TALLINN UNIVERSITY OF TECHNOLOGY

MASTER THESIS

Development of a cybersecurity evaluation test bed for autonomous self-driving vehicles

ANDREW ROBERTS

TALTECH PRESS

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Software Science

The dissertation was accepted for the defence of the degree of Master of Science (cybersecurity) on 4 August 2020

Supervisor: Prof. Dr. Olaf Maennel
Department of Software Science, School of Information Technologies
Tallinn University of Technology
Tallinn, Estonia

Defence of the thesis: 17 August 2020, Tallinn

Declaration:

I hereby certify that I am the sole author of this thesis and this thesis has not been presented for examination or submitted for defence anywhere else. All used materials, references to the literature and work of others have been cited.

Andrew Roberts ______________________

signature

Copyright: Andrew Roberts, 2020

TALLINNA TEHNIKAÜLIKOOL

Magistritöö

XX/202X

Isesõitvate autode küberturvalisuse hindamiseks loodava katsekeskonna arendamine

ANDREW ROBERTS

KIRJASTUS

Abstract

Autonomous self-driving vehicles crash because there is a lack of rigorous testing of their systems and autonomous cognition. Threats from cyber attacks, which have been proven on legacy vehicle architectures, present a fundamental challenge to the safety and security of autonomous self-driving vehicles, their passengers and pedestrians in the driving environment. There is lack of testing for cybersecurity of autonomous self-driving vehicles. Existing processes support testing in simulators which are unrealistic and scope-limited and real-world operational vehicles which are costly and resource intensive. For autonomous self-driving vehicles to operate in real-world traffic they need to ensure to the public, safety and security from cyber threats. To resolve this problem, this thesis develops a low-cost, small-factor autonomous self-driving vehicle test bed for cybersecurity testing. Evaluation of the test bed is conducted through applied practical experiments using real-world cyber threat test cases contributed by experts from autonomous system designers, operators and component providers. The results of the evaluation demonstrated that a low-cost, small-factor test bed can support cybersecurity testing of real-world threats against sensors and perception, communication channels and hardware and compute. These findings can be used to improve the defensive mechanisms of autonomous vehicles in areas such as the Robotic Operating System (ROS) communication, network intrusion detection and monitoring and the resiliency of autonomous cognition. However, limitations in the small-factor test bed design were identified in the lack of computational resources to support on-board training and processing of neural networks and inability to include the diverse profile of vehicular electronic components. This thesis emphasises the need for autonomous self-driving vehicle operators to utilise small-factor test beds that can emulate the systems and functionality of their operational vehicles to improve cybersecurity testing and ensure the public of safe and secure autonomous transportation.

This thesis is written in English and is 110 pages long, including 5 chapters, 78 figures and 4 tables.

Annotatsioon

Isesõitvad autod satuvad liiklusõnnetustesse, sest nende süsteeme ja isemõtlemist ei testita piisavalt. On tõestatud, et küberründed ohustavad ka tava-autode süsteeme, seetõttu on põhiline väljakutse, mida on vaja ületada, isesõitvate autode, nendega reisijate, jalakäijate ning üldise liikluskeskkonna ohutuse tagamine. Autonoomsete isesõitvate autode küberturbetestimist ei tehta piisavalt. Hetkel sooritatakse antud teste kasutades kas simulaatoreid, mille tulemused on ebarealistlikud või mille testimise mastaap on limiteeritud, või kasutades reaalseid autosid, mis aga on kallis ning ressursimahukas. Isesõitva auto igapäevases liikluses kasutamiseks peab ühiskond olema kaitstud autot mõjutavate küberohtude eest. Selle saavutamiseks on antud magistritöös välja töötatud odava maksumusega ning väikesemõõtmeline katsekeskkond küberturbe testide läbiviimiseks. Keskkonna väljatöötamisel on sooritatud mitmeid katseid, kasutades reaalelulisi küberrünnakuid vastavalt isesõitvate autode ekspertide, süsteemidisanerite, operaatorite ning komponentide tootjate poolsele sisendile. Testimise tulemused näitavad, et odava maksumusega väikesemõõtmeline katsekeskkond on piisav selleks, et testida reaalelulisi küberrünnakuid, mis on sooritatud sensorite taju, sidekanalite, riist- ning tarkvara vastu. Neid tulemusi kasutades on võimalik parandada isesõitvate autode kaitsevõimet eri valdkondades: robotiarendusplatvormi kommunikatsioonis, võrgu sissetungituvastuses ja monitooringus ning autonoomse taju vastupidavuses. Piiravateks asjaoludeks väikesemõõtmelise katsekeskkonna korral olid isesõitva auto arvutusressursi puudus, mis oli vajalik pardal toimuvaks neurovõrkude töötlemiseks ja väljaõpetamiseks, ning võimetus kaasata auto elektroonikakomponentide laia valikut. Käesolev magistritöö rõhutab isesõitvate autode tootjate poolse väikesemõõtmeliste katsekeskkondade, mis suudavad jäljendada töötavate autode süsteeme ja funktsionaalsust, kasutamise vajadust, et parandada küberturbe testimist ja tagada avalikkusele turvaline ja ohutu autonoomne transport.

Magistritöö on kirjutatud inglise keeles, on 110 lehekülge pikk, koosneb 5 peatükist, sisaldab 78 joonist ning 4 tabelit.

