



UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**DEVELOPMENT OF A REAL-TIME ALERT SYSTEM OF
TRAFFIC SIGN DETECTION AND RECOGNITION**

This report is submitted in accordance with the requirement of the Universiti Teknikal Malaysia Melaka (UTeM) for the Bachelor of Electronics Engineering Technology (Telecommunications) with Honours.

اونيورسي تيكنيكل مليسيا ملاك
by
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

CHOY JA YEONG

FACULTY OF ELECTRICAL AND ELECTRONIC ENGINEERING

TECHNOLOGY

2021

DECLARATION

I hereby, declared this report entitled DEVELOPMENT OF A REAL-TIME ALERT SYSTEM OF TRAFFIC SIGN DETECTION AND RECOGNITION is the results of my own research except as cited in references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.



Signature:

Author: CHOY JA YEONG

Date: 15/02/2021

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

APPROVAL

This report is submitted to the Faculty of Mechanical and Manufacturing Engineering Technology of Universiti Teknikal Malaysia Melaka (UTeM) as a partial fulfilment of the requirements for the degree of Bachelor of Mechanical Engineering Technology (Industrial Automation & Robotics) with Honours. The member of the supervisory is as follow:



The image shows the official logo of Universiti Teknikal Malaysia Melaka (UTeM) on the left, which consists of a circular emblem with a gear and a sun-like shape, surrounded by the university's name in Malay and English. To the right of the logo is a large, stylized 'UTeM' acronym. Overlaid on the 'UTeM' text is a handwritten signature in black ink. Below the signature, the word 'Signature:' is printed, followed by a dotted line. Underneath this, the name 'DR. IDA SYAFIZA BINTI MD ISA' is printed in a smaller font. At the bottom of the image, the full name 'UNIVERSITI TEKNIKAL MALAYSIA MELAKA' is printed in a light blue, semi-transparent font.

Signature:
Supervisor : DR. IDA SYAFIZA BINTI MD ISA
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

DEDICATION

This thesis is dedicated to my supervisor for guiding me along the developing of this project. I would like to thank every lecturer who has taught me the knowledges, skills and tips for my research. I would like to dedicate my parents and family member who give me moral support and encouragement during completing this report. I also would like to dedicate my friends that always possibly help me when I have trouble with this project.



ABSTRACT

Nowadays, the road accident in Malaysia is increasing expeditiously. Therefore, the implementation of the road sign recognition system is necessary to reduce the number of road accidents. In this work, a road sign recognition system has been developed to ensure road safety and driver awareness. The system has low power consumption and can be easily installed in the car for driver awareness. Previous study on the development of Advanced Driving Assistance System (ADAS) shows that TensorFlow object recognition machine learning has higher accuracy for recognizing the road signs. Raspberry Pi 3 Model B+ is chosen as the microprocessor and microcontroller in this project due to its low power consumption and convenience. Meanwhile, the Raspberry Pi Camera NoIR Module V2 is used to record the real-time video and send to the Raspberry Pi 3 Model B+ for processing and analysis using TensorFlow machine learning. The result of the analysis that includes the type of the road sign and the percentage of accuracy when compared to its original sign image in its database, will be sent to the Raspberry Pi Display Module for display purpose. Also, to alert the driver, a speaker will be installed together with the Raspberry Pi 3 Model B+ to give an alert sound.

ABSTRAK

Pada masa ini, kemalangan jalan raya di Malaysia semakin meningkat. Oleh itu, pelaksanaan sistem pengenalan papan tanda jalan raya adalah penting untuk mengurangkan jumlah kemalangan jalan raya. Dalam projek ini, sistem pengenalan papan tanda jalan raya telah dikembangkan untuk memastikan keselamatan jalan raya dan meningkatkan kesedaran pemandu. Sistem ini mempunyai penggunaan kuasa yang rendah dan boleh dipasang dengan mudah di dalam kereta. Kajian sebelumnya yang mengenai perkembangan *Advanced Driving Assistance System* (ADAS) menunjukkan bahawa perisian pengenalan objek TensorFlow mempunyai ketepatan yang lebih tinggi untuk mengenali papan tanda jalan raya. *Raspberry Pi 3 Model B +* dipilih sebagai mikropemproses dan mikrokontroler dalam projek ini kerana penggunaan kuasa yang rendah. Sementara itu, *Raspberry Pi Camera NoIR Module V2* digunakan untuk merakam video masa nyata dan menghantar ke *Raspberry Pi 3 Model B +* untuk pemprosesan dan analisis menggunakan perisian TensorFlow. Hasil analisis yang merangkumi jenis papan tanda jalan raya dan peratusan ketepatan jika dibandingkan dengan gambar tanda asalnya dalam pangkalan data, akan dihantar ke Modul Paparan Raspberry Pi untuk tujuan paparan. Selain itu, untuk memberi amaran kepada pemandu, pembesar suara akan dipasang bersama dengan *Raspberry Pi 3 Model B +* untuk memberi suara amaran.

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my deepest gratitude to all the outstanding people who have provided continuous support, guidance, experience, understanding and commitment to my successful project. I would like to express my sincere appreciation to my supervisor, Dr. Ida Syafiza Binti Md Isa for her support, guidance and motivation in completing the development of this project. I would like to thank you very much for your support and understanding. I would like to thank every lecturer who has taught me the knowledges, skills and tips for my research especially to TS. Mohd Faizal Bin Zulkifli. Moreover, I would like to express my sincere gratitude and deepest thankfulness to my parents for their support and encouragement. I would also like to thank all my friends of BEET, BEEA, BEEE and BEEC in particular for helping me and encouraging me on my project.

TABLE OF CONTENTS

	PAGE
DECLARATION	
APPROVAL	
DEDICATION	
ABSTRACT	I
ABSTRAK	II
ACKNOWLEDGEMENTS	III
TABLE OF CONTENTS	IV
LIST OF TABLES	VIII
LIST OF FIGURES	IX
LIST OF APPENDICES	XIII
LIST OF SYMBOLS	XIV
LIST OF ABBREVIATIONS	XV
CHAPTER 1 INTRODUCTION	1
1.1 Introduction	1
1.2 Background	1
1.3 Problem Statement	2
1.4 Objective	3
1.5 Project Scope	3
1.6 Expected Result	4

1.7	Thesis Organization	4
1.8	Summary	5
CHAPTER 2 LITERATURE REVIEW		6
2.1	Introduction	6
2.2	Road Sign Recognition	6
2.3	Technology	9
2.4	Algorithm	10
2.5	Convolutional Neural Networks (CNN)	11
2.6	Boundary Estimation	12
2.7	Comparison of Literature Review	14
2.8	Hardware	15
2.8.1	Microprocessor	15
2.8.2	Raspberry Pi Camera NoIR Module V2	17
2.8.3	TFT Display Module	18
2.8.4	Speaker 8R 1W	19
2.9	Expected Result	20
CHAPTER 3 METHODOLOGY		22
3.1	Introduction	22
3.2	Overview of Project	22
3.3	Methodology of Road Sign Recognition System	24
3.4	Framework of Road Recognition System Proposed Project	24
3.4.1	Block Diagram of the Proposed System	24

3.4.2	List of Hardware Components	25
3.4.3	Flowchart of Operation	27
3.4.4	Hardware Implementation	28
3.4.5	Software Implementation	30
	3.4.5.1 Python Programming Language	30
	3.4.5.2 TensorFlow Machine Learning	31
	3.4.5.3 Labeling	32
CHAPTER 4 RESULTS AND DISCUSSION		34
4.1	Introduction	34
4.2	Hardware Implementation	34
4.3	Circuit Design	36
4.4	Result of Coding	38
4.5	Findings of the Project	40
4.6	Analysis of the Training Result	45
4.7	Traffic Sign Recognition System with TensorFlow	48
4.8	Analysis of the Accuracy of the System	52
4.9	Analysis of the Reliability of the System	56
	4.9.1 Reliability of system on recognition of incomplete traffic sign	59
	4.9.2 Reliability of system based on recognition at day and night	61
	4.9.3 Reliability of system based on recognition of colour faded traffic sign	63
4.10	Project Limitation	64

CHAPTER 5	CONCLUSION AND RECOMMENDATION	65
5.1	Introduction	65
5.2	Conclusion	65
5.3	Recommendation	67
5.4	Project Potential	68
REFENRECES		69
APPENDIX		73



LIST OF TABLES

TABLE	TITLE	PAGE
Table 2.1:	Comparison of literature review on machine learning algorithm	14
Table 2.2:	Specification of Raspberry Pi 3 Model B+	16
Table 3.1:	Hardware components list	26
Table 4.1:	Accuracy of the total 20 test of images with different classes	54
Table 4.2:	Accuracy vs Time Delay of the total 20 test of images with different classes	57

LIST OF FIGURES

FIGURE	TITLE	PAGE
Figure 1.1:	Dataset sample of road signs	2
Figure 2.1:	Occlusion of the road sign problem	7
Figure 2.2:	Categories of Malaysian Traffic Signs	8
Figure 2.3:	Hardware architecture configuration in TensorFlow	9
Figure 2.4:	The classification model	10
Figure 2.5:	Estimation of boundary using 2D pose and shape-class label	13
Figure 2.6:	Raspberry Pi Camera NoIR Module V2	18
Figure 2.7:	Raspberry Pi 2.2' inch TFT Display Module	19
Figure 2.8:	Speaker 8R 1W	20
Figure 2.9:	Expected result of the recognition of road sign	20
Figure 2.10:	Expected result of the recognised road sign after zoomed in	21
Figure 3.1:	Flow chart of the project	23
Figure 3.2:	Block diagram of Road Sign Recognition System	25
Figure 3.3:	The operation process of the system	27
Figure 3.4:	Hardware connection of Raspberry Pi 3 Model B+ with Raspberry Pi Camera V2.1 and Raspberry Pi Display Module	29

Figure 3.5: Coding for installing Python	31
Figure 3.6: Coding for starting detection using TensorFlow machine learning	32
Figure 3.7: Labelling the road sign sample image	33
Figure 4.1: Front view of the project	34
Figure 4.2: Side view of the project	35
Figure 4.3: Rear view of the project	35
Figure 4.4: Prototype of schematic circuit of Road Sign Recognition system	37
Figure 4.5: Coding for training road sign images	38
Figure 4.6: The training result of 500 road sign images which the loss is above 0.5	39
Figure 4.7: Training result showed that the loss is below 0.5 after 30 minutes	39
Figure 4.8: Original view of Speed Bump traffic sign	40
Figure 4.9: Screenshot of 'Speed Bump' traffic sign is recognised from the front view	41
Figure 4.10: Original view of 'Stop' traffic sign	41
Figure 4.11: Screenshot of 'Stop' traffic sign is recognised from the side view	42
Figure 4.12: Screenshot of 'Stop' traffic sign is recognised from the far view within the distance range of 3 metre	42
Figure 4.13: Original view of 'Give Away' traffic sign	43
Figure 4.14: Screenshot of 'Give Away' traffic sign is recognised with 90 percent of accuracy	43
Figure 4.15: Original view of 'Curve to Right' traffic sign	44
Figure 4.16: Screenshot of 'Curve to Right' traffic sign is recognised	44
Figure 4.17: First training process of the dataset	45
Figure 4.18: Second training process of the dataset	45

Figure 4.19: Total loss against step in the training process	47
Figure 4.20: Learning rate against step in the training process	47
Figure 4.21: TensorFlow 2.2.0 is installed on Raspberry Pi 3	48
Figure 4.22: TensorFlow Lite version 2.5.0 is installed	49
Figure 4.23: Conversion of TensorFlow pre-trained model into TensorFlow Lite version	50
Figure 4.24: Pre-trained model run by TensorFlow normal version 2.2.0	50
Figure 4.25: Pre-trained model loaded with TensorFlow Lite version 2.5.0	51
Figure 4.26: Detection of ‘Speed Bump’ warning traffic sign with the output accuracy of 96 percent	52
Figure 4.27: ‘Stop’ warning traffic sign is recognised with the highest 96 percent of accuracy	53
Figure 4.28: ‘Give Away’ warning traffic sign is recognised with 93 percent of high accuracy	53
Figure 4.29: Graph of accuracy versus the test sample images	55
Figure 4.30: Graph of accuracy versus time delay of the test traffic sign sample	58
Figure 4.31: Missing part of incomplete ‘Stop’ warning traffic sign is recognised with 91% of accuracy	59
Figure 4.32: Missing part of the ‘Stop’ warning traffic sign is recognised with 94 percent of accuracy	60
Figure 4.33: The missing part of ‘Speed Bump’ incomplete warning traffic sign is recognised	60
Figure 4.34: Recognition of ‘Speed Bump’ traffic sign with 93 percent of accuracy at night	61

Figure 4.35: Recognition of ‘Stop’ traffic sign with 97 percent of accuracy at night	62
Figure 4.36: Recognition of ‘Speed Bump’ traffic sign from the front view at night	62
Figure 4.37: Colour fading of ‘Speed Bump’ traffic sign is recognised	63



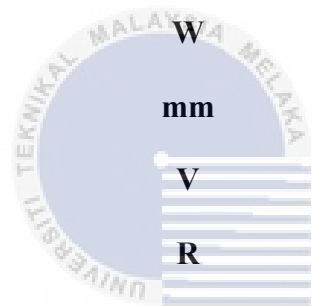
LIST OF APPENDICES

APPENDIX	TITLE	PAGE
Appendix A	Gantt Chart	73
Appendix B	Coding of training dataset for traffic sign images	74
Appendix C	Coding of test traffic sign recognition using Pi Camera	80



LIST OF SYMBOLS

s	Second
GHz	Giga Hertz
FPS	Frames Per Seconds
GB	Gigabyte
MP	Megapixel
W	Watt
mm	Millimetre
V	Volt
R	Ohm
%	Percent



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

LIST OF ABBREVIATIONS

ADAS	Advanced Driving Assistance System
NoIR	No Infrared
CNN	Convolutional Neural Networks
TFT	Thin Film Transistor
TSDR	Traffic Sign Detection and Recognition
GTSDRB	German TSDR Benchmark
STSD	Swedish Traffic Signs Dataset
UKOD	United Kingdom Online Dataset
RTSD	Russian Traffic Sign Dataset
MTSD	Malaysian Traffic Sign Dataset
DCNN	Deep Artificial Neural Networks
GPU	Graphics Processing Units
CPU	Central Processing Unit
XML	eXtensible Markup Language
HSV	Hue, Saturation and Value

CHAPTER 1

INTRODUCTION

1.1 Introduction

Nowadays, engineers are focusing on developing the Road Sign Recognition System to increase awareness among the driver for their safety purpose. Since the road accident in Malaysia is increasing expeditiously, a reliable driving assistance system is needed to lower the number of accidents. Road sign recognition system is a pivotal platform that can be implemented in the autonomous driving system. Sometimes, the driver may overlook the road sign due to tiredness and this will indirectly cause road accidents. Besides, the unpredictable or unwanted case may happen due to the blockage of the road sign by big vehicle and tree thus the driver may overlook the road sign. Hence, a road sign recognition system that able to detect different types of road signs is necessary to alert the driver.

1.2 Background

The implementation of Road Sign Recognition System is important to ensure road safety to reduce the number of road accidents. Besides, the system can also increase awareness among driver especially in highways. According to [1], it is reported that Malaysian has the third highest fatality rate of road accidents in ASEAN, with 7,512 Malaysian citizens death in 2016. It also reported that the fourth most common cause of death for Malaysian people is road accidents [1].

Therefore, the development of the road sign recognition system has been considered in the past few years to reduce the number of accidents. The first Traffic Sign Detection and Recognition (TSDR) system was developed in Japan since 1984 [2]. The TSDR system used computer vision with eight standard datasets recognition techniques included German TSDR Benchmark (GTSDRB), KUL Belgium Traffic Sign Dataset (KULD), Swedish Traffic Signs Dataset (STSD), The Netherlands RUG Traffic Signs Database (RUGD), France Stereopolis Database, United States LISA Dataset (LISAD), United Kingdom Online Dataset (UKOD) and Russian Traffic Sign Dataset (RTSD) for fast detection of the road signs. The sample of road signs was taken during daylight, night and raining condition as shown in Figure 1.1.

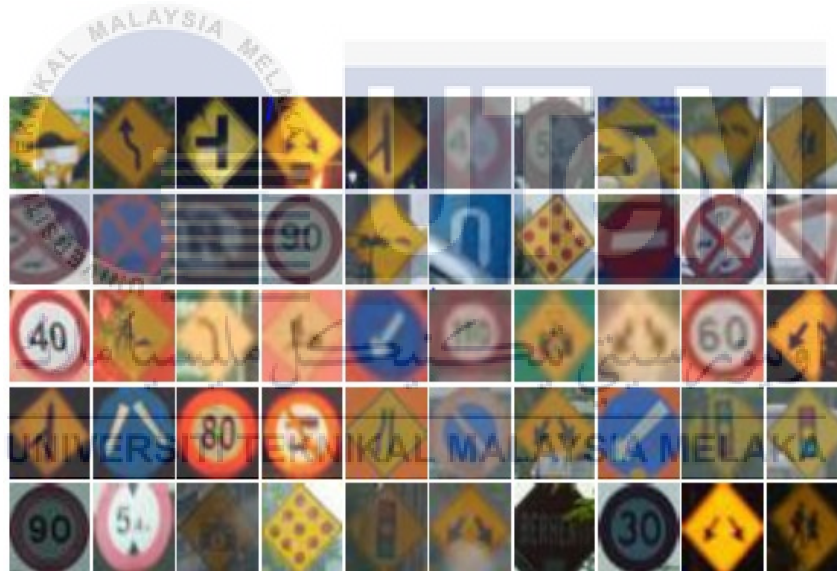


Figure 1.1: Dataset sample of road signs [2]

1.3 Problem Statement

Advanced Driving Assistance System (ADAS) is a system developed by professional engineers. ADAS aims to reduce the number of road accidents that happened. As the number of road accidents is increasing from day to day which causes the number of deaths of both car drivers and passengers increases, hence the implementation of the road

sign recognition system is necessary to overcome this issue. However, the type of machine learning algorithm used in the road sign recognition system to efficiently detect the road sign is also important. Sometimes the weather condition, and the reflection of light on the road sign reduce the quality of the captured road sign image. Therefore, an efficient algorithm is required to perform the processing and analysis of the captured image. In this work, TensorFlow machine learning is used to detect the road sign due to its short processing time.

1.4 Objective

The objectives of the project are:

1. To develop a road sign recognition system using Raspberry Pi and TensorFlow machine learning software.
2. To analyze the accuracy of the developed road sign recognition system in determining the road signs.
3. To evaluate the reliability of the developed system in real-time implementation.

1.5 Project Scope

The scope of this project is mainly focused on the accuracy of the road sign recognition system to detect and identify the road signs while driving. The effect of distance between the road sign and camera with the accuracy of the road sign recognition system is also tested. The machine learning that is implemented in this system is the TensorFlow software. TensorFlow machine learning is used to process and analyze the sample model of road signs. The processing and analysis using TensorFlow software is performed by Raspberry Pi and is connected to the camera in the car. Besides, the reliability of the system to determine how long the system can work without failure. Different weather condition

when recognizing the road sign is also tested, for example, raining, fog condition and normal condition. The reliability of the system is also depending on the algorithm, machine learning, the microprocessor and microcontroller used. Also, the system can send out a notification regarding the detected road sign through the display screen and speaker.

1.6 Expected Result

The proposed road sign recognition system using Raspberry Pi with TensorFlow machine learning is expected to give more than 90 per cent accuracy in determining the type of the road signs. This system is expected to recognise the target road sign at a distance up to 100 meters from the front of the car and send out a notification when road sign is detected.

1.7 Thesis Organization

Chapter 1:

Chapter 1 introduces and discusses the idea and overview of the project. It includes the background, objectives, problem statement, scope and expected outcome of this project.

Chapter 2:

Chapter 2 presents the summarization of the previous research from scholarly articles, journals, books and other sources on road sign recognition. This includes the relevant theories and methods that are being implemented in the current road sign detection system.

Chapter 3:

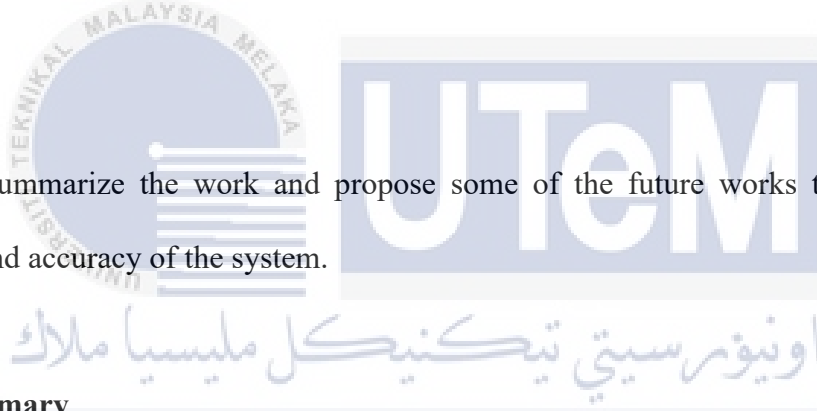
Chapter 3 describes the methodologies of the developed system. This includes the development of TensorFlow algorithm using Python Programming language to classify the presence of the road sign and the type of road signs.

Chapter 4:

Chapter 4 presents the preliminary results obtained from the experiment. The performance of the developed system in terms of reliability and accuracy to detect the road signs are evaluated in this chapter.

Chapter 5:

Chapter 5 summarize the work and propose some of the future works to increase the efficiency and accuracy of the system.



