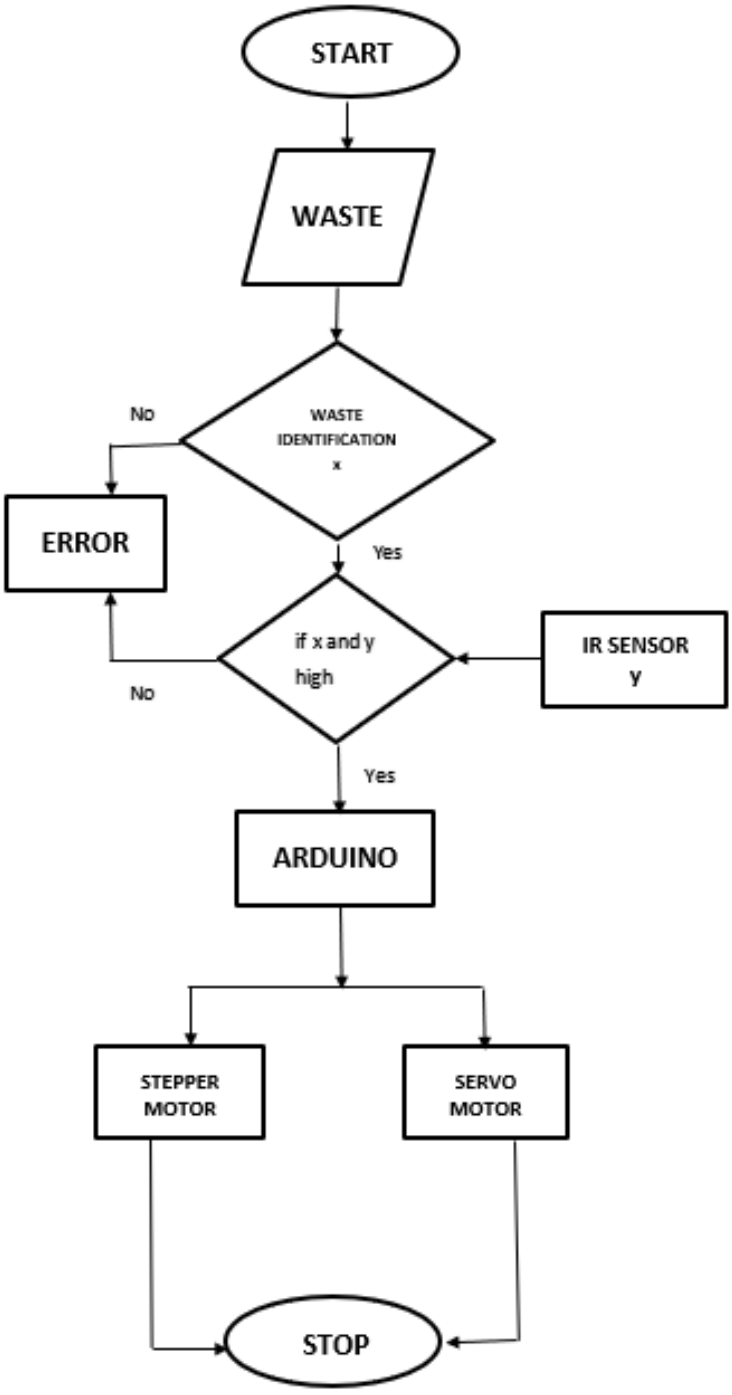


Fig 3. 1 Flowchart

3.3 DATASET CREATION

The crucial part of the project is the object detection using deep learning. Therefore, dataset creation becomes an important task. A wide variety of dataset is one of the essential conditions for training a robust detection model. In spite of all the data availability, fetching the right type of data which matches the exact use-case of the experiment is a daunting task. To overcome this issue, the dataset has to be manually generated by the team members. Moreover, the data has to have good diversity as the object of interest needs to be present in varying sizes, lighting conditions and poses if the desire is that the network generalizes well during the deployment phase. To overcome this problem of limited quantity and limited diversity of data, data augmentation is performed on the collected data. Data augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models, without actually collecting new data.

In the case of images, this means:

1. Cropping, flipping, scaling, see Fig. 3.3.1

2. Change lighting, contrast, etc.

3. Inject (overlay) objects to be found into images

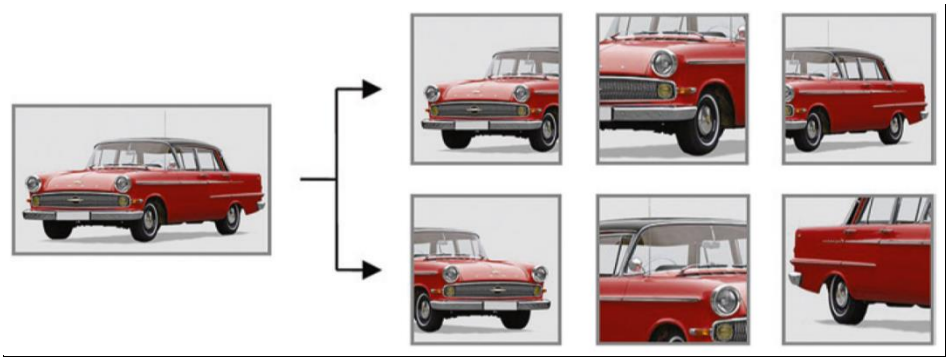


Fig 3. 2 An example of Data Augmentation

The first group of operations is often included in training frameworks for easily augmenting the data by a factor of 10 or more. The other two operations need to be carefully implemented to prevent the network from learning the differences between natural and artificial images. All these efforts have in common that they intend to get more data without manual labelling.

Training all layers of a deep network requires substantially more data than other machine learning methods. The availability of large labelled data sets is central for the advancements in deep learning. Such collections are prepared in funded projects or by research groups and made

available to the public to be used for algorithm development and assessment. The decisive aspect is that this data is labelled (or annotated), i.e., it contains class labels, object positions, translations, or whatever algorithms should predict. For visual tasks, annotations range from plain class descriptions and bounding boxes to complete semantic labelling. Accurate image labelling/annotation of the dataset is equally important to create an efficient object detection model. Among the various annotation types, bounding box will be used to label the dataset for this project. Bounding boxes are the most commonly used type of annotation in computer vision. Bounding boxes are rectangular boxes used to define the location of the target object. They can be determined by the x and y axis coordinates in the upper-left corner and the x and y axis coordinates in the lower-right corner of the rectangle. Bounding boxes are generally used in object detection and localisation tasks. Bounding boxes are usually represented by either two coordinates (x1, y1) and (x2, y2) or by one coordinate (x1, y1) and width (w) and height (h) of the bounding box.

The dataset after labelling is divided into training and testing data (usually in the ratio of 4:1).

3.4 TRANSFER LEARNING

Transfer learning is a technique particularly useful for quickly adapting a system to new requirements. The theory is very similar as for the auxiliary tasks mentioned above: When, e.g., introducing a new class in object classification, the deep layers do not change much as the features remain similar for the new task. In the simplest case, the convolutional part of a CNN is kept fixed, only the final layers are re-trained on a rather small number of images. For adding a new object, it may then be enough to have 100 images instead of 100000. This can be used for teaching an object detection system new traffic signs, for adding new people to face recognition or for adapting speech recognition to local dialects. In a more general sense, transfer learning means using a network pre-trained on a certain task as a starting point and adapting it for one's own applications. Benefit is always the reduction of data and training time.

3.5 OBJECT DETECTION AND LOCALIZATION

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Detecting objects in scenes is a step in complexity for which the system needs to determine both position and class. A straightforward way would be using a segmentation method or a sliding window to define regions of interest (ROIs) and then feed them into a CNN. However, this is inefficient at run time as the CNN

needs to be evaluated for each ROI. Instead, the ROI can simply be learned. As discussed, features are usually similar for different tasks, thus ROI and class probability can even be predicted with a common network. There are two paths to do this: ROI proposals and single-shot detectors.

The ROI proposal approach is associated with a series of algorithms called R-CNN, Fast R-CNN and Faster R-CNN. Based on a common stack of convolutional layers, two tasks are performed:

• Estimating the ROI coordinates in a region proposal network. The number of proposals is critical to the network's performance; common choices are 300 or more.

• Extracting the ROIs (= ROI pooling) and passing them to the final classifier.

Typical single-shot detectors are YOLO and SSD. Taking the feature maps from the convolutional layers, they simultaneously estimate the classes and offsets for a fixed number of bounding boxes. SSD improves this by additional convolutional layers helping to locate objects on different scales. This produces a large number of confidence values for box/class pairs, from which the final decision is taken by non-maximum suppression.

Single-shot detectors are generally faster but less precise compared to ROI proposal methods, both using the CNNs as underlying structure.

After training a model with an optimum level of accuracy and speed of detection, the model will be integrated onto Raspberry pi. The Raspberry pi will send information about the object detected to an Arduino connected to the pi module. This will be done via serial communication. Serial communication is simply a way to transfer data. The data will be sent sequentially, one bit at a time (1 byte = 8 bits), contrary to parallel communication, where many bits are sent at the same time. When using Serial with Arduino and Raspberry Pi, the UART protocol is used. UART means "Universal Asynchronous Reception and Transmission". It is an asynchronous multi-master protocol based on the Serial communication, which allows communication between the 2 boards. There are libraries that will handle all the low layers. The Arduino Uno board has one UART that can be used either with a USB cable or from the RX/TX pins. On the Raspberry Pi, many Serial devices can be connected on the USB ports. Each will have a different device name. On the Raspberry pi, a piece of code for carrying out the serial communication will be incorporated into the main python program which detects the object.