Acknowledgements

“Midagi pole võimatu, niipea kui inimene hakkab sellest kord tõsiselt mõtlema.” [1] - Anton Hansen Tammsaare

I would specifically like to thank Prof. Dr. Olaf Maennel for supervising my academic journey of the last 2 years. I am grateful for the opportunities he has given me, the wisdom he has imparted and the patience and understanding he has shown.

I’d like to acknowledge Prof. Dr. Tobias Eggendorfer and Hochschule Ravensburg-Weingarten for hosting me for the Erasmus + internship and for providing access to facilities. I’d specifically like to thank Prof. Dr. Eggendorfer who was able to arrange that opportunity at short notice and his support and understanding during the COVID-19 events. I’d like to thank Barbara Wildenhain, International Coordination Officer of RWU, for assisting me when i was unwell in Germany.

I’d like to thank MIT CSAIL and the DuckieTown foundation for providing hardware and Dr. Jacopo Tani, ETH Zurich, for ensuring that I got the robotics equipment delivered during COVID-19. I’d like to acknowledge Prof. Liam Paull, University of Montreal, for answering my questions about DuckieTown.

I’d like to thank the ISEAUTO team of Prof. Raivo Sell, Prof. Juhan-Peep Ernits and Ehsan Malayjerdi for providing expert opinion and supporting me with this work.

I’d like to thank the security and safe driving team at Starship Robotics team for providing expert opinion and support. I would like to thank Jaan Priisalu for making that connection possible.

I’d like to thank the autonomous security engineers at ZF for providing expert opinion and always being helpful in motivating me to pursue an active research career in automotive cybersecurity. I’d like also to acknowledge their support in hosting me at ZF in Germany, even though the visit couldn’t occur due to COVID-19.

I’d like to thank my fellow student Nikita Snetkov for teaching me how to use a multimeter and providing me knowledge on robotics.

I’d like to thank the Government and people of Estonia for providing me with a scholarship and opportunities for education. I’d like to thank Prof. Rain Ottis for the opportunity to study at TalTech. It is an honour and a pleasure to be the first Australian to graduate from the Master of cybersecurity program at Tallinn University of Technology/University of Tartu.

Principally, i'd like to thank Martha Jung for continually supporting me and motivating me in my studies and life in Estonia. I doubt whether i would've survived my first Estonian winter without Martha's reassurance and helpful guidance. It's a pleasure to work with Martha and I am a better person for having known her.

List of Abbreviations and Terms

AEB	Autonomous Emergency Braking
FCW	Foward Collision Warning
NTSB	U.S National Transportation Safety Board
CAN	Controller Area Network
AI	Artificial Intelligence
MIT	Massachusetts Institute of Technology
CSAIL	Computer Science and Artificial Intelligence Laboratory
TARA	Threat and Risk Assessment
DSRM	Design Science Research Methodology
LED	Light Emitting Device
GPS	Global Positioning System
ITS	Intelligent Transportation Systems
ECU	Electronic Control Unit
IoT	Internet of Things
DDoS	Distributed Denial of Service
DNN	Deep Neural Network
ML	Machine Learning
ENISA	European Union Agency for Cybersecurity
TPU	Tensor Processing Unit
OTA	Over-the-air
OWASP	The Open Web Application Security Project
OEDR	Object event detection response
SLAM	Simultaneous localisation and mapping
LKAS	Lane-Keeping assistance systems
PBAD	Physics-based anomaly detection
ROS	Robotic Operating System
TPMS	Tire pressure monitoring system
ISO	International Standards Organisation
SAE	Society of Automotive Engineers
ETSI	European Telecommunications Standards Institute
SAE	Society of Automotive Engineers
IEC	International Electrotechnical Commission
BSI	British Standards Institute

Contents

List of Figures

List of Tables

1 Introduction

Autonomous self-driving vehicles represent the future of transportation for modern technologically enhanced cities. The benefits of automation of driving include lower fatality rates from the elimination of human-driver error caused by drink-driving, poor decision making and medical emergencies. The importance of autonomous transportation has been reinforced by the recent occurrence of the COVID-19 pandemic which requires social distancing in passenger transportation and contactless logistics. Ensuring autonomous vehicles are designed with safety and security is of fundamental importance for societal adoption. Cyber resiliency and survivability are key components of safety and security of autonomous vehicles. To ensure the development of autonomous vehicles in real-life traffic scenarios and adoption by society, testing and certification for cybersecurity is essential[2].

1.1 Real-World Problems of Autonomous Transportation Platforms

The last five years has seen an increase in accidents of semi-autonomous and autonomous transportation platforms. The transformation of vehicles from human control to control by algorithms and advanced sensor and perception technology has increased the complexity for autonomous system designers and road traffic authorities [3].

On May 7, 2016, in Florida, a Tesla Model S travelling at 74 mph collided with the the trailer of a truck turning in the opposite direction. The Tesla drove underneath the trailer, tearing the roof off and killing the driver instantly. The Tesla was using a semi-autonomous software mode, "auto-pilot", to assume the driving functions, whilst, allowing manual human intervention. Post-incident analysis by Tesla identified that the object detection algorithm had failed to identify the trailer as an obstacle due the similar colour of the side-panel of the trailer with the lane markings, coloured white. The failure of the automation logic to correctly interpret the images from the sensors affected the object and event detection response (OEDR) and neither the autonomous electronic braking (AEB) or forward collision warning (FCW) were activated[4].