1.8 Summary

In this chapter, we proposed a road sign recognition system using the Raspberry pi and TensorFlow algorithm to detect the road sign and display a precaution notification for driver awareness. The problem statements that motivate the development of this system are highlighted in this chapter. The objectives and scope of the work are also provided.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In this chapter, a summary of the previous projects and research related to the development of the road sign recognition are presented. The type of hardware such as microcontrollers and communication modules used to develop such system are reviewed in this chapter. Also, discussion on the difference of the algorithm used to recognise the road sign model is presented.

2.2 Road Sign Recognition

Nowadays, the number of people who died in a road accident is increasing gradually. The average road accidents occurred in Malaysia also increases from year to year. Due to this, the idea to implement the automatic road sign recognition system to assist the driver besides alerting the driver if there are any road signs in that area for precaution. Based on the literature, the road sign recognition system can be divided into two stages which are detection and classification of the road signs [3]. Detection of the road sign can be done using two methods, first is colour-based detection while the second method is shape-based detection [3]. Hough transform and fast radial symmetry methods are usually implemented in the shape-based method [3]. The purpose of this project is to increase the accuracy of detecting the road signs that have occlusion problem as shown in Figure 2.1.

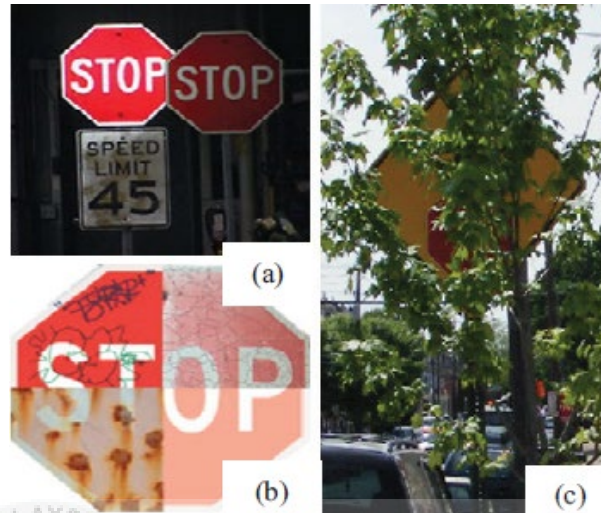


Figure 2.1: Occlusion of the road sign problem [3]

Furthermore, the algorithm used to determine the type of road sign is very important in the road sign recognition system. The algorithm will be used to perform the detection and classification on the road signs process. In [2], the authors mentioned that the dataset collection of the road sign is very important to design a road sign recognition system because the dataset will be used to train the algorithm to improve the accuracy of the system.

However, the Malaysian Traffic Sign Dataset (MTSD) is slightly different than the German TSDR Benchmark, Swedish Traffic Signs Dataset and United Kingdom Online Dataset. The Malaysian Traffic Sign Dataset are collected for different weather condition, during a rainy day and at night. Most of the Malaysian Traffic Signs are the same as the United States Traffic Sign, where the warning signs are in diamond-shaped with yellow and black colour as shown in Figure 2.2 (a) [4]. Meanwhile, the Regulatory signs are in a circle-shaped and the backgrounds are in white and red colour with black pictograms as shown in

Figure 2.2 (b). Meanwhile, the mandatory instruction signs are in a circle-shape with blue backgrounds and white pictogram except for ‘Stop and Give Way’ sign. This is because the shape of this sign for Malaysia Traffic Sign is in octagon shape with red backgrounds and white pictogram, which is also known as Priority signs as shown in Figure 2.2 (c) and Figure 2.2 (d) [4].

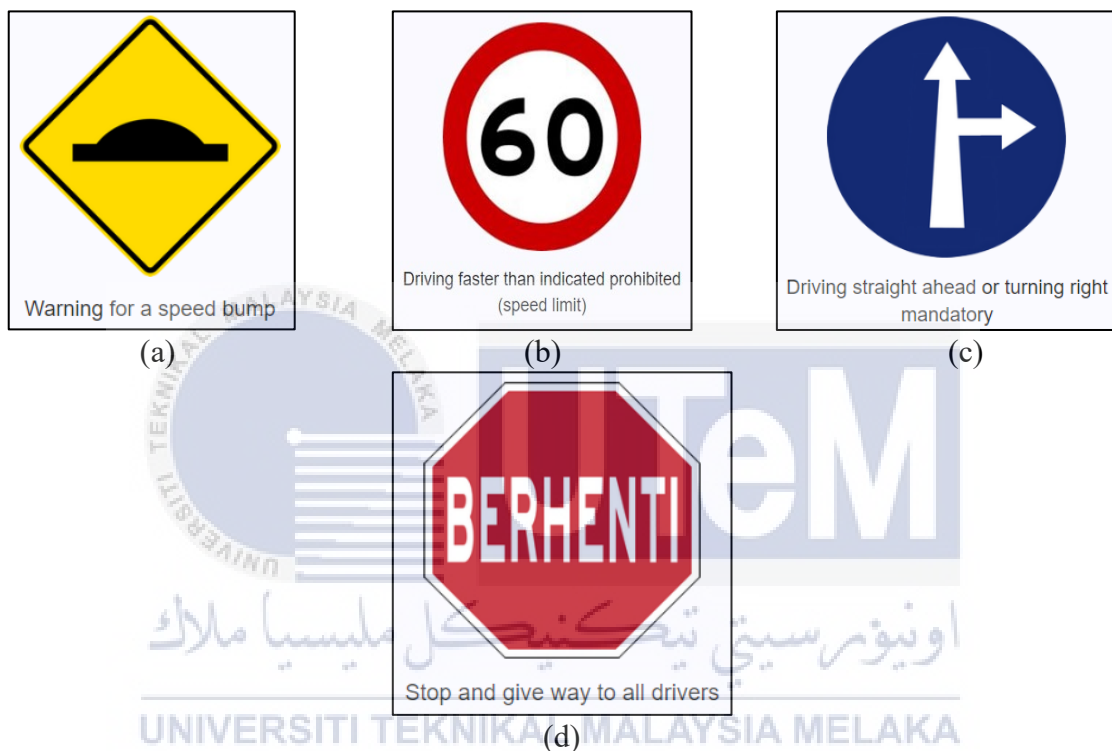


Figure 2.2: Categories of Malaysian Traffic Signs, (a) Warning signs (b) Regulatory signs, (c) Mandatory signs and (d) ‘Stop and Give Way’ [4]

Hence, it can be concluded that to develop an efficient road sign recognition system that has high accuracy, the shape’s corner and the colour of the road sign will be the main focused to detect the type of the traffic sign.

2.3 Technology

Road sign recognition system has been developed since 1980, however, the project did not achieve its objectives due to low accuracy [5]. Due to the rapid development of modern technology, the road sign recognition is developed using TensorFlow machine learning algorithm which is an open-source software library. For instance, in [5], a Deep Learning Model called Deep Artificial Neural Networks (DCNN) is developed using TensorFlow algorithm cooperated with Graphics Processing Units (GPU) to classify the road signs. Figure 2.3 shows the hardware structure of TensorFlow that is used to carry on a test for the annotated road sign image sample after segmented by slide window algorithm [5]. Besides, TensorFlow also carries on the test for the random road sign image without segmented by the algorithm. The result showed that the DCNN TensorFlow managed to identify the road sign with more than 94% of accuracy although the image samples were blurred, occlusion and have low resolution.

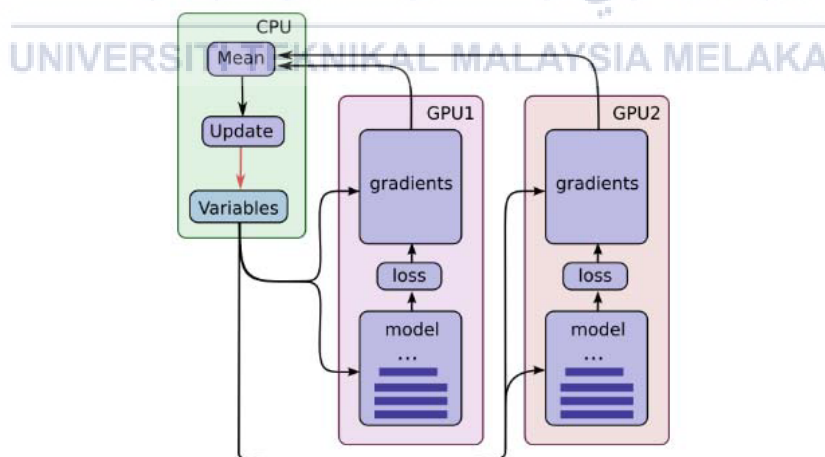


Figure 2.3: Hardware architecture configuration in TensorFlow [5]

Besides that, in [6], the author claims that the types of algorithms used inside TensorFlow such as transfer learning increases the efficiency of the road sign recognition when compared to traditional machine learning. Not that, the transfer learning is developed using information or knowledge from the surrounding environment [6]. The training samples and test samples are randomly selected which can be a large amount of data or a small amount of labelled data. Transfer learning allows training only the last layer of the whole Inception-v3 network and this can help to reduce the training time and maintain the accuracy of recognition [6]. Figure 2.4 presents the classification model of the whole system in [6].

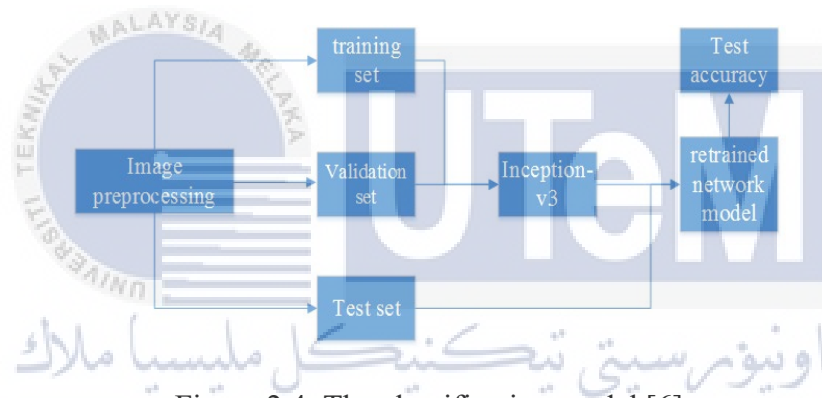


Figure 2.4: The classification model [6]

2.4 Algorithm

The road sign can be detected through its colour. The road signs can be classified using the conversion of colour space and the application of colour-based segmentation. Neural Network algorithm is used to recognise the traffic sign by performing three main processes. The first process is to perform the pre-process of the static images of road signs. The second process is to detect the sample module of road sign samples. At this process, the road signs are segmented and extracted. The third process is the processing and analysis of

the road sign sample module using the neural network algorithm to classify the type of the road sign [7].

According to [8], a new machine learning algorithm, AdaBoost has been proposed to reduce the noise of the segmented road sign images. Due to AdaBoost algorithm is using the learning-based method, thus a big dataset of the sample images is used for machine learning to accurately detect the road sign. Edge Drawing Circles algorithm is also used to detect the circle shape of the road sign. The results show that the proposed algorithm is able to identify the edges on that road sign in grayscale sample images [8].

Based on the conducted literature review above, it shows that the type of algorithm used to process and analyse the captured image of the road sign is important in developing a road sign recognition system. Moreover, a machine learning such as TensorFlow and the algorithm it used can recognise low-resolution sample images, incomplete or occlusion images and increase the accuracy of detecting the road sign. Besides, TensorFlow machine learning is suitable to be used due to its less time required to complete a full recognition process compared to the other neural networks.

2.5 Convolutional Neural Networks (CNN)

To ensure road safety and driver awareness, an application of automatic road sign recognition using machine learning likes TensorFlow, Convolutional Neural Networks (CNN) and Keras are proposed in [9] and [10]. According to [9], a CNN ensemble is used in the road sign smart detection system which focuses on the colour-based detection, shape-based detection and sign validation. For colour-based detection, the original model of Red Blue Green (RGB) image is converted into Hue, Saturation and Value (HSV) image so that it is easier to be detected by the machine learning program. This is because HSV image can

detect a various range of colour and is easier to be equalized so that the contrast of the image is adjustable.

According to [10], CNN can be used to annotate the sample images. CNN is a class of models that is easier to be trained with the sample images. Besides, data segmentation method is also used to reduce the image data overfitting by performing the process of label-preserving transformation.

The convolutional neural network is very helpful in the classification and identification of the road signs. The focus of this neural network system is to identify if the current sample image contains a road sign or not based on the specific colour or certain shape in that image. However, the speed of the recognition performed using the neural network is slow. The main challenge of CNN is the changing of the road sign's colour, lack of lights or dark condition especially detection at night. Text-to-voice module can also be applied in the system to increase awareness to the drivers as drivers do not need to take a longer time to read the alert message, especially during an emergent situation.

2.6 Boundary Estimation

For a good and efficient road sign recognition system, the accuracy of the system to recognise the actual shape of the road sign is important. With the estimation of the actual shape of the road sign, the captured image of the road sign can be classified into different categories such as 'Warning signs' category, 'Regulatory signs' category and 'Mandatory signs' category. By classifying the road sign, it will be easier to determine the actual type of road sign.

The actual location or precise boundary of the road signs are usually estimated and confirmed using a neural network algorithm. The boundary boxes of the road sign are

important in a recognition system to determine the type of the road sign. In [11], the contour estimation is used as the main method used to estimate the boundary of the road sign by formulating the road sign into a 2D pose and shape-class prediction. From the 2D pose and shape class of the road sign, the boundary of a road sign can be estimated by designing the boundary of the corresponding template road sign images and converting the images into the input image plane as shown in Figure 2.5 [11]. These images are then trained and processed by the neural network algorithm.

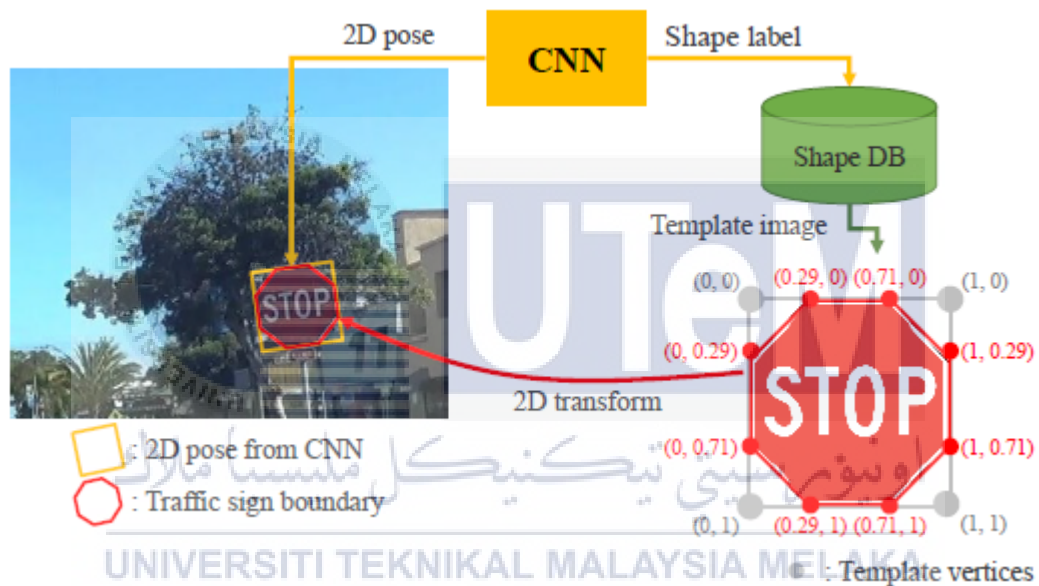


Figure 2.5: Estimation of boundary using 2D pose and shape-class label [11]

2.7 Comparison of Literature Review

Title	Machine Learning Algorithm	Author	Advantages	Disadvantages
Facial Expression Recognition Based on TensorFlow Platform	Transfer learning algorithm used by TensorFlow	X.-L. Xia, et al. (2017)	<ul style="list-style-type: none"> -Open-source library -Less training time -High accuracy of recognition 	<ul style="list-style-type: none"> -Negative transfer
Traffic sign detection and classification using colour feature and neural network	Neural Network algorithm	M. A. A. Sheikh, et al. (2016)	<ul style="list-style-type: none"> -Faster to classify the type of the road sign 	<ul style="list-style-type: none"> -Longer training time
An overview of traffic sign detection and classification methods	AdaBoost algorithm	Y. Saadna, et al. (2017)	<ul style="list-style-type: none"> -Latest algorithm -Reduce the noise of the segmented road sign images 	<ul style="list-style-type: none"> -A big dataset of the sample images is required

Table 2.1: Comparison of literature review on machine learning algorithm

2.8 Hardware

In order to develop an efficiency road sign recognition system, the selection of processors is very important. This is because, different types of processor have different processing capability, performance and power consumption. However, to increase the awareness of driver, the system should also include a good video streaming camera, large display screen and speaker to alert driver.

2.8.1 Microprocessor

The type of microprocessor used to develop a road sign recognition system is important so that it can work efficiently with the selected machine learning algorithm. Broadcom BCM2837B0 microprocessor that is integrated inside the Raspberry Pi 3 Model B+ device which is the latest product among the Raspberry Pi 3's series. This processor is designed by the Raspberry Pi Foundation with a 64-bit Quad-Core processor running at 1.4GHz. It can also connect to the Dual Band 2.4GHz and 5GHz Wi-Fi network. Broadcom BCM2837B0 processor has 50 to 60 per cent better performance compared to the Broadcom BCM2836 processor which is used in Raspberry Pi 2 [12]. Besides the booting time for the Broadcom BCM2863 is much faster compared to other processors with 36 seconds to start up [13]. The specifications of Raspberry Pi 3 B+ are shown in Table 2.1.

Features	Raspberry Pi3 Model B+
Release Date	14 th March 2018
Processor	Broadcom BCM2837B0
Central Processing Unit (CPU) Clock	1.4 GHz

Core	Cortex-A53 64-bit
Cores Number	4
Graphic Processing Unit	VideoCore IV
Random Access Memory (RAM)	1 GB LPDDR2
Memory storage	microSD card
Operating System	Latest version Raspbian
Ethernet port	Gigabit over USB 2.0 (Max 300 Mbps)
Wi-Fi	802.11 b/g/n/ac (2.4GHz and 5GHz)
HDMI	1 x full-size HDMI
Antenna	PCB Antenna
Bluetooth	4.2
USB Port	4 x USB 2.0
Dimension	85mm x 56mm
GPIO	40 pins
PoE (Power over Ethernet)	Yes, require PoE HAT

Table 2.2: Specifications of Raspberry Pi3 Model B+ [14]

On the other hand, Rockchip RK3328 microprocessor that is used in Rock64 has 64-bit quad-core A53 processor running on a 1.5GHz CPU. It came with a Mali-450 MP2 Graphic Processing Unit but unfortunately it does not have wireless board [15]. In addition, they also reported that the thermal management plate and voltage regulation of the Rockchip RK3328 is not as good as compared to Raspberry Pi 3 B+ [15]. Meanwhile, Orange Pi PC

Plus processor with 1.6GHz Quad-Core, 1 GB DDR3 RAM and Mali400MP2 GPU which has better specification than Raspberry Pi 3 B+. However, Orange Pi Pc Plus processor does not have a Wi-Fi function and it has a small community of information and support compared to the Raspberry Pi 3 Model B+. Hence, Raspberry Pi 3 Model B+ is chosen to be implemented to this recognition system as it has better performance, compact and simplicity use and commonly used in project nowadays

2.8.2 Raspberry Pi Camera NoIR Module V2

Raspberry Pi Camera NoIR Module V2 as shown in Figure 2.6 has 8MP Sony IMX 219 image sensor which is suitable to capture sample image at night. It can also capture an image with high resolution of 3280 x 2464 pixel. It is compatible with video recording for 1080p at 30 frames per second (fps) and 720p at 60 fps [16]. The other camera that is also being used in a recognition system is Logitech C310HD webcam and Go Pro Hero 3 as these devices have lower video recording specification. Logitech C310HD webcam can only records video of 720p at 30fps and its photo sensor resolution is 5MP [17]. Go Pro Hero 3 has only 5MP of photo sensor resolution [18].

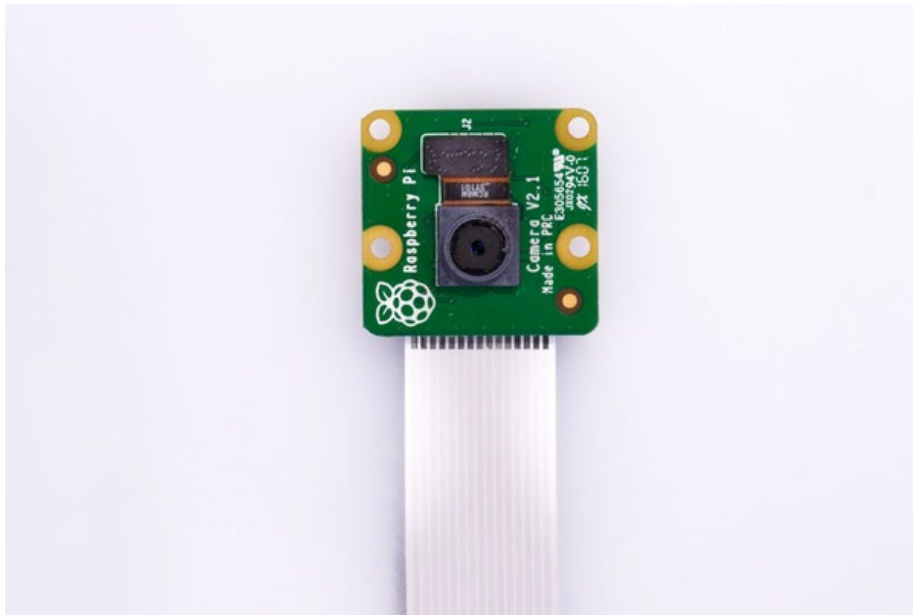


Figure 2.6: Raspberry Pi Camera NoIR Module V2 [19]



2.8.3 TFT Display Module

A display module is needed to display the output result of a specific system. However, the type and the size of the display module may vary based on the type of application system. For instance, a touchscreen display may not be necessary to display a simple notification from a system. The common screen size of a display module is around 2.2 inches to 3.5 inches displayed with 320 x 240 16-bit colour pixels. TFT display module is one of the modules that is widely used and compatible with many microcontroller boards. Figure 2.7 shows the Raspberry Pi 2.2' inch attached with a TFT Display Module that is used for display purposes [20].