The Arduino will be programmed such that it accepts input from the Raspberry pi and rotates the stepper motor so that the appropriate bin will come under the opening. This is to be followed by the rotation of the servo motor, which controls the platform onto which the waste falls initially, to allow the waste to fall into the bin. The platform containing the bins will be mounted on the stepper motor. Therefore, the motor should have a stall torque that allows the platform to rotate even when the bins have reached maximum capacity. The motor specifications should be chosen accordingly.

CHAPTER 4

COMPONENTS AND SOFTWARES USED

4.1 LIST OF COMPONENTS AND SOFTWARES

- Anaconda Python 3.6
- OpenCV 4.2.0
- TensorFlow 1.13.0
- CREO Parametric
- Arduino Uno

Fig 4. 1 Arduino Uno

- Raspberry Pi Model 3B

Fig 4. 2 Raspberry Pi

- NEMA 17 4.2 kg/cm stepper motor

Fig 4. 3 Stepper Motor

- A4988 stepper motor driver module

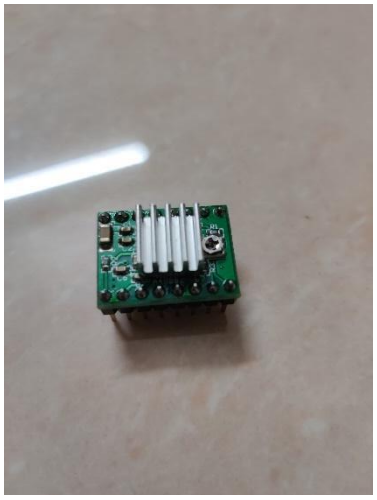


Fig 4. 4 A4988 Stepper motor driver module

- 100 uF Capacitor

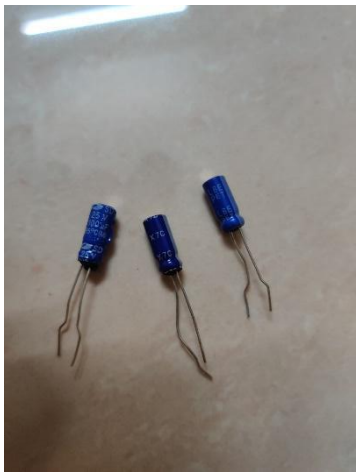


Fig 4. 5 Capacitor

- SG90 Servo Motor

Fig 4. 6 Servo Motor

- Raspberry Pi Camera Module

Fig 4. 7 Raspberry pi Camera Module

4.2 THEORY

4.2.1 Stepper motor

A stepper motor is an electric motor whose main feature is that its shaft rotates by performing steps, that is, by moving by a fixed amount of degrees. This feature is obtained thanks to the internal structure of the motor, and allows to know the exact angular position of the shaft by simply counting how many steps have been performed, with no need for a sensor.

Stepper motors have a stationary part (the stator) and a moving part (the rotor). On the stator, there are teeth on which coils are wired, while the rotor is either a permanent magnet or a variable reluctance iron core.

The basic working principle of the stepper motor is the following: By energizing one or more of the stator phases, a magnetic field is generated by the current flowing in the coil and the

rotor aligns with this field. By supplying different phases in sequence, the rotor can be rotated by a specific amount to reach the desired final position. Fig 3.4.1.1 shows a representation of the working principle. At the beginning, coil A is energized and the rotor is aligned with the magnetic field it produces. When coil B is energized, the rotor rotates clockwise by 60° to align with the new magnetic field. The same happens when coil C is energized. In the pictures, the colours of the stator teeth indicate the direction of the magnetic field generated by the stator winding.

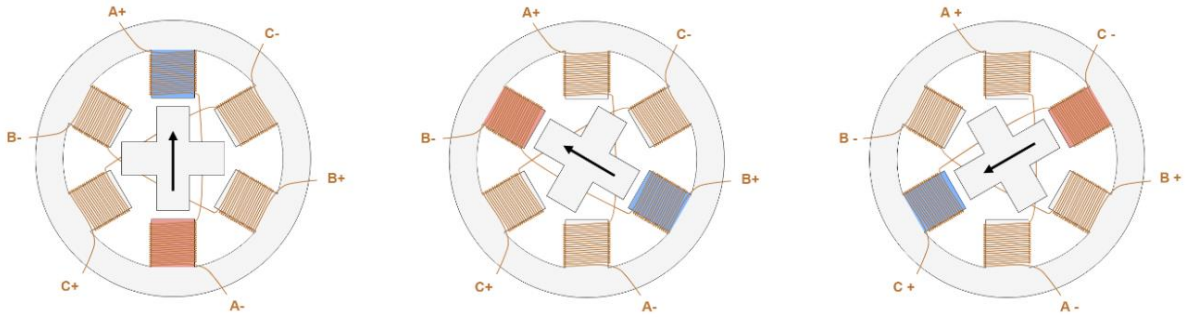


Fig 4. 8 Working principle of stepper motor

4.2.2 Servo Motor

A servo motor is an electrical device which can push or rotate an object with great precision. It is used if an object needs to be rotated at some specific angles or distance. Servo motors are rated in kg/cm (kilogram per centimetre). Most hobby servo motors are rated at 3kg/cm or 6kg/cm or 12kg/cm. This kg/cm gives the weight the servo motor can lift at a particular distance. For example: A 6kg/cm Servo motor should be able to lift 6kg if the load is suspended 1cm away from the motors shaft, the greater the distance the lesser the weight carrying capacity.

A servo consists of a Motor (DC or AC), a potentiometer, gear assembly and a controlling circuit. First of all, we use gear assembly to reduce RPM and to increase torque of the motor. Say at the initial position of the servo motor shaft, the position of the potentiometer knob is such that there is no electrical signal generated at the output port of the potentiometer. Now an electrical signal is given to another input terminal of the error detector amplifier. Now the difference between these two signals, one comes from potentiometer and another comes from other source, will be processed in feedback mechanism and output will be provided in term of error signal. This error signal acts as the input for motor and motor starts rotating. Now motor shaft is connected with potentiometer and as motor rotates so the potentiometer and it will generate a signal. So as the potentiometer's angular position changes, its output feedback signal changes. After sometime the position of potentiometer reaches at a position that the output of

potentiometer is same as external signal provided. At this condition, there will be no output signal from the amplifier to the motor input as there is no difference between external applied signal and the signal generated at potentiometer, and in this situation the motor stops rotating.

Servo motor is controlled by PWM (Pulse width Modulation) which is provided by the control wires. There is a minimum pulse, a maximum pulse and a repetition rate. Servo motor can turn 90 degree from either direction form its neutral position. The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90° position, such as if pulse is shorter than 1.5ms shaft moves to 0° and if it is longer than 1.5ms than it will turn the servo to 180°.

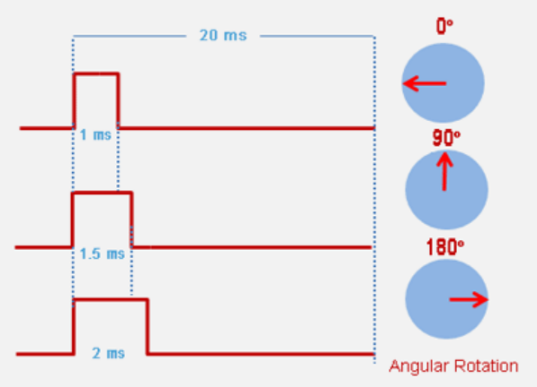


Fig 4. 9 Timing diagram of Servo motor

4.2.3 Infrared (IR) Sensor

An infrared sensor is an electronic instrument that is used to sense certain characteristics of its surroundings. It does this by either emitting or detecting infrared radiation. Infrared sensors are also capable of measuring the heat being emitted by an object and detecting motion. IR is invisible to the human eye, as its wavelength is longer than that of visible light (though it is still on the same electromagnetic spectrum). Anything that emits heat (everything that has a temperature above around 5K) gives off infrared radiation.

There are two types of infrared sensors: active and passive. Passive infrared sensors are basically infrared detectors. Passive infrared sensors do not use any infrared source and detect energy emitted by obstacles in the field of view. They are of two types: quantum and thermal.

Active infrared sensors both emit and detect infrared radiation. In our project we have used an active infrared sensor. Active infrared sensors consist of two elements: infrared source and infrared detector. Infrared sources include a LED or infrared laser diode. Infrared detectors include photodiodes or phototransistors. Active IR sensors act as proximity sensors, and they are commonly used in obstacle detection systems (such as in robots).

The principle of an IR sensor working as an Object Detection Sensor can be explained using the following figure.

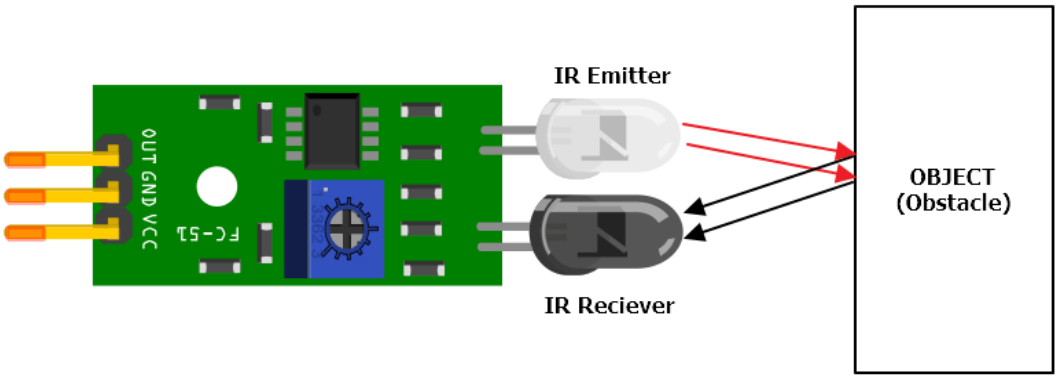


Fig 4. 10 Basic working of IR senor

When the IR transmitter emits radiation, it reaches the object and some of the radiation reflects back to the IR receiver. Based on the intensity of the reception by the IR receiver, the output of the sensor is defined.