On March 19, 2018, in Tempe, Arizona, a Volvo XC90 sports utility vehicle fitted with a sensing kit and operating in autonomous mode, struck and killed a pedestrian. The vehicle was travelling at 43 miles per hour when it struck the pedestrian, who was crossing with a bicycle at an unmarked crossing. The U.S National Transportation Safety Board (NTSB) investigation found that the radar detected the pedestrian 6 seconds before impact followed by the laser-ranging lidar sensor. The autonomous cognition, however, did not have capability to classify an object as a pedestrian, unless, they were near a crosswalk. As the vehicle approached the pedestrian it's classification of the object switched between a vehicle, bicycle and unknown object. It made a prediction that the object, as a vehicle or bicycle, would move faster than the capability of the pedestrian and as an unknown object it interpreted the pedestrians movement as static. Eventually the harm minimisation of the autonomous cognition reverted control back to the human in the vehicle. The driver was not focused on the driving environment as they inherently trusted in the autonomy to navigate safely. The distracted state of the driver resulted in a delay in regaining situational awareness which resulted in the brakes being applied only after impact[5].

In 2015, cybersecurity researchers, Chris Valasek and Dr. Charlie Miller demonstrated that the internal-vehicle network, controller area network (CAN), of a 2014 Jeep Cherokee could be remotely exploited and used by a malicious cyber adversary to stop or alter the course of the vehicle[6]. This event led to the establishment of test beds for cybersecurity testing of automotive networks, centered on the CAN bus protocol. Autonomous self-driving vehicles offer a more diverse profile of cyber threats as their use of artificial intelligence (AI) with sensor and perception technologies open new attack surfaces and enable new methods for adversarial activity. As with the epoch of automotive test beds of CAN bus, autonomous vehicle test beds which are accessible to smaller vehicle developers and researchers are required for testing and research.

1.2 Research Problem

The FinEst twins project aims to connect the cities of Tallinn and Helsinki through establishment of a shared urban digital architecture. Autonomous self-driving vehicles are a salient part of this aim [7, p.2]. To ensure safety and security of autonomous self-driving vehicles on the roads of Tallinn and Helsinki, the autonomous cognition, the algorithms and sensors that assume the human driving action must be rigorously tested for vulnerability to cyber attacks. Currently, there is limited testing for cybersecurity due to the costs associated with potential damage to an operational vehicle and resources required to supervise the testing and repair systems and components[8].

Test beds such as the Massachusetts Institute of Technology (MIT) Computer Science and Artificial Intelligence Laboratory (CSAIL), DuckieTown, provide a low-cost, small-factor environment accessible to autonomous self-driving vehicle developers and quality assurance testers. These environments, which utilise the same software and network interfaces as vehicles in the FinEst project have the potential to be used for cybersecurity testing and research. [9, p.3]. The research problem this thesis investigates is;

Is it possible for a low-cost, small-factor, autonomous self-driving vehicle test bed to support realistic scenarios for cybersecurity testing?

1.3 Research Questions

The research question this thesis answers is: **How can a low-cost, small-factor, autonomous self-driving test bed be used for cybersecurity testing?**

The thesis also provides insight and answers to several questions:

1. How can a low-cost, small-factor autonomous self-driving vehicle and driving environment be designed?
2. How can cybersecurity testing of a small-factor autonomous self-driving vehicle test bed used to improve cybersecurity of the FinEst autonomous self-driving vehicles.
3. What are the limitations of test beds for autonomous self-driving vehicle cybersecurity testing?
4. Can automation and sensor failures caused by cyber attacks be identified using an experimental test bed?

1.4 Purpose

This thesis seeks to improve the effectiveness of cybersecurity testing of autonomous self-driving vehicles. It seeks to provide a basis for the use of low-cost, small-factor autonomous self-driving vehicle test beds in cybersecurity testing. The wider purpose is to increase safety and security of the autonomous self-driving vehicles.

1.5 Objectives

The primary objectives of this thesis are:

- Development of a low-cost, small-factor test bed for cybersecurity testing of autonomous self-driving systems
- Evaluation of the test bed to understand if it can support cybersecurity test cases from the FinEst project and industry.

The secondary objective is to identify enhancements for test bed environments for security research for autonomous systems.

1.6 Novelty

This thesis provides the first evaluation of a low-cost, small-factor security test bed for autonomous self-driving vehicles. Whilst this thesis was being written, Zelle et al. [8] released a study which detailed a design of a small-factor security test bed for autonomous vehicles. Their paper did not include in the scope, the evaluation of the test bed, the design of the driving environment or an analysis of the autonomous cognition utilised in the test bed. This thesis investigates a solution to a pressing real-world issue, the safety and security of autonomous self-driving vehicles to cyber attacks.