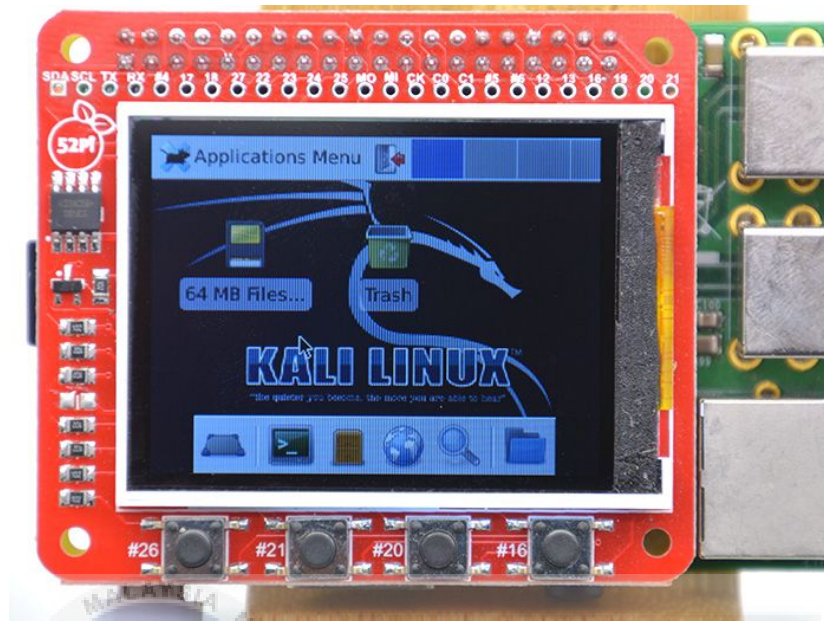


Figure 2.7: Raspberry Pi 2.2' inch TFT Display Module [21]



UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2.8.4 Speaker 8R 1W

In some application system, a speaker is needed to send out an alert notification sound. This is also necessary for a road sign recognition system where an alert notification sound can help to increase the awareness of the driver. Figure 2.8 shows the overview of Speaker 8R 1W, which can give a louder and clearer sound [21]. This type of speaker can also be connected to Raspberry Pi 3 B+ by using cable.



Figure 2.8: Speaker 8R 1W [21]

2.9 Expected Result

After the TensorFlow finished the training process from the sample of the dataset that contained multiple images of the road sign, the system should be able to recognise the road sign accurately. Figure 4.5 shows the scene of detected road sign from the real-time video recorded by Pi Camera and Figure 4.6 shows the recognised road sign after zoomed.



Figure 4.9: Expected result of the recognition of road sign [22]



Figure 4.10: Expected result of the recognised road sign after zoomed in [23]



CHAPTER 3

METHODOLOGY

3.1 Introduction

In this chapter, the methodology of this project is being focused and discussed to ensure the objective of this project are achieved. The flow of the project from the starting point until the completed project is being explained. Moreover, the implementation of hardware and software and the flow chart of the proposed system are also being discussed.

3.2 Overview of Project

In this section, the procedures that have been applied to ensure that the development of road sign recognition system projects can be completed on time and the objectives of this project are achieved are presented in a flow chart. Also, the details of every stage start from the beginning of the project until the end of the project is discussed. Figure 3.1 shows the flow chart of the project.

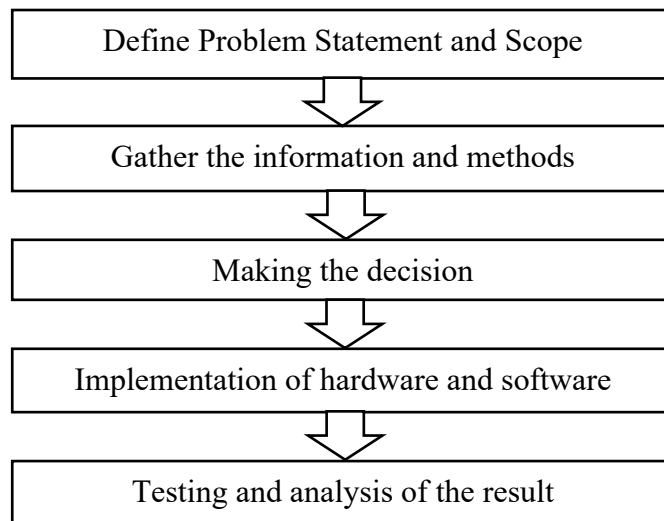


Figure 3.1: Flow chart of the project

First is defining the problem statement. This is important to help to start the progress of the project. The scope of the project can be analysed based on the problem statement. The second step is performing the literature review. Here, the research on the previous studies that are related to the project is conducted to determine the type of algorithm, method, working system and hardware component used to develop the system. The third step is to make decision and finalised the working system together with the type of algorithm and hardware used for the developed system. The information is also being analysed and summarized in Chapter 2 of the project which is Literature Review. The fourth step is the implementation of the prototype, the hardware and software. Lastly, is the testing and analysis of the results. Here the performance of the developed system will be evaluated through a testbed implementation.

The project planning is important to perfectly manage the progress of the project so that the project can be completed on time. Project management is used to make sure the flow of the project is fluent. Thus, the Gantt chart is a method that is used for project management. Gantt chart will list out all the tasks that have to completed and allocate the time estimated

for every task to be completed. Gantt chart can be referred all the time while on the process of developing the project and the next task that is needed to complete can also be checked on the track in the Gantt chart. Hence, the time needed to complete the task can be estimated as a reminder to finish the task on time. By using the Gantt chart, the planning of the budget used for the project can be done also to make sure that the money spent is not exceeding the limit. The Gantt chart of the work is shown in Appendix A.

3.3 Methodology of Road Sign Recognition System

In this section, the block diagram of the developed road sign recognition system is presented. Also, the working flow of the system and the selection of hardware and software used in the system are presented.

3.4 Framework of Road Recognition System Proposed Project

3.4.1 Block Diagram of the Proposed System

In this project, the Raspberry Pi Camera V2.1 and Raspberry Pi Display Module are connected to the Raspberry Pi 3 Model B+ as shown in Figure 3.2. The Raspberry Pi Camera V2.1 is used to record the video and send it to Raspberry Pi 3 Model B+ for processing and analysis using a machine learning algorithm. Then, Raspberry Pi 3 Model B+ sends the result of the processed image to the Raspberry Pi Display Module for display and alert user.

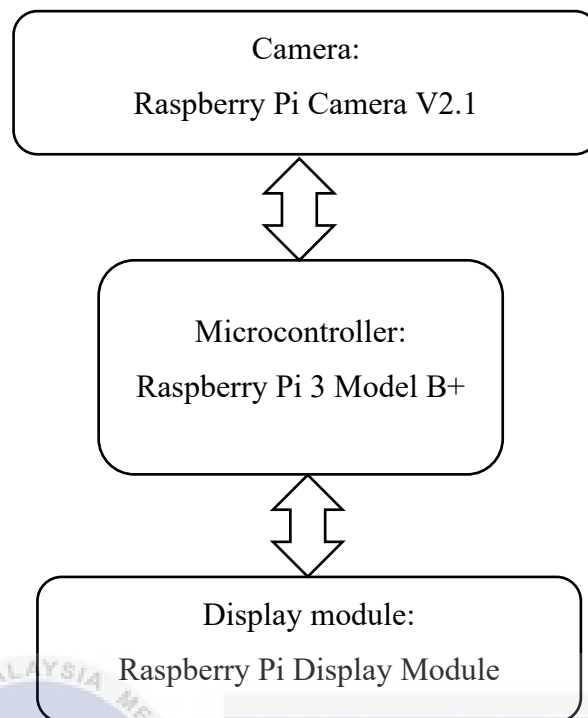


Figure 3.2: Block diagram of Road Sign Recognition System

3.4.2 List of Hardware Components

In this section, the selection of hardware includes the microcontroller, the display module and the video recorder to develop the road sign recognition system is presented as in Table 3.1. The components that are required in the road sign recognition system are Raspberry Pi 3 Model B+, Raspberry Pi Camera V2.1 and Raspberry Pi Display Module. The function of using Raspberry Pi Camera V2.1 is to record the real-time video with higher resolution. It can record the road sign within 100 meters. A Raspberry Pi 3 Model B+ is used to process, classify and recognise the road sign image retrieve from the recorded video, to perform the real-time recognition of the road sign. The Raspberry Pi Display Module is used to display the output result and the percentage of accuracy of the detected sign.

No	Name of hardware	Units	Function	Features
1	Raspberry Pi3 Model B+	1	To execute and process the computing tasks	Higher performance and consumes less power
2	Raspberry Pi Camera V2.1	1	Able to record videos while the car is driving	Manage to record video while in motion
3	Raspberry Pi Display Module	1	To display output result	Able to display high resolution object and image

Table 3.1: Hardware components list [15, 17]

3.4.3 Flowchart of Operation

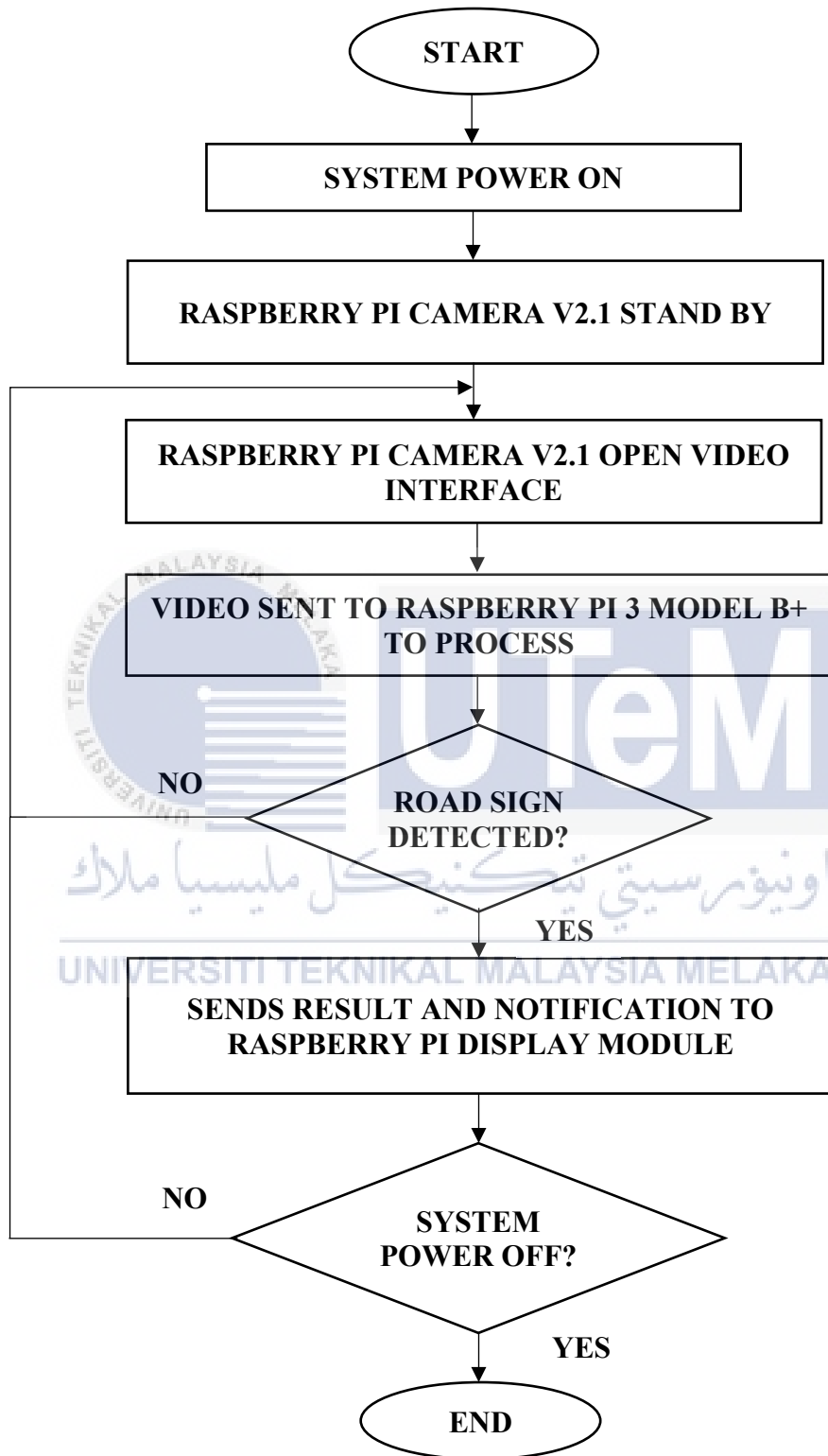


Figure 3.3: The operation process of the system

Figure 3.3. shows the flow chart related to the working principle of the developed road sign recognition system. First, the Raspberry Pi 3 Model B+ and Raspberry Pi Camera V2.1 are put in a stand mode once the system is powered on. Then, the Raspberry Pi Camera V2.1 will start records the video when the car is moving. The Raspberry Pi Camera V2.1 will share the video to the Raspberry Pi 3 Model B+ synchronously. The Raspberry Pi 3 Model B+ will process the video and recognizing the road sign. After that, the results are sent out by the Raspberry Pi 3 Model B+ to the Raspberry Pi Display Module. When there is detection of the road sign, Raspberry Pi Display Module will show out the class of that road sign is. Pi Camera will continue to record video and the processor will keep on doing processing until they detect the road sign. Then only it will do further processing to determine the type of road sign. If road sign is not detected the process of recording and processing at the processor will be continued

3.4.4 Hardware Implementation

For the interfacing of hardware components in this road sign recognition system, it is important that understand first for the specification and application of the hardware such as Raspberry Pi 3 Model B+, Raspberry Pi Camera V2.1 and Raspberry Pi Display Module that will be implemented in the proposed system. Generally, Raspberry Pi 3 Model B+ is also known as a small embedded computer or a small single-board computer, hence Raspberry Pi 3 Model B+ is a microcontroller and also a microprocessor because it has the Broadcom BCM2837B0 processor embedded with the VideoCore IV Graphic Processing Unit and 1 GB LPDDR2 RAM. The suitable power capacity needed to supply to the Raspberry Pi 3 Model B+ is 5.1V. Any DC power supply that is higher than 5.1V is not

allowed to be supplied to Raspberry Pi 3 Model B+. Raspberry Pi Camera V2.1 is connected to the Raspberry Pi 3 Model B+ through the CSI Camera Port so that the real-time video can be sent to Raspberry Pi 3 Model B+ to be processed. Besides, the HDMI video output port is used to connect the Raspberry Pi Display Module with the main board of Raspberry Pi 3 Model B+. In this work, the output result, the type of road sign and the percentage of accuracy of the detected road sign will be displayed on the screen. Figure 3.4 shows the connections of the related devices of the road sign recognition system on the Raspberry Pi board.

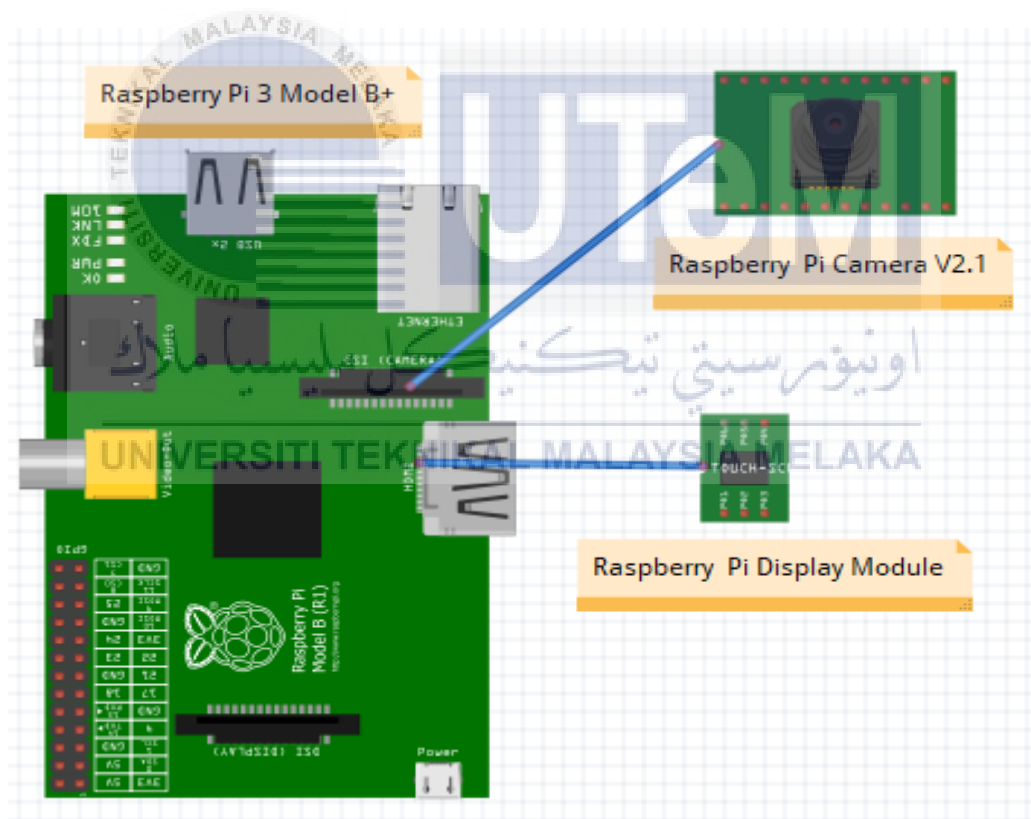


Figure 3.4: Hardware connection of Raspberry Pi 3 Model B+ with Raspberry Pi Camera V2.1 and Raspberry Pi Display Module

3.4.5 Software Implementation

In the road sign recognition system, the software implemented is Raspbian software which is the operating system to access and run the Raspberry Pi 3 Model B+. Moreover, TensorFlow is an artificial intelligence machine learning which is founded by Google. TensorFlow software allows the Raspberry Pi 3 Model B+ to carry out classification processes and recognise the road sign that is recorded by Raspberry Pi Camera V2.1 [22]. In this work, two hundred of road sign sample images will be used as a dataset. The dataset is being arranged and organized into a variety of different directories according to the type of road signs for example speed limit signs, warning signs such as caution, pedestrian crossing, obstacles ahead and the regulatory signs including the stop, give away, no parking, etc. These sample images are used to train the road sign model in the TensorFlow algorithm. It is known that as the number of trained road sign images increases, the higher the accuracy on recognition of road signs. A small amount of sample images is chosen and used as the testing sample set to test the TensorFlow algorithm. This will reduce the time taken for Raspberry Pi 3 Model B+ to recognise the real-time road sign with high accuracy.

3.4.5.1 Python Programming Language

Python coding is a type of advanced C++ programming language. It is commonly used for artificial intelligence robots, controlling machines, IoT devices and others worldwide. Python programming language is easier to learn and implemented in many professional software developers such as TensorFlow platform, Keras platform and OpenCV software. Python coding shows the good management of auto memory. The coding also provides different programming paradigms, for example, well-structured programming which has a huge reliable library source. Python programming is also being widely used for

many software and different platform [23]. In this project, Python coding is being used to write the commands in Raspberry Pi 3 Model B+ which is the Linux OS. Figure 3.5 shows the coding used to install Python programming language in Raspbian software.

```
lets=4.3.2 wwidth=0.1.7 widgetshextension=3.2.1
pi@raspberrypi:~/tf $ sudo apt-get install python-tk
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-tk is already the newest version (2.7.13-1).
The following packages were automatically installed and are no longer required:
  lxkeymap python-cairo python-gobject python-gobject-2 python-gtk2
  python-xklavier
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@raspberrypi:~/tf $
```

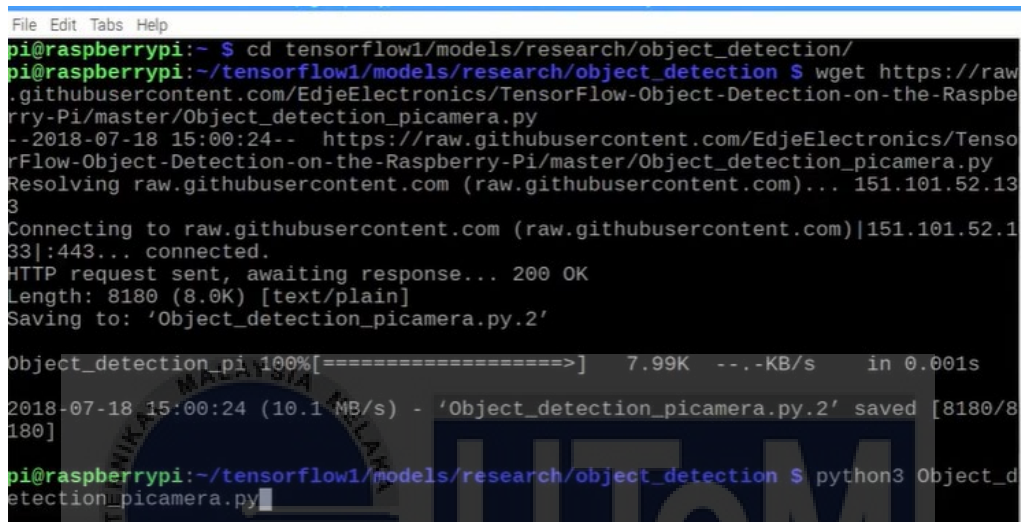
Figure 3.5: Coding for installing Python

3.4.5.2 TensorFlow Machine Learning

TensorFlow is the new machine learning software that is commonly used on artificial intelligence object detection nowadays. TensorFlow needs a suitable model to run the object recognition process. With the use of inception-V3 model, the models which are trained before may work and useful for the different variety of performance tasks. As more sample road sign images of the dataset are being trained by TensorFlow, the higher the accuracy of the recognition on road sign is being analyzed. Recently, TensorFlow machine learning software is easier to be implemented in the Raspberry Pi 3 after the new updated version of TensorFlow software. The compiler is not used anymore when installing TensorFlow into the Raspberry Pi 3 Model B+ and the way to install TensorFlow is much easier as the step for compiling TensorFlow can be skipped.