4.2.4 Singleshot Multibox Detector (SSD)

SSD is designed for object detection in real-time. Faster R-CNN uses a region proposal network to create boundary boxes and utilizes those boxes to classify objects. While it is considered the start-of-the-art in accuracy, the whole process runs at 7 frames per second, far below what a real-time processing need. SSD speeds up the process by eliminating the need of the region proposal network. To recover the drop in accuracy, SSD applies a few improvements including multi-scale features and default boxes. These improvements allow SSD to match the Faster R-CNN's accuracy using lower resolution images, which further pushes the speed higher. According to the following comparison, it achieves the real-time processing speed and even beats the accuracy of the Faster R-CNN.

4.2.5 Arduino UNO

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. Along with ATmega328P, it consists of other components such as crystal oscillator, serial communication, voltage regulator, etc. to support the microcontroller. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts.

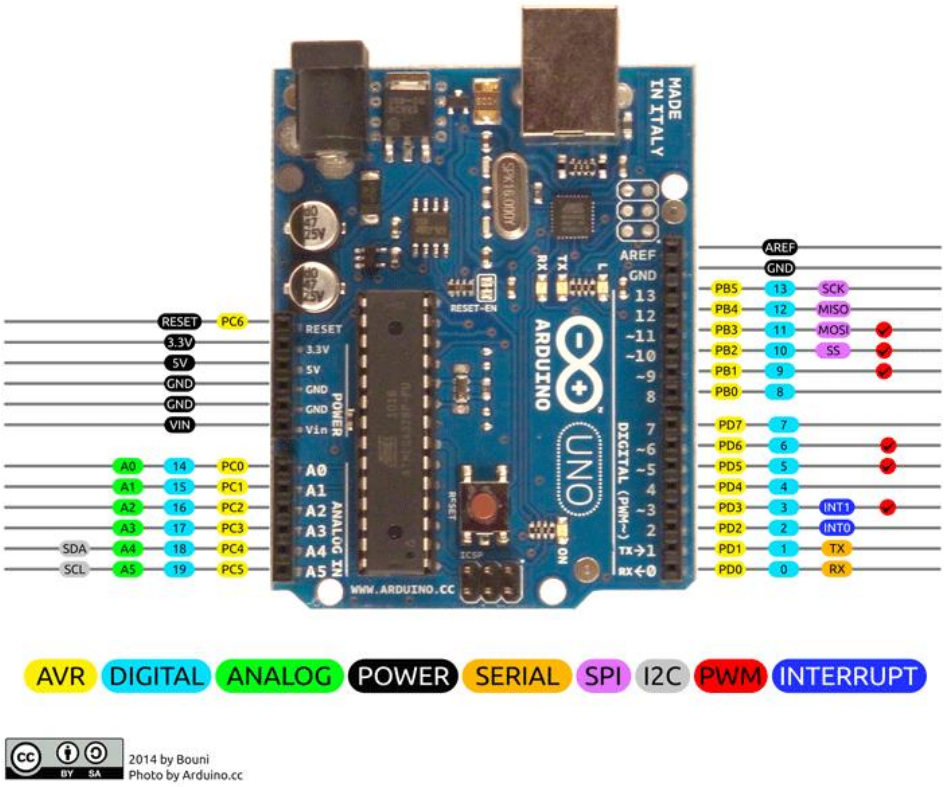


Fig 4. 11 Pin structure of Arduino Uno

(i) Pin Description

The 14 digital input/output pins can be used as input or output pins by using *pinMode(), digitalRead()* and *digitalWrite()* functions in arduino programming. Each pin operates at 5V and can provide or receive a maximum of 40mA current, and has an internal pull-up resistor of 20-50 kohms which are disconnected by default. Out of these 14 pins, some pins have specific functions as listed below:

- Serial Pins 0(Rx) and 1(Tx): Rx and Tx pins are used to receive and transmit TTL serial data. They are connected with the corresponding ATmega328P USB to TTL serial chip.
- External Interrupt Pins 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 Pins 3, 5, 6, 9 and 11: These pins provide an 8-bit PWM output by using *analogWrite()* function.
- SPI Pins 10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK): These pins are used for SPI communication.
- In-built LED Pin 13: This pin is connected with an built-in LED, when pin 13 is HIGH – LED is on and when pin 13 is LOW, its off.

Along with 14 Digital pins, there are 6 analog input pins, each of which provide 10 bits of resolution, i.e. 1024 different values. They measure from 0 to 5 volts but this limit can be increased by using AREF pin with *analogReference()* function.

- Analog pin 4 (SDA) and pin 5 (SCA): also used for TWI communication using Wire library.

Arduino Uno has a couple of other pins as explained below:

- AREF: Used to provide reference voltage for analog inputs with *analogReference()* function.

- Reset Pin: Making this pin LOW, resets the microcontroller.

Pin Category	Pin Name	Details
Power	Vin, 3.3V,5V, GND	Vin: Input voltage to Arduino when using an external power source. 5V: Regulated power supply used to power microcontroller and other components on the board.

		3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA. GND: ground pins.
Reset	Reset	Resets the microcontroller.
Analog Pins	A0-A5	Used to provide analog input in the range of 0-5V.
Input/output Pins	Digital Pins 0-13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Used to receive and transmit TTL serial data.
External Interrupts	2, 3	To trigger an interrupt.
PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13(SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuilt LED.
TWI	A4 (SDA), A5(SCA)	Used for TWI communication.
AREF	AREF	To provide reference voltage for input voltage.

Table 4. 1 Pin Description

(ii) Communication

Arduino can be used to communicate with a computer, another Arduino board or other microcontrollers. The ATmega328P microcontroller provides UART TTL (5V) serial communication which can be done using digital pin 0 (Rx) and digital pin 1 (Tx). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The ATmega16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple

textual data to be sent to and from the Arduino board. There are two RX and TX LEDs on the Arduino board which will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (not for serial communication on pins 0 and 1). A Software Serial library allows for serial communication on any of the Uno's digital pins. The ATmega328P also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus.

4.2.6 Raspberry pi Model 3B

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games. All over the world, people use Raspberry Pis to learn programming skills, build hardware projects, do home automation, and even use them in industrial applications.

The Raspberry Pi is a very cheap computer that runs Linux, but it also provides a set of GPIO (general purpose input/output) pins that allow you to control electronic components for physical computing and explore the Internet of Things (IoT).There are a few versions of the Raspberry Pi, but we have used the model B for our project. This model has improved upon its predecessor in terms of both form and functionality.

The Raspberry Pi Model B features:

• More GPIO

• More USB

• Micro SD

• Lower power consumption

• Better audio

• Neater form factor

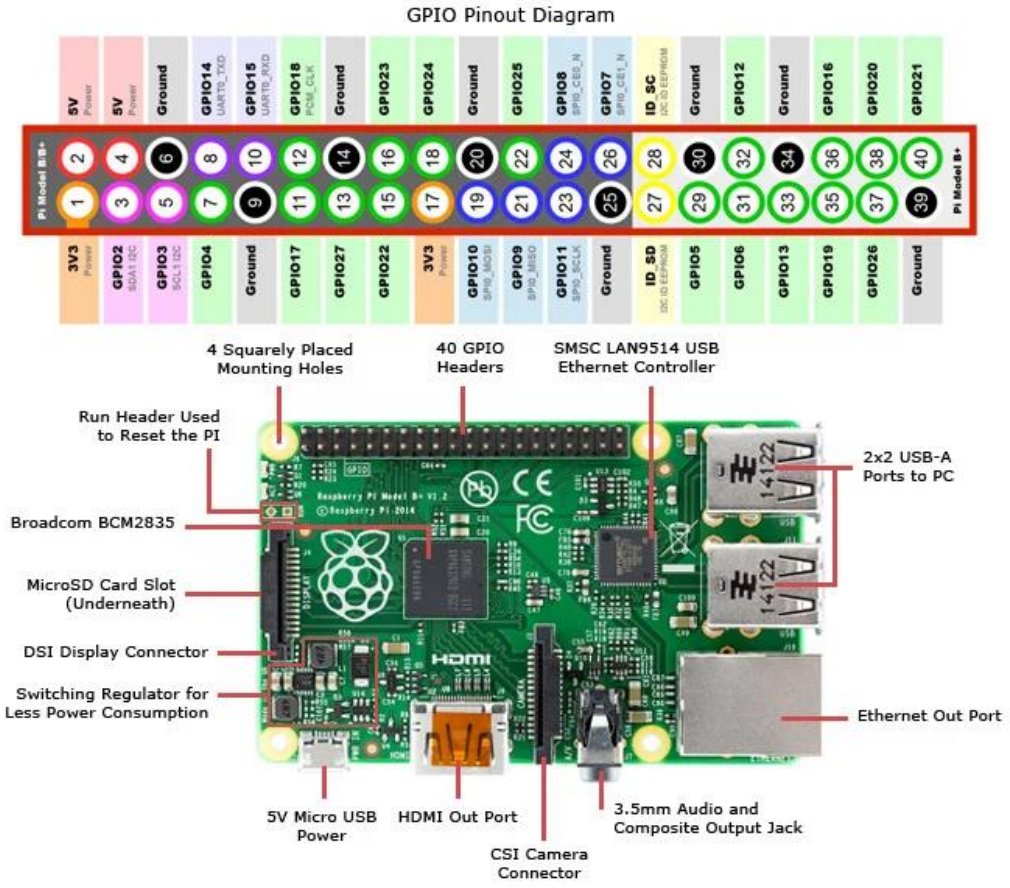


Fig 4. 12 Pin structure of Raspberry Pi Model 3B

4.2.7 OpenCV

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

4.2.8 TensorFlow 1.13.0

TensorFlow is a free and open source library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is used for machine learning applications such as neural networks. TensorFlow provides stable Python and C APIs; and without API backwards compatibility guarantee: C++, Go, Java, JavaScript and Swift.

