**Development Project:**

**Drive AwAI**

**CTEC 3451: Development Project**

**Nathan Lloyd**

**De Montfort University**

**Word Count: 7717**



Contents

[List of Figures 3](#_Toc40623610)

[Acknowledgements 3](#_Toc40623611)

[Abstract 4](#_Toc40623612)

[Introduction 4](#_Toc40623613)

[The Problems & Objectives 4](#_Toc40623614)

[The System & Development 5](#_Toc40623615)

[Development Lifecycle 5](#_Toc40623616)

[Chassis & Electronics 7](#_Toc40623617)

[Suitable GUI 8](#_Toc40623618)

[Security 9](#_Toc40623619)

[Object Detection 10](#_Toc40623620)

[Critical Analysis 16](#_Toc40623621)

[Tools Utilised 16](#_Toc40623622)

[Raspberry Pi 16](#_Toc40623623)

[GitHub 16](#_Toc40623624)

[Integrated development environment’s 16](#_Toc40623625)

[Electronics 17](#_Toc40623626)

[Coral TPU 17](#_Toc40623627)

[Google Collaboratory, TensorFlow & TensorBoard 17](#_Toc40623628)

[Programming Libraries 18](#_Toc40623629)

[VNC 19](#_Toc40623630)

[Problems Faced 19](#_Toc40623631)

[Product Appraisal 19](#_Toc40623632)

[Conclusion 20](#_Toc40623633)

[Future Work 20](#_Toc40623634)

[Bibliography 22](#_Toc40623635)

# List of Figures

[Figure 1: Kanban Board for March to May 5](#_Toc40623636)

[Figure 2: Waterfall General Structure (Kemp and Smith, 2010) 6](#_Toc40623637)

[Figure 3: Acrylic base and 3D printed motor case 7](file:///C:\Users\Nath\OneDrive\Documents\Drive%20AwAI%20second%20deliverable%2017.docx#_Toc40623638)

[Figure 4: Camera Stand Mount 8](#_Toc40623639)

[Figure 5: UI Designs 8](file:///C:\Users\Nath\OneDrive\Documents\Drive%20AwAI%20second%20deliverable%2017.docx#_Toc40623640)

[Figure 6: GUI Prototype 9](#_Toc40623641)

[Figure 7: Explorer-Hat Electronical Layout 9](#_Toc40623642)

[Figure 8: Change password feature 10](#_Toc40623643)

[Figure 9: Model 1’s Inaccurate Detection 11](#_Toc40623644)

[Figure 10: Model 1v2’s Loss (Red: M1, Light Blue: M2) 12](#_Toc40623645)

[Figure 11: Model 3’s Improvement (D-Blue: M1, L-Blue: M2, Green: M3) 13](#_Toc40623646)

[Figure 12: Model’s Loss Compared (D-Blue: M1, L-Blue: M2, Green: M3) 13](#_Toc40623647)

[Figure 13: Model 3’s Distance 14](#_Toc40623648)

[Figure 14: Model 3’s Similar Signs 14](#_Toc40623649)

[Figure 15: Model 3 Confusion Matrix 15](#_Toc40623650)

[Figure 16: Improved Framerate 15](#_Toc40623651)

# Acknowledgements

Throughout this dissertation I have received an incredible amount of support, patience and assistance from friends, family and many of the CEM faculty at De Montfort University; without these qualities shown to me, my final year of undergraduate studies would not have been as enjoyable.

I would first like to thank my supervisor Dr. H. Malekmohamadi, who’s guidance, candidness and support throughout my project has been invaluable. Thank you so much for all the advice you’ve given me along the way.

Similarly, I would also like to thank Dr. A. Khuman for his support and advice through his curriculum and beyond.

I would also like to acknowledge all of the people I met and worked alongside whilst upon my placement at Loughborough Grammar School, with particular note to the Computer Science and Central Services teams for their great support and friendship during my post. Similarly, a big thank you to the great team I worked with at Raspberry Pi. I would particularly like to single our M. Calleja for being an incredible boss and being especially helpful towards my endeavours post role.

I would like to thank some of my past tutors: R. Bunn, L. Feeney, D. Holliday, S. Turner and H. Sohal, for great guidance early on in my education. The skills you built up have undoubtedly contributed to my success. I would also like to thank: A. Minshall, L. Redden and H. Thorpe, for any contributions they’ve made to this project; my design skills are heavily lacking.

Finally, I would like to thank my friends and family for any support or welcomed distractions they’ve provided throughout my studies. Specifically, my partner, ‘editor and chief’ K. Ingram, thank you for all your help; reminding me to add comma’s and to use ‘however’ less. Thank you to my parents who have supported me more so in my final year, funding my ever-growing coffee and takeaway addiction.

# Abstract

This report covers a prototyped semi-autonomous driving system developed using a Raspberry Pi 4 and a Coral USB accelerator for improved tensor computation. The purpose of the system is to provide research to support the use of such technology in modern transportation systems. In parallel, an application will be created to enable manual control as well as including security features; artefacts of which are pivotal to a vehicle. This paper follows on from the previous and discusses this project through its development processes and result.

# Introduction

Vehicles are an integral part to the modern way of living; they are relied upon heavily by both industry and the general public for the main method of travel. Their excessive use day to day generally improves the quality of life for the many people who have access to personal vehicles, there is, however, an infamous drawback to their use. Motor vehicles are recognised as being one of the leading causes of death, the World Health Organization estimating that it accounts for 1.35 million deaths per year; making it the 8th leading cause of mortality (World Health Organization, 2018). This worrisome estimation is enforced by current statistics found within the UK (Department for Transport, 2019). Whilst the supporting literature varies in strength from country to country, the notoriety of vehicle fatalities is clearly a global issue. It is an ever-important consideration to then mitigate the risk involved with motor travel, solutions of which are can be attributed to the incorporation of advance systems; often birthed from the Intelligent Transportation System field.

# The Problems & Objectives

Modern vehicles not only permit travel but include many amenities that improve upon safety, entertainment and comfort. These new features are often a response to key technological advancements, with the addition of machine learning algorithms and remote vehicle control being the most notable. Machine learning has been propelled by the recent advancements in the accessibility of data, improved algorithms (LeCun et al., 1998), as well as the monumental advancements in computational processing (Moore, 1965). The combination of which have enabled the application of machine learning models to be applied within embedded systems, processing live data and returning actionable information that can inform system outputs in real time. A clear benefit of these advancements is within computer vision and deep learning; as they are now being applied within commercial vehicles.

Drive AwAI has been designed to utilise these technologies whilst preserving the core elements found within a standard road vehicle. Whilst automated driving elements are an important aspect of modern vehicles, as seen with Tesla (Tesla.com, 2019) or Waymo (Waymo, 2018), manual control is still a necessity for drivers. As Drive AwAI is not to scale, a remote-controlled approach to the vehicle will be needed.

Although remote control is a necessity within this project, it is also becoming a staple with many modern vehicles possessing cameras and sensors. This feature is often accessed through the vehicles key or a downloadable phone application, both of which enable the user to manoeuvre the vehicle at a slow pace; completing simple functions and improving the user experience. Such connections allow for further integration of personal technologies to improve both control and the entertainment systems within a car; with Fords patent displaying enhanced vehicle control (Steven, 2018).

It is the aim of this project to provide a prototype that reduce the risk of accidents, naturally improving upon safety as a by-product. The project can therefore be decomposed into multiple elements, spanning from the core functionality expected within a vehicle to the complex assisted driving mechanisms within the computer vision model. The core functionality of this project can be broken down further into the: User interface and control, the electronics and vehicle design, and finally the security systems to protect the software and vehicle.

# The System & Development

## Development Lifecycle

The project management of Drive AwAI took many forms during the development lifecycle, the project itself encompassed many disciplines and fields which themselves require specific procedures for successful application. Agile prototyping was employed for a vast majority of the development lifecycle, the short sprints for individual features synergises well with the backlog of tasks. A project backlog can be defined as a list of functional and non-functional requirements, that when put together deliver upon the projects vision (Schwaber, 2004). An agile approach is often implemented within software development, as the pragmatic yet flexible processes involved allow for development to maintain the levels of freedom required when working through features.

The agile manifesto prioritises change over a plan (Fowler and Highsmith, 2001). This factor is befitting of a project in which the sole developer is still gaining the skills necessary for the project's completion. This lack of knowledge at the start of the project would hinder project management methodologies such as waterfall, where extensive planning would not be possible. Instead through an agile approach, the ability to research, build knowledge and develop periodically allows for this knowledge gap to be circumvented entirely.

Through agile methodologies, SCRUM and Kanban, a bastardized combination of the two was implemented to provide guidelines for sprints, backlog management and self-reviews as mock stand-ups. As there was only one developer and no end customer, the requirement for self-reviews were at the forefront of the development. Through these stand-ups and reviews, features could be analysed with empathy to visualise their effect upon end users and their feasibility within the final project.