The novelty of this thesis resides in the design of low-cost, small-factor, experimental test bed for cybersecurity testing of autonomous self-driving vehicles and the evaluation using real-world test cases. This is the first evaluation of a small-factor test bed for cybersecurity using applied methods

1.7 Contribution

This research contribution can be defined as a combination of basic science and applied research, as termed by Louis Pasteur, Use-inspired basic research. Figure 1 presents Pasteur Quadrant, it defines use-inspired basic research as research that uses basic scientific research methods, such as that typified by Niels Bohr, with the contribution of producing a tangible artifact, such as those produced by Thomas Edison[10, p.104]. The scientific contribution of this thesis is the applied experimental method for security analysis and evaluation testing of autonomous self-driving vehicles. The primary practical contribution of this thesis is the establishment of a test bed for self-driving vehicle cybersecurity testing that can be applied to the Tallinn to Helsinki smart connected cities [11] and organisations that exist in this ecosystem such as; Starship Technologies and TalTech IseAuto.

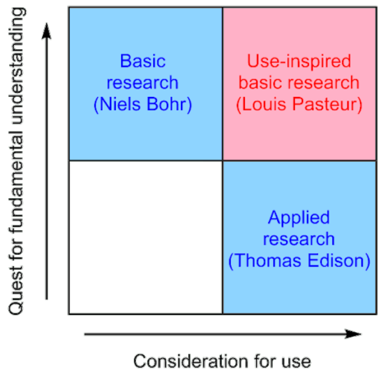


Figure 1 – Pasteur's Quadrant

[12]

1.8 Scope

Two autonomous self-driving vehicles and an autonomous driving city are developed and designed as the test bed. As the objective is to create a low-cost, small-factor test bed, the cost of the robotic components required for the design is less than €1000 and small-factor is defined as able to be fit within a small classroom or laboratory environment.

The focus of the test bed is the ability to replicate autonomous driving cognition, systems and networks of real-world operation autonomous vehicles. The scope of the test bed does not include replicating vehicular components such as an engine or electronics.Also, back-end corporate systems such as customer databases are not included in the scope.

The evaluation of test bed is conducted through security test cases. Test Cases is limited to realistic test cases and methods provided by real-world autonomous driving organisations. A Threat and Risk Assessment (TARA) is not part of the scope of this project, rather, threats are identified and prioritised based on expert opinion. A method for generating test cases for cybersecurity testing of real-world scenarios is included as part of this work.

Evaluation of the test bed is conducted through applied experiments. All experiments were conducted in a controlled environment in TalTech Robotics Laboratory.

1.9 Methodology

This thesis is problem centered. autonomous self-driving vehicles need protection from cyber attacks and there is a lack of cybersecurity testing due to cost, time and resources. To investigate this problem, this thesis designs and develops an artifact, a test bed, and evaluates the ability of the artifact to support the problem definition. The optimal methodology to achieve this is the Design Science Research Methodology (DSRM) as defined by Peffers et al. [13].Figure 2 presents the DSRM as it is applied to this thesis.

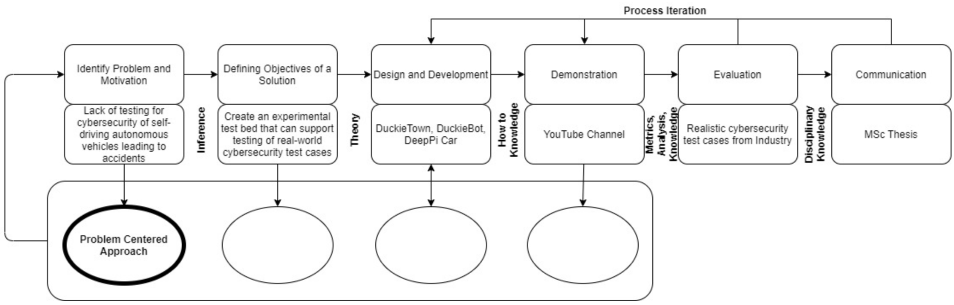


Figure 2 – Design Science Research Methodology - Cybersecurity Evaluation Test Bed

[13, Figure 2]

The knowledge base which supports each phase of the DSRM is as below:

1. **Identification of Problem, Objectives and Design of Test bed**

   Related Work: The related work reviews the existing knowledge of cybersecurity of autonomous self-driving vehicles, standards and test beds.

2. **Demonstration**

   Scientific communication: Demonstration of the testbed environment in workshops and presentations using YouTube and in real-life in the Tallinn University of Technology Robotics Laboratory.

3. **Evaluation**

   Expert Interviews: Interviews with real-world autonomous vehicles operators: Starship Technologies, TalTech IseAuto and ZF. Expert opinion was provided for identification and prioritisation of real-world cybersecurity threats and feedback on the results of the experiment tests.

   Applied experimental methods: Applied experimental testing is conducted on the test bed using cybersecurity test cases.

   Behavioural Observation: Analysis of the applied experiments is conducted using behavioural observation. As the focus of the cybersecurity testing is the autonomous cognition, behavioural observation is a primary method to derive how the vehicle behaviour changed to manipulation by cyber attack.

4. **Communication**

   Scientific communication: Publication of MSc thesis and workshops.

1.10 Limitations

- Experiments were conducted in a controlled environment due to the limitations of supervising resources and length of the process for seeking permission for testing in an outdoor environment with pedestrians.
- The design and development of the small-factor autonomous vehicles were limited due supply chain issues arising from the COVID-19 emergency. Delayed delivery of equipment reduced the in-scope vehicles from 3 to 2.