4.2.9 Creo Parametric

Creo Parametric is a cad-cam software widely used for product design and assembly. It is different from AutoCAD in such a way that it is based on Parametric which means all the dimensions are Parametric which affect the actual part. Creo Parametric, as Pro/ENGINEER and Wildfire, is a solid modelling or CAD, CAM, CAE, and associative 3D modelling application, running on Microsoft Windows.

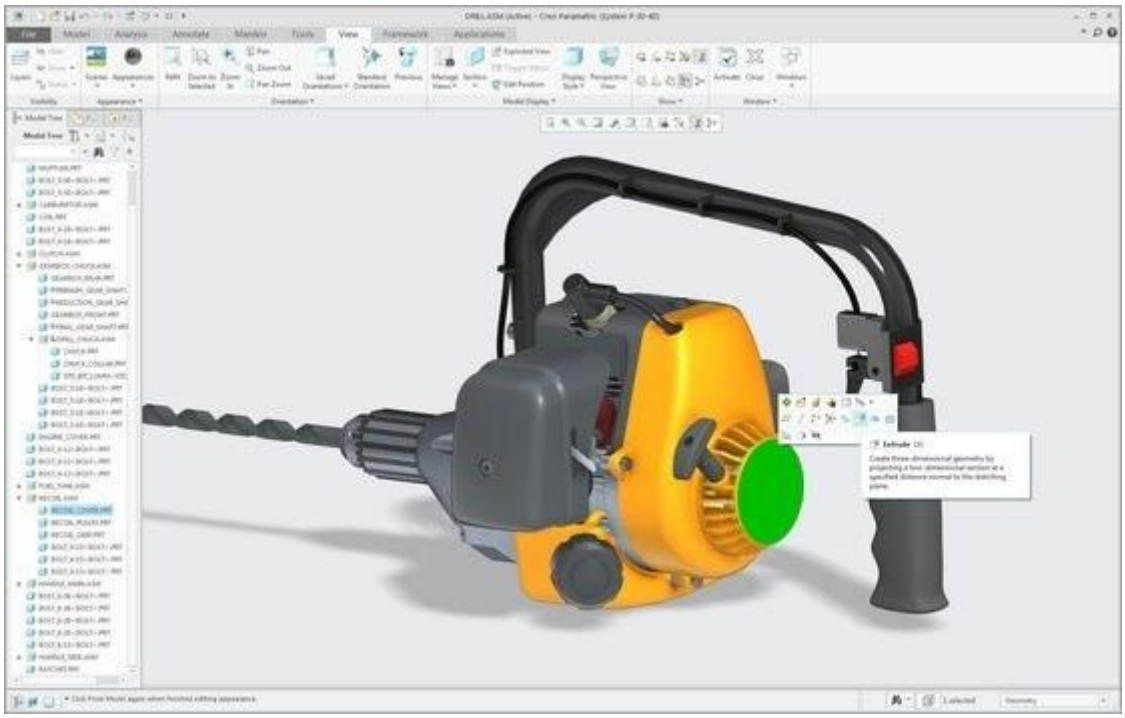


Fig 4. 13 Creo Parametric

4.2.10 Arduino IDE

The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards, but also, with the help of 3rd party cores, other vendor development boards.

The Arduino Software (IDE) contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for

cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor. Serial monitor displays serial sent from the Arduino or Genuino board over USB or serial connector.

Fig 4. 14 Arduino IDE

CHAPTER 5

IMPLEMENTATION

5.1 OBJECT DETECTION AND ARDUINO PROGRAMMING

The primary objective of creating an object detection model is to develop a program that can identify and draw boxes around specific objects in pictures, videos, or in a webcam feed. Our application requires the use of capturing a video stream in real time, and identifying the object that falls in the field of view of the webcam.

The steps involved in developing the model are explained below:

- Installation of Anaconda and setting up of virtual environment

Anaconda is a free and open-source distribution of the Python and R programming languages for data science and machine learning applications. Setting up of a virtual environment is necessary because it creates an isolated environment for Python projects. This means that each project can have its own dependencies, regardless of what dependencies every other project has. Since our model has been developed on Windows OS, we had to use more complex commands which are different from the Linux OS

- Setting up TensorFlow directory structure

The TensorFlow Object Detection API requires using the specific directory structure provided in its GitHub repository. It also requires several additional Python packages, specific additions to the PATH and PYTHONPATH variables.

The TensorFlow object detection repository was downloaded from the GitHub object detection repository located at https://github.com/tensorflow/models

- Selection of pre-trained model

The selection of a pre-trained model depends on the application for which object detection is being used. Speed and accuracy are two main parameters to be considered while choosing the model. Some models (such as the SSD-MobileNet model) have an architecture that allows for faster detection but with less accuracy, while some models (such as the Faster-RCNN model) give slower detection but with more accuracy. Since our application requires speed, we chose Singleshot Multibox Detector (SSD) as our base model. The model is required to run on a Raspberry Pi, which has comparatively less computational power than a PC. As a result, we decided to choose a quantized version of the SSD MobileNet model.

- Data creation and Labelling

TensorFlow needs hundreds of images of an object to train a good detection classifier. To train a robust classifier, the training images should have random objects in the image along with the desired objects, and should have a variety of backgrounds and lighting conditions

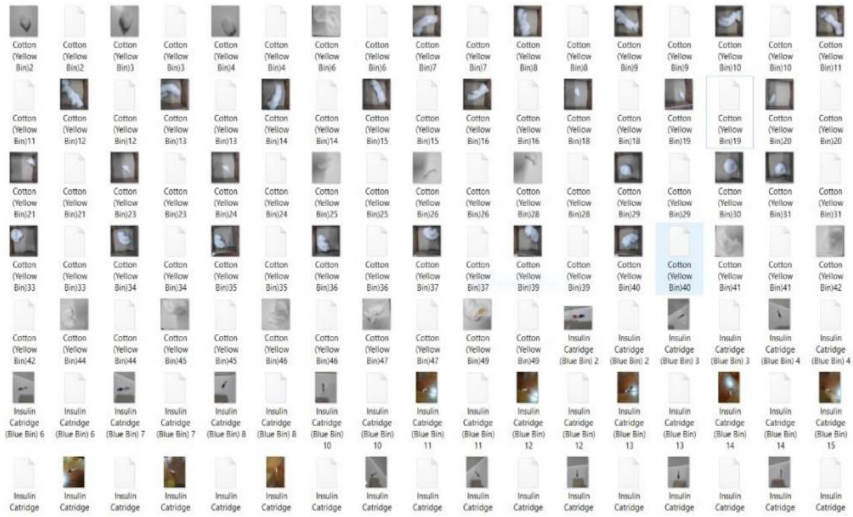


Fig 5. 1 Dataset

The four objects we selected were one object from each category of hospital waste - Cotton (Yellow), Syringe, Needle (White) and Insulin Cartridge. Around 200 images of each category were collected, with some images captured using a Raspberry Pi camera module while others on a mobile phone camera. The images should be less than 200KB each, and their resolution shouldn't be more than 720x1280. The larger the images are, the longer it will take to train the classifier.

We created the dataset required to train the neural network model and separated the dataset into training and testing images. 80% of the images were used for training and 20% of the images were for testing the trained model.

Labelling of the images was done using LabelImg(labelimg.exe). Each image is opened in this application and bound using bounding boxes and the category of the object in the image is also specified. This is done for all the training as well as testing images. The coordinates of the bounding boxes is saved in an excel sheet which is used during the training and testing phase.

- Generating training data

After labelling, an xml file is generated for each image. These files are then converted to .csv, which results in two files train_labels.csv and test_labels.csv corresponding to training and

testing data respectively. Since object detection involves using a pre-trained model, certain changes are to be made in the existing code so as to retrain it with the new training data. One such modification is to alter the labels that are currently assigned to the model. Since our model is to detect four objects, we added four new labels to the generate_tfrecords.py file. The newly assigned labels are:

1. Syringe Without Needle (Red bin)
2. Cotton (Yellow Bin)
3. Insulin Cartridge (Blue bin)
4. Needle (White Bin)

Now that the file has been modified to suit our needs, record files are generated corresponding to training and testing data: train.record and test.record

- Create Label Map and Configure Training

The last thing to do before training is to create a label map and edit the training configuration file. The label map tells the trainer what each object is by defining a mapping of class names to class ID numbers. An important point to be notes is that the ID numbers that are assigned in the label map must be equivalent to that used while generating the record files. We created this using Notepad and saved it as a .pbtxt file.

```
item {
  id: 1
  name: 'Cotton (Yellow bin)'
}

item {
  id: 2
  name: 'Syringe without Needle (Red bin)'
}

item {
  id: 3
  name: 'Needle (White Bin)'
}

item {
  id: 4
  name: 'Insulin Catridge (Blue Bin)'
}
```

Fig 5. 2 Label Map

Finally, the object detection training pipeline was configured. It defines which model and what parameters will be used for training. In order to do this, we made the following changes to the quantized SSD config file:

1. Number of classes was set to 4
2. Paths to the record files and label map were configured
3. Number of examples was set to 128 which is equal to the number of test images in the test directory.
4. Batch size was reduced to 8. Batch size refers to the number of images that are taken simultaneously in order to train the model. If the model is being trained on powerful GPUs such as NVIDIA, higher batch sizes such as 128 can be afforded. Since we trained our model on a CPU with low processing capability, we had to reduce the batch size.