In addition to these self-reviews, bi-weekly evaluations were undertaken through a project supervisor. Again, due to their being a sole developer with a knowledge gap, evaluations allowed for meaningful feedback and suggestions on what approaches should be taken. Bi-weekly evaluations also followed a similar structure to a stand-up, where problems, progression and objectives were discussed for the next sprints.

Through utilising a Kanban board, features and prerequisite tasks could be managed with ease in a visual format. SCRUM and Kanban differ in how roles are assigned and how features and prioritised and managed, however, due to the development team consisting of one member, these differences within the methodologies can be bypassed. Due to both being derivatives of a dynamic or agile workflow many of the core concepts were similar, but the Kanban board itself is a unique benefit that undoubtedly aided project management, see Figure 1.

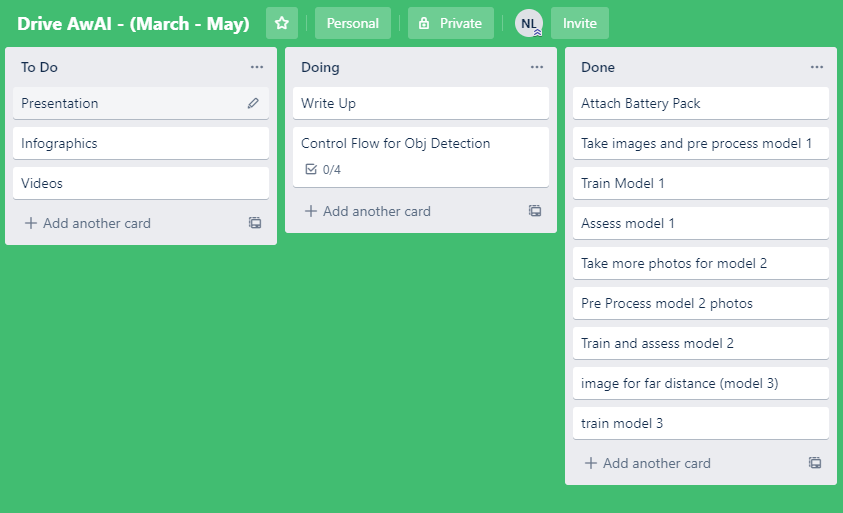


Figure 1: Kanban Board for March to May

Prototyping was used in cohesion with the above methods to rapidly develop single features into a working prototype. A prototype enables testing and evaluation of not only the features present within a build release, but also the methods and processes that went into that build; enabling further self-reflection. An assortment of low fidelity prototypes such as paper sketches and design mock-ups were used in specific areas that required feedback or interdisciplinary collaboration.

On account of the cooperation needed between a product designer and the developer, a waterfall model was adhered to. As the developer this permitted the requirements to be at the focus of the design, enabling a rigorous structure to be conformed to. This methodology is often viewed as outdated within the software development industry due to its linearity (Clear, 2003), nevertheless, as the chassis has requirements that are fixed and clearly specified, a more traditional management approach was deemed necessary; see figure 2.

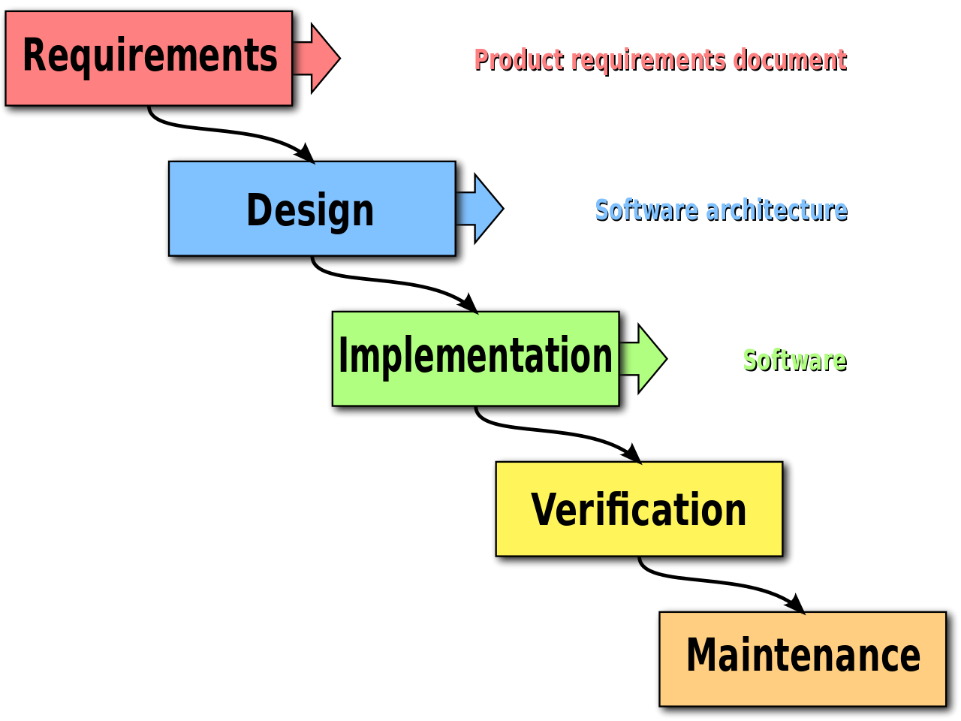


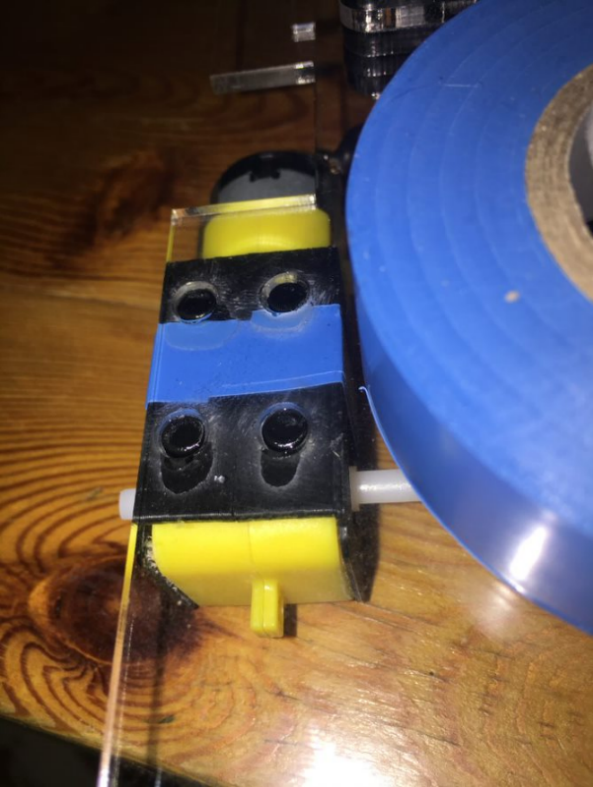
Figure 2: Waterfall General Structure (Kemp and Smith, 2010)

## Chassis & Electronics

Evidentially a feature that builds the foundations to the rest of the project, the chassis has been designed to incorporate all the electronics and devices that are required for successful vehicle control. The chassis is an example of a hybrid between product design and electronics. These two outlooks have ensured simplicity as well as functionality, allowing for the underlying technological systems to be displayed and maintained with ease. The chassis has been designed with the electronical systems in mind, enabling fabrications to be developed in a compartmentalized fashion. This decomposed approach to its development has not only improved the ability to provide suitable project management when outsourcing the model fabrications, but, has also enabled the back and forth discussions during the designing phase to improve the proposed systems further.

The utilisation of a clear acrylic base has enabled further presentation of the underlying electronics, whilst providing the suitable rigid structure required within a remote-control vehicle. The component specific elements have been incorporated via 3D printed models, keeping costs low and enabling remodelling where needed. The motor case allows for the devices to be fixed to the chassis with ease, providing a functional way to replace broken motors or to re-solder broken connections. See figure 3.

Figure : Acrylic base and 3D printed motor case



The Raspberry Pi is the core electronic component for the Drive AwAI project, its position and mounting within the vehicle is therefore, of utmost importance. The Pi is responsible for running the application, hosting a remote desktop and interfacing with both the USB Accelerator and the ExplorerHat. Its positioning requires suitable access to the PIN system for human interaction whilst not complicating wiring for other electronical components. A clear decision was to incorporate the Pi centrally, as to facilitate all sub-components whilst maintaining an appropriate weight distribution.

The inclusion of a 3D printed camera stand was an important consideration for this project, the camera needed adequate height to ensure that input from the camera stream was suitable. Not only is the camera stream used by the driver within the GUI, but it is also integral to the machine learning algorithm for detecting objects. The higher vantage point provided by the camera stand enables a larger field of view on road ahead, enhancing the detection ability of the computer vision algorithm. Whilst this feature remains incomplete due to the impact of COVID-19, it was an important consideration throughout; see figure 4.