Hence, for the road sign recognition, TensorFlow machine learning starts to process and recognise the road sign while the Raspberry Pi Camera NoIR Module V2 is recording the real-time video. Raspberry Pi Model B+ will eventually send the output of the result to the Raspberry Pi Display Module for display purposes. When TensorFlow algorithm detects

the road signs with the help of the Python coding for road sign classification, Raspberry Pi Display Module will show the type of that road sign and the percentage of accuracy. The speaker will also trigger to voice out the alert sound. Figure 3.6 shows the coding to run TensorFlow software on object recognition.



```
File Edit Tabs Help
pi@raspberrypi:~ $ cd tensorflow1/models/research/object_detection/
pi@raspberrypi:~/tensorflow1/models/research/object_detection $ wget https://raw.githubusercontent.com/EdjeElectronics/TensorFlow-Object-Detection-on-the-Raspberry-Pi/master/Object_detection_picamera.py
--2018-07-18 15:00:24-- https://raw.githubusercontent.com/EdjeElectronics/TensorFlow-Object-Detection-on-the-Raspberry-Pi/master/Object_detection_picamera.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.52.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.52.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8180 (8.0K) [text/plain]
Saving to: 'Object_detection_picamera.py.2'

Object_detection_pi 100%[=====] 7.99K --.-KB/s in 0.001s
2018-07-18 15:00:24 (10.1 MB/s) - 'Object_detection_picamera.py.2' saved [8180/8180]
pi@raspberrypi:~/tensorflow1/models/research/object_detection $ python3 Object_detection_picamera.py
```

Figure 3.6: Coding for starting detection using TensorFlow machine learning

3.4.5.3 Labeling

Labeling also known as label image, is an open-source image labelling tool that is used to label the size of the road sign image dataset. Each road sign image needs to be labelled first before the training process of the road sign images with TensorFlow machine learning software. The function of labelling the road sign images is to show and confirm where the road signs are located. Hence, after labelling the road sign images, TensorFlow is easier to detect the road sign.

In addition, Labeling carries out the process of segmentation of road sign images and continues with the process of annotation and interpretation for the road sign images. Annotation of road sign images is important as the bounding outside boxes will be shown on the road sign images and thus the images are easier to be recognised as shown in Figure

3.7. The coding used for Labeling is Python programming language. After Labeling annotates the sample set of road sign images, it saves the images as 'XML' file format and the images are ready to be trained and tested by TensorFlow.



Figure 3.7: Labelling the road sign sample image

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

In this chapter, the output and findings of the system will be discussed. The software implementation and hardware implementation for this recognition system will be shown. The analysis of the training result before it is implemented into the real-time alert system, analysis of the accuracy of this system and the reliability of the system will be discussed.

4.2 Hardware Implementation



Figure 4.1: Front view of the project



Figure 4.2: Side view of the project

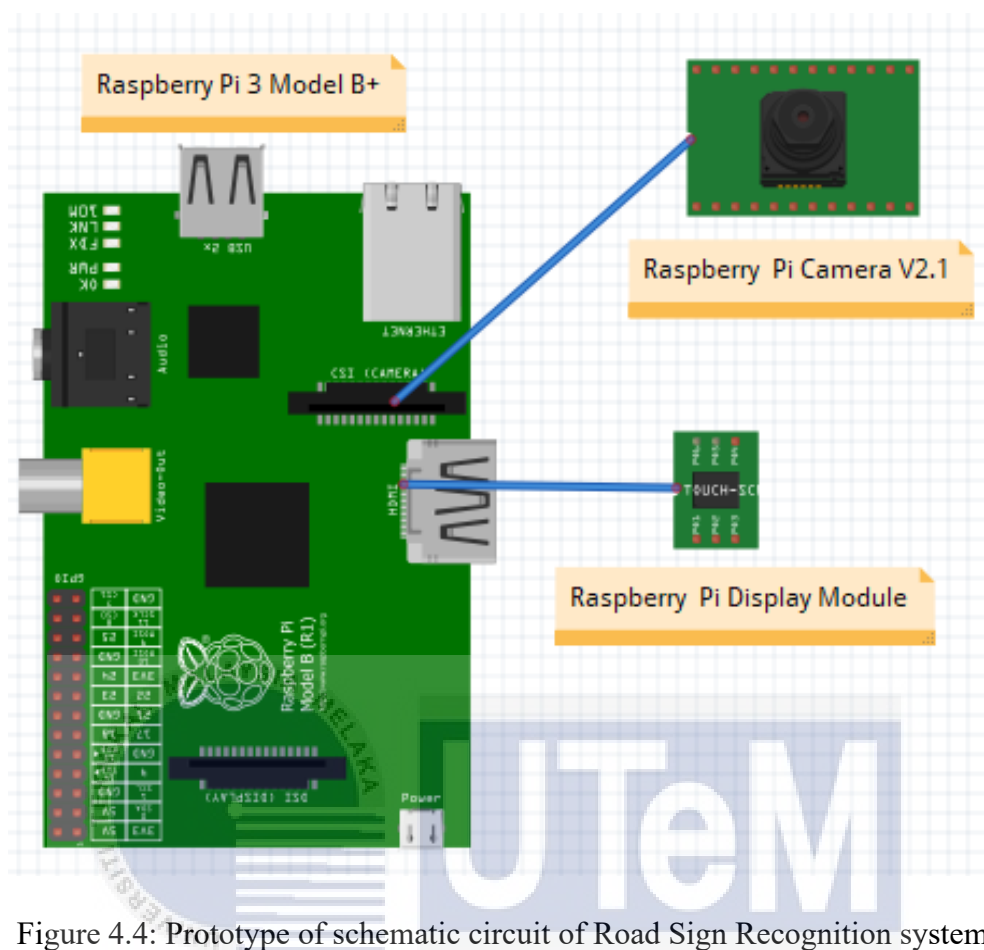


Figure 4.3: Rear view of the project

The hardware implementation of the real-time traffic sign recognition system includes the main device which is the Raspberry Pi 3 Model B+ which acts as the Central Processing Unit (CPU) of this system. Raspberry Pi 3 Model B+ has the Broadcom BCM2837B0 processor embedded to process the programming interface. Raspberry Pi 3 Model B+ connects with Raspberry Pi Camera V2.1 through the CSI Camera Port. Raspberry Pi Camera V2.1 is used to perform real-time video. Other than that, Raspberry Pi Display Module is connected to the HDMI port of the Raspberry Pi 3 Model B+ to display the output of the real-time traffic sign recognition system and alert the user when there is detection on the traffic sign.

4.3 Circuit Design

The schematic circuit of the road sign recognition system includes the Raspberry Pi 3 Model B+, Raspberry Pi Camera V2.1 and Raspberry Pi Display Module as shown in Figure 4.4. Raspberry Pi Camera V2.1 is connected to Raspberry Pi 3 Model B+ through the CSI Camera Port. Raspberry Pi Display Module is connected to the Raspberry Pi 3 Model B+ through HDMI port.



اونيورسيتي تيكنيكل مليسيا ملاك
 UNIVERSITI TEKNIKAL MALAYSIA MELAKA

4.4 Result of Coding

In this work, Python programming language is used to develop coding for TensorFlow software to train the sample set of the road sign images to increase the accuracy of the recognition system. This coding will be embedded in the Raspberry Pi 3 Model B+ microcontroller. Figure 4.5 shows the coding for the training road sign image. Meanwhile, Figure 4.6 and Figure 4.7 show the training result of 500 road sign images in which the loss is above 0.5 and the training result which shows that the loss is below 0.5 after 30 minutes of training respectively.

```
(tensorflow1) C:\tensorflow1\models\research\object_detection>
(tensorflow1) C:\tensorflow1\models\research\object_detection>
(tensorflow1) C:\tensorflow1\models\research\object_detection>
(tensorflow1) C:\tensorflow1\models\research\object_detection>python train.py --logtostderr --train_dir=training/ --pipe
line_config_path=training/faster_rcnn_inception_v2_pets.config
INFO:tensorflow:Scale of 0 disables regularizer.
INFO:tensorflow:Scale of 0 disables regularizer.
WARNING:tensorflow:From C:\tensorflow1\models\research\object_detection\trainer.py:210: create_global_step (from tensorf
low.contrib.framework.python.ops.variables) is deprecated and will be removed in a future version.
Instructions for updating:
Please switch to tf.train.create_global_step
INFO:tensorflow:Scale of 0 disables regularizer.
INFO:tensorflow:Scale of 0 disables regularizer.
INFO:tensorflow:depth of additional conv before box predictor: 0
WARNING:tensorflow:From C:\tensorflow1\models\research\object_detection\core\box_predictor.py:380: calling reduce_mean (
from tensorflow.python.ops.math_ops) with keep_dims is deprecated and will be removed in a future version.
Instructions for updating:
keep_dims is deprecated, use keepdims instead
WARNING:tensorflow:From C:\tensorflow1\models\research\object_detection\core\losses.py:339: softmax_cross_entropy_with_l
ogits (from tensorflow.python.ops.nn_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Future major versions of TensorFlow will allow gradients to flow
into the labels input on backprop by default.
See tf.nn.softmax_cross_entropy_with_logits_v2.
```

Figure 4.5: Coding for training road sign images

```

INFO:tensorflow:Saving checkpoint to path training/model.ckpt
INFO:tensorflow:Starting Queues.
INFO:tensorflow:global_step/sec: 0
INFO:tensorflow:Recording summary at step 0.
INFO:tensorflow:global step 1: loss = 4.9249 (7.167 sec/step)
INFO:tensorflow:global step 2: loss = 5.1224 (0.213 sec/step)
INFO:tensorflow:global step 3: loss = 4.4063 (0.197 sec/step)
INFO:tensorflow:global step 4: loss = 4.3019 (0.202 sec/step)
INFO:tensorflow:global step 5: loss = 4.4028 (0.211 sec/step)
INFO:tensorflow:global step 6: loss = 3.8368 (0.236 sec/step)
INFO:tensorflow:global step 7: loss = 3.6195 (0.204 sec/step)
INFO:tensorflow:global step 8: loss = 3.4207 (0.200 sec/step)
INFO:tensorflow:global step 9: loss = 3.0984 (0.203 sec/step)
INFO:tensorflow:global step 10: loss = 2.6750 (0.206 sec/step)
INFO:tensorflow:global step 11: loss = 2.5898 (0.211 sec/step)
INFO:tensorflow:global step 12: loss = 2.2851 (0.195 sec/step)
INFO:tensorflow:global step 13: loss = 2.0338 (0.206 sec/step)
INFO:tensorflow:global step 14: loss = 1.9776 (0.212 sec/step)
INFO:tensorflow:global step 15: loss = 1.3794 (0.197 sec/step)
INFO:tensorflow:global step 16: loss = 1.5916 (0.195 sec/step)
INFO:tensorflow:global step 17: loss = 2.1680 (0.201 sec/step)
INFO:tensorflow:global step 18: loss = 1.1794 (0.199 sec/step)
INFO:tensorflow:global step 19: loss = 1.2081 (0.204 sec/step)
INFO:tensorflow:global step 20: loss = 0.7990 (0.213 sec/step)
INFO:tensorflow:global step 21: loss = 1.0784 (0.198 sec/step)
INFO:tensorflow:global step 22: loss = 0.8025 (0.201 sec/step)
INFO:tensorflow:global step 23: loss = 1.7400 (0.205 sec/step)
INFO:tensorflow:global step 24: loss = 1.8792 (0.201 sec/step)
INFO:tensorflow:global step 25: loss = 0.9136 (0.203 sec/step)

```

Figure 4.6: The training result of 500 road sign images which the loss is above 0.5

```

INFO:tensorflow:global step 56153: loss = 0.0305 (0.208 sec/step)
INFO:tensorflow:global step 56154: loss = 0.0354 (0.186 sec/step)
INFO:tensorflow:global step 56155: loss = 0.0638 (0.187 sec/step)
INFO:tensorflow:global step 56156: loss = 0.0580 (0.198 sec/step)
INFO:tensorflow:global step 56157: loss = 0.0157 (0.199 sec/step)
INFO:tensorflow:global step 56158: loss = 0.0681 (0.192 sec/step)
INFO:tensorflow:global step 56159: loss = 0.0767 (0.190 sec/step)
INFO:tensorflow:global step 56160: loss = 0.0197 (0.183 sec/step)
INFO:tensorflow:global step 56161: loss = 0.0606 (0.194 sec/step)
INFO:tensorflow:global step 56162: loss = 0.0173 (0.195 sec/step)
INFO:tensorflow:global step 56163: loss = 0.0204 (0.188 sec/step)
INFO:tensorflow:global step 56164: loss = 0.0113 (0.187 sec/step)
INFO:tensorflow:global step 56165: loss = 0.0180 (0.193 sec/step)
INFO:tensorflow:global step 56166: loss = 0.0117 (0.191 sec/step)
INFO:tensorflow:global step 56167: loss = 0.0320 (0.188 sec/step)
INFO:tensorflow:global step 56168: loss = 0.0154 (0.211 sec/step)
INFO:tensorflow:global step 56169: loss = 0.0290 (0.198 sec/step)
INFO:tensorflow:global step 56170: loss = 0.0730 (0.199 sec/step)
INFO:tensorflow:global step 56171: loss = 0.0252 (0.193 sec/step)
INFO:tensorflow:global step 56172: loss = 0.0584 (0.195 sec/step)
INFO:tensorflow:global step 56173: loss = 0.0325 (0.189 sec/step)
INFO:tensorflow:global step 56174: loss = 0.0301 (0.196 sec/step)
INFO:tensorflow:global step 56175: loss = 0.0298 (0.200 sec/step)
INFO:tensorflow:global step 56176: loss = 0.0132 (0.193 sec/step)
INFO:tensorflow:global step 56177: loss = 0.0126 (0.198 sec/step)
INFO:tensorflow:global step 56178: loss = 0.0487 (0.194 sec/step)
INFO:tensorflow:global step 56179: loss = 0.0302 (0.195 sec/step)
INFO:tensorflow:global step 56180: loss = 0.0428 (0.190 sec/step)
INFO:tensorflow:global step 56181: loss = 0.0805 (0.200 sec/step)

```

Figure 4.7: Training result showed that the loss is below 0.5 after 30 minutes

4.5 Findings of the project

Five different classes of warning-type traffic sign sample images are collected as the sample images dataset and trained with TensorFlow software as stated above. The class of warning-type included the 'Speed Bump' traffic sign, 'Stop' traffic sign, 'Give Away' traffic sign, 'Curve to Right' traffic sign and 'No U-turn' traffic sign. After the pre-trained model was prepared, the real-time recognition system was taken outside the field to test the traffic sign from different classes. The pre-trained model contained the size, colour, shape and boundary of the sample traffic sign images that used to recognised.



Figure 4.8: Original view of Speed Bump traffic sign

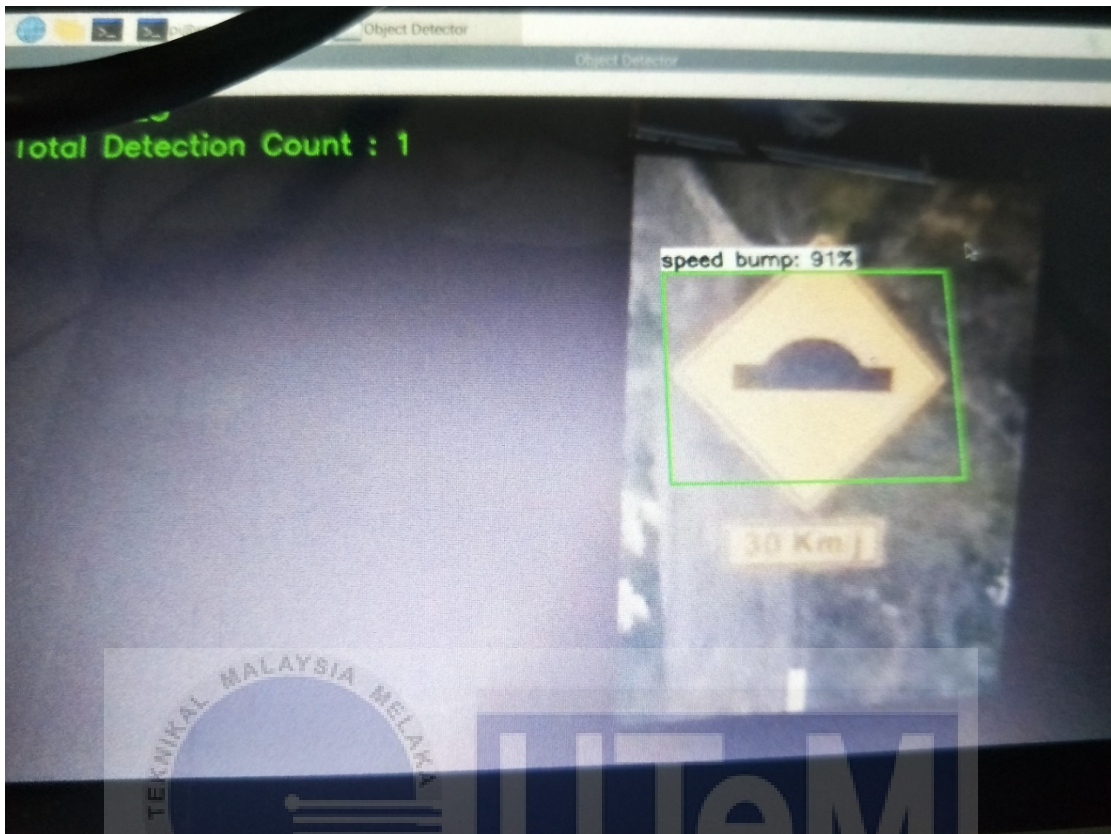


Figure 4.9: Screenshot of 'Speed Bump' traffic sign is recognised from the front view