- Training

Once the config file has been modified, the model was trained. The effectiveness of training is indicated by the loss value. An SSD model performs well if the loss value remains less than 1. The model was trained for around 48 hours. Initially, the loss value started at 28 which then followed an exponentially decreasing curve until the value oscillated between 1.9 and 0.7. We stopped further training of the model in order to prevent overfitting. Overfitting is an undesirable case in which the loss value starts increasing again as a result of which it detected both train and test images, but unable to detect new objects belonging to the trained classes.

Once the training is done, an inference graph is created using the checkpoints generated during the training process. The inference graph is a file with .pb extension

- Conversion to TensorFlow Lite

Since our object detection model is to run on a Raspberry Pi, there was a need to convert our model to TensorFlow Lite. This is because TensorFlow Lite gives a faster detection than TensorFlow when executed on the Raspberry Pi. The conversion was done by executing a few commands which resulted in a file: detect.tflite. This was followed by creating a label map for the TFLite model. Initially, we tried to perform the conversion on our local machine. However, this was a tedious process which involved building TensorFlow from Source using Bazel. In order to build the package, the following command was issued:

bazel build --config=opt //tensorflow/tools/pip_package:build_pip_package

This step, which supposedly takes around 70 minutes to complete, kept on running for more than 12 hours even after multiple attempts. The conversion when performed on Google Colab

notebook successfully generated the detect.tflite file. The dataset was trained on the quantized SSD Mobilenet model for around 9500 steps until the loss reached a value less than 2. The labelmap.txt file was manually created and saved as a Notepad file. Both these files were then imported into the Raspberry Pi for further programming. The generation of these files indicates that the model has been successfully converted to TensorFlow Lite.

- Integration of object detection model onto Raspberry Pi

On the Raspberry pi, we created an environment tflite1 and installed TensorFlow and OpenCV along with their libraries. The GitHub repository for integrating the model onto the raspberry pi was cloned. The folder, 'TFLite_model', containing the detect.tflite and labelmap.txt files were moved to the tflite1 folder and the following command was typed on the environment terminal to run the python code:

python3 TFLite_detection_webcam.py --modeldir=TFLite_model

This opens a terminal showing the video feed from the pi camera connected to the Raspberry pi module. The objects, when placed in front of the camera, were detected correctly. The python code also contains a block of code which sends the information about the category of waste detected to the Arduino, which is connected to the Pi via a USB cable.

- Arduino UNO coding and hardware integration

Arduino UNO is used as the microcontroller to control the working of three components: stepper motor, servo motor and IR sensor, all of which are connected to the Arduino. The IR sensor is used to detect the presence of an object in the chamber into which the waste initially falls. The Arduino has been programmed in such a way that when the inputs from the Raspberry Pi and IR sensor are high, it prompts the stepper motor and servo motor to rotate based on the input from Raspberry pi.

- The stepper motor is rotated 90 degree clockwise, 90 degree anticlockwise, 180 degree clockwise or not rotated at all based on the category of waste that has been deposited. This is done in order to rotate the platform mounted on the stepper motor such that the

bins placed on top of the platform rotate in such a way that the appropriate bin comes under the platform containing the waste.

- Servo motor is used to rotate the platform onto which the waste falls initially. After the stepper motor rotates to bring the appropriate bin under the platform, the Arduino prompts the servo motor to rotate by an angle of 90 degrees which in turn causes the platform to open up, facilitating the waste to fall into the bin underneath. After a 1 second delay it returns back to its initial position.

The step pin of the stepper motor driver is connected to pin 8 of the Arduino and the direction pin is connected to pin 7, both are set as output ports. When the direction pin is set to HIGH, the motor rotates in anticlockwise direction and if set to LOW, it rotates in clockwise direction. Pulses are given at the step pin to rotate the motor; one pulse moves the motor at an angle of 1.8 degree. So, in order to rotate at an angle of 90 degrees we have to give 50 pulses. Information about the detected object will be sent from the Raspberry Pi to the Arduino and gets saved as the new object, as per the written code, corresponding bin comes under the opening as the motor rotates. After that, if the object is detected by the IR sensor and the detected object is one of the labelled objects, the servo motor rotates and the waste falls into the bin, the opening closes after 1 second delay. The signal from the IR sensor is taken as input through pin 2 and the servo motor is controlled by using pin 9. This process continues for every case.

5.2 HARDWARE

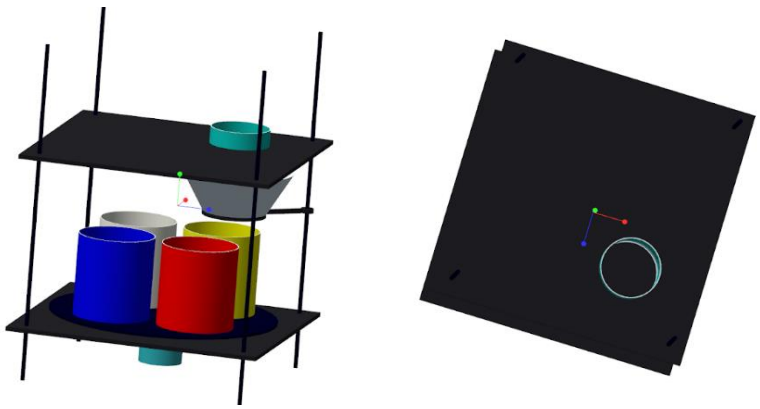


Fig 5. 3 3D Model

The 3D modelling of the hardware section of our project was done in Creo Parametric. The model has been designed such that there is a rotating base above which the four differently coloured bins are placed. The main components that are used to develop this hardware are

plywood, PVC pipe (3/4 inch and 4 inch), PVC pipe bends (90° elbow, equal tee, coupler) and recycled paint bins. The basic design has four legs made out of PVC pipe, supported with a coupler in each of them. The bends are used to connect the legs together to form the required structure. The plywood was cut into strips of four and attached to the legs at the bottom, in order to establish a firm support. The circular rotating platform is made out of a hardboard of 3mm thickness, with the bins placed on top of it. This hardware is driven using NEMA17 stepper motor and SG90 servo motor. The upper area is given a plywood covering with a circular opening, through which waste is dumped into the system platform, underneath the opening. The platform is controlled using a SG90 servo motor fixed on one of its legs. The Pi camera is also placed adjacent to the opening to capture the image of the waste that has landed on the platform. An IR sensor is also fixed just above the platform to check if the waste is dumped to the system or not.

CHAPTER 6

RESULTS AND DISCUSSION

6.1 OBJECT DETECTION MODEL

As object detection is one of the core components of the system, the dataset was trained using two different object detection models (non-quantized and quantized SSD-MobileNet) for varying number of steps so as to compare the results and select a model that would be the most efficient for our application. We selected quantized SSD-MobileNet model for our project as quantized models use 8-bit integer values instead of 32-bit floating values within the neural network, allowing them to run much more efficiently on GPUs or specialized TPUs (TensorFlow Processing Units). Four wastes- Cotton (Yellow bin), Syringe without needle (Red bin), Needle (White bin), Insulin Cartridge (Blue bin)- were chosen, one from each category. The training of the model to identify these four objects with an optimum accuracy was successfully completed. The integration of the object detection model onto Raspberry Pi was successfully completed. Arduino was successfully programmed to accept the inputs from Raspberry Pi and IR sensor and rotate the stepper motor and servo motor according to the type of waste deposited into the unit.

6.1.1 Comparing non-quantized and quantized SSD-MobileNet models

(i) With pre-trained SSD model (10000 steps)

Initially, the non-quantized SSD model was used to train around 200 images from each category and the model was trained for 10000 steps.

The fig 4.1.1.1.1 clearly depict that the model, during testing, showed varying percentages of accuracy. For some images it went as low as 40%, which was inadequate for the application. The FPS was also low.

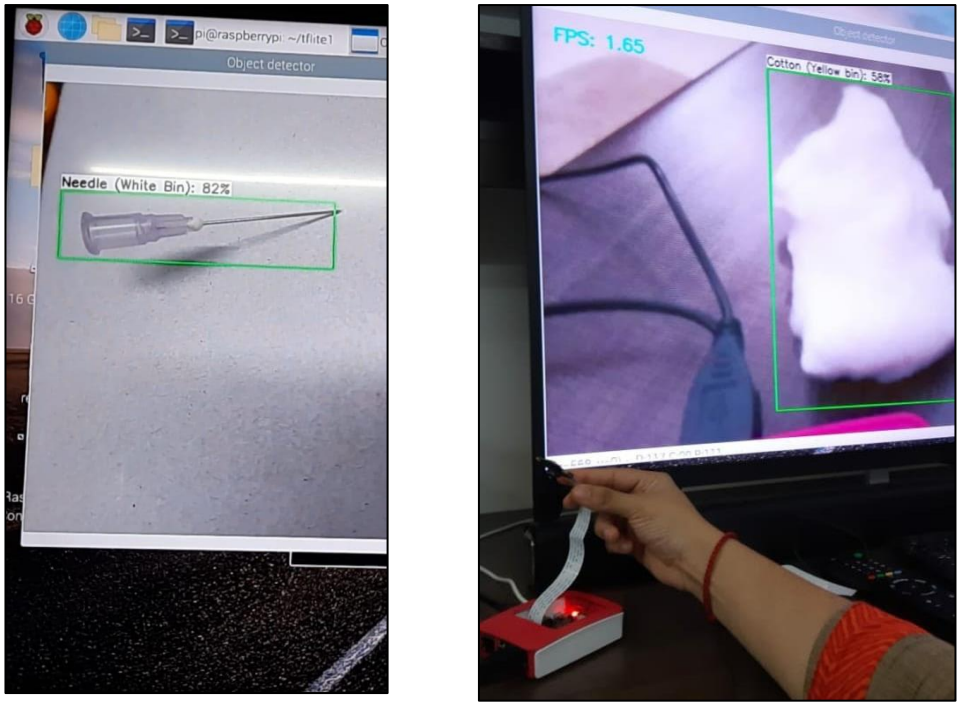


Fig 6. 1 Figure showing the accuracy of cotton (58%) and needle (82%) during detection

(ii) With pre-trained Quantized SSD (18000 steps)

To rectify the fallbacks that occurred in using the previous model, the quantized SSD model was used to train the same dataset for an increased number of steps.