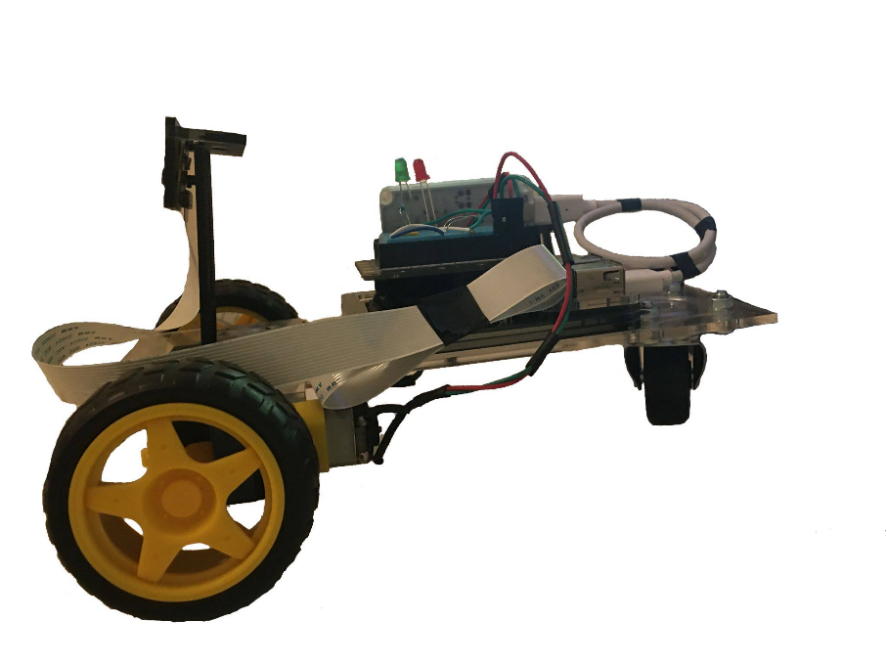
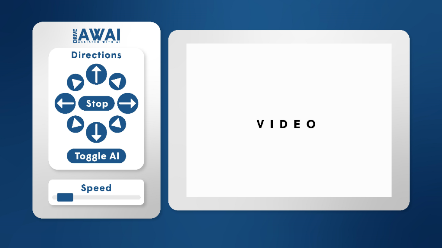
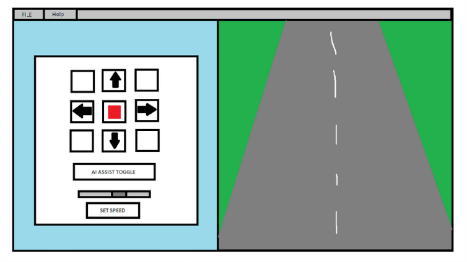
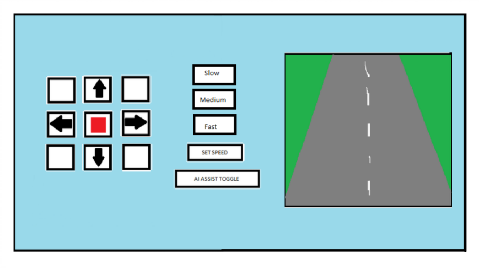


Figure 4: Camera Stand Mount

## Suitable GUI

The Graphical User Interface is a core element to the application, as without a suitable interface, control would not be possible. The interface is comprised of two key elements: motor control and the visual stream; these two components allow for the most basic of control functions to be applied effectively. The first of the two panels, motor control, primarily provides the user with the ability to control the movement of the vehicle via eight directional indicators and one stop button. This range of directions allows for subtle forward and reverse directional changes as well as, on the spot turning; features present in modern vehicles. When designing this interface an important consideration was the ease of use, thus many iterations of designs were created, see figure 5.

Figure : UI Designs



These core design principals have echoed throughout the developmental process, championing functionality and simplicity to ensure a parsimonious system. This clear and minimalist approach to the design of the GUI is both practical and reminiscent of a real vehicle's dashboard display; presenting only the most important features. As well as directional changes, this panel provides the user with the ability to alter the vehicles speed using a scale; allowing for minimal and rapid changes if need be. These two functionalities underpin the systems found in common road vehicles.

The second panel provides the user with the ability to stream the video feed directly from the onboard camera; located at the front of the vehicle. The onboard camera is crucial, as it not only informs the driver of their position in the environment but is also a key component within the computer vision system. This panel has been designed to complement its predecessor, allowing for inputs to be monitored by the user; permitting informed decisions of control. Whilst the prototypes designs have not fulfilled the stylised goal, the theme of function over form and the simplicity for the user are clearly at the forefront, see figure 6.

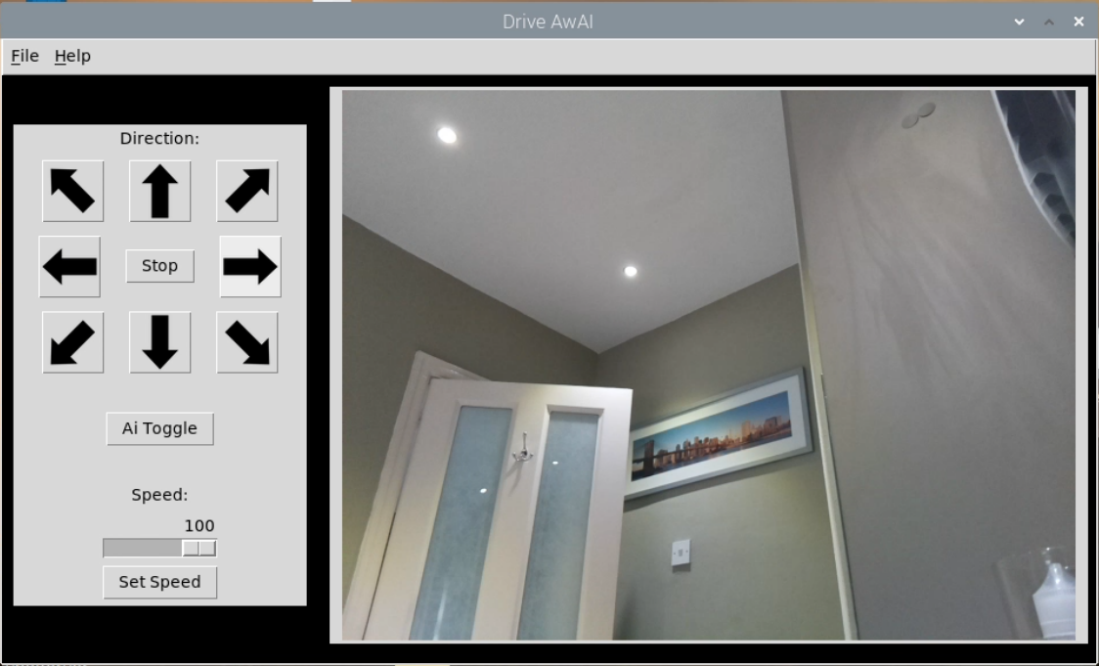


Figure 6: GUI Prototype

## Security

With the: ever-increasing price of vehicles, their ability to store personal items and the expensive underlying technologies, the requirement for an appropriate security system was at the forefront of this prototype. The Drive AwAI project since its inception has understood the need for adequate systems to be incorporated to protect the vehicle and its driver, and so, security has taken shape in many forms. From a security system that has been designed to mirror modern day lock and keys by preventing unauthorised access, to alternate systems which manage access to features such as speed control; ensuring enhanced features are protected behind setting configurations.

At the start of development, the ability to repurpose the motor controller system ‘ExplorerHat’ for use within a PIN system was a clear advantage to developing a secure application. Most PIN systems take four digits and to keep in line with this societal norm a four key pin was implemented; limiting access to the vehicle. Whilst this number could be increased if required, for this initial stage of the prototype and for the purpose of testing, following this accepted level was a natural step. A four-digit PIN provides 10,000 possible combinations and when combined with a counter that provides the user with three attempts before denied access, it was clear the adequate security levels to protect unauthorised access had been implemented, see figure 7.