Figure 4.10: Original view of 'Stop' traffic sign

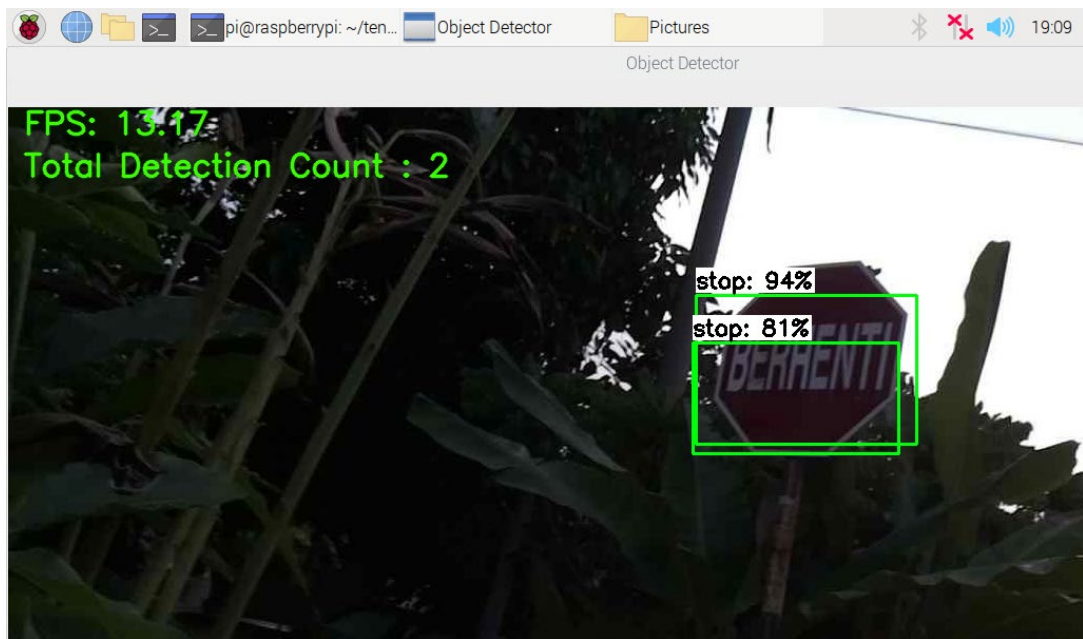


Figure 4.11: Screenshot of 'Stop' traffic sign is recognised from the side view

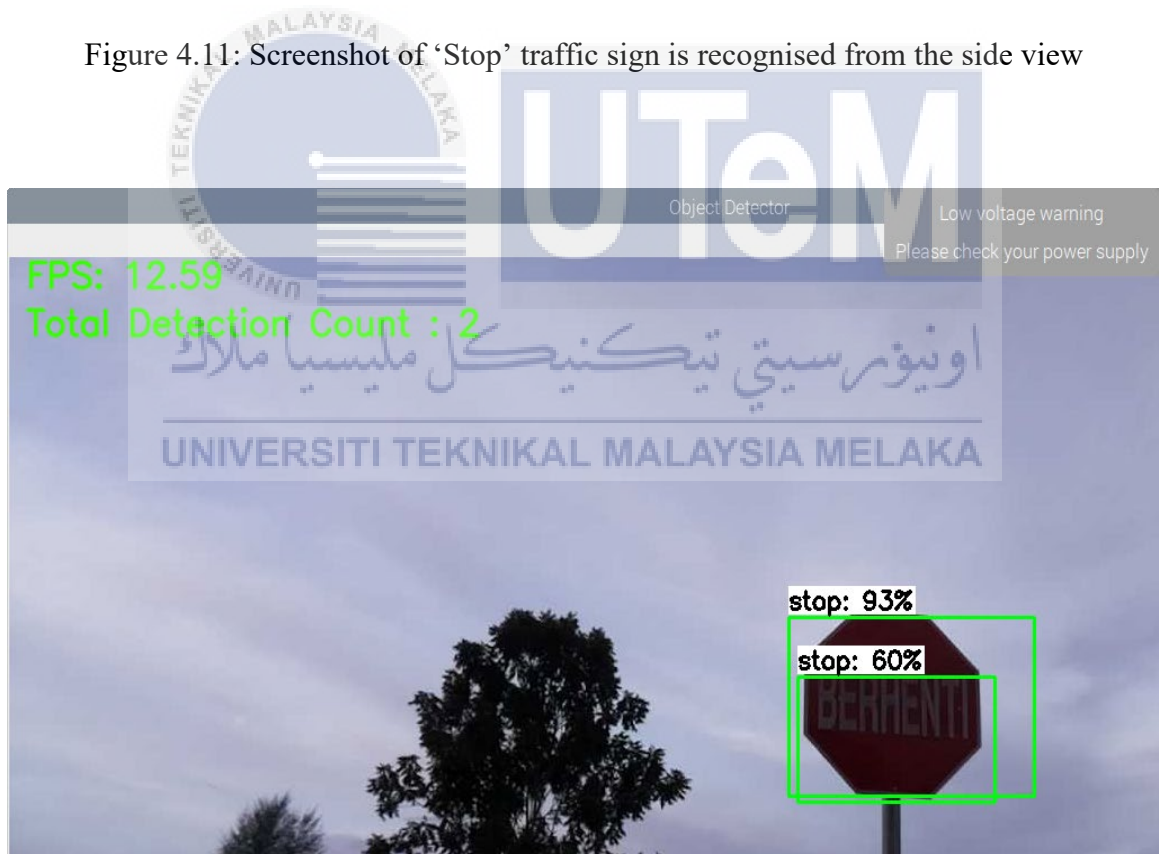


Figure 4.12: Screenshot of 'Stop' traffic sign is recognised from the far view within the distance range of 3 metre



Figure 4.13: Original view of 'Give Away' traffic sign



Figure 4.14: Screenshot of 'Give Away' traffic sign is recognised with 90 percent of accuracy



Figure 4.15: Original view of 'Curve to Right' traffic sign

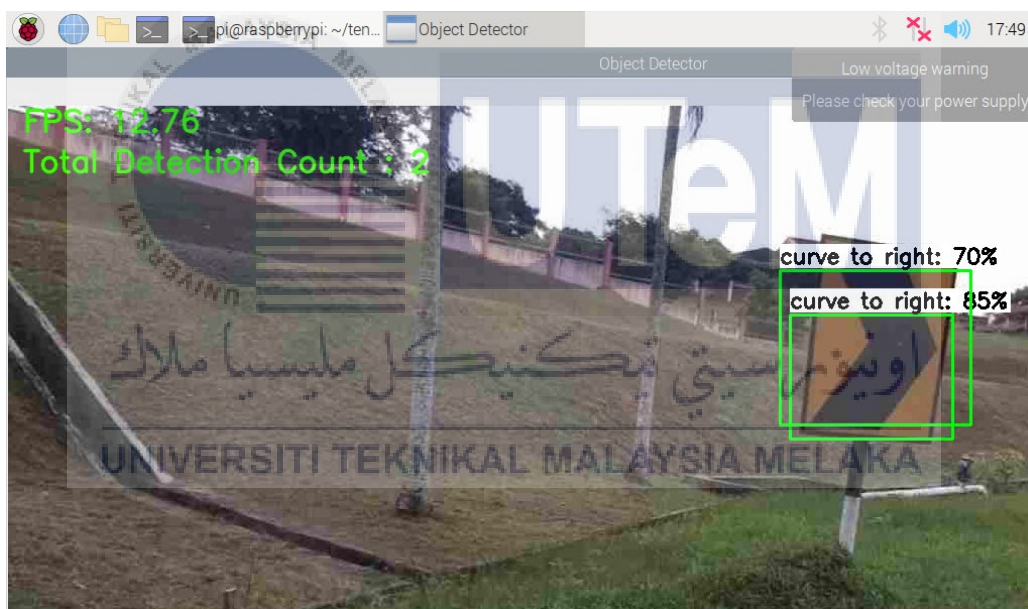


Figure 4.16: Screenshot of 'Curve to Right' traffic sign is recognised

The real-time traffic sign recognition system is implemented and installed in the car in front of the wide car window. The car was then driven slowly around the residential area to test the recognition of that targeted and specific warning-type traffic sign in the real-time mode. The result above showed that the specific warning-type of traffic signs which are from

that five classes of traffic sign that are trained before is recognised successfully with the percentage accuracy of 90 percent and above.

4.6 Analysis of the training result

A dataset that consists of the different classes of traffic signs is prepared for the TensorFlow machine learning to train and learn from the dataset sample traffic sign. The training process is important before the implementation of the real-time recognition system. The number of steps used to train the dataset will affect the loss of the training process.

```
INFO:tensorflow:Step 48700 per-step time 0.537s loss=0.055
I1209 07:11:27.034243 5516 model_lib_v2.py:648] Step 48700 per-step time 0.537s loss=0.055
INFO:tensorflow:Step 48800 per-step time 0.545s loss=0.059
I1209 07:12:19.104255 5516 model_lib_v2.py:648] Step 48800 per-step time 0.545s loss=0.059
INFO:tensorflow:Step 48900 per-step time 0.530s loss=0.057
I1209 07:13:12.261588 5516 model_lib_v2.py:648] Step 48900 per-step time 0.530s loss=0.057
INFO:tensorflow:Step 49000 per-step time 0.568s loss=0.061
I1209 07:14:05.748207 5516 model_lib_v2.py:648] Step 49000 per-step time 0.568s loss=0.061
INFO:tensorflow:Step 49100 per-step time 0.557s loss=0.047
I1209 07:15:00.226881 5516 model_lib_v2.py:648] Step 49100 per-step time 0.557s loss=0.047
INFO:tensorflow:Step 49200 per-step time 0.527s loss=0.057
I1209 07:15:53.769811 5516 model_lib_v2.py:648] Step 49200 per-step time 0.527s loss=0.057
INFO:tensorflow:Step 49300 per-step time 0.599s loss=0.047
I1209 07:16:47.281497 5516 model_lib_v2.py:648] Step 49300 per-step time 0.599s loss=0.047
INFO:tensorflow:Step 49400 per-step time 0.541s loss=0.067
I1209 07:17:41.830239 5516 model_lib_v2.py:648] Step 49400 per-step time 0.541s loss=0.067
INFO:tensorflow:Step 49500 per-step time 0.578s loss=0.070
I1209 07:18:36.238093 5516 model_lib_v2.py:648] Step 49500 per-step time 0.578s loss=0.070
INFO:tensorflow:Step 49600 per-step time 0.663s loss=0.074
I1209 07:19:29.844349 5516 model_lib_v2.py:648] Step 49600 per-step time 0.663s loss=0.074
INFO:tensorflow:Step 49700 per-step time 0.547s loss=0.055
I1209 07:20:23.445427 5516 model_lib_v2.py:648] Step 49700 per-step time 0.547s loss=0.055
INFO:tensorflow:Step 49800 per-step time 0.540s loss=0.060
I1209 07:21:17.711602 5516 model_lib_v2.py:648] Step 49800 per-step time 0.540s loss=0.060
INFO:tensorflow:Step 49900 per-step time 0.531s loss=0.056
I1209 07:22:10.092414 5516 model_lib_v2.py:648] Step 49900 per-step time 0.531s loss=0.056
INFO:tensorflow:Step 50000 per-step time 0.482s loss=0.047
I1209 07:23:03.695848 5516 model_lib_v2.py:648] Step 50000 per-step time 0.482s loss=0.047
```

Figure 4.17: First training process of the dataset

```
INFO:tensorflow:Step 5300 per-step time 10.972s loss=0.165
I1229 23:43:27.514392 7324 model_lib_v2.py:648] Step 5300 per-step time 10.972s loss=0.165
INFO:tensorflow:Step 5400 per-step time 10.685s loss=0.192
I1230 00:01:32.412905 7324 model_lib_v2.py:648] Step 5400 per-step time 10.685s loss=0.192
INFO:tensorflow:Step 5500 per-step time 10.962s loss=0.175
I1230 00:19:56.910857 7324 model_lib_v2.py:648] Step 5500 per-step time 10.962s loss=0.175
INFO:tensorflow:Step 5600 per-step time 11.041s loss=0.180
I1230 00:38:29.410203 7324 model_lib_v2.py:648] Step 5600 per-step time 11.041s loss=0.180
INFO:tensorflow:Step 5700 per-step time 11.129s loss=0.166
I1230 00:57:01.829581 7324 model_lib_v2.py:648] Step 5700 per-step time 11.129s loss=0.166
```

Figure 4.18: Second training process of the dataset

The dataset that includes 5 different classes of sample traffic sign images is prepared. This dataset consists of 500 sample images which each class has 100 sample images respectively. 80 percent of the total sample images which are 400 images are pre-saved into the train directory where the rest 20 percent, 100 sample images are pre-saved into the test directory. The dataset is trained with TensorFlow machine learning using SSD MobileNet V2 FPNLite 640x640 and the result of the first training process is shown in Figure 4.17. The result showed one step time and the loss of the model for every 100 steps of the training process. The sample dataset model is being trained from Step 100 to Step 50000 and it is shown that at Step 50000, the loss is 0.047. Whereas for the second training process that is shown in Figure 4.18, the same dataset is used to train with the same model which is SSD MobileNet V2 FPNLite 640x640. The training process started at Step 100 and stopped at Step 5700 and the loss of the model is 0.166. Thus, it can be analyzed that the more steps of the training process to be carried out, the less the loss of the model. The more steps the machine learning took to learn the model, the loss of the training model to be learned is reduced. One step time for every 100 steps of the training process can be represented by the training time duration.

The time duration will be only affected by the hardware Graphics Processing Unit (GPU) and Central Processing Unit (CPU) that is used to train the sample dataset model. In Figure 4.17, more powerful, high performance CPU and GPU are used to train the model whereas the lower performance of CPU and GPU are used to train the model in Figure 4.18, therefore the per-step time for every 100 steps of the training process for high performance hardware, which is about 0.550 seconds is much less than the per-step time for low performance hardware, which is about 10.900 seconds. The graph of total loss against the step of the training process and graph of learning rate against step is shown below. The total loss is decreased when the step for the training process is increased which is shown in Figure 4.19. The learning rate is increased steeply from Step 100 until Step 1000 and then remains constant until Step 2800 and slightly decreased from Step 3000 until the training process stopped as shown in Figure 4.20.

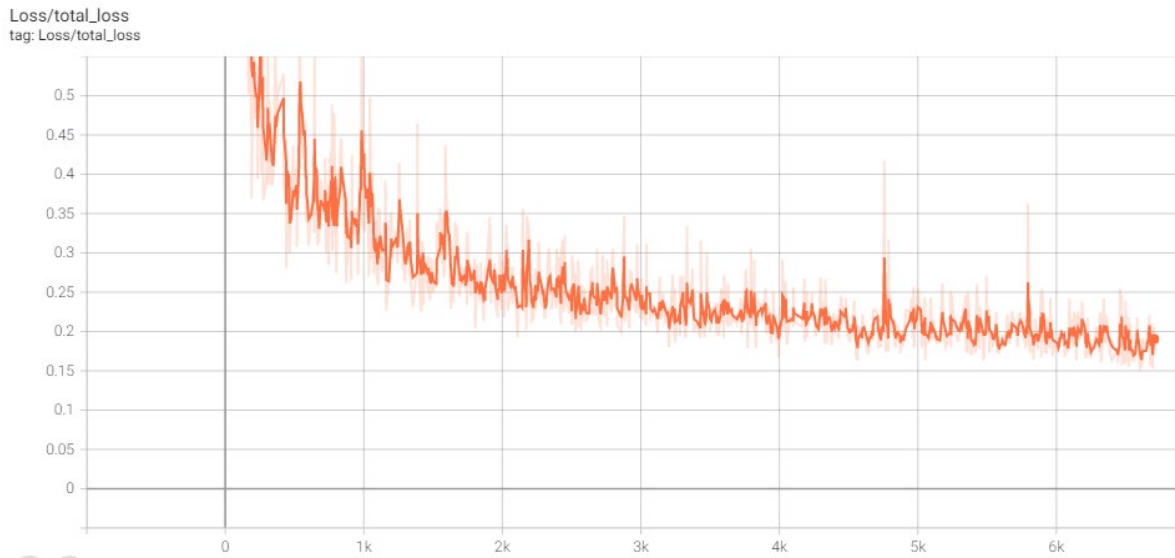


Figure 4.19: Total loss against step in the training process

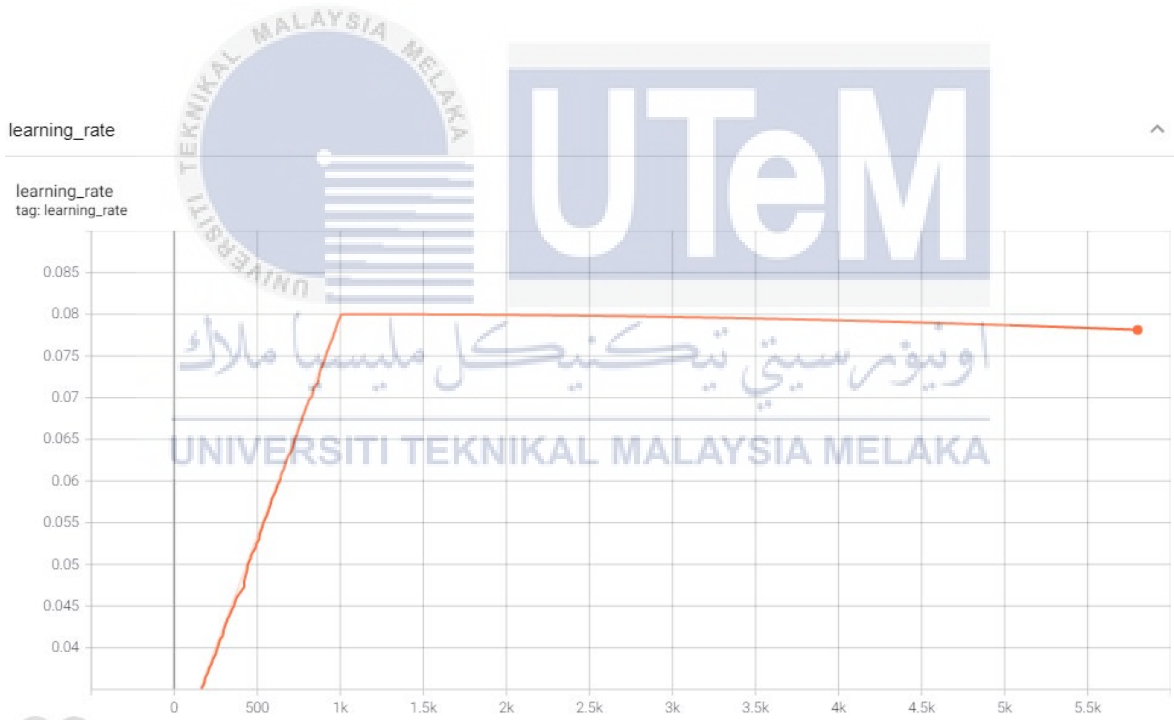
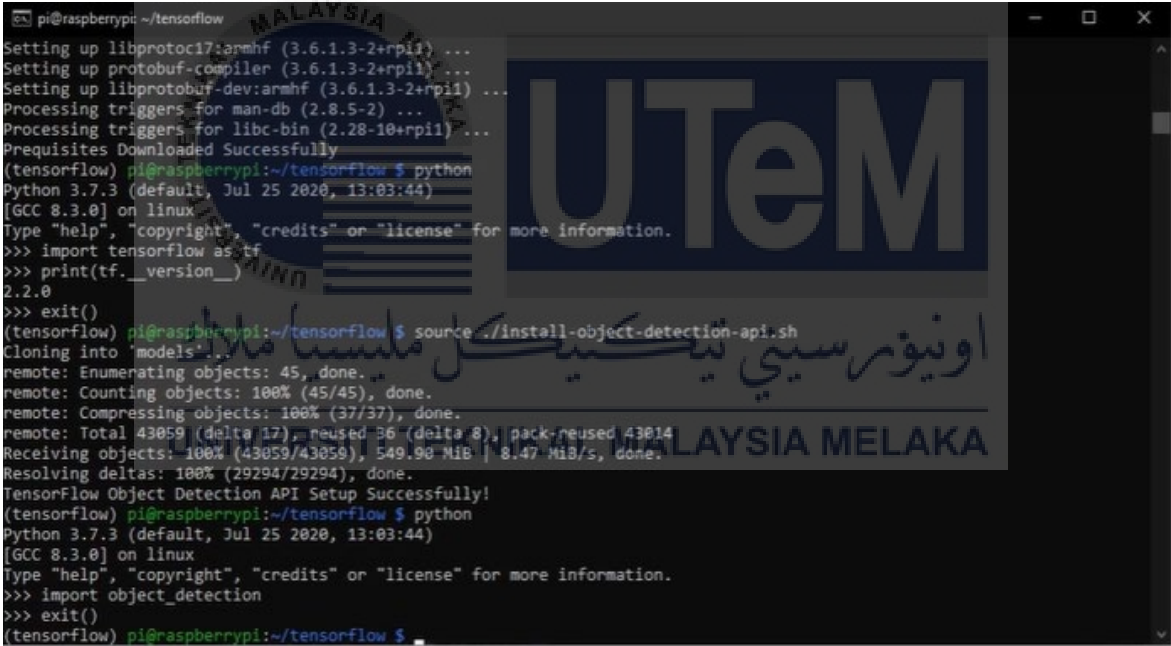


Figure 4.20: Learning rate against step in the training process

4.7 Traffic Sign Recognition System with TensorFlow

After the dataset model is being trained, TensorFlow machine learning software is being set up in the Raspberry Pi Model 3 Model B+ hardware as shown in the figure below. There are many versions of TensorFlow software that are suitable to be installed in the Raspberry Pi 3 Model B+. The latest version of TensorFlow software is recommended to be installed on the Raspberry Pi 3 Model B+ is the latest version of TensorFlow software which is supported by Raspbian, which is the operating system of the Raspberry Pi. However, the latest version of TensorFlow which is TensorFlow 2.4.0 is not stable and hence TensorFlow 2.2 is chosen to be installed on Raspberry Pi 3 Model B+ as it is a stable and new version of TensorFlow. TensorFlow version 2.2 is suitable to be installed because this version of software is smoother and more advance to be used on object recognition function.



```
pi@raspberrypi: ~/tensorflow
Setting up libprotoc17:armhf (3.6.1.3-2+rpil) ...
Setting up protobuf-compiler (3.6.1.3-2+rpil) ...
Setting up libprotobuf-dev:armhf (3.6.1.3-2+rpil) ...
Processing triggers for man-db (2.8.5-2) ...
Processing triggers for libc-bin (2.28-10+rpil) ...
Prerequisites Downloaded Successfully
(tensorflow) pi@raspberrypi:~/tensorflow $ python
Python 3.7.3 (default, Jul 25 2020, 13:03:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> print(tf.__version__)
2.2.0
>>> exit()
(tensorflow) pi@raspberrypi:~/tensorflow $ source ./install-object-detection-api.sh
Cloning into 'models'...
remote: Enumerating objects: 45, done.
remote: Counting objects: 100% (45/45), done.
remote: Compressing objects: 100% (37/37), done.
remote: Total 43059 (delta 17), reused 36 (delta 0), pack-reused 43014
Receiving objects: 100% (43059/43050), 549.90 MiB | 8.47 MiB/s, done.
Resolving deltas: 100% (29294/29294), done.
TensorFlow Object Detection API Setup Successfully!
(tensorflow) pi@raspberrypi:~/tensorflow $ python
Python 3.7.3 (default, Jul 25 2020, 13:03:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import object_detection
>>> exit()
(tensorflow) pi@raspberrypi:~/tensorflow $
```