1.11 Ethics

The related work section contains the ethical considerations for autonomous self-driving vehicles. As applicable to this thesis, the test bed is a controlled environment and testing is able to be controlled to the extent of removing variables that might raise ethical concerns such as collection of personally identifiable information by vehicle sensor and cameras.

This thesis includes security testing. It is possible during the course of the testing to find vulnerabilities in systems used in real-world operational vehicles. This thesis utilised a vulnerability disclosure process, if vulnerabilities of software or systems are found, they will be disclosed to the product owner first.

1.12 Thesis Organisation

The thesis has been organised in the structure of the design science research methodology. There are 5 chapters. Chapter 1 is the introduction which identifies the problem and motivation. Chapter 2 presents the related work. Chapter 3 details the design and development of the autonomous self-driving vehicle test bed and it's demonstration. Chapter 4 presents the evaluation of the testbed using cybersecurity test cases. Chapter 5 contains the conclusions drawn from this research and direction for future research.

2 Related Work

The related work is categorised into three sections:

1. State-of-the-art for cyber attacks on autonomous self-driving vehicles
2. Standards for cybersecurity testing and certification
3. Legal, ethical and social environment for autonomous self-driving vehicles

2.1 State-of-the-art for cyber attacks on autonomous self-driving vehicles

Scientific Literature Reviews:

- **Petit and Shladover**used practical evaluation of a real-world autonomous self-driving vehicle to analyse and prioritise attack surfaces. The highest priority attack surfaces are identified as: GPS spoofing and jamming and camera blinding by infra-red LEDs and lasers [14]. The full list of potential attack surfaces are presented in Appendix 1.
- **Affia et al.** conducted a systematic literature review of scientific papers relating to security risk management in cooperative intelligent transportation systems. The study found a gap in the conduct of analysis of security risks to ITS platforms. The study advocates for approaches to risk management in ITS that considers the connected nature of ITS systems, were, for example, a perception layer attack can inhibit application processes. The study also finds a lack of studies for application security for ITS platforms in contrast to perception and network layer[15].
- **Parkinson et al.** conducted a literature survey with the purpose of presenting a paper-based evaluation of knowledge gaps to autonomous and connected vehicle cybersecurity. The study identified the following high profile cyber challenges:
  - GPS integrity
  - Sensor (IMU, ECU) data integrity
  - Resiliency of LiDAR and camera sensor to cyber manipulation and environmental impacts such as sunlight
  - Human aspects; privacy and ownership of data.

  The study concludes with a list of future research questions that are directed at exploring defensive mechanisms against cyber attacks. These include defence against adversary automation of offensive tools and developing mechanisms for intrusion detection to trigger vehicle processes such as passing control back to the human driver on detection of cyber attack [16].

- **Checkoway et al** [17] investigate attack surfaces for remote exploitation of vehicles. The study argues, existing threat modelling of automotive cyber threats are incomplete as they presuppose access to in-vehicular networks has already achieved. The remote attack surfaces identified in the study are:

  - Direct Access: On-board diagnostics port
  - Indirect Access: Telematics unit
  - Short-range wireless: Bluetooth
  - Long-range wireless: WiFi, Cellular

  The threat model in the study is practically evaluated using test case scenarios. The study concluded, in threat modelling, the importance of connected, end-to-end attacks, for instance; A CD with a malicious firmware is uploaded into the vehicular telematics unit which provides remote access to an in-vehicular network.

- **Thing & Wu** propose a taxonomy of cyber attacks and defences against autonomous vehicles. The proposed taxonomy, derived from literature review, categorises cyber attacks against autonomous vehicles as being either physical attacks (side-channels, code modification, code injection) or remote attacks (signal spoofing, jamming). Defensive mechanisms are categorised as passive detection of attack, response to attack such as isolation of systems and active defence which includes security monitoring. The study also proposes that adaptive security such as cyber deception (honeypots), will become a prevalent option for autonomous vehicles [18].

- **Meryem & Mazri** categorises cyber attacks against autonomous self-driving vehicles as either attacks which impact the control of the vehicle or passive attacks. Their classification model prioritises spoofing and jamming attacks against the vehicular sensors, LiDAR and camera, as the highest risk. The study also identifies low-level sensors and IoT devices in the smart city environment as a feasible attack surface for spoofing, blinding attacks as well potential entry points for network communication attacks[19].

- **Ren et al.** [20] provide a systematic study of security threats to autonomous vehicles. The study categorises two threat profiles; existing threats and new threats. Existing threats are denoted as:

  - Sensor attacks: Jamming and spoofing

- Passive keyless entry and ignition manipulation: Jamming, relay, replay and cryptographic analysis.
- Voice controllable systems: manipulation using machine learning.
- Vehicular networks: Spoofing, DDoS.

New threats are categorised in the study as threats to the deep neural network (DNN) from adversarial machine learning(ML), leakage of ML training models and manipulation of ML compute components such as an edge Tensor Processing Unit (TPU). The study proposes defensive mechanisms such as multi-sensor fusion, sensor redundancy and implementation of cryptographic protocols for secure communication.