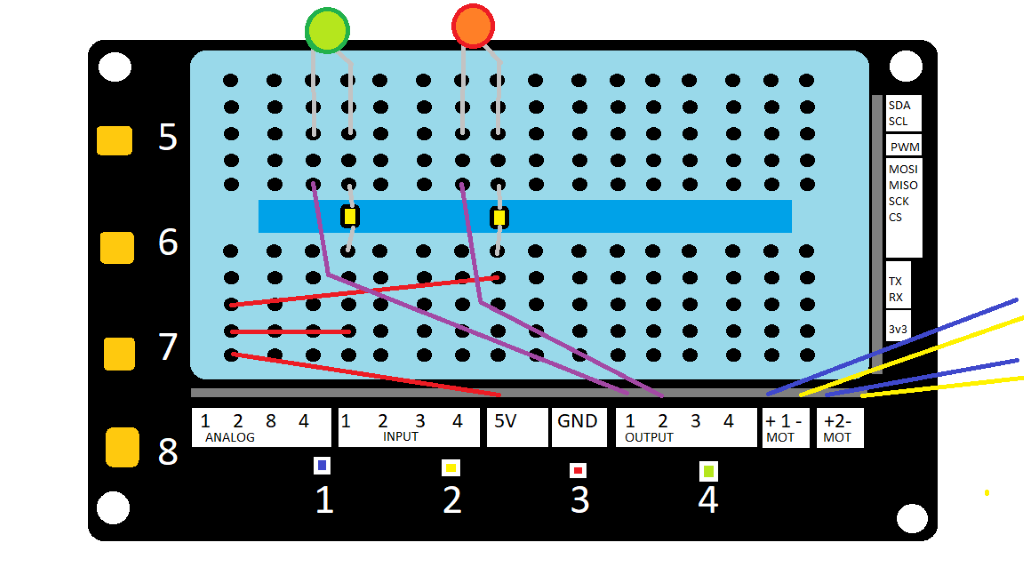


Figure 7: Explorer-Hat Electronical Layout

Although the PIN password provides direct security, at this time the PIN password is stored locally, and so, if attempts were made to access the file in which the password is stored, the user would be able to achieve unauthorised access. To ensure this was not the case, the PIN was hashed and then stored. Upon the applications launch, the hashed password would be compared against the attempted PIN entry; further protecting against unauthorised access. As the password is stored locally, it was imperative that a change password feature be included within the application itself. This feature can be accessed through the menu and caters for bad users by only allowing 4 digits, disabling alternate entries and throwing warning alerts if they are breached; see figure 8.

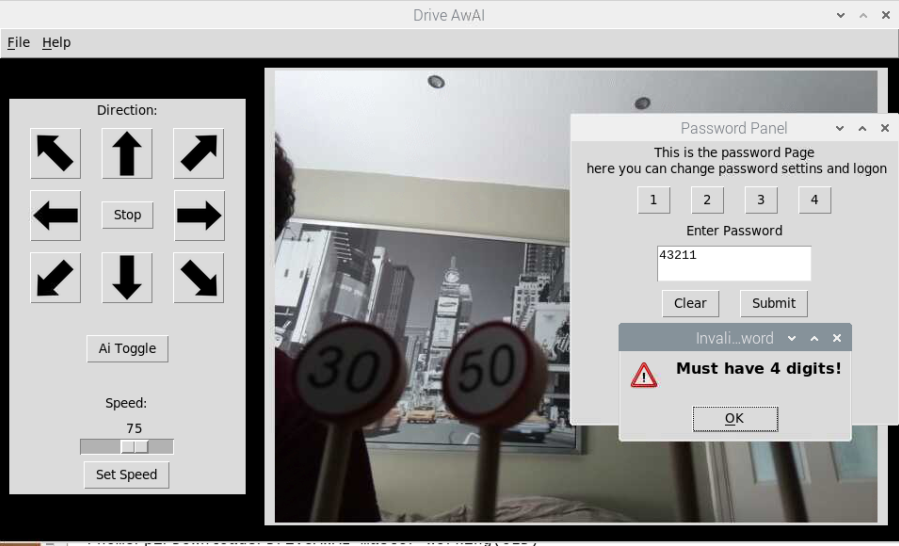


Figure 8: Change password feature

Further security was taken into mind to protect the driver rather than the vehicle. Often vehicles do not cater for the drivers needs or requirements, I.e. providing them access to features they should not have access to according to their ability. Through this application and its menu settings, the driver's access to speed modification can be toggled. The ability to toggle this feature requires the user to re-load the application, subsequently re-entering the password, perhaps again limiting unauthorised modifications of these settings. Whilst these settings can be changed easily if the user holds the password, it is not by design that enhanced features should be hard to change for the driver, instead they are meant to aid a driver who may not have the skill or confidence to operate the vehicle with these features selected.

## Object Detection

A well-known combination for a successful object detection model is MobileNet (Howard et al., 2017), and SSD (Liu et al., 2015). MobileNet is applied as a base convolutional neural network for extracting features such as lines and edges, whereas SSD (Single Shot Detection) is a detection neural network for identifying objects; utilising bounding boxes for detected objects. Each detection block is comprised of 3 branches, box generation, classification and localization and each can be used to calculate ‘goodness’ of a model. Whilst other algorithms were available, ResNet50, YOLO and R-CNN to name a few, this combination is well documented and requires less computational power to run when applying transfer learning (Pradhan, 2019); ideal for embedded devices.

Although these algorithms play a crucial role within the final model, selecting an appropriate dataset for training is known to be more important; garbage in, garbage out (Halevy, Norvig and Pereira, 2009). This predisposition towards better data is in fact due to the data hungry algorithms utilised within deep learning, it is the many parameters that these algorithms encapsulate that demand such a taxing amount of data. As there are many suitable datasets to train from it is important to consider what objects can be found within the dataset.

As the application of the model is to run alongside a vehicle to assist the driver, a dataset containing objects that would be seen on the road was a natural decision. Upon research into available datasets, the COCO dataset was found (Lin et al., 2014). COCO, a well-documented and tested dataset is a highly suited to the task due to its inclusion of many items expected whilst driving upon a road. The COCO dataset, ‘Common Objects in Context’, contains 80 object categories and over 200 thousand labelled images, including: humans, bicycles, cars, motorcycles, trucks, traffic lights and street signs. COCO contains many useful objects for this application, but it is the combination of MobileNet v2, SSD and COCO that ensure a quality model is created.

Speed is clearly an important consideration for this application, when Drive AwAI is run in the real world it could mean the difference between life and death. The Raspberry Pi 4, although much more powerful than its predecessor (Heath, 2019), is still held back within its system on chip architecture. This limitation in its design hinders its processing capabilities; running object detection/recognition at a much lower framerate. What the Pi 4 lacks in power, it more than makes up for it in its ability to interface with additional components. The Pi 4’s USB3.0 ports allow for an improved data transfer speed and when combined with a coral USB Accelerator tensor processing power is in abundance (Coral.ai, 2019).

Despite having a suitable model with training data sourced from COCO, specific nuances such as the difference between road signs have not been captured; requiring the model to be adjusted and expanded upon. This was done via fine tuning, otherwise known as transfer learning. Transfer learning enables knowledge gained from one solution to be applied to another related problem, I.e. recognising a sign object to recognising variations of a sign. To implement this though, new images that represent the objects were in need of collection. Initially only 120 images containing 268 object instances of: a person, stop sign, no entry sign, 30 and 50 signs were collected. These were then labelled, parsed into a CSV file and then generated into a TFRecord. This, in combination with a 90:10 training and validation split produced a working model, but, undesirable one. The model trained for over 4 hours and inevitably struggled to identify the differences between the signs 30 and 50 as well as the no entry and stop signs. This issue was clearly due to the similarities between numerical values and red backgrounds, see figure 9.

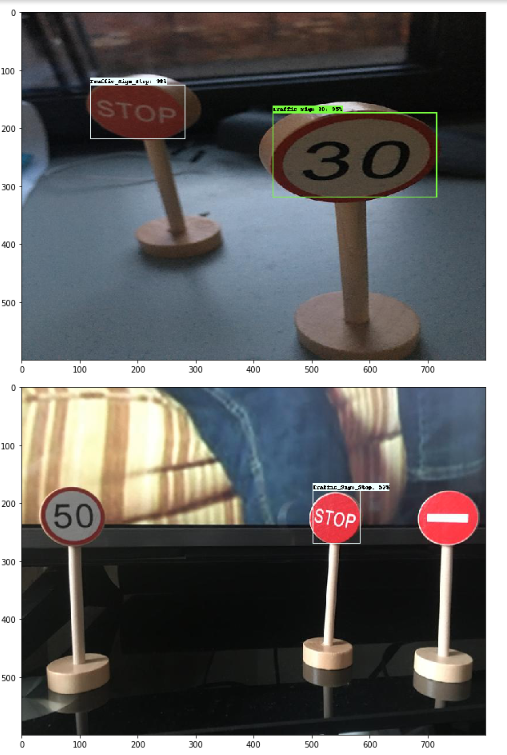
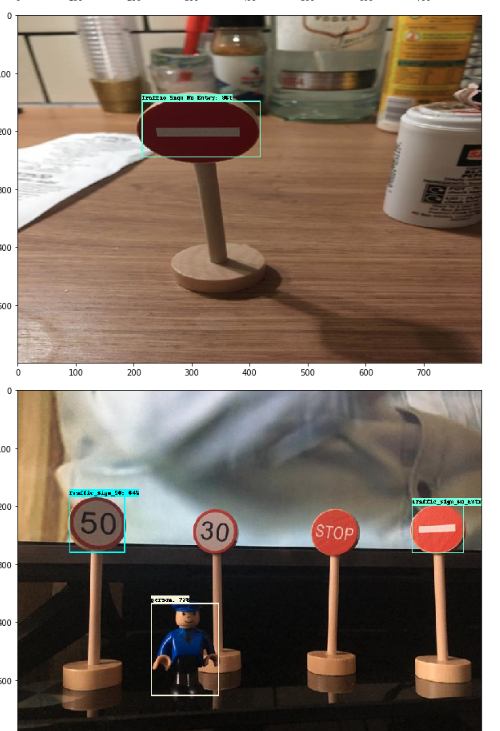


Figure : Model 1’s Inaccurate Detection

This problem only occurred when the similar sign was within the same image. Overall, the training loss decreased steadily, but this issue had left the Total loss (validation) relatively high. Clearly the model performed well in some scenarios but there was room for improvement in others, and as this device has been created to improve safety it was necessary to improve the reliability of the model.