Figure 4.21: TensorFlow 2.2.0 is installed on Raspberry Pi 3

Moreover, there is also another version of TensorFlow called TensorFlow Lite version which is the simple version of the TensorFlow software. TensorFlow Lite normally applies on the embedded devices and mobile IoT devices due to it has low latency and a faster interpreter to save the models in a small size. Thus, the TensorFlow Lite version is

installed on Raspberry Pi 3 Model B+ as shown in Figure because it can run models faster and has high performance on object detection.

```
Setting up libjasper-dev (1.900.1-debian1-2.4+deb8u1) ...
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting opencv-python==4.1.0.25
  Downloading https://www.piwheels.org/simple/opencv-python/opencv_python-4.1.0.25-cp37-cp37m-linux_armv7l.whl (9.8MB)
    100% |#####| 9.8MB 46kB/s
Collecting numpy>=1.16.2 (from opencv-python==4.1.0.25)
  Downloading https://www.piwheels.org/simple/numpy/numpy-1.19.4-cp37-cp37m-linux_armv7l.whl (10.5MB)
    100% |#####| 10.5MB 43kB/s
Installing collected packages: numpy, opencv-python
Successfully installed numpy-1.19.4 opencv-python-4.1.0.25
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting tflite-runtime==2.5.0 from https://github.com/google-coral/pycoral/releases/download/release-frogfish/tflite_runtime-2.5.0-cp37-cp37m-linux_armv7l.whl
  Downloading https://github.com/google-coral/pycoral/releases/download/release-frogfish/tflite_runtime-2.5.0-cp37-cp37m-linux_armv7l.whl (1.1MB)
    100% |#####| 1.1MB 391kB/s
Requirement already satisfied: numpy>=1.12.1 in ./lib/python3.7/site-packages (from tflite-runtime==2.5.0) (1.19.4)
Installing collected packages: tflite-runtime
Successfully installed tflite-runtime-2.5.0
Prerequisites Installed Successfully
(tensorflow) pi@raspberrypi:~/tensorflow $ python
Python 3.7.3 (default, Jul 25 2020, 13:03:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tflite_runtime as tf
>>> print(tf.__version__)
2.5.0
```

Figure 4.22: TensorFlow Lite version 2.5.0 is installed

In this traffic sign recognition system, TensorFlow version 2.2.0 machine learning software is installed first on the Raspberry Pi 3 Model B+ and then run the pre-trained model which consists of the dataset that was trained before. Likewise, TensorFlow Lite version 2.5.0 is also installed on the Raspberry Pi and tested with the pre-trained model. For the TensorFlow Lite version, the pre-trained model which is in the '.pb' file format model must be converted into the '.tflite' file format model. For example, my pre-trained model which is named as 'saved_model.pb' file is converted into 'model.tflite' file. This conversion did not change the image boundary data and the classification information of the dataset that is trained by the algorithm before. After the pre-trained model is converted into '.tflite' file format, then it can be read by TensorFlow Lite version software that is installed on the Raspberry Pi 3 Model B+. The process of conversion of TensorFlow pre-trained model into TensorFlow Lite pre-trained model is shown in Figure 4.23.

```

(tflite) C:\Users\armaa>python
Python 3.7.9 (default, Aug 31 2020, 17:10:11) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
2020-11-27 13:12:48.359555: W tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2020-11-27 13:12:48.359781: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
>>> print(tf.__version__)
2.5.0-dev20201127
>>> exit()

(tflite) C:\Users\armaa>cd C:\TensorFlow\workspace\training_demo

(tflite) C:\TensorFlow\workspace\training_demo>python convert-to-tflite.py\
python: can't open file 'convert-to-tflite.py': [Errno 22] Invalid argument

(tflite) C:\TensorFlow\workspace\training_demo>python convert-to-tflite.py
2020-11-27 13:14:30.006225: W tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2020-11-27 13:14:30.006383: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.

```

Figure 4.23: Conversion of TensorFlow pre-trained model into TensorFlow Lite version

By using the TensorFlow Lite version pre-trained model, the Raspberry Pi 3 Model B+ is turned on and tested the traffic sign recognition system. The pre-trained model can be tested by running the code to open the Pi Camera. The result of the loaded TensorFlow Lite version pre-trained model is shown in Figure 4.24. After that, the pre-trained model before the conversion was tested with the normal TensorFlow version. The result of the comparison is shown below.

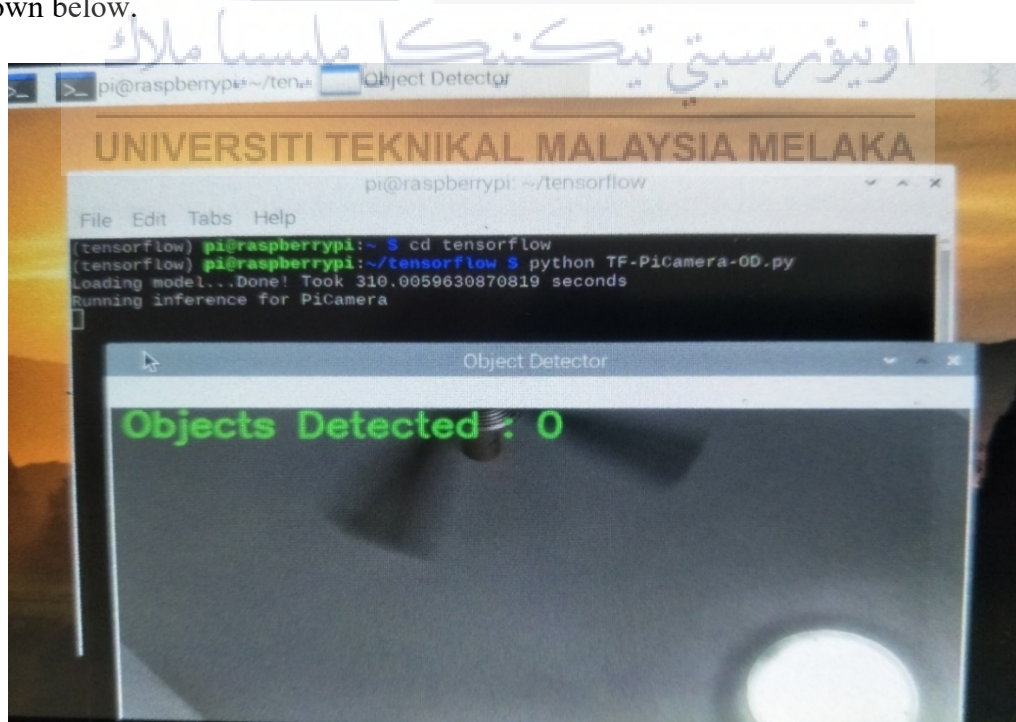
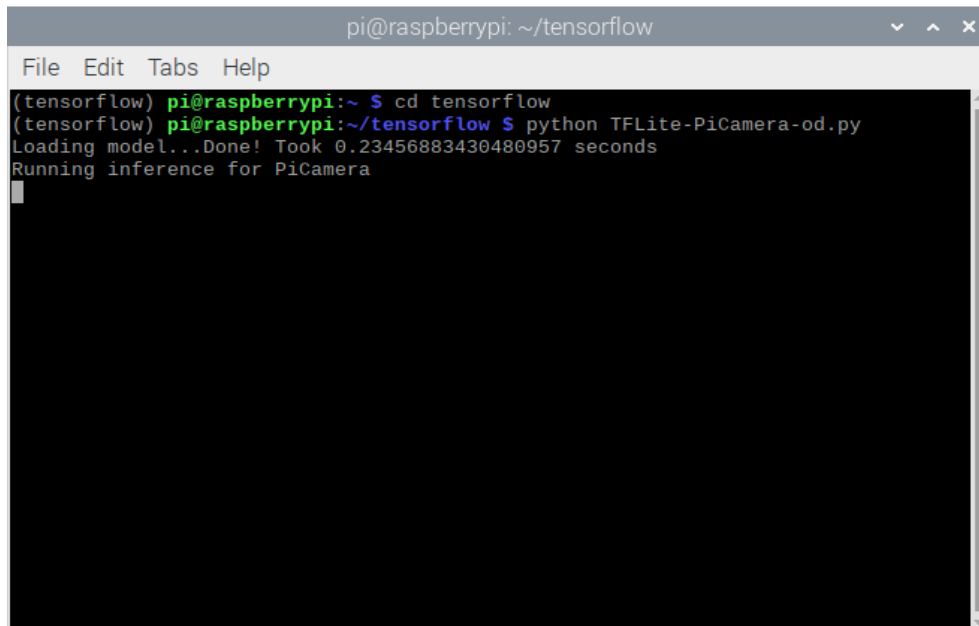


Figure 4.24: Pre-trained model run by TensorFlow normal version 2.2.0

A terminal window titled 'pi@raspberrypi: ~/tensorflow' with a menu bar (File, Edit, Tabs, Help). The terminal output shows the following commands and results:

```
(tensorflow) pi@raspberrypi:~ $ cd tensorflow
(tensorflow) pi@raspberrypi:~/tensorflow $ python TFLite-PiCamera-od.py
Loading model...Done! Took 0.23456883430480957 seconds
Running inference for PiCamera
```

Figure 4.25: Pre-trained model loaded with TensorFlow Lite version 2.5.0

It can be shown from the figure above about the time taken for the TensorFlow to load the pre-trained model of the dataset. Full version TensorFlow 2.2.0 machine learning software took about 310 seconds with is 5 minutes and 10 seconds the successfully loads the pre-trained model. Whereas for the TensorFlow Lite version 2.5.0, the TensorFlow Lite pre-trained model is loaded by TensorFlow Lite version 2.5.0 and it only took about 0.235 seconds to successfully load. Hence, TensorFlow Lite is 1319 times faster than TensorFlow full version and the performance is much higher for the TensorFlow Lite version software compared to the full version of TensorFlow. TensorFlow Lite saves much time in loading the pre-trained model before running the Pi Camera to perform the real-time traffic sign recognition. Hence, TensorFlow Lite machine learning software is implemented in this system due to it is more reliable and high efficiency.

4.8 Analysis of the accuracy of the system

The accuracy of the real-time traffic sign recognition system is very important as it is used to classify a certain class of images from the real-time video which is the warning traffic sign images. The warning traffic sign images are needed to recognise so that the road user is notified and alert with that warning traffic sign.

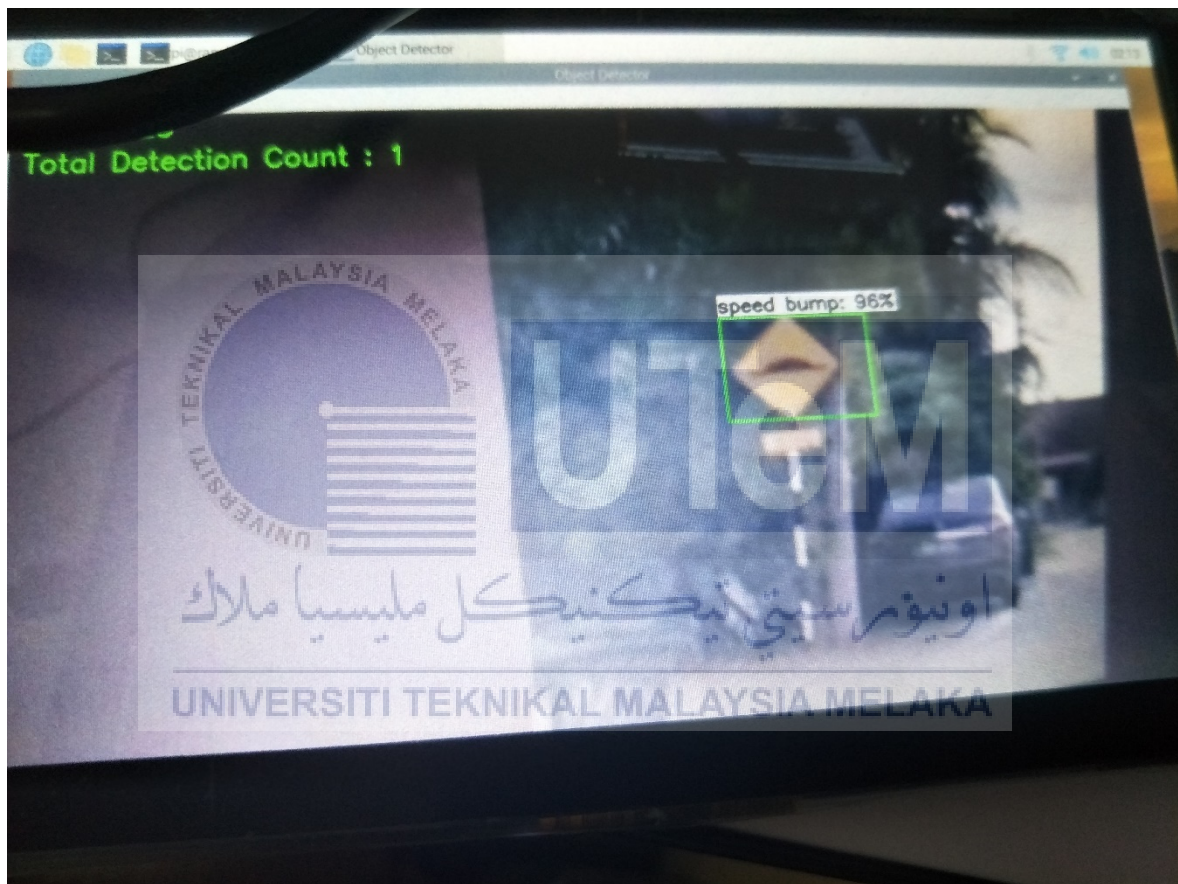


Figure 4.26: Detection of ‘Speed Bump’ warning traffic sign with the output accuracy of 96 percent



Figure 4.27: ‘Stop’ warning traffic sign is recognised with the highest 96 percent of accuracy

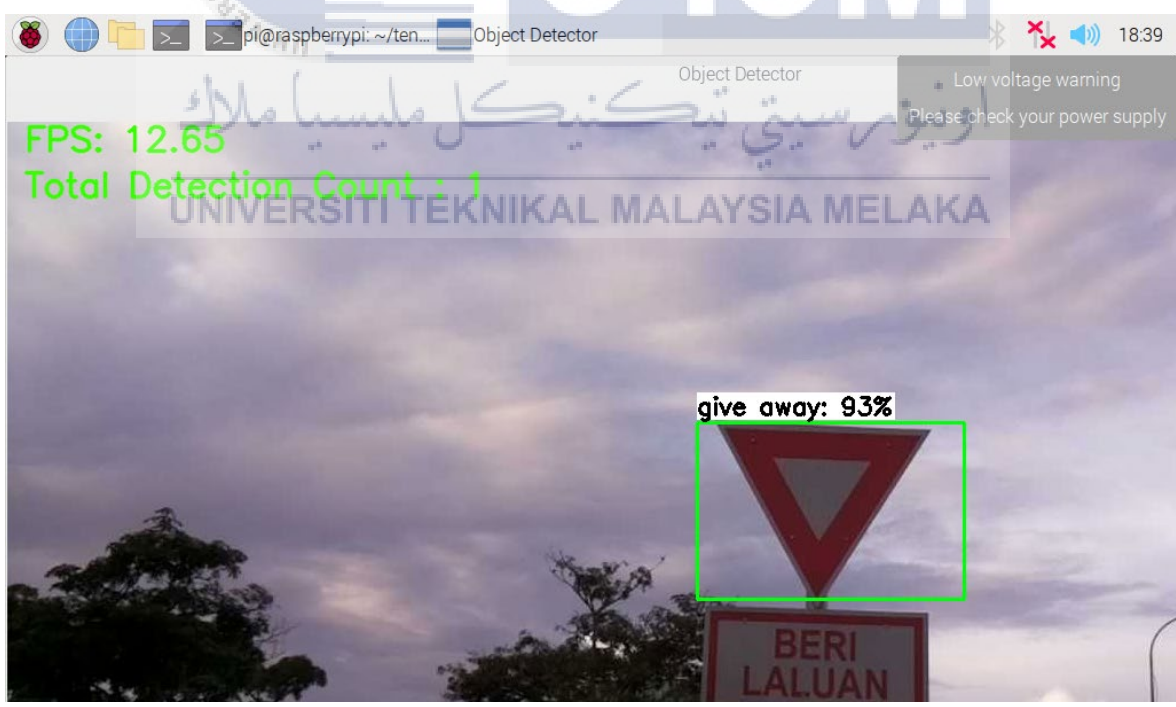


Figure 4.28: ‘Give Away’ warning traffic sign is recognised with 93 percent of high accuracy

From the figure above, the Stop warning traffic sign achieves 94 percent of accuracy as the real-time traffic sign recognition system manages to detect the class of warning signs precisely and accurately. The figure above is the recognition of stop signs in the evening condition and the system can recognise them with a high percentage of accuracy.

Class of Warning Traffic Sign	Test Sample	Accuracy
Stop	1	93
Speed Bump	2	91
Give Away	3	94
Speed Bump	4	93
Stop	5	95
Speed Bump	6	96
No U-turn	7	95
Give Away	8	94
Speed Bump	9	93
Give Away	10	93
Speed Bump	11	93
Curve to Right	12	92
Stop	13	91
Speed Bump	14	93
Curve to Right	15	92
Give Away	16	91
Speed Bump	17	92
No U-turn	18	93
Stop	19	91
Give Away	20	92

Table 4.1: Accuracy of the total 20 test of images with different classes

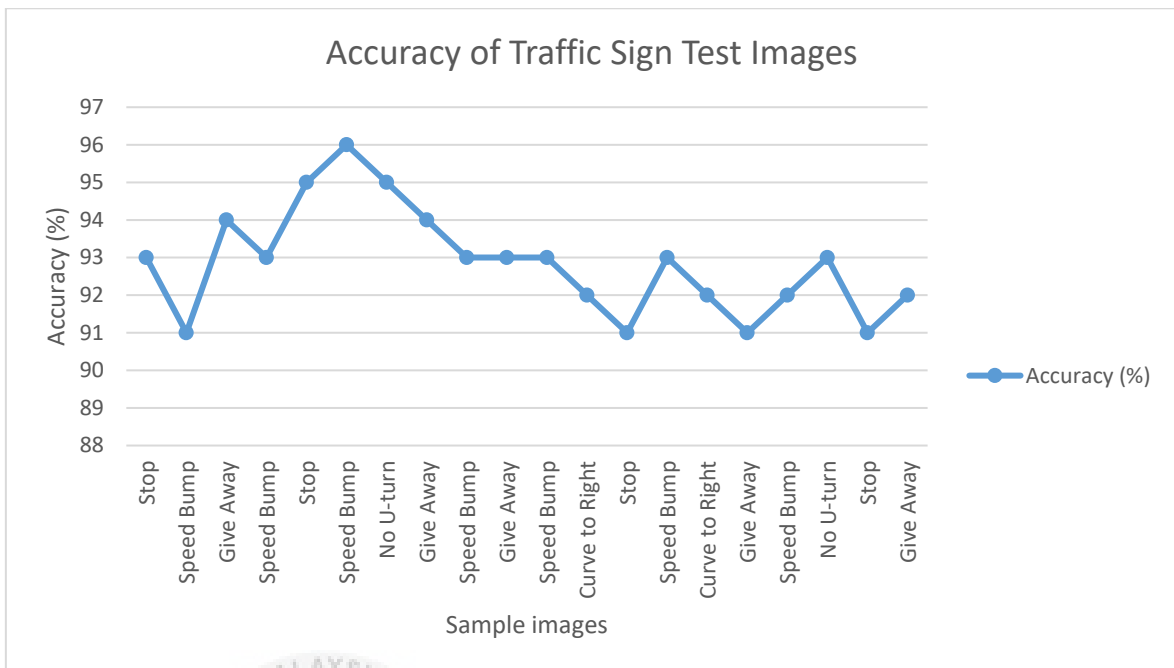


Figure 4.29: Graph of accuracy versus the test sample images

The result of accuracy for 20 test sample images for a different class of warning traffic signs is recorded in the table above. These 20 test images are taken randomly from different classes of the warning traffic sign. In this traffic sign test analysis, the warning traffic sign images consist of ‘speed bump’ warning traffic sign, ‘stop’ warning traffic sign and ‘give away’ warning traffic sign, ‘warning for children’ warning traffic sign and ‘speed limit’ warning traffic sign. This is because these classes of warning traffic signs are commonly found at the roadside and it is categorized as the important alert to the road user to keep safe from danger. From that, it can be analyzed that the accuracy of the traffic sign recognition system is above 90 percent which this recognition system is accurate and manages to classify the traffic sign object and non-traffic sign object. Hence, each class of detected warning traffic sign is high in accuracy to identify whether it is categorized as the right type of traffic sign.

4.9 Analysis of the reliability of the system

The reliability of the real-time traffic sign recognition system is the focus part of this project as it is the dependency of this system for the awareness of the road user. This traffic sign recognition can detect the warning traffic sign from the side view instead of the detection from the front view to the warning traffic sign.

The Frame Per Second (FPS) value for this real-time recognition is showed at the range of 12 to 13.50 FPS, which the average FPS value is at 12.75 FPS. The FPS value of this real-time recognition system depended on the performance of the hardware that is implemented on this system, for example, the input power source of the system. The FPS value is dropped when there is a lack of power input or low voltage that is not enough to support the system and hence it caused the latency to become higher and the system lagged. Meanwhile, the FPS value is also affected by the condition of the running real-time video by the Raspberry Pi Camera V2.1 and loaded together with the pre-trained model.

The algorithm inside the TensorFlow Lite software ran the pre-trained model to detect the traffic sign and presented the output on the display screen, which is the Raspberry Pi Display Module. The system needed a couple of seconds to run the real-time video and at the same time loaded the real-time video with the pre-trained model. The pre-trained model contained the size and boundary of the specific traffic sign that is needed to be detected. Hence, the running algorithm caused a little latency of the system as it needed time to read the real-time video. This latency of the system was the time delay that happened on the Raspberry Pi Display Module after the Pi Camera was moving from one point of view to another point of view. The result for 20 different classes of traffic sign images was collected to record the time delay and percentage of accuracy.