Grey Literature Review Survey:

- **ENISA** published a guide for security of smart cars. They used a methodology of expert interviews and literature review to determine the state-of-the-art for cyber attacks and requirements for defence-by-design of automotive systems[21]. The highest rated attacks based on severity are listed as:
  - Communication attacks which block or manipulate in-vehicular network traffic used to send messages to ECUs for vehicular control.
  - Manipulation of open-source maps which support construction of LiDAR maps for navigation.
  - Data leakage from back-end systems such as databases and remote servers.
  - Attacks on mobile applications, especially in car share and rental applications.
  - Rogue Firmware updates and exploiting software over-the-air(OTA) updates[21, p.19].

  ENISA propose over 50 defensive controls to implement to mitigate the risk of cyber attack. These include; cryptography, multi-sensor redundancy, implementing best practice technical standards such as OWASP and ISO27001 for risk management. The smart cars attack scenarios are presented in Appendix 2.

Whilst each reviewed work had different conclusions, the adversarial cyber threats to autonomous vehicles that were omnipresent in all were categorised as threats to:

1. Sensors and Perception: LiDAR, Camera, Radar, Sonar, Neural Networks.

2. Hardware & Compute: Operating system, Vehicle Code, On-board control PC, Embedded components.

3. Connected Vehicle: Vehicle-to-Vehicle (v2v), Vehicle-to-Infrastructure (v2i), Vehicle-to-Everything (v2x), WiFi, In-vehicular networks, IoT networks, Back-end Infrastructure.

Sensors and Perception

Attacks on sensors and perception aim to manipulate the object event detection response (OEDR) and simultaneous localisation and mapping (SLAM) to alter the behaviour of the autonomous vehicle to take an action not expected by the passenger or according to the traffic laws. Adversarial machine learning attacks exploit the reliance that autonomous systems have on machine vision and neural networks[22] [23][24]. As part of traditional model of information security, adversarial machine learning and sensor perturbation impact the integrity of the machine learning training model and sensor data to induce the neural network to alter the driving state of the vehicle.

Eykholt et al. developed an adversarial attack algorithm, Robust Physical Perturbations (RP2), against DNN to generate robust physical adversarial perturbations [22, p.1]. They used a real-world case study of a stop sign to demonstrate that a DNN could be manipulated by their perturbed stop sign to incorrectly classify the object and cause an autonomous vehicle to advance through the stop sign. The results of their laboratory testing were 100 percent success rate for incorrect classification and the field test in a real-world environment generated 84 percent success. The test case used variables of distance, noise and angle to test the perception of the sensor and the DNN[22, p.6-10].

Figure 3 – Physical perturbation of Stop Signs

[22, p.2]

The study is considered a seminal work in adversarial machine learning for autonomous transportation as it demonstrated a low-cost and easy to produce physical attack could manipulate a control system in a more efficient manner than a software or communication attack. The findings of the study motivated algorithm designers to improve the robustness of methods such as object detection using filtering and probabilistic methods[22, p.6-10].

The study also contributes an evaluation methodology which consists of selecting a test case and then experimenting, firstly in a laboratory simulation environment and secondly in a real-world environment[22, p.1].

Sato et al. developed an attack on deep neural DNN based lane-keeping assistance systems (LKAS). The study proposes that an attacker can reverse engineer the logic of a neural network and use the knowledge to design a malicious road patch. Reverse-engineering the driving logic involves gaining an understanding of the driving path, the camera angle and inputs and the predictive behaviour of the vehicle[25].

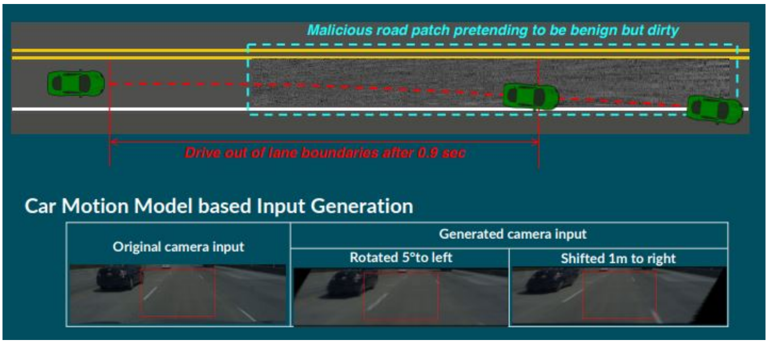


Figure 4 – LKAS Attack

[25, p.3]

The intent of the attacker is to manipulate the logic of the DNN to drive the vehicle off the road or involve the vehicle with a collision with another vehicle. The authors demonstrated the success of the attack on simulators; OpenPilot and LGSVL-1. Figure 4 shows the use of a discreet, perturbed, adversarial road patch that caused the simulated vehicle to drive out of the road lanes. The study is limited as only one test case for LKAS spoofing was evaluated and no real world tests were conducted[25].The study is also limited to the Tesla vehicle which do not use LiDAR for sensing of the driving environment.

Nassi et al. demonstrated that an attacker could use a projector to project an image on the road that would be recognised by the vehicles camera's as a real object (Figure 5). In their experiment, the Mobile 630 Pro camera and Tesla Model X with Hardware 2.5 detected and perceived the projected image as a physical object and took driving actions such as swerving, braking and accelerating. The study also demonstrated mobility use-cases by engineering a drone to carry a projector to project images on the road. The control variables for the experiment were that the projection needed to occur at nighttime and and the projector needed to be within close proximity of the projection surface. Lighting of the environment and attenuation of the projection image impact the success of the attack [23].