To improve upon the first iteration of the Drive AwAI model, a further 60 images were taken containing another 217 object instances; almost doubling the amount of inputs. This wider dataset also enabled for the validation set to now cover 20% of the total object instances; a more desirable split (Richards, 2020). Furthermore, as the data grew larger the model had more training examples to learn from, and so, it had become more accurate driving down the total loss.

As stated, the detection block is comprised of 3 branches: box generation, classification and localization. The latter two can be summed to find the total loss and it is through this that the model can be assessed. Classification indicates whether the box contained the correct label whereas, localization defines whether the box was displayed on the correct proportions of the frame, the combination of which clearly hold substantial importance for the accuracy of an object detection model.

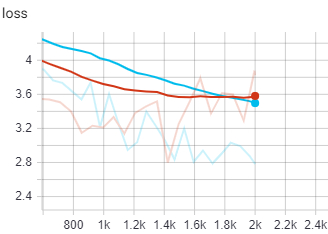


Figure 10: Model 1v2’s Loss (Red: M1, Light Blue: M2)

With the addition of more data within model 2 the loss steadily decreases becoming more accurate than its predecessor, see figure 10 which displays both the actual [faded] and smoothed [bold] results. To improve the system further, live testing was conducted. From these tests, as displayed within OneDrive testing folder: *TPU&Model\_Working\_Poor\_depth*, it was clear that the model still maintained an inability to recognise traffic signs at a distance. This is an issue exacerbated by the lack of images representing this scenario where the signs may be at a further distance from the camera. In addition to this, the second model was pushed for another 600 steps to test the effectiveness of further training and whilst some areas of the model improved, the total loss became erratic and often diverged from the goal of a decreased loss.

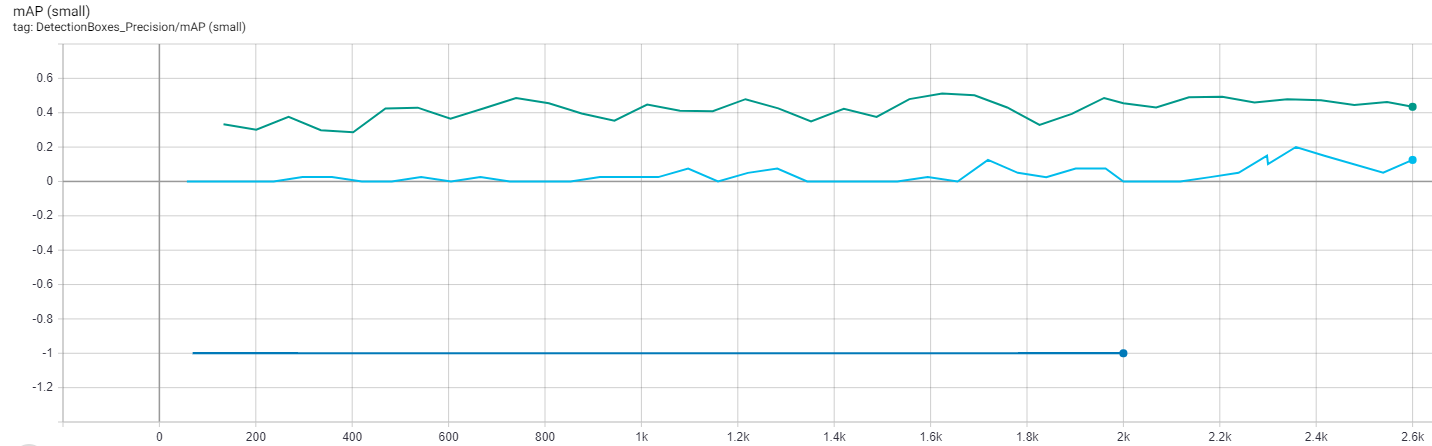


Figure 11: Model 3’s Improvement (D-Blue: M1, L-Blue: M2, Green: M3)

From the findings of models 1 and 2, model 3 was created to rectify any outstanding issues, incorporating another 100 images to again double the labelled objects from 400+ to 800+. This increase not only provides more data for both training and validation but prioritised adding images of signs from a distance. As a result, model 3 saw vast improvements in recall and accuracy for the small detection boxes; see figure 11.

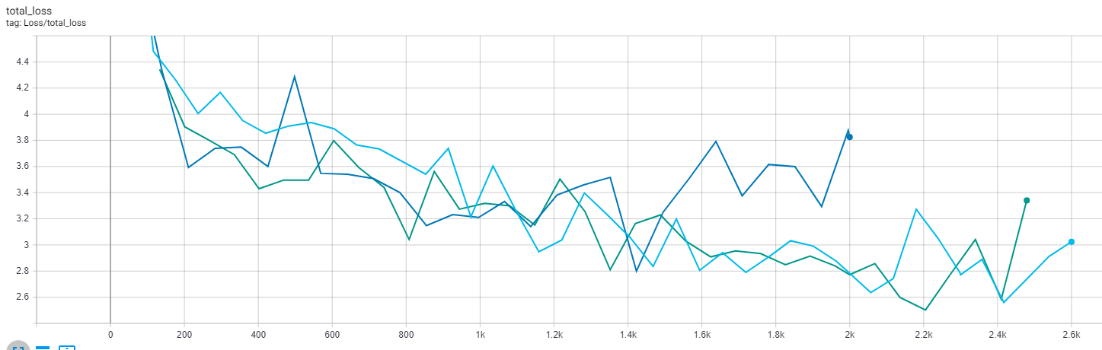


Figure 12: Model’s Loss Compared (D-Blue: M1, L-Blue: M2, Green: M3)

Model 3 was similarly run for an additional 600 steps to ensure that the final model was as effective as possible. As displayed in figure 12, the inclusion of another 200 steps maximized the model's loss before becoming as similarly erratic as model 2. Whilst further training may decrease the loss further, it is important to note that the following erratic behaviour could be a signal of overfitting. The model at 2,200 steps was then converted into an appropriate format for the Coral TPU and a live test was conducted. As expected, due to the inclusion of appropriate images there was a more desirable object recognition model for the traffic signs at a distance; see figure 13. Noticeably, the additional images also improved the confidence of the signs that had similar traits, figure 14 highlighting the model’s ability to identify both solid red and numbered signs with ease; unlike model 1 and figure 9.



Figure 13: Model 3’s Distance

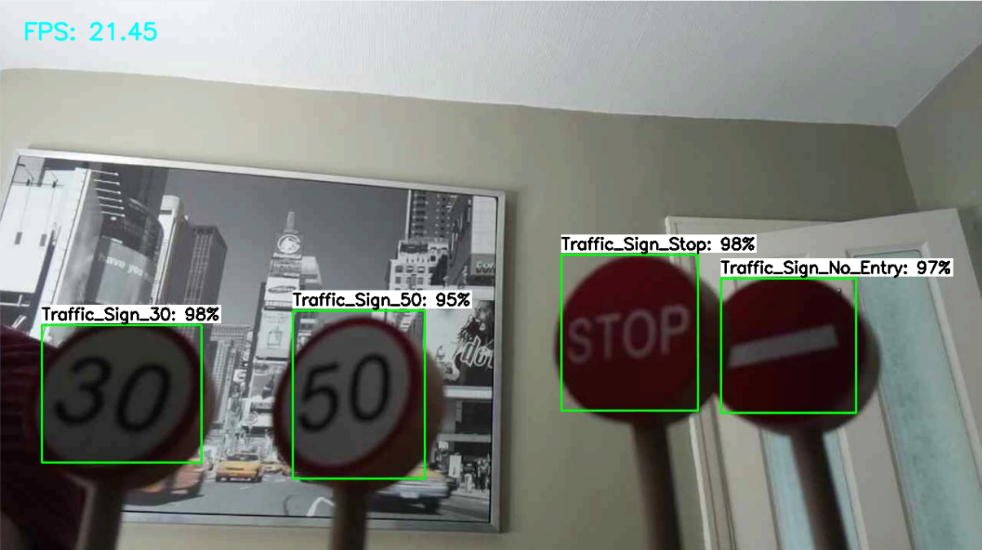


Figure 14: Model 3’s Similar Signs

Figure 15 presents further analysis of model 3, using the test data (20%) and a confusion matrix. A confusion matrix is a tool commonly implemented to analyse the performance of a classification model, allowing for a visual representation of the success of an algorithm. The horizontal rows in this table represent the target values (ground truth) whereas the vertical columns indicate the values which have been predicted by the model. Each row and column represent one of the classes included within the model, this does not provide insights on false negatives, consequently, an additional unclassified column is added to indicate any missed detections.