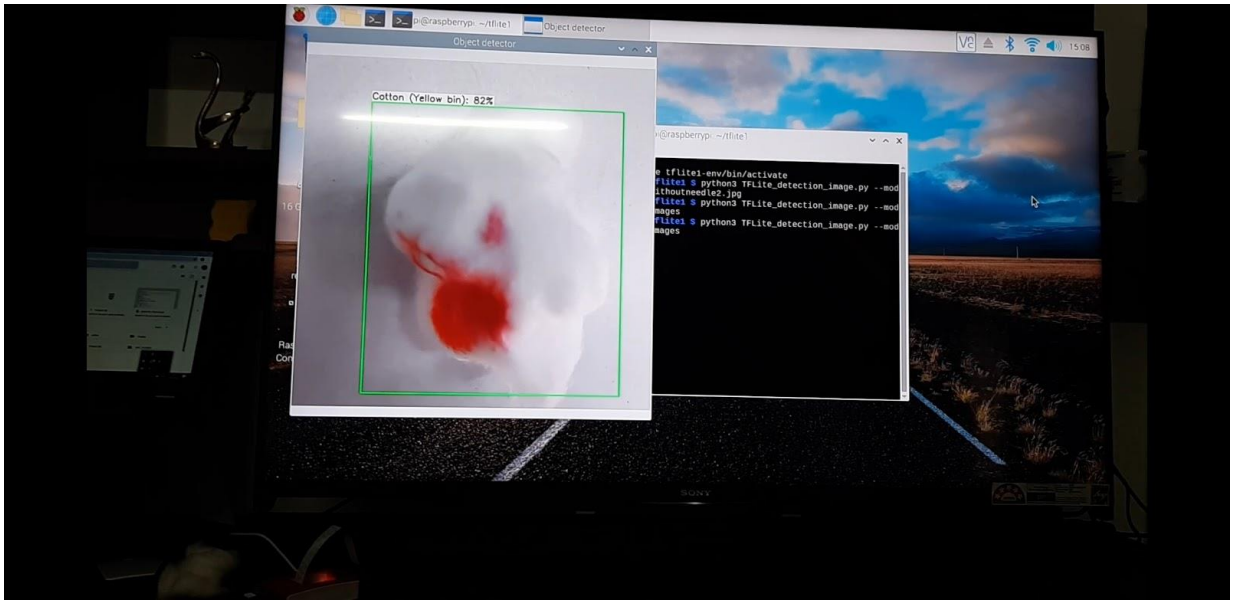


Fig 6. 2 Figure showing the accuracy of cotton (82%) during detection

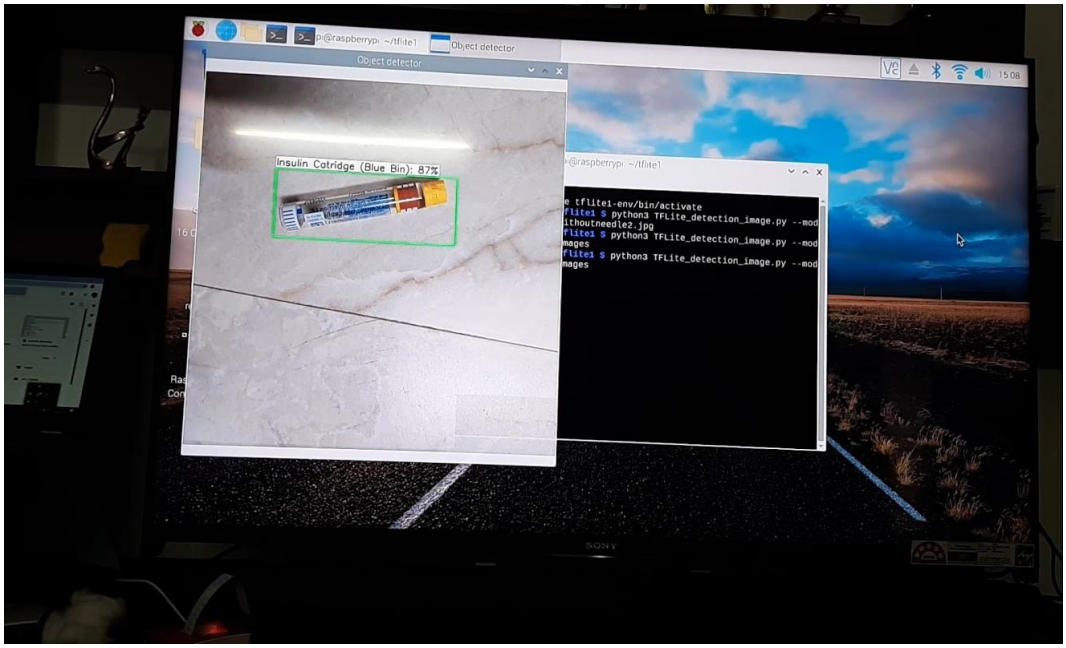


Fig 6. 3 Figure showing the accuracy of insulin cartridge (87%) during detection

During testing, the trained model showed increased speed of detection and increased percentages of accuracy which remained above an optimum for most of the test images

(iii) With Quantized SSD (25000 steps)

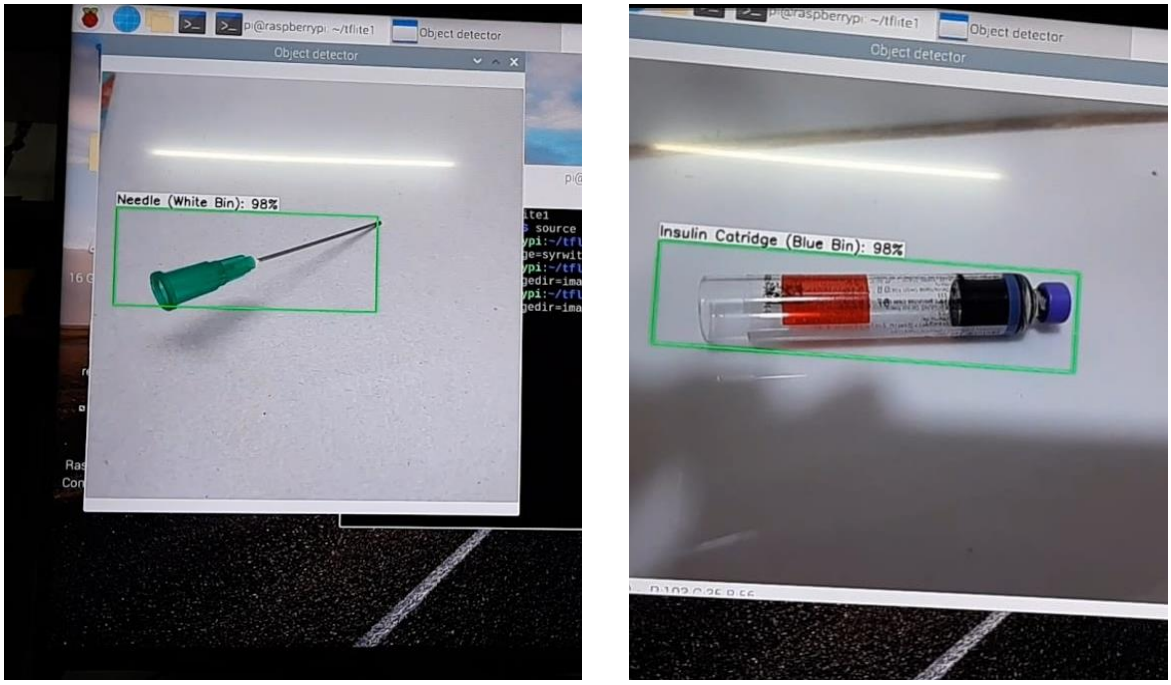


Fig 6. 4 Figure showing the accuracy of insulin cartridge (98%) and needle (98%) during detection

Training the model for an increased number of steps increased the accuracy of detection as is evident from the percentages shown in the above images whereas the speed of detection remained the same.

Therefore, it was concluded that the quantized SSD-Mobilenet model was to be chosen for our application.

6.1.2 Training of model and integration onto Raspberry pi

The training was done on Google Colab, which is a free Jupyter notebook environment that runs entirely in the cloud. Training process required 25,000 steps and when trained on a CPU it was estimated that each step would take up to 60 seconds. Thus, the need for training on GPU was necessary which was difficult due to the unavailability of NVIDIA graphics card. Thus, Google Colab was chosen for the training of the model.