Class of Warning Traffic Sign	Accuracy (%)	Time Delay (Seconds)
Stop	93	3.13
Speed Bump	91	3.12
Give Away	94	3.02
Speed Bump	93	3.25

Stop	95	3.16
Speed Bump	96	3.48
No U-turn	95	3.36
Give Away	94	3.28
Speed Bump	93	3.55
Give Away	93	3.19
Speed Bump	93	3.29
Curve to Right	92	3.08
Stop	91	3.67
Speed Bump	93	3.58
Curve to Right	92	3.75
Give Away	91	3.36
Speed Bump	92	3.82
No U-turn	93	3.51
Stop	91	3.45
Give Away	92	3.31

Table 4.2: Accuracy vs Time Delay of the total 20 test of images with different classes

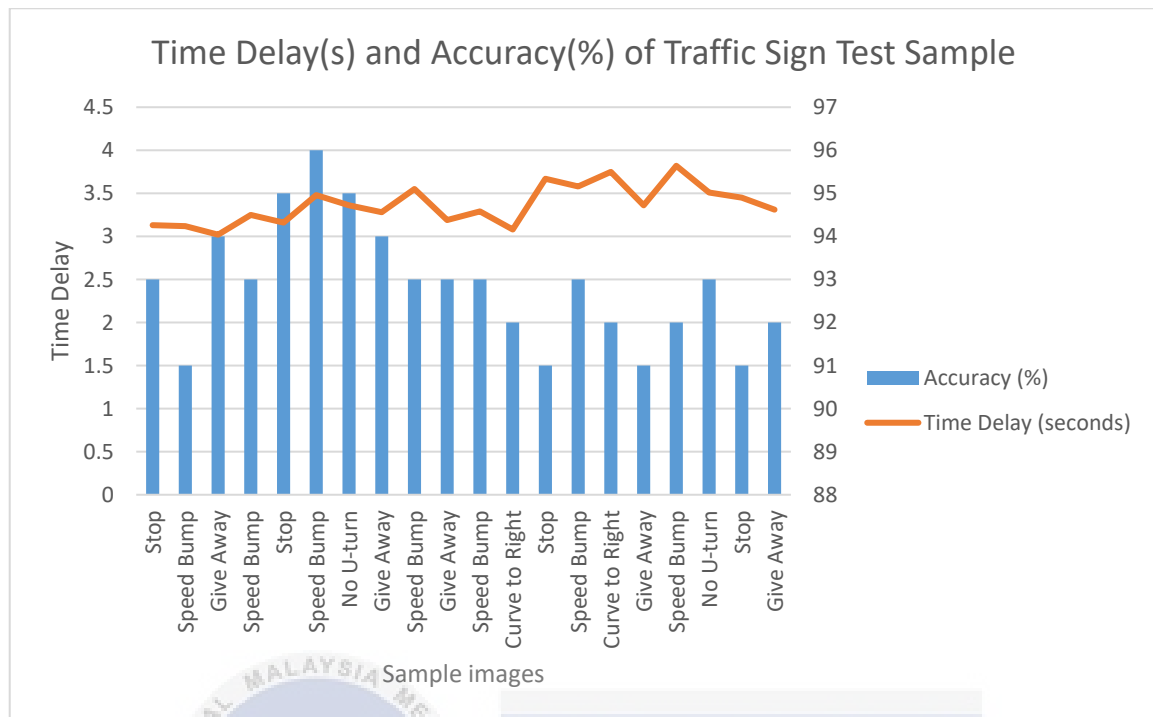


Figure 4.30: Graph of accuracy versus time delay of the test traffic sign sample

From the tabulated graph above, it can be summarized that the average time delay for 20 different class of traffic sign sample test images is 3.37 seconds which the lowest time delay recorded is 3.02 seconds and the highest time delay is 3.82 seconds. Thus, there was some delay of time due to the performance of the system hardware and the condition of the algorithm to load and read the pre-trained model when the real-time video was running. Therefore, this real-time traffic sign recognition system is suitable for normal speed and slow speed moving cars that are being driven at 50 kilometres per hour and below. This recognition system is not suitable for fast-moving cars as the latency of the system may cause danger to the driver. Hence, the reliability of this real-time recognition system is quite good due to the Pi Camera that is run by TensorFlow Lite can detect the class of traffic signs in the real-time mode.

4.9.1 Reliability of system on recognition of incomplete traffic sign

The reliability of the real-time traffic sign recognition system also depends on the ability of the recognition system to detect the incomplete part of the traffic sign or the damaged traffic sign. Sometimes, there are some damaged traffic signs found on the roadside which have not been repaired by the authorities. These damaged traffic signs are missing some parts of their real shape that it should be looked like. The figure below shows that this real-time traffic sign recognition system manages to recognise the missing part of the incomplete traffic sign with a high percentage of accuracy.

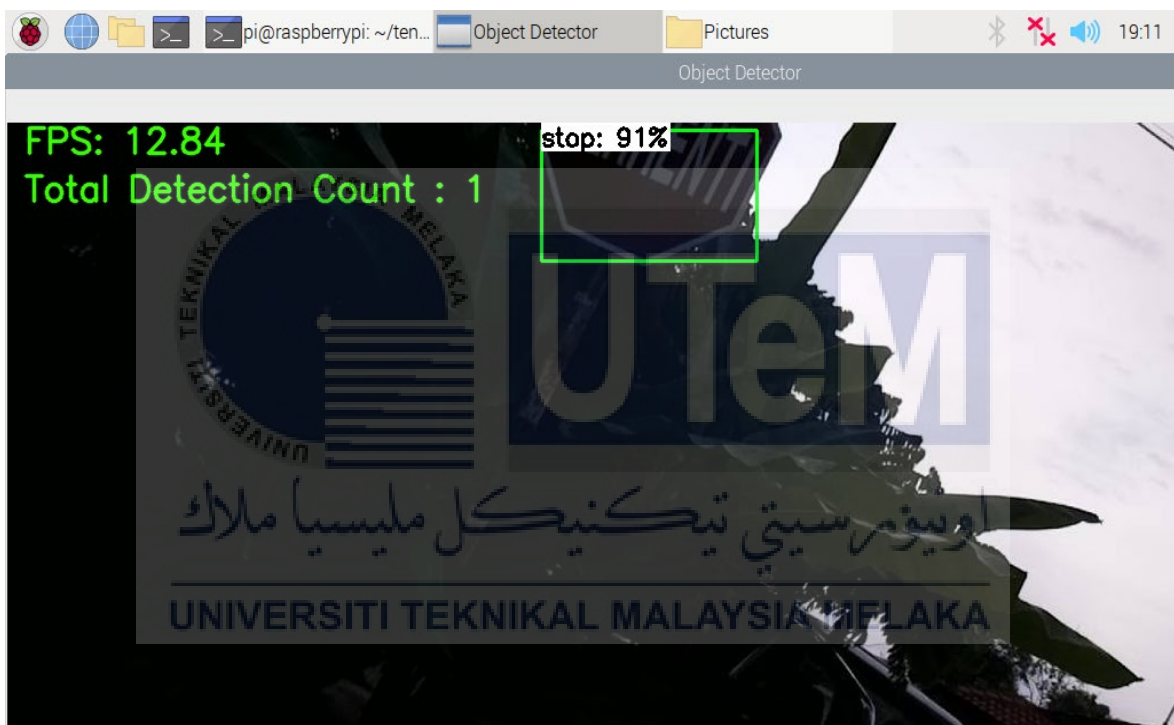


Figure 4.31: Missing part of incomplete 'Stop' warning traffic sign is recognised with 91% of accuracy

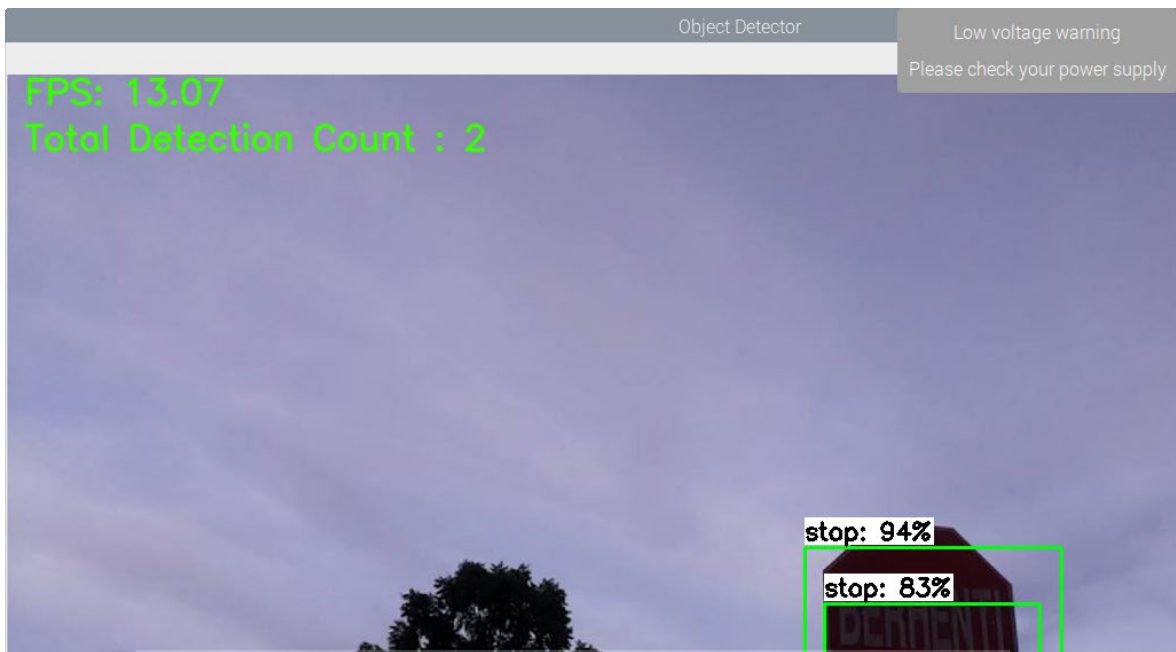


Figure 4.32: Missing part of the ‘Stop’ warning traffic sign is recognised with 94 percent of accuracy

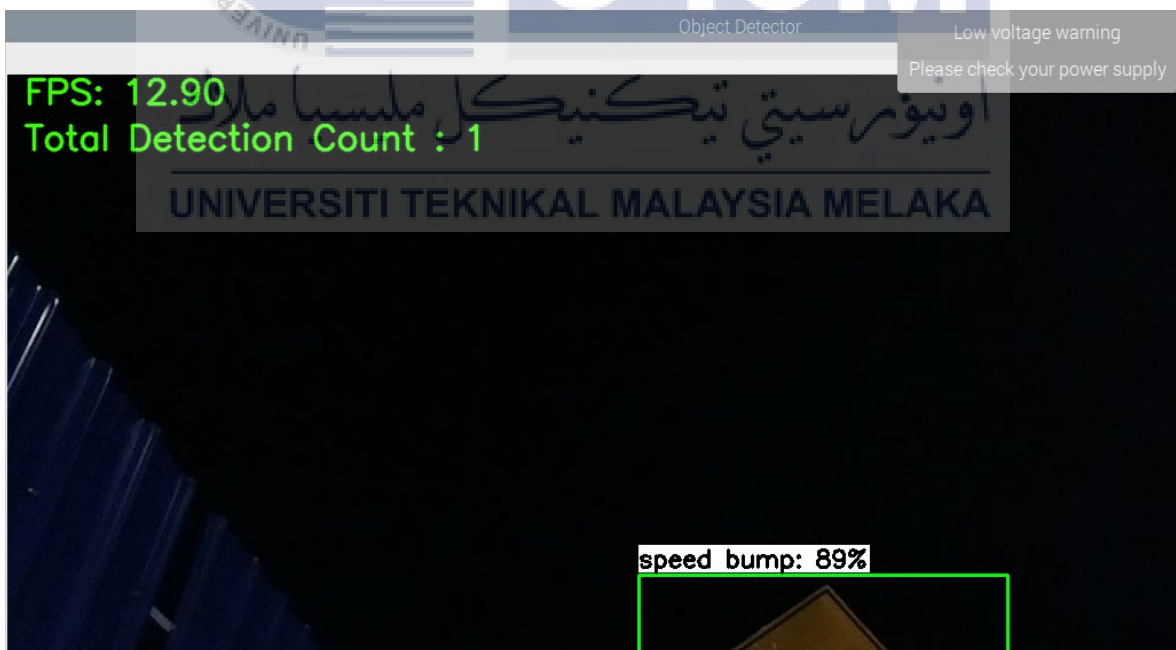


Figure 4.33: The missing part of ‘Speed Bump’ incomplete warning traffic sign is recognised

4.9.2 Reliability of system based on recognition at day and night

The reliability of the real-time traffic sign recognition system depends on the ability of that system to recognise the warning-type traffic sign in different conditions. Commonly, vehicle transport is being driven not only in the daytime but also at the night-time. Thus, the requirement for the real-time traffic sign recognition system at night-time is high. This is because driving at night is more dangerous and the driver should more alert to the warning-type traffic sign. This real-time traffic sign recognition system can recognise the different classes of warning-type traffic signs which are from the pre-trained model during the day and night. This system manages to recognise and detect the traffic sign with high accuracy at night-time as shown in the figure below.



Figure 4.34: Recognition of 'Speed Bump' traffic sign with 93 percent of accuracy at night

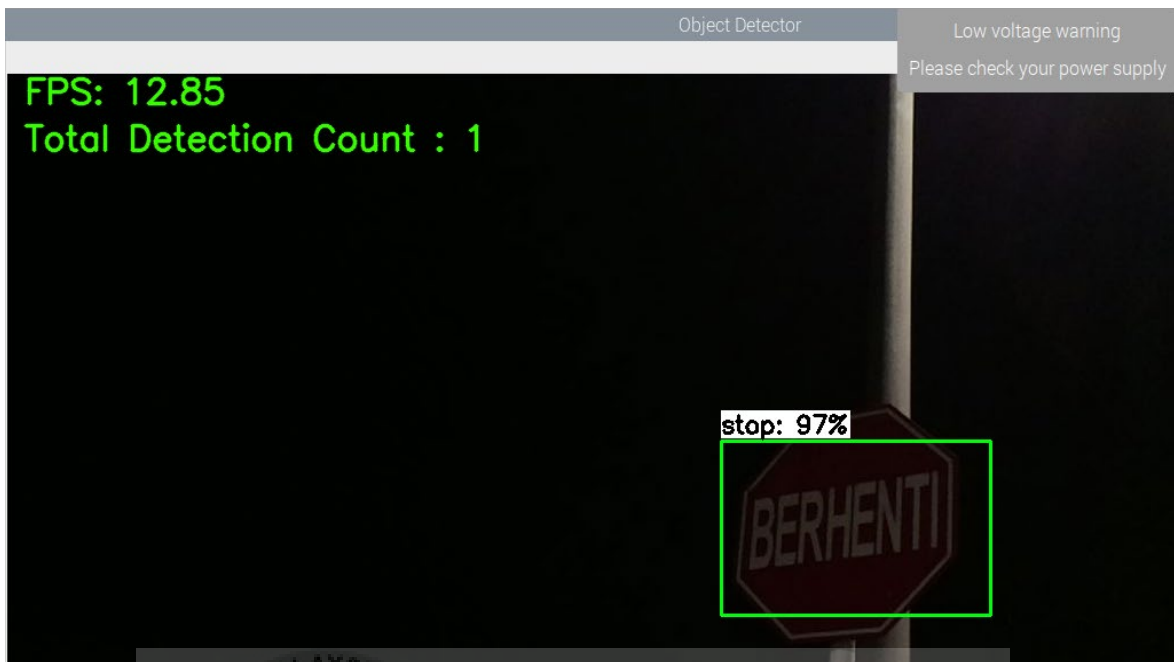


Figure 4.35: Recognition of 'Stop' traffic sign with 97 percent of accuracy at night



Figure 4.36: Recognition of 'Speed Bump' traffic sign from the front view at night

4.9.3 Reliability of system based on recognition of colour faded traffic sign

A good real-time traffic sign recognition system manages to recognise traffic signs with high accuracy although the traffic sign itself is in bad condition. Bad condition of traffic sign includes the colour fading traffic sign. Commonly, the colour fading condition occurs on the old traffic sign that can be found everywhere by the roadside nowadays. Colour fading normally occurred on the traffic sign that not yet renovate by the authorities for many years. However, this real-time traffic sign recognition system manages to detect and classify the class of the colour fading traffic sign and the output result showed the high percentage of accuracy. Figure 4.37 showed the recognition of colour fading traffic signs.

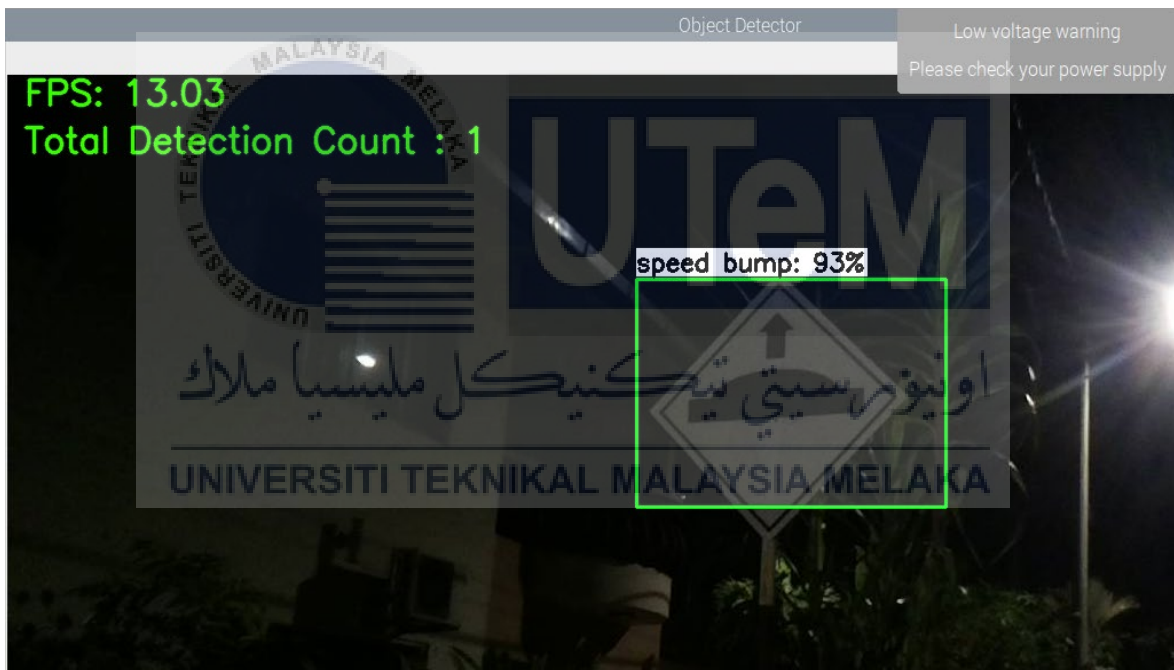


Figure 4.37: Colour fading of ‘Speed Bump’ traffic sign is recognised

4.10 Project Limitation

In the real-time traffic sign recognition system with TensorFlow Lite, there were also some limitations and difficulties for the development of this system. The limitations were the pixel of the Raspberry Pi Camera V2.1 is not enough high to detect the traffic signs from the far distance range. The Pi Camera was not able to run high resolution real-time video to recognise the warning traffic sign. This system had the limitation of detection for warning traffic signs within the distance range of more than 5 metres between the Pi Camera lens and the traffic sign. Besides, the FPS value for this real-time traffic sign recognition system using TensorFlow Lite machine learning software is not enough high to perform smooth real-time video display. There is also the latency of the system caused this real-time traffic sign recognition system is not able to be implemented on fast-moving car due to the time delay of about 3.37 seconds. Therefore, it was slow for the TensorFlow Lite to react when there was detection on the warning-type traffic signs especially when this real-time recognition system was applied on the fast-moving vehicle.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Introduction

This chapter showed the part of the conclusion of the output on this real-time traffic sign recognition system. This chapter also discussed the recommendation for future improvements to enhance this real-time traffic sign recognition system.

5.2 Conclusion

From Chapter 1 as mentioned earlier before, the objectives of this real-time traffic sign recognition system that required to be achieved for the well-developed of this project. The three objectives were achieved since the real-time traffic sign recognition system was successfully developed using the Raspberry Pi 3 processor with the corporation of the TensorFlow machine learning software. TensorFlow machine learning software is used and was able to train the dataset sample images of the warning-type traffic sign. Moreover, TensorFlow Lite version software was used to load the pre-trained detection model and run Raspberry Pi Camera for the real-time recognition of traffic signs. This project had been completed according to the flowchart of the project and the Gantt chart.

Besides, the purpose of this real-time traffic sign recognition system was to analyze the accuracy of the developed road sign recognition system in determining the road signs. This real-time recognition system was successfully worked and managed to recognised different classes of warning-type traffic signs with above 90 percent of accuracy. Thus, this real-time recognition system was precise and accurate. Moreover, the reliability of the developed system in real-time implementation was successfully evaluated. This reliable real-time recognition system managed to recognise the missing part and incomplete traffic sign and alert the driver. Besides, this system also can recognise colour fading warning-type traffic signs. This recognition system manages to recognise traffic signs during daytime and at night with high accuracy of recognition output results. This system was also able to run a video interface with good FPS for the detection of the traffic sign.