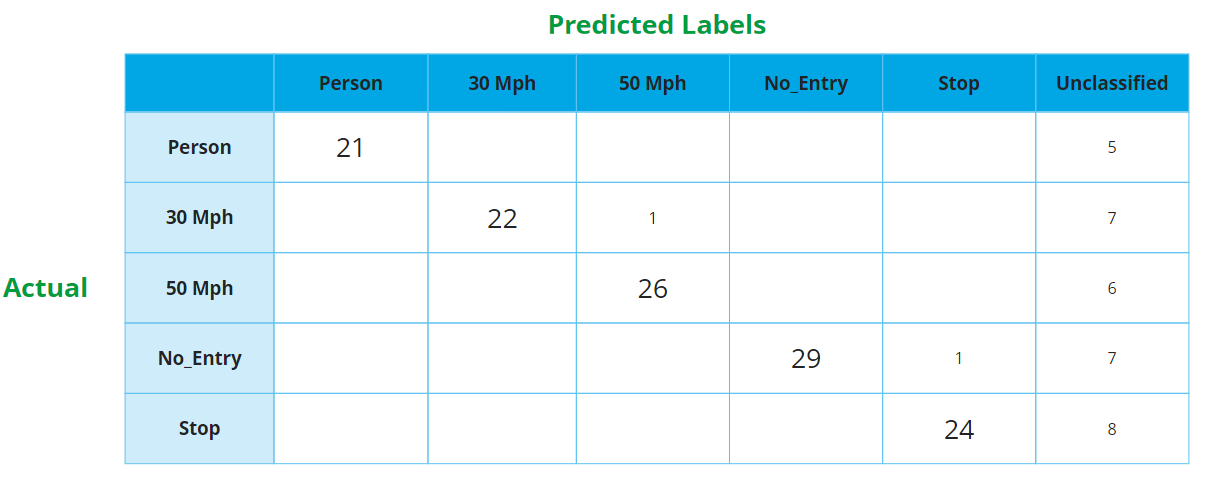


Figure 15: Model 3 Confusion Matrix

The confusion matrix displays that there was a total of 124 predictions, 122 of which were correctly predicted (True Positives); according the model with a 98% precision score. Whilst this presents the model as being successful, this measurement is not a sole indication of the model’s performance, as it does not take into consideration the classes which should have been detected but were missed. These unclassified objects are likely a result of the training data not encapsulating specific aspects that were included within the test data, this could include various angles and lighting differences as well as occluded objects. These false negatives can be used with the false positives to calculate the misclassification rate; an indication of how often the model is wrong. The misclassification rate is 20%, which for final application to real vehicles is not an acceptable score, yet for the purpose of this prototype and its current generation of the model, it shows a large improvement from the examples in figure 9.

Finally, the model is to be applied, recycling the functions created within the RCCar class for dynamic and automated vehicle manoeuvres. Getting the model to interact with the motors is achieved by implementing a control flow statement to observe which class is being detected and respond accordingly. The library used to process the onboard camera feed does not allow for the data to be multithreaded, in response to this, an additional camera was mounted to the vehicle. This solution allows the vehicle a wider field of view for object recognition, a higher definition of footage, as well as, USB3.0 improved data transfer; removing the aforementioned delay and increasing framerate; see figure 16.



Figure 16: Improved Framerate

# Critical Analysis

## Tools Utilised

### Raspberry Pi

Through previous work experience as a Digital Hackathon Facilitator for the Raspberry Pi Foundation, familiarity with the Pi and its Raspbian operating system was to be expected. Although the Raspberry Pi 4 provided new specifications, the hardware and operating system layout were typical of that of previously released models. The main benefits of a Pi 4 to this project was the 4GB of RAM, improved CPU speed and the USB 3.0 upgrade from the previous model. These all vastly effected the performance of the application, whether through improved data transfer and processing or the ability to run much larger tasks.

In addition to this, the Pi 4 supports dual monitors which was an exceptional tool for programming. When not making use of this feature, the Pi could be used as a remote desktop with pre-installed software. This would enable the use of both the desktop and Pi at once; greatly improving development and testing.

The main benefit of a Raspberry Pi is the ability to interface with various technologies. The Pi supports various electronics perfect for prototyping and is the element that handles all individual components. Furthermore, as the Pi handles these components, it also allows you to interact with them in many pre-installed languages. Naturally this project has relied upon python due to the wide support of it within the data science community, it is also the language heavily championed by the Raspberry Pi Foundation (Python - Raspberry Pi Documentation, 2016), consequently, at the forefront of the Pi itself.

### GitHub

Throughout development, GitHub was a primary tool to the management and success of the project. GitHub provided a solution to centralise code and training data over multiple devices; a necessity as development occurred on both desktop and the Raspberry Pi. This ability to manage code over multiple systems allowed for rapid development, testing and deployment from a desktop to the Raspberry Pi. Managing these new features for deployment was done through the inclusion of branches, which are GitHub’s method of adding and managing features without overwriting the master repository.

By implementing branches, the developer can add to the repository without threat of code clashes, thereby elements can be tested in solitude with the rest of the already working system. Through the ability to create branches the project management methodology of agile prototyping has been realised; the branch repository management system has permitted features to be prototyped with ease, rolling back to stable builds if not suitable.

GitHub also allows for the project to be structured in such a way where stable builds can be run out of the box with no modifications to the code. This has enabled speedy testing and deployment of other features; particularly when working with TensorFlow and building datasets. Furthermore, GitHub supports the use of secondary documents that can contain their own markdown, this has been beneficial for read me files as to inform the reader, as well as creating documents that contain prerequisites for pip installs; improving setup installations.

### Integrated development environment’s

The two main IDE’s used were PyCharm and Mu respectively. Due to previous experience writing pythonic code for a Raspberry Pi, initially the project was mainly developed on a personal desktop and pushed to GitHub for testing upon the Raspberry Pi. This choice was simply due to the number of features provided within PyCharm, however, most of the additional options were never utilised fully and the additional processes required moving from a desktop to Pi was a taxing endeavour. To mitigate this, larger features were worked upon through PyCharm and the Pi was accessed via remote desktop to utilise Mu.

Mu is a preinstalled IDE that supports python development and is a development environment previously used during work. A benefit of using the local IDE was that it enabled micro changes to be completed a lot faster than pushing changes through GitHub. Whilst Mu is clearly a more lightweight IDE and does not include many of the features when compared to PyCharm, its slim and simple design allowed for small changes to not be bothered by small clutter.

### Electronics

Throughout physical prototyping many components and tools were used to ensure the electronics worked together as expected. A key tool was a soldering iron for attaching the motors to their respective wires. The experience was relatively simple, however, due to some wear and tear, wires were re-soldered once again once the chassis was assembled; as to ensure the decayed solder did not impact the vehicles functions. To prevent future decay upon the solder, the wires were fitted with shrink wrap to protect it from the elements.

### Coral TPU

Through research, it was clear that the Raspberry Pi did not possess the computational power necessary for a fast response time when implementing a neural network (Allan, 2019). To overcome this issue, the aptly named USB accelerator was purchased. The accelerator is developed by Google and utilises the edge computing paradigm to improve the speed of data computation (Coral.ai, 2019). The TPU used is encompassed within a USB device, the USB itself supports the 3.0 architecture and therefore synergises well with the Pi 4’s recent upgrade to USB 3.0; allowing for the Pi to take full advantage of high data transfer speeds (Spector, L., 2014).

This device provides clear benefits to the project as faster computation naturally improves the safety of the vehicle. Unfortunately, there were minor issues with the installation of TPU and the integration with the Pi 4. Early in development there was a solution to enable the Pi 4 to interface with the accelerator that required an extensive overhead of package installs; this was a tedious process for what should have been a simple installation. This was an issue faced by many, and so, Google produced a method that would support the Pi 4 directly.

Whilst this improved the installation time and removed clutter, this solution was not implemented immediately although the previous solution was no longer supported; leaving a period where no solution would work. For a fortnight, this halted any attempt at progressing the object detection portion of the project, once implemented the benefits were surely noticed when running virtual machines or setting up clone SD cards.

Out-of-date documentation and installation guides were a constant issue for the coral device, upon training a quantized model, which the USB accelerator could support, it was evident that the method to configure the “Tflite” model to an “edgetpu” model was no longer supported. Instead, further steps had to be taken to install and run the edge compiler, which is currently only supported on Linux machines. These updates were clearly designed to make development more efficient for the end user, although for the Drive AwAI project it was unfortunate timing. This was an issue that was echoed throughout the use of the supporting libraries for the TPU, with multiple errors occurring through the application of the ‘libedgetpu’.