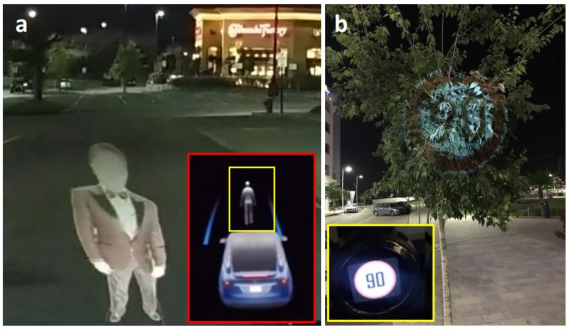


Figure 5 – Spoofed/Phantom Image Attack

[23, p.5]

The experiments were limited to Tesla vehicles and in some experimental test cases the radar, rather than the camera, detected the projection image as an obstacle. The study contributed a machine-learning solution to detect projection images as spoofs, however, as the attack was only trialled on Tesla, which use a customised object detection algorithm, it is left to conjecture whether this attack would work on other object detection algorithms[23].

Attackers can mount remote attacks on LiDAR and camera sensors using blindsiding, shielding and jamming techniques with infra-red lasers and other noise generating tools. The success of these attacks relies on the attacker understanding the machine learning model and sensory technology in order to exploit their limitations. For different LiDAR models the viewing angle, distance and horizontal angle of the laser beam required for successful manipulation will differ as will the sequence of laser beam flashes. A machine learning model may be trained to ignore messages received from a steady laser beam, however, a dot point laser may successfully inject false sensory data. [24].

Cao et al. evaluated blinding and shielding of LiDAR sensors using laser pointing devices. The experiments used the Apollo Baidu autonomous driving simulator to evaluate adversarial test cases. Real-World autonomous driving data was inputted into the simulator and the laser attack was simulated by inputting LiDAR sensor data in form expected of the laser manipulation. The first test case, an attacker points a laser at the LiDAR sensor to manipulate it to perceiving it as a obstacle, failed. This was due to the angle of the laser and the speed of the vehicle. The speed of the vehicle didn't allow the laser point enough receiving time to be interpreted by the LiDAR sensor nor did the angle of the adversary laser manage to focus on one of the laser points of the LiDAR sensor[24].

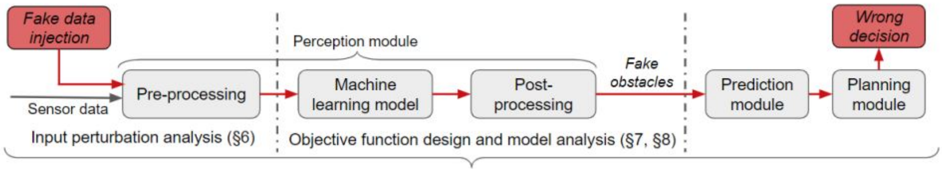


Figure 6 – Overview of the Adversarial-LiDAR methodology.

[24, p.5]

The study developed a successful attack based on creating a spoof 3D point cloud map that is generated by the LiDAR sensor. However, this attack relies on access to the data of the vehicle sensor and understanding of the machine learning model for object detection[24].

Davidson et al. evaluated a sensor input spoofing attack against unmanned aerial vehicles (UAVs). Projectors and lasers are used to spoof the UAVs sensors to input incorrect data in the optical flow to alter the flight path of the UAV. The experiment conducted in the paper, uses a test bed of small factor UAVs operating in diverse environmental conditions; tiles, carpet, concrete and grass. The study found a lack of robustness in the existing optical flow algorithm, the Lukas-Kanade method, which averages flow over all detected images. The vulnerability of the method which Davidson et al. successfully exploits is that Lukas-Kanade method assumes the difference between two consecutive image frames is small and approximately constant within the range. The study develops a new optical flow method, RANSAC, which works by forming a hypothesis of each image, developing a ground truth and assessing each image based on the ground truth [26].

Quinonez et al. [27] propose a new architecture for securing against robust physical invariants caused by attacks such as laser, projector attacks. The study investigates the use of physics-based anomaly detection (PBAD) in control system environments (water, energy, autonomous systems) were cyber attacks impact the physical processes. PBAD works by baselining expected correlations between sensors and actuators and triggering alerts on observation of unexpected behaviour.

The problem this study seeks to solve is stealthy attacks which manipulate the behaviour of control systems below the threshold of detection. The study's contribution is SAVIOR, a PBAD based on the extended kalman filter. The premise of SAVIOR is to train a anomaly detection system with pre-processed sensor data and use an algorithm, extended kalman filter to make predictions of the expected state of the next sensor observation.

To validate the success of their architecture the authors conducted experiments using test cases on a small-factor autonomous self-driving vehicle and drone. The threat model used for the test cases assumed the attackers had full access to the systems in the vehicle and the attack was conducted by uploading sensor data which contained the physical manipulation. The contribution demonstrates the usefulness of physics based anomaly detection for cyber attacks, however, the SAVIOR solution is limited as it will show false positives for environmental impacts to sensors such as wind gushes and rain.

Hardware & Compute

Autonomous self-driving vehicles contain a diverse array of hardware and compute components. This extends from operating systems, middleware, computational hardware and the code base used for operation of the vehicle.