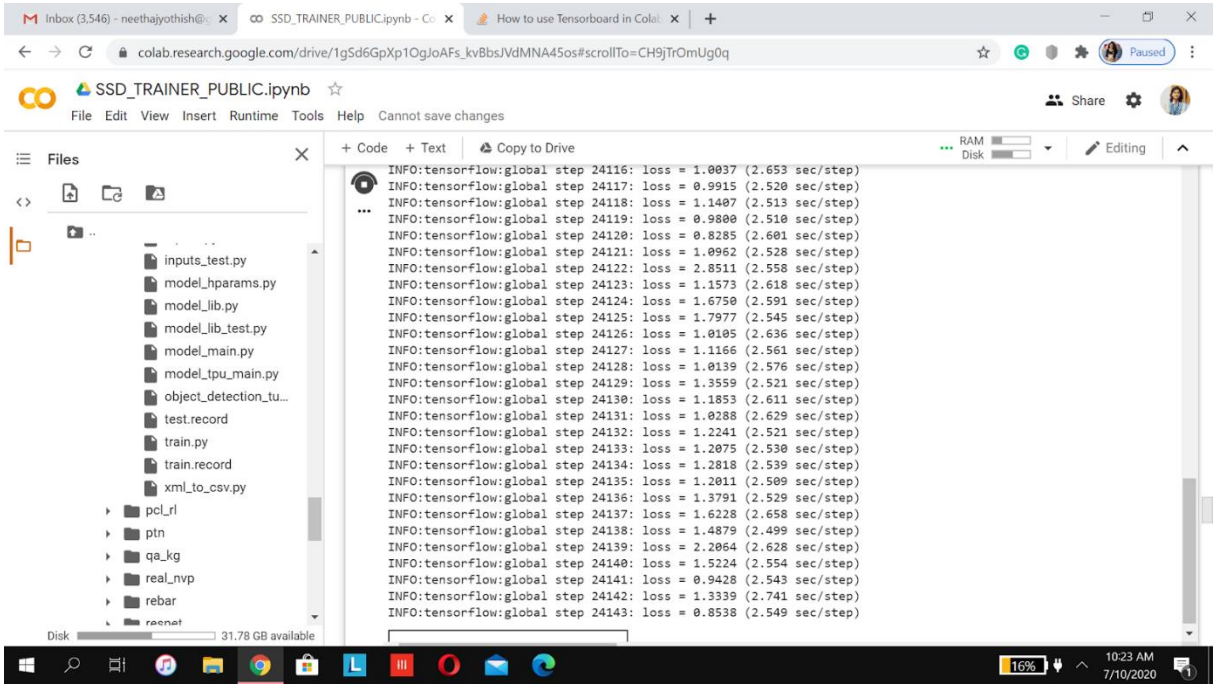


Fig 6. 5 Training the model in Google Colab

The Fig 6.6 shows a loss graph as generated by the tensorboard during training. It is a measure of the accuracy of the model. Since the loss graph is exponentially decreasing, it shows the model is being trained well and there is no overfitting.

After the training was completed successfully, the efficiency of the model was checked with test images. The model detected all the test images. When other images (which were not part of the training or testing dataset) were given to the model, it was seen that 42 out of 50 images

were detected correctly with a high percentage of accuracy thus giving the model an 84% accuracy.

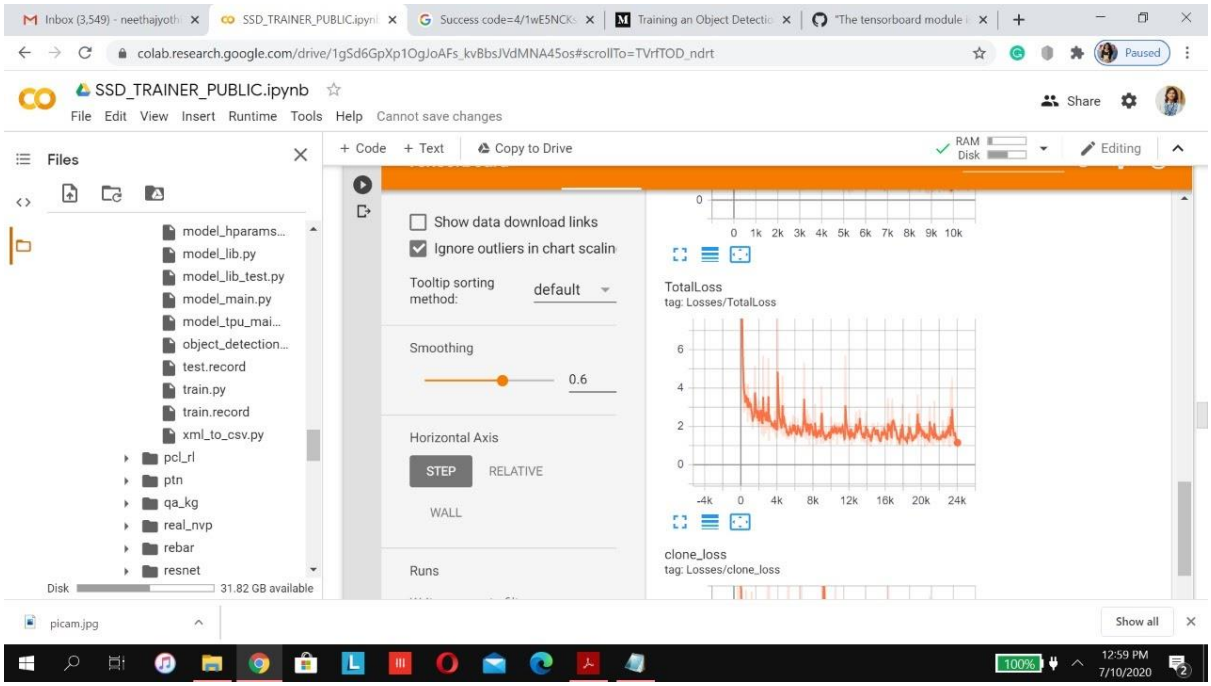


Fig 6. 6 Visualization of Total Loss Graph in TensorBoard

The model was then converted to TensorFlow Lite in order to integrate it to Raspberry pi as TensorFlow Lite models require less processing power and Raspberry pi is a device with resource constraints. The two files generated after the conversion, detect.tflite and labelmap.txt, were transferred to the Raspberry Pi. The object detection model was integrated onto the Raspberry Pi and the model was tested on live stream video from the pi camera connected to the pi as well as pre-recorded images and videos. In all the cases, the accuracy of the model remained the same. The information about the type of waste detected was also successfully sent to the Arduino

6.1.3 Arduino Programming

The Arduino on receiving inputs from the IR sensor and Raspberry pi successfully rotated the stepper motor and servo motor according to the type of waste detected in Realtime.

Fig 6. 7 Arduino IDE

Fig 6.7 shows the Arduino program after compilation and ready to be uploaded to the Arduino UNO board

Fig 6.1.3.2 show the various components connected together and working properly. The waste is being detected by the Raspberry pi and the information about the same is sent to the Arduino which rotates the stepper motor, depending on the category of waste, followed by the servo motor so that the waste falls into the bin.

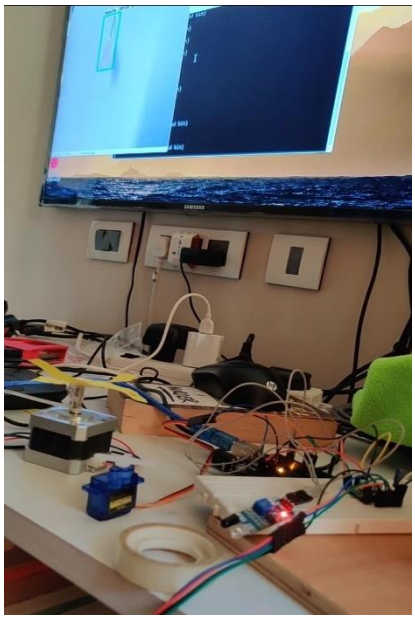


Fig 6. 8 Stepper motor and servo motor rotating in response to the output from Raspberry Pi and IR Sensor