### Google Collaboratory, TensorFlow & TensorBoard

TensorFlow has been the main library utilised to train the traffic sign object detection model as it is a free open source research tool created by Google; who also develop the TPU. The requirement to use TensorFlow due to the TPU streamlined the search for an appropriate module to be selected. Whilst this could be argued to be a negative due to the lack of freedom in selecting a library for machine learning, TensorFlow his an incredibly intuitive language.

TensorFlow is an intuitive language as a response to its vast community use. In 2016, Jeff Dean the Google AI lead stated that there are over 1,500 GitHub repositories mentioning TensorFlow and only five of these were official Google Repositories (<https://www.youtube.com/watch?v=Rnm83GqgqPE> @ 730). This wide community acceptance of the library has enabled online learning through various mediums, speeding up access to training of a bespoke model.

Again, through this community acceptance, various blueprint scripts could be modified to support transfer learning of bespoke object detection models. These scripts enable many of the key steps to be automated through Google Collab or Jupyter Notebook’s run all methods, enabling set-up and training to be conducted with ease.

A clear negative of using TensorFlow was the update from 1.x to 2.0. Whilst this upgrade comes with many useful features, such as parsimonious code and smaller functions, it had unfortunately deprecated many of the functions that were used within the blueprint script in use. As training was not conducted locally, this was an incredibly impactful issue as Google Collaboratory automatically uses the most up-to-date modules, consequently, not supporting a roll-back to 1.x due to dependency issues. To remedy this, a roll back of the modules through a “runtime” hack was used to ensure that the script could work as intended.

Once the training data had passed the pre-requisite steps, TensorBoard could be utilised to monitor how the model was responding to the training data over each step. A step is a batch of the total data used and can be used to view the progress over each epoch; a complete cycle of all training data. Through TensorBoard, an assortment of graphs is utilised to display various measurements, such as loss and accuracy. This visualization of data enables easy tracking of the model and how it changes over time. TensorBoard also includes the matplotlib library which enables images to be drawn and annotated as graphs. This essential tool ensures that the models can be assessed with validation data upon the completion of training; allowing the user to see the models “goodness of fit” on unseen data.

Similarly, to using Google Collaboratory for training scripts, TensorBoard allows development to lightweight when compared to downloading the additional modules to a desktop or rather training through the Pi itself. Both methods remove a large installation overhead whilst ensuring the end user can return to the content with ease.

### Programming Libraries

Throughout development many libraries such as Tkinter, OpenCV and ExplorerHat were utilised, each library representing a requirement for an individual element of the project. The first library implemented was the ExplorerHat library which interfaces with its physical components; haptics, motor controllers and I/O channels. This library is at the core of the Drive AwAI project as it enables the implementation of core features through a simple python library. The ExplorerHat (Hardware Attached on Top) is a very simple component to install, as well utilise through menial syntax, that can be implemented in an object orientated style, thereby allowing for testing a development to occur rapidly; as seen through the initial testing within OneDrive: *MotorControl\_T1.MOV.*

In addition to the motor controllers, the haptic sensors supported by the ExplorerHat were employed for a security system, although suitable for a prototype, this security could be improved upon further, consequently hashing was used. Bcrypt was exploited to bolster the security of the application through hashing of stored passwords. Whilst this required the code to be refactored, the final outcome improves security tenfold; a core element within the project contract.

Tkinter is likely the library that possesses much of the code base. Tkinter is essential for the graphical user interface and all subsequent elements, essentially at the core of the project; tying individual elements to one location. It is known to be the de-facto GUI toolkit for python and again comes with wide community support. The GUI itself was designed to be as straight forward and as organised as possible as to not distract the driver and with Tkinter’s grid system this functional layout was easily implemented (Shipman, 2013, pp.5-8).

An issue during development was initially importing the Pi-camera module within a Tkinter element. Typically, when the Pi-camera is activated a new window must be created rather than this element being solely nested within a GUI. For the main application this was not an acceptable solution as it would require manual configuration of the window size and positions for each instance of the application. Again, due to the wide community support of python and its subsequent libraries a method to circumvent this issue was found within the OpenCV library.

Through OpenCV, the Pi-Camera feed could easily be passed into a Tkinter window. Unfortunately, this sidestep was accompanied with a slight input lag as the feed is passed through multiple stages. Whilst this presents a drawback of using OpenCV, it is the only solution that could be found when researching upon the issue. Although this solution is suboptimal, the input lag is minor and the benefits of using OpenCV, such as colour correction, black and white and colour blinds modes could be realised in future applications. OpenCV is also used within the object detection feature, allowing for bounding boxes to be drawn on a visual stream in real time, thus balancing out the negative.

### VNC

Through running both a VNC server and a VNC viewer, a device can be accessed remotely; operating as a graphical desktop sharing system (VNC (Virtual Network Computing) - Raspberry Pi Documentation, 2016). VNC can enable the user to interact with the Raspberry Pi desktop without explicit I/O devices, allowing for keyboard, mouse and touch events to be passed through the VNC server for remote control of the Raspberry Pi. Clearly, this benefits prototype testing of vehicle control, permitting mobile or desktop control of the Raspberry Pi. This freedom from wires within testing is clearly a great benefit, however, there are a few minor negatives to this tool.

As a consequence of the events being processed by the VNC server, input lag is a by-product of the extra layers of computation. Drive AwAI is an application focused upon the user experience and most of all, safety, input lag therefore presents imposes a large negative impact to both. This tool is solely used for prototyping to display the vehicles ability to be controlled, whilst this is a hinderance for the prototype, through further development different methods of control could be implemented to remove this issue.

## Problems Faced

In addition to the issues faced throughout development, COVID-19 also had a substantial impact upon the project. Unfortunately, due to the lockdown put in place by the UK government, the ability to collaborate and fulfil designs for the camera and battery mounts were not possible. This issue leaves the chassis incomplete and a “hacked” approach was used to ensure the remaining components could be added to the vehicle. Although not ideal, this allows for the prototype to be displayed.

On account of the “hacked” togetherness of the chassis, a majority of the weight rests upon the powered wheels; requiring more power for the vehicle to get going. As a result of this excess weight and its poor distribution, the speed selection slider cannot be fully utilised as many of the integers of speed do not possess the power to initially move the vehicle. This almost removes the need for this feature entirely, however, upon further development and access to the missing resources this issue can be mitigated altogether.

## Product Appraisal

A key issue echoed noted throughout is that people spend an ever-increasing amount of time within their vehicles, consequently, issues such as safety, security and simplicity are key aspects expected within a modern vehicle. Through these expectations a baseline of objectives was formed.

AI assistance was at the core of this project as human mortality is an infamous trait to personal vehicles, ensuring that a suitable model was created, of which, progress could be tracked to display the model’s improving accuracy over each generation. The auxiliary features included were to mimic elements such as a lock and key and dashboard options. These secondary features, such as a security system, not only help to develop the user experience but also provide a realistic solution to an important problem.

From this it could be argued that Drive AwAI has not only met the minimum requirements set within the project contract, but for many of its features, development has surpassed initial expectations; through the addition of more complex systems. Through the utilisation of cutting-edge technologies and a flexible management structure the prototype is not only successful but also inspires further research into additional systems, allowing this prototype to become a more accurate representation of a modern vehicle.

# Conclusion

Typically, research conducted within this field has focused upon a single concept, problem or solution, and whilst this allows for the components and systems to be maximised within their scenarios, vehicles by their design encompass many interlocking systems. Although prior research has provided excellent results for individual systems, the vehicles themselves are not represented within their entirety.

The proposed system aims to join these interdependent features into one working prototype. The access to such rapidly developing technologies has allowed for the amalgamation of many libraries, electronics and components to come together to prototype solutions for real world issues. It is therefore the goal of the Drive AwAI project, to encompass the traditional aspects of a vehicle, as well as the features that modern technology has introduced.

The natural progression of technology has allowed for the independent features within this project to work in harmony; often sharing hardware or code. In addition, the research undertaken throughout this project has undoubtedly aided the advancements of the specific systems promised within the project contract; becoming a superior system than initially designed. The consistent testing of the prototyped features also provides a noticeable contribution to the success of the project; displaying the benefits of agile management strategy.

Motor vehicles are the cornerstone to modern travel, the freedom they provide the inhabitants and the time they spend in their vehicles is of extreme importance. Needless to say, the auxiliary features such as security and modifications through settings play a principal part in the user experience. Unfortunately, motor vehicles are notoriously backed by worrisome statistics, showing that there is a high risk to human life.

Road safety and the improvement of vehicles will continue to be an important research domain, developments of which will have a global impact. The ability to not only improve the quality of life for passengers but more importantly develop systems which reduce the risk to human life will maintain to be an important issue within the ITS field. Research suggests that less economically developed countries are more affected by road collisions, and such technologies as traffic sign recognition could be utilised to improve the vehicle for the safety of its inhabitants.