Choi et al investigated vulnerabilities of the robotic operating system (ROS) middleware on a personal robotic system. ROS is used ubiquitously in autonomous systems and robotic platforms including autonomous self-driving vehicles. The vulnerabilities discovered in the study exploited the lack of authentication in the ROS architecture. A robotic platform must execute a number of simultaneous processes in order to achieve a task. To manage these diverse processes the ROS master acts as a central management point. In ROS there is no secure communication. Choi et al demonstrates a variety of exploits including; ROS Master spoofing, intercepting and replaying ROS log files and insertion of malicious robotic processes. The ease of the success of the attacks is assisted by the architecture of ROS having no cryptography and messages are passed in plain-text. The novelty of this research for autonomous self-driving vehicles is that many research development projects such as the Tallinn University of Technology, ISEAUTO also use ROS and so this attack is relevant to the security of those vehicles[28].

Weiss et al. created a model for the characterisation of automotive ransomware. The study conducted a literature review and analysis of automotive ransomware samples to derive common characteristics. The study validated the model using practical methods, implementing a proof-of-concept ransomware in a real car. The properties of automotive ransomware characterised in the study are;

- Self-distribution mechanism to spread through network
- Download functionality
- Infects automotive components
- Impacts vehicle processes
- Persistence
- Encryption of data

- Request of payment

More advanced malware functionality include the ability to protect itself against reverse engineering and countermeasures and controlled infection, which means, if a victim has paid for the decryption, the malware will no longer exist in that system[29, p.6-7] .

The study implemented a ransomware malware on a real vehicle. The initial infection was achieved through manipulation of a firmware update file and the malware was successful in encrypting the data on the real-time operating system of an electronic control unit (ECU) used for vehicle control [29, p.8-9]. The studies relevance to cybersecurity testing is that it demonstrates that malware attacks can be achieved easily and can have severe impact to the operational processes of the vehicle.

Connected Vehicle
Rouf et al. assessed the privacy and security of external network communication interfaces of vehicles. The research problem the study investigated was whether the integration of wireless network connectivity in vehicles had made vehicles more vulnerable to remote exploitation. To investigate this, the authors performed an attack using a software radio platform on a real cars tire pressure monitoring system (TPMS). The attack consisted of monitoring the vehicles networks, capturing it's traffic and then reverse engineering the message id of the protocol used by the TPMS. The outcome of the study was that an attacker, with a software radio attack platform from 40m away from a vehicle, could capture traffic and inject malicious packets causing TPMS update alarms[30].The importance of this study to testing of autonomous self-driving vehicles is that the same scenario can easily translate to a vehicle which utilises more communication interfaces.

Tbatou et al. [31] profiled attacks on communication channels of connected vehicles. The study analysed the attack surfaces of connected vehicles and found a lack of encryption and authentication mechanisms for external communication interfaces. The study recommends the increased use of cryptography to secure internal and external networks of connected vehicles.

2.2 Standards for cybersecurity testing and certification

There are numerous international and national standards for cybersecurity of autonomous vehicles and supporting critical infrastructure. Table 1 lists applicable standards collected in the literature search.

Standards for cybersecurity in Vehicles			
Standardisation Body/Authority	**Country**	**Standard Code**	**Standard**
ISO	International	PAS 21448:2019	Road vehicles — Safety of the intended functionality [32]
ISO	International	26262	Road Vehicles - Functional Safety (Superseded by ISO/PAS 21448:2019) [33]
ISO/SAE	International	DIS 21434	Road vehicles — Cybersecurity engineering [34]
ISO/IEC	International	15408-1:2009	Information technology — Security techniques — Evaluation criteria for IT security (Common Criteria) [35]
SAE	International	J3101	Hardware Protected Security for Ground Vehicles [36]
SAE	International	J3061	Cybersecurity Guidebook for Cyber-Physical Vehicle Systems [37]
ETSI	International	TS 102 940 - 102 943	Intelligent Transport Systems; Security [38]
VDA-QMC	Germany	AK ACSMS	Automotive Cybersecurity Management System Audit [39]
BSI	United Kingdom	PAS 1885:2018	The fundamental principles of automotive cybersecurity [40]
BSI	United Kingdom	PAS 11281:2018	Connected automotive ecosystems. Impact of security on safety. Code of practice [41]

Table 1 – Standards for cybersecurity in Vehicles

From review of each of the standards, automotive cybersecurity is consistently divided into three layers of responsibility;

- Ensuring the protection of the vehicle
- Ensuring secure design, engineering, testing and governance standards of the automaker and automotive suppliers (embedded device manufacturers etc.)
- Ensuring security of service providers such as car service providers (Uber, Bolt).

Automotive cybersecurity standards provide guidance on models, methods and requirements that can be implemented to manage cyber risk. Automakers often combine standards to optimise processes for cybersecurity risk management. Forster et al. provide a new model for including in TARA, inputs from Hazard and Risk Assessment (HARA), using a combination of ISO15408-1:2009 (Common Criteria) standard, EVITA (E-safety vehicle intrusion protected applications) standard and ASIL (Automotive Safety Integrity Level).

This approach recognises the interdependent relationship of security and safety. A cybersecurity incident can affect the safety of the vehicle, whilst, a safety incident can impact the cybersecurity of a vehicle [42].

The interrelationship between standards is visualised in Figure 7 which maps the Forster et al. method.