6.2 HARDWARE

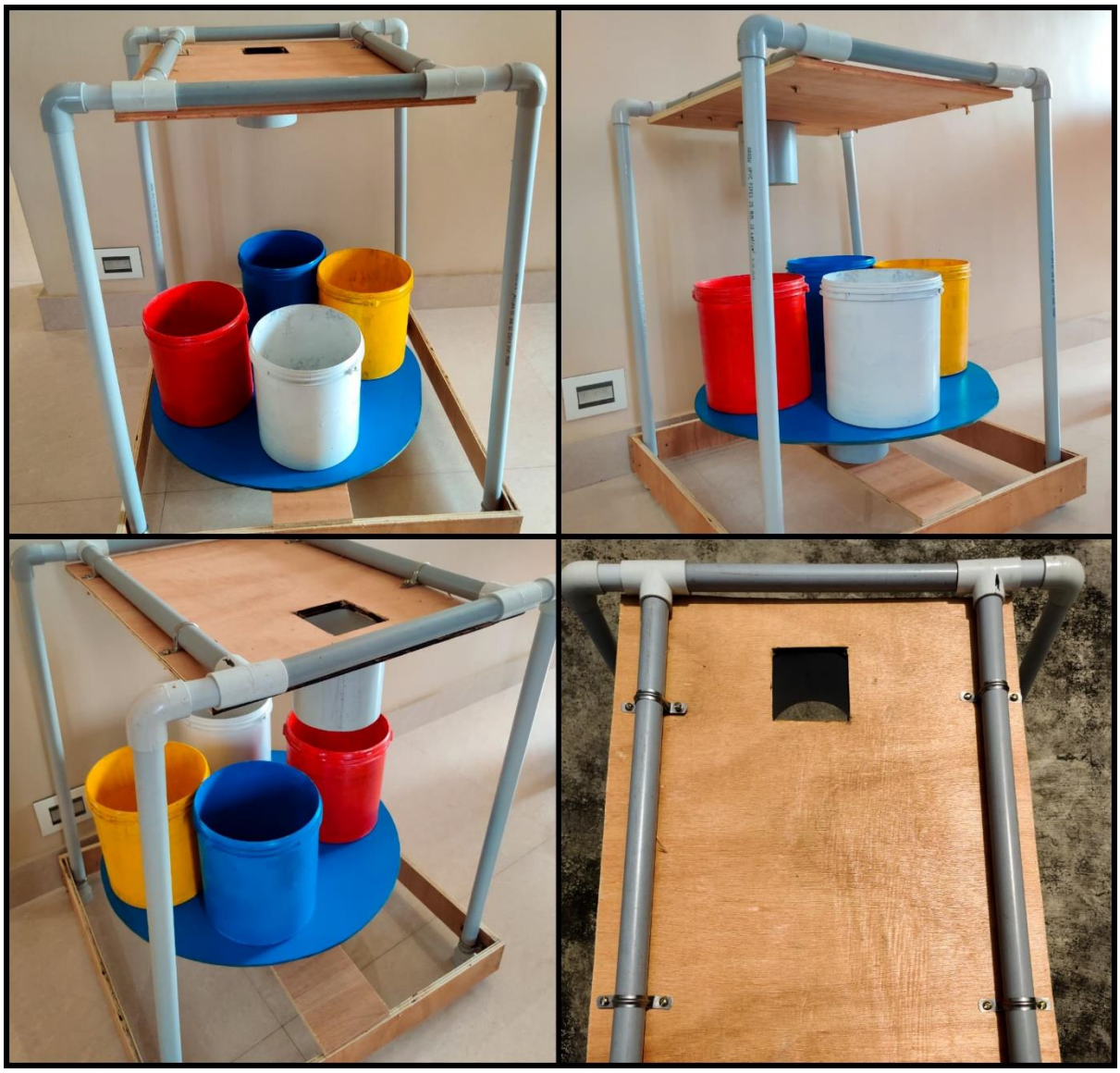


Fig 6. 9 Prototype

A prototype of the Automatic Biomedical Waste Segregation Bin was made using everyday materials.

CHAPTER 7

CONCLUSION

The currently existing system of hospital waste disposal employs colour coded bins for different types of wastes and requires hospital staff to manually categorise the waste and dispose them into the appropriate coloured bin. This method is susceptible to human errors. Accidentally disposing the waste into a wrong bin may lead to grave consequences as retrieval of the same is impractical and unhygienic. Thus, automating the system only leads to ensuring the avoidance of such human errors. The valuable time of the hospital staff which could be used to save a life is also saved in the process. We hope to alleviate these issues by replacing this system with the Automatic Biomedical Waste Segregation Bin, which is a single unit, that can be placed at different locations in the hospital as necessary. It automatically identifies the category of the waste disposed into the unit with the help of object detection techniques using Convolutional Neural Networks and disposes it into the appropriate bin. By translating our project into a working model and implementing the same in hospitals, we hope to provide a safer, error-free, time-saving waste segregation system which would improve the overall efficiency of the hospitals.

CHAPTER 8

FUTURE SCOPE

Even though the waste generated from hospitals varies in abundance, our prototype has been developed by selecting on type of waste from each of the four categories. We decided to do so, as increasing the number of detection classes would not only result in an increased training period, but it would also result in greater complexities which would not be feasible considering the time constraints. As a further expansion, this work can therefore be modified by incorporating all the possible waste types generated from hospitals.

Another addition that can be included in future is a mechanism to detect the level of waste in each bin so as to notify the people in charge, thereby avoiding the bins from overflowing.

CHAPTER 9

FUNDING RECEIVED

Our project has been selected as one among the best innovative ideas for YIP 2019-2022. Young Innovators Programme (YIP) is a specially designed programme under Kerala Development and Innovation Strategic Council (K-DISC), which aims to empower future innovators to innovate new products and existing market needs of the society more effectively through an innovative challenge.

REFERENCES

[1] **Priya**, **D. Neela**, et al. “Current Scenario of Biomedical Waste Management in India: A Case Study”, *Waste Valorisation and Recycling*. Springer, Singapore, 2019. 147-158.

[2] **Al Mamun, Md Abdulla**, et al. “Integrated sensing systems and algorithms for solid waste bin state management automation” *IEEE Sensors Journal* 15.1 (2014): 561-567.

[3] **Hegde, Veda, R. D. Kulkarni, and G. S. Ajantha**, “Biomedical waste management" *Journal of Oral and Maxillofacial Pathology* 11.1 (2007).

[4] **Ren S, He K, Girshick R, Sun J**, “Faster r-cnn: Towards real-time object detection with region proposal networks”, Advances in neural information processing systems 2015 (pp. 91-99).

[5] **Redmon, Joseph**, et al. “You only look once: Unified, real-time object detection”, *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[6] **Vogt, Michael**, “An Overview of Deep Learning and Its Applications”, *Fahrerassistenzsysteme 2018*. Springer Vieweg, Wiesbaden, 2019. 178-202.

[7] **Sudha, S**., et al. “An automatic classification method for environment: Friendly waste segregation using deep learning” *2016 IEEE Technological Innovations in ICT for Agriculture and Rural Development (TIAR)*. IEEE, 2016.

[8**] Li, Guanbin, and Yizhou Yu**, “Contrast-oriented deep neural networks for salient object detection”, *IEEE transactions on neural networks and learning systems* 29.12 (2018): 6038-6051.

[9] **Sze, Vivienne**, et al. “Efficient processing of deep neural networks: A tutorial and survey”, *Proceedings of the IEEE* 105.12 (2017): 2295-2329.

[10] **Zhao, Zhong-Qiu**, et al. “Object detection with deep learning: A review”, *IEEE transactions on neural networks and learning systems* 30.11 (2019): 3212-3232.

[11] **Liu, Wei**, et al. “Ssd: Single shot multibox detector”, *European conference on computer vision*, Springer, Cham, 2016.

APPENDIX

Arduino code to rotate the motors wrt output from Raspberry Pi

```
#include <Servo.h>
Servo myservo;
int current_object = 1;
const int stepPin=8;
const int dirPin=7;
int new_object;
void setup() {
  // put your setup code here, to run once:
  digitalWrite(13,HIGH);

  pinMode(stepPin,OUTPUT);
  pinMode(dirPin,OUTPUT);
  pinMode(13,OUTPUT);
  pinMode(2,INPUT);
  myservo.attach(9);//attachs the servo on pin 9 to servo object
  myservo.write(0);//back to 0 degrees
  delay(1000);
  Serial.begin(9600);


}

void loop() {
  // put your main code here, to run repeatedly:
  if(Serial.available()>0){
    int new_object=Serial.readStringUntil('\n').toInt();
    Serial.print(new_object);
    if (new_object==2){
     digitalWrite(dirPin,HIGH); // Enables the motor to move in a particular direction
      // Makes 200 pulses for making one full cycle rotation
      for(int x = 0; x < 50; x++) {
        digitalWrite(stepPin,HIGH);
        delayMicroseconds(500);
        digitalWrite(stepPin,LOW);
        delayMicroseconds(500);
      }
     delay(1000); // One second delay
     digitalWrite(dirPin,LOW); // Enables the motor to move in a particular direction
       // Makes 200 pulses for making one full cycle rotation
     for(int x = 0; x < 50; x++) {
       digitalWrite(stepPin,HIGH);
       delayMicroseconds(500);
       digitalWrite(stepPin,LOW);
       delayMicroseconds(500);
     }
```

```
   delay(1000); // One second delay
  }
  else if(new_object==3){
   digitalWrite(dirPin,HIGH); // Enables the motor to move in a particular direction
   // Makes 200 pulses for making one full cycle rotation
    for(int x = 0; x < 100; x++) {
      digitalWrite(stepPin,HIGH);
      delayMicroseconds(500);
      digitalWrite(stepPin,LOW);
      delayMicroseconds(500);
    }
    delay(1000); // One second delay
    digitalWrite(dirPin,LOW); // Enables the motor to move in a particular direction
    // Makes 200 pulses for making one full cycle rotation
    for(int x = 0; x < 100; x++) {
      digitalWrite(stepPin,HIGH);
      delayMicroseconds(500);
      digitalWrite(stepPin,LOW);
      delayMicroseconds(500);
   }
   delay(1000); // One second delay
 }
 else if (new_object==4){
  digitalWrite(dirPin,LOW); // Enables the motor to move in a particular direction
   // Makes 200 pulses for making one full cycle rotation
    for(int x = 0; x < 50; x++) {
      digitalWrite(stepPin,HIGH);
      delayMicroseconds(500);
      digitalWrite(stepPin,LOW);
      delayMicroseconds(500);
    }
    delay(1000); // One second delay
    digitalWrite(dirPin,HIGH); // Enables the motor to move in a particular direction
    // Makes 200 pulses for making one full cycle rotation
    for(int x = 0; x < 50; x++) {
      digitalWrite(stepPin,HIGH);
      delayMicroseconds(500);
      digitalWrite(stepPin,LOW);
      delayMicroseconds(500);
    }
    delay(1000); // One second delay
 }
 else{
   delay(1000);
 }



}
if(digitalRead(2)== LOW &&(new_object==1 or new_object==2 or new_object==3 or
```

```
new_object==4) ){
  for(int num=0;num<=100;num++)
      {
        myservo.write(num);//back to 'num' degrees(0 to 180)
        delay(10);//control servo speed
      }

  for(int num=100;num>=0;num--)
      {
        myservo.write(num);//back to 'num' degrees(180 to 0)
        delay(10);//control servo speed
      }
      delay(1000);

 }


}
```