# Future Work

As previously stated, OpenCV allows for many visual features to be customised from the Pi-camera input. The video stream upon display could therefore be altered for particular situations. Key situations could include corrections for the driver, such as colour-blind features that can be selected through the settings within the application. Furthermore, processes could be automated through light sensors or pixel detection to remove sun-glare from the video stream (Paul and Chung, 2018); a feature which could be mirrored to increase the gamma of an image when in low light conditions. Such a feature would boost the accessibility of the vehicle for people with visual impairments; impairments which not only reduce safety but also remove the benefits such as mobility altogether (Suen and Mitchell, 2000).

As object detection was a primary feature within the project, it is therefore important to consider further applications of similar techniques. As seen through Tesla’s recent Autopilot update, a beta version of traffic light recognition has been implemented to automatically slow down vehicles at intersections (Moon, 2020). This echoes the principles of the traffic sign detection found within Drive AwAI, instead developing it further.

Furthermore, another common feature found within Tesla’s Autopilot is the ability to self-navigate between lanes on the road. This feature would undoubtedly assist the driver, improving safety whilst reducing the mental effort required. This assistive technology is a common addition within many intelligent vehicle systems and has wide support within OpenCV (Ye et al., 2010) (Rathore, 2019); making further use of this essential library.

In addition to a wider application of machine learning, various datasets or algorithms could have been used in place of the ones within this project. The German Traffic Sign Recognition Benchmark (Stallkamp et al., 2011), for example, contains over 50,000 traffic sign images with various occlusions and distances to build a well-rounded dataset. As this is a multiclass classification and detection problem, many algorithms could have been used to train an effective model. Although typically used as a binary classifier, a support vector machine could be used for such effect when implemented with a one-vs-all or one-vs-one approach.

From previous electronics experience, implementing an ultrasonic distance sensor could be used to great effect. A UDS could enable an automatic cruise control system; a popular system within intelligent vehicles. Automatic cruise control echoes the core principles of intelligent transport, to bolster safety, improve comfort and increase simplicity (Simões et al., 2016).

Moving away from the improvement of safety towards additional features that improve the user experience is another possibility. Through reiterating the use of machine learning models, further layers of control or security could be added. For example, voice command control could be used as a method for both unlocking the vehicle as well as passing simple directional instructions; features not too dissimilar from pi-controlled home systems (Hidayat and Firmanda, 2015). Biometric authentication and security are ever increasing in popularity and could be used within this system, through taking advantage of facial scans or voice activated verification.

Diverging from the use of machine learning entirely, additional features could be added to improve the prototype. Multiple logins were an initial consideration, with the spare USB operating as a lock that could read a user’s login from a thumb drive key. The combination of a key with a user’s identity, with their locally stored password undoubtedly improves the layers of security.

A final consideration was the method of mobile control. For this project the VNC application was used for remote control of the desktop and subsequent applications of the Raspberry Pi. Through this software the Drive AwAI application could be used from another desktop or a mobile phone, improving the prototypes ability to display its features. As stated, this software could be temperamental and so, specific network control could have been implemented, taking advantage of the onboard Bluetooth or Wi-Fi to provide direct control.

# Bibliography

*Adaptive Cruise Control Stage.* 2019. [Image] Available at: <https://www.bosch-mobility-solutions.com/en/products-and-services/passenger-cars-and-light-commercial-vehicles/driver-assistance-systems/adaptive-cruise-control/>

Allan, A., 2019. *Benchmarking Machine Learning On The New Raspberry Pi 4, Model B*. [online] Hackster.io. Available at: <https://www.hackster.io/news/benchmarking-machine-learning-on-the-new-raspberry-pi-4-model-b-88db9304ce4> [Accessed 4 November 2019].

Clear, T., 2003. The waterfall is dead.. long live the waterfall!!. *ACM SIGCSE Bulletin*, *35*(4), pp.13-14.

Coral.ai. (2019). [online] Available at: https://coral.ai/docs/accelerator/datasheet/ [Accessed 8 Jan. 2020].

Department for Transport (2019). Reported road casualties in Great Britain: 2018 annual report. [online] Department for Transport, p.1. Available at: <https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/834585/reported-road-casualties-annual-report-2018.pdf> [Accessed 7 Feb. 2020].

Fowler, M. and Highsmith, J., 2001. The agile manifesto. *Software Development*, *9*(8), pp.28-35.

Halevy, A., Norvig, P. and Pereira, F., 2009. The unreasonable effectiveness of data.

Heath, N. (2019). Raspberry Pi 4 Model B review: This board really can replace your PC. [online] TechRepublic. Available at: https://www.techrepublic.com/article/raspberry-pi-4-model-b-review-this-board-really-can-replace-your-pc/ [Accessed 20 Nov. 2019].

Hidayat, S. and Firmanda, S.F., 2015, May. Scheduler and voice recognition on home automation control system. In *2015 3rd International Conference on Information and Communication Technology (ICoICT)* (pp. 150-155). IEEE.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Kemp, P. and Smith, P., 2010. *Waterfall Model Of System Development*. [Digital].

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), pp.2278-2324.

Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C.L., 2014, September. Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740-755). Springer, Cham.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. and Berg, A.C., 2016, October. Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham.  
Moore, G., 1965. Moore’s law. *Electronics Magazine*, *38*(8), p.114.

Moon, M., 2020. *Engadget Is Now A Part Of Verizon Media*. [online] Engadget.com. Available at: <https://www.engadget.com/tesla-traffic-light-stop-sign-autopilot-183724262.html?guccounter=1&guce\_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce\_referrer\_sig=AQAAAFmKdwFg0G6COqv4wtHbvhgTiTrSd8unRv27pRxsuJVNuYv016un720sT1zcFUt\_Y9D8RroULcUNafSwrJCmjE26dBgCgq7xY1o1woDcqpr5zK9Rjq5Taz9SNPvMDDqAGNemTE2\_NCGGRkuvFqoHF3JVL0yNr09Kj94nMdjO041-> [Accessed 4 May 2020].

Paul, N. and Chung, C., 2018. Application of HDR algorithms to solve direct sunlight problems when autonomous vehicles using machine vision systems are driving into sun. *Computers in Industry*, *98*, pp.192-196.

Pradhan, M., 2019. *Image Classification Using Mobilenet In The Browser*. [online] Medium. Available at: <https://medium.com/analytics-vidhya/image-classification-using-mobilenet-in-the-browser-b69f2f57abf> [Accessed 1 May 2020].

Raspberrypi.org. 2016. *Python - Raspberry Pi Documentation*. [online] Available at: <https://www.raspberrypi.org/documentation/usage/python/> [Accessed 4 May 2020].

Raspberrypi.org. 2016. *VNC (Virtual Network Computing) - Raspberry Pi Documentation*. [online] Available at: <https://www.raspberrypi.org/documentation/remote-access/vnc/> [Accessed 4 May 2020].

Rathore, A.S., 2019. Lane Detection for Autonomous Vehicles using OpenCV Library.

Richards, J., 2020. *Finally: Why We Use An 80/20 Split For Training And Test Data Plus An Alternative Method (Oh Yes…)*. [online] Medium. Available at: <https://towardsdatascience.com/finally-why-we-use-an-80-20-split-for-training-and-test-data-plus-an-alternative-method-oh-yes-edc77e96295d> [Accessed 1 May 2020].

Schwaber, K., 2004. *Agile project management with Scrum*. Microsoft press.

Shipman, J.W., 2013. Tkinter 8.4 reference: a GUI for Python. *New Mexico Tech Computer Center*.

Simões, J., Jaco, D., Gomes, R., Araujo, P., Fernandes, A. and Seabra, E., 2016. Evolution of the cruise control. In International Conference on Regional Triple Helix Dynamics (HELIX2016). Instituto Politécnico de Castelo Branco (IPCB).

Spector, L., 2014. USB 3.0 speed: real and imagined. PC World, 26.

Stallkamp, J., Schlipsing, M., Salmen, J. and Igel, C., 2011, July. The German traffic sign recognition benchmark: a multi-class classification competition. In *The 2011 international joint conference on neural networks* (pp. 1453-1460). IEEE.

Steven, R., Ford Global Technologies LLC, 2018. *Non-autonomous steering modes*. U.S. Patent 10,081,387.

Suen, S.L. and Mitchell, C.G.B., 2000. Accessible transportation and mobility. *Transportation in the New Millennium*.

Tesla.com. (2019). Autopilot. [online] Available at: <https://www.tesla.com/en_GB/autopilot> [Accessed 25 Nov. 2019].

Waymo. 2018. Home – Waymo. [online] Available at: <https://waymo.com/> [Accessed 30 April 2020].

Who.int. (2019). WHO | World Health Organization. [online] Available at: <https://www.who.int/ith/other_health_risks/injuries_violence/en/> [Accessed 05 Jan. 2020].

Ye, W., Yuetian, S., Yunhe, X., Shu, W. and Yuchen, Z., 2010, December. The implementation of lane detective based on OpenCV. In *2010 Second WRI Global Congress on Intelligent Systems* (Vol. 3, pp. 278-281). IEEE.