Meanwhile, the design of this system is compact, simple and convenient and contains the ability to recognise an object and correctly name the class of the object in the real-time mode is good enough for the current implementation of traffic sign detection and recognition system for the road safety and awareness. This real-time traffic sign recognition system only requires a very low power consumption and it enables drivers to implement this system on their vehicles and travels everywhere they want to recognise the presence of warning-type traffic signs.

5.3 Recommendation

In this part, the project is proposed to have some enhancement to increase its functionality in the future time. As this is a new project and not a complete system that can be applied on the autonomous driving system is proposed with the improvements below:

1. Applying the Google Coral Accelerator hardware which can highly increase the FPS value when running the video interface for traffic sign recognition system. With the high frame per second or high frame rate, this real-time recognition system manages to recognise the presence of traffic signs faster in the real-time mode.

2. Modifying the current camera module to a better camera module that has a higher megapixel camera lens and functionality.

A camera with a higher megapixel lens enables the system to have a high resolution of video interface for the process of traffic sign recognition.

3. Improve the system by replacing the Raspberry Pi with the Android-based or Linux-based Tablet device to be able to run the recognition system faster.

Since the processor of Raspberry Pi 3 Model B+ which is Broadcom BCM2837B0 has the limitation on performing high performance object recognition. The Snapdragon processor and Kirin processor that have been implemented on Android-based Tablet device have better performance to

perform a real-time recognition task, for example, increase FPS of video interface at the same time load the pre-trained model faster.

5.4 Project Potential

In addition, this project can be applied as the part of the autonomous driving system or the Advanced Driving Assistance System (ADAS) as it can automatically recognise and classify the warning-type traffic sign. This recognition system can be modified into another type of traffic sign instead of the warning-type traffic sign. For example, the information type of traffic sign can be applied into this real-time recognition system. The Regulatory-type traffic sign and the mandatory instruction traffic sign can be implemented into this recognition system. With the addition of other types of traffic signs into the Advanced Driving Assistance System (ADAS), different model of traffic sign dataset can be applied into the OpenCV platform on the TensorFlow machine learning software. Hence, this advanced real-time traffic sign recognition system can be implemented on tourism transport as it can detect and read the information and instruction traffic signs. This is suitable for the vehicle that is driven by users who are first travel and explore to the new places.

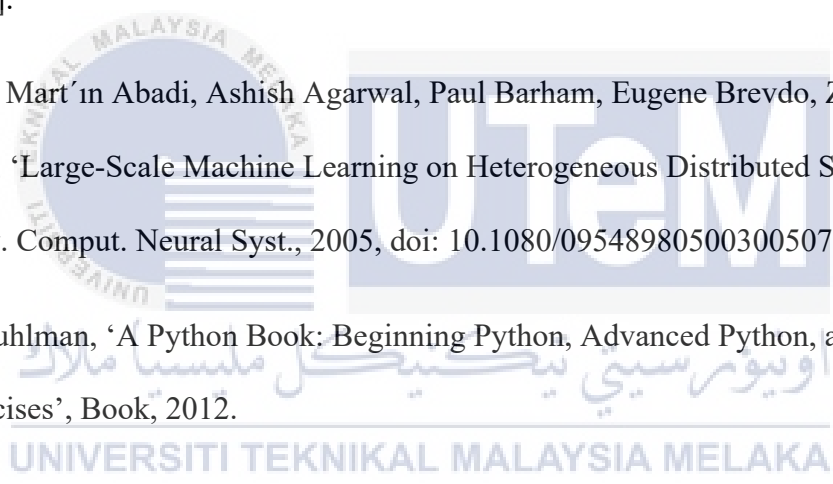
REFERENCES

- [1] M. Lum, 'We have the third highest death rate from road accidents', Star Online, 2019.
- [2] A. Madani and R. Yusof, 'Malaysian traffic sign dataset for traffic sign detection and recognition systems', J. Telecommun. Electron. Comput. Eng., 2016.
- [3] M. M. Lau, K. H. Lim, and A. A. Gopalai, 'Malaysia traffic sign recognition with convolutional neural network', in International Conference on Digital Signal Processing, DSP, 2015, doi: 10.1109/ICDSP.2015.7252029.
- [4] W. Road and R. Work, 'Malaysian Road Traffic Signs Warning signs Regulatory signs', 2019.
- [5] D. R. Bruno and F. S. Osorio, 'Image classification system based on deep learning applied to the recognition of traffic signs for intelligent robotic vehicle navigation purposes', Proc. - 2017 LARS 14th Lat. Am. Robot. Symp. 2017 5th SBR Brazilian Symp. Robot. LARS-SBR 2017 - Part Robot. Conf. 2017, vol. 2017-Decem, pp. 1–6, 2017, doi: 10.1109/SBR-LARS-R.2017.8215287.
- [6] X.-L. Xia, C. Xu, and B. Nan, 'Facial Expression Recognition Based on TensorFlow Platform', ITM Web Conf., vol. 12, p. 01005, 2017, doi: 10.1051/itmconf/20171201005.

- [7] M. A. A. Sheikh, A. Kole, and T. Maity, 'Traffic sign detection and classification using colour feature and neural network', 2016 Int. Conf. Intell. Control. Power Instrumentation, ICICPI 2016, pp. 307–311, 2017, doi: 10.1109/ICICPI.2016.7859723.
- [8] Y. Saadna and A. Behloul, 'An overview of traffic sign detection and classification methods', Int. J. Multimed. Inf. Retr., vol. 6, no. 3, pp. 193–210, 2017, doi: 10.1007/s13735-017-0129-8.
- [9] A. Vennelakanti, S. Shreya, R. Rajendran, D. Sarkar, D. Muddegowda, and P. Hanagal, 'Traffic Sign Detection and Recognition using a CNN Ensemble', 2019 IEEE Int. Conf. Consum. Electron. ICCE 2019, pp. 1–4, 2019, doi: 10.1109/ICCE.2019.8662019.
- [10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, 'ImageNet classification with deep convolutional neural networks', Commun. ACM, 2017, doi: 10.1145/3065386.
- [11] H. S. Lee and K. Kim, 'Simultaneous Traffic Sign Detection and Boundary Estimation Using Convolutional Neural Network', IEEE Trans. Intell. Transp. Syst., vol. 19, no. 5, pp. 1652–1663, 2018, doi: 10.1109/TITS.2018.2801560.
- [12] N. Heath, "What is the Raspberry Pi 3? Everything you need to know about the tiny, low-cost computer," ZDNet, 30-Nov-2017. [Online]. Available: <https://www.zdnet.com/article/what-is-the-raspberry-pi-3-everything-you-need-to-know-about-the-tiny-low-cost-computer/>. [Accessed: 08-Apr-2020].

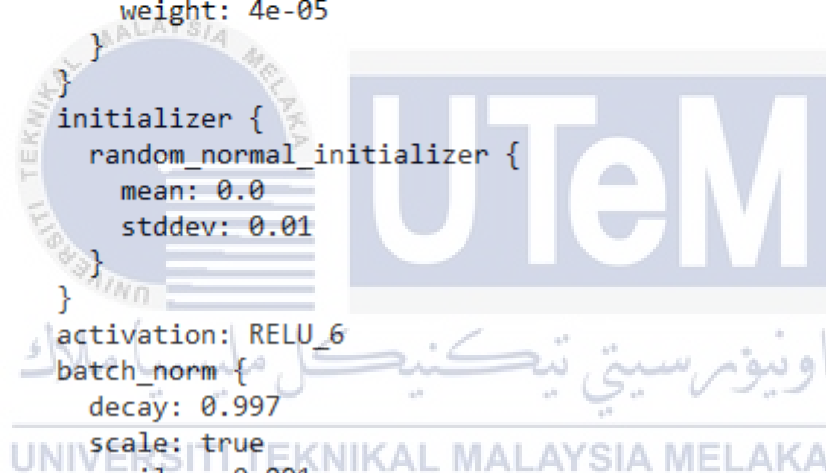
- [13] B. Westover, "Raspberry Pi Model B Review: An Awesomely Versatile \$35 Mini PC," Tom's Guide, 04-Dec-2018. [Online]. Available: <https://www.tomsguide.com/us/raspberry-pi-3-model-b-plus,review-5983.html>. [Accessed: 08-Apr-2020].
- [14] "Raspberry Pi 3 Model B," Cytron Technologies Malaysia, 21-Feb-2020. [Online]. Available: <https://my.cytron.io/p-raspberry-pi-3-model-b>. [Accessed: 08-Apr-2020].
- [15] M. Long, M. L. is an editor, and Follow, "Libre Computer ROC-RK3328-CC vs Raspberry Pi 3," Electromaker, 02-Apr-2019. [Online]. Available: <https://www.electromaker.io/blog/article/raspberry-pi-3-vs-libre-computer-roc-rk3328-cc>. [Accessed: 08-Apr-2020].
- [16] "Camera Module," Camera Module - Raspberry Pi Documentation, 15-Oct-2019. [Online]. Available: <https://www.raspberrypi.org/documentation/hardware/camera/>. [Accessed: 08-Apr-2020].
- [17] "Logitech C310 HD Webcam," 960-000585 B&H Photo Video, 09-Mar-2019. [Online]. Available: https://www.bhphotovideo.com/c/product/709599-REG/Logitech_960_000585_C310_HD_Webcam.html. [Accessed: 08-Apr-2020].
- [18] "GoPro Hero3 Specs," CNET, 16-Nov-2019. [Online]. Available: <https://www.cnet.com/products/gopro-hero3/specs/>. [Accessed: 08-Apr-2020].
- [19] Buy a Pi NoIR Camera V2 – Raspberry Pi, 16-Feb-2020. [Online]. Available: <https://www.raspberrypi.org/products/pi-noir-camera-v2/>. [Accessed: 08-Apr-2020].

- [20] “2.7inch E-Ink display HAT for Raspberry Pi & Arduino red/black/white 3 color e paper,” Seeed Studio, 09-Jun-2019. [Online]. Available: <https://www.seeedstudio.com/2-7inch-E-Ink-display-HAT-for-Raspberry-Pi-Arduino-red-black-white-3-color-e-paper-p-3094.html>. [Accessed: 08-Apr-2020].
- [21] “Speaker 8R 1W,” Cytron Technologies Malaysia, 21-Aug-2018. [Online]. Available: https://my.cytron.io/p-speaker-8r-1w?r=1&gclid=Cj0KCQjwz4z3BRCgARIsAES_OVdNQTWpScd0LWfYeTpS7Zwh4bmnFtzf_v4GWq0VItCrafxeIjiC2AaAqkHEALw_wcB. [Accessed: 08-Apr-2020].
- [22] C. C. Mart'in Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen et al., ‘Large-Scale Machine Learning on Heterogeneous Distributed Systems’, Netw. Comput. Neural Syst., 2005, doi: 10.1080/09548980500300507.
- [23] D. Kuhlman, ‘A Python Book: Beginning Python, Advanced Python, and Python Exercises’, Book, 2012.



Appendix B – Coding of training dataset for traffic sign images

```
model {
  ssd {
    num_classes: 22
    image_resizer {
      fixed_shape_resizer {
        height: 640
        width: 640
      }
    }
  }
  feature_extractor {
    type: "ssd_mobilenet_v2_fpn_keras"
    depth_multiplier: 1.0
    min_depth: 16
    conv_hyperparams {
      regularizer {
        l2_regularizer {
          weight: 4e-05
        }
      }
      initializer {
        random_normal_initializer {
          mean: 0.0
          stddev: 0.01
        }
      }
    }
    activation: RELU_6
    batch_norm {
      decay: 0.997
      scale: true
      epsilon: 0.001
    }
  }
}
```



```

use_depthwise: true
override_base_feature_extractor_hyperparams: true
fpn {
  min_level: 3
  max_level: 7
  additional_layer_depth: 128
}
}
box_coder {
  faster_rcnn_box_coder {
    y_scale: 10.0
    x_scale: 10.0
    height_scale: 5.0
    width_scale: 5.0
  }
}
matcher {
  argmax_matcher {
    matched_threshold: 0.5
    unmatched_threshold: 0.5
    ignore_thresholds: false
    negatives_lower_than_unmatched: true
    force_match_for_each_row: true
    use_matmul_gather: true
  }
}
similarity_calculator {
  iou_similarity {

```



```

box_predictor {
  weight_shared_convolutional_box_predictor {
    conv_hyperparams {
      regularizer {
        l2_regularizer {
          weight: 4e-05
        }
      }
    }
    initializer {
      random_normal_initializer {
        mean: 0.0
        stddev: 0.01
      }
    }
    activation: RELU_6
    batch_norm {
      decay: 0.997
      scale: true
      epsilon: 0.001
    }
  }
  depth: 128
  num_layers_before_predictor: 4
  kernel_size: 3
  class_prediction_bias_init: -4.6
  share_prediction_tower: true
  use_depthwise: true
}

```

اونیورسیتی تکنیکل مالیزیا ملاکا

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```

anchor_generator {
  multiscale_anchor_generator {
    min_level: 3
    max_level: 7
    anchor_scale: 4.0
    aspect_ratios: 1.0
    aspect_ratios: 2.0
    aspect_ratios: 0.5
    scales_per_octave: 2
  }
}
post_processing {
  batch_non_max_suppression {
    score_threshold: 1e-08
    iou_threshold: 0.6
    max_detections_per_class: 100
    max_total_detections: 100
    use_static_shapes: false
  }
  score_converter: SIGMOID
}
normalize_loss_by_num_matches: true
loss {
  localization_loss {
    weighted_smooth_l1 {
    }
  }
  classification_loss {
    weighted_sigmoid_focal {
      gamma: 2.0
      alpha: 0.25
    }
  }
}

```



```
    classification_weight: 1.0
    localization_weight: 1.0
  }
  encode_background_as_zeros: true
  normalize_loc_loss_by_codesize: true
  inplace_batchnorm_update: true
  freeze_batchnorm: false
}
}
train_config {
  batch_size: 4
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
}
data_augmentation_options {
  random_crop_image {
    min_object_covered: 0.0
    min_aspect_ratio: 0.75
    max_aspect_ratio: 3.0
    min_area: 0.75
    max_area: 1.0
    overlap_thresh: 0.0
  }
}
}
```

اونیورسیتی تکنیکل ملیسیا ملاک

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

```

sync_replicas: true
optimizer {
  momentum_optimizer {
    learning_rate {
      cosine_decay_learning_rate {
        learning_rate_base: 0.08
        total_steps: 50000
        warmup_learning_rate: 0.026666
        warmup_steps: 1000
      }
    }
    momentum_optimizer_value: 0.9
  }
  use_moving_average: false
}
fine_tune_checkpoint: "pre-trained-models/ssd_mobilenet_v2_fpnlite_640x640_coco17_tpu-8/checkpoint/ckpt-0"
num_steps: 50000
startup_delay_steps: 0.0
replicas_to_aggregate: 8
max_number_of_boxes: 100
unpad_groundtruth_tensors: false
fine_tune_checkpoint_type: "detection"
fine_tune_checkpoint_version: V2
}
train_input_reader {
  label_map_path: "annotations/label_map.pbtxt"
  tf_record_input_reader {
    input_path: "annotations/train.record"
  }
}

```

```

eval_config {
  metrics_set: "coco_detection_metrics"
  use_moving_averages: false
}
eval_input_reader {
  label_map_path: "annotations/label_map.pbtxt"
  shuffle: false
  num_epochs: 1
  tf_record_input_reader {
    input_path: "annotations/test.record"
  }
}

```

Appendix C – Coding of test traffic sign recognition using Pi Camera

```
import tflite_runtime.interpreter as tflite
import os
import argparse
import cv2
import numpy as np
import sys
import time
from threading import Thread
import importlib.util

class VideoStream:
    """Camera object that controls video streaming from the Picamera"""
    def __init__(self, resolution=(640,480), framerate=30):
        # Initialize the PiCamera and the camera image stream
        self.stream = cv2.VideoCapture(0)
        ret = self.stream.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*'MJPG'))
        ret = self.stream.set(3, resolution[0])
        ret = self.stream.set(4, resolution[1])

        # Read first frame from the stream
        (self.grabbed, self.frame) = self.stream.read()

        # Variable to control when the camera is stopped
        self.stopped = False

    def start(self):
        # Start the thread that reads frames from the video stream
        Thread(target=self.update, args=()).start()
        return self

    def update(self):
        # Keep looping indefinitely until the thread is stopped
        while True:
            # If the camera is stopped, stop the thread
            if self.stopped:
                # Close camera resources
                self.stream.release()
                return

            # Otherwise, grab the next frame from the stream
            (self.grabbed, self.frame) = self.stream.read()
```

```

def read(self):
    # Return the most recent frame
    return self.frame

def stop(self):
    # Indicate that the camera and thread should be stopped
    self.stopped = True

parser = argparse.ArgumentParser()
parser.add_argument('--model', help='Provide the path to the TFLite file, default is models/model.tflite',
                    default='models/model.tflite')
parser.add_argument('--labels', help='Provide the path to the Labels, default is models/labels.txt',
                    default='models/labels.txt')
parser.add_argument('--threshold', help='Minimum confidence threshold for displaying detected objects',
                    default=0.5)
parser.add_argument('--resolution', help='Desired webcam resolution in WxH. If the webcam does not support the resolution entered, errors may occur.',
                    default='1280x720')

args = parser.parse_args()

# PROVIDE PATH TO MODEL DIRECTORY
PATH_TO_MODEL_DIR = args.model

# PROVIDE PATH TO LABEL MAP
PATH_TO_LABELS = args.labels

# PROVIDE THE MINIMUM CONFIDENCE THRESHOLD
MIN_CONF_THRESH = float(args.threshold)

resW, resH = args.resolution.split('x')
imW, imH = int(resW), int(resH)
import time
print('Loading model...', end='')
start_time = time.time()

```



```

# LOAD TFLITE MODEL
interpreter = tflite.Interpreter(model_path=PATH_TO_MODEL_DIR)
# LOAD LABELS
with open(PATH_TO_LABELS, 'r') as f:
    labels = [line.strip() for line in f.readlines()]
end_time = time.time()
elapsed_time = end_time - start_time
print('Done! Took {} seconds'.format(elapsed_time))

interpreter.allocate_tensors()
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

height = input_details[0]['shape'][1]
width = input_details[0]['shape'][2]

floating_model = (input_details[0]['dtype'] == np.float32)

input_mean = 127.5
input_std = 127.5

# Initialize frame rate calculation
frame_rate_calc = 100
freq = cv2.getTickFrequency()
print('Running inference for PiCamera')
# Initialize video stream
videostream = VideoStream(resolution=(imW, imH), framerate=30).start()
time.sleep(1)

#for frame1 in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
while True:
    # Start timer (for calculating frame rate)
    current_count=0
    t1 = cv2.getTickCount()

    # Grab frame from video stream
    frame1 = videostream.read()

```



```

# Acquire frame and resize to expected shape [1xHxWx3]
frame = frame1.copy()
frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
frame_resized = cv2.resize(frame_rgb, (width, height))
input_data = np.expand_dims(frame_resized, axis=0)

# Normalize pixel values if using a floating model (i.e. if model is non-quantized)
if floating_model:
    input_data = (np.float32(input_data) - input_mean) / input_std

# Perform the actual detection by running the model with the image as input
interpreter.set_tensor(input_details[0]['index'],input_data)
interpreter.invoke()

# Retrieve detection results
boxes = interpreter.get_tensor(output_details[0]['index'])[0] # Bounding box coordinates of detected objects
classes = interpreter.get_tensor(output_details[1]['index'])[0] # Class index of detected objects
scores = interpreter.get_tensor(output_details[2]['index'])[0] # Confidence of detected objects
#num = interpreter.get_tensor(output_details[3]['index'])[0] # Total number of detected objects (inaccurate and not needed)

# Loop over all detections and draw detection box if confidence is above minimum threshold
for i in range(len(scores)):
    if ((scores[i] > MIN_CONF_THRESH) and (scores[i] <= 1.0)):#

        # Get bounding box coordinates and draw box
        # Interpreter can return coordinates that are outside of image dimensions, need to force them to be within image using max() and min()
        ymin = int(max(1, (boxes[i][0] * imH)))
        xmin = int(max(1, (boxes[i][1] * imW)))
        ymax = int(min(imH, (boxes[i][2] * imH)))
        xmax = int(min(imW, (boxes[i][3] * imW)))

        cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 2)

```

```

# Draw label
object_name = labels[int(classes[i])] # Look up object name from "labels" array using class index
label = '%s: %d%%' % (object_name, int(scores[i]*100)) # Example: 'person: 72%'
labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.7, 2) # Get font size
label_ymin = max(ymin, labelSize[1] + 10) # Make sure not to draw label too close to top of window
cv2.rectangle(frame, (xmin, label_ymin-labelSize[1]-10), (xmin+labelSize[0], label_ymin+baseLine-10), (255, 255, 255), cv2.FILLED) # Draw white box to put
cv2.putText(frame, label, (xmin, label_ymin-7), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2) # Draw label text
current_count+=1

# Draw framerate in corner of frame
cv2.putText(frame, 'FPS: {0:.2f}'.format(frame_rate_calc), (15,25),cv2.FONT_HERSHEY_SIMPLEX,1, (0,255,55),2,cv2.LINE_AA)
cv2.putText (frame, 'Total Detection Count : ' + str(current_count), (15,65),cv2.FONT_HERSHEY_SIMPLEX,1, (0,255,55),2,cv2.LINE_AA)
# All the results have been drawn on the frame, so it's time to display it.
cv2.imshow('Object Detector', frame)

# Calculate framerate
t2 = cv2.getTickCount()
time1 = (t2-t1)/freq
frame_rate_calc= 100/time1

# Press 'q' to quit
if cv2.waitKey(1) == ord('q'):
    break

# Clean up
cv2.destroyAllWindows()
videostream.stop()
print("Done")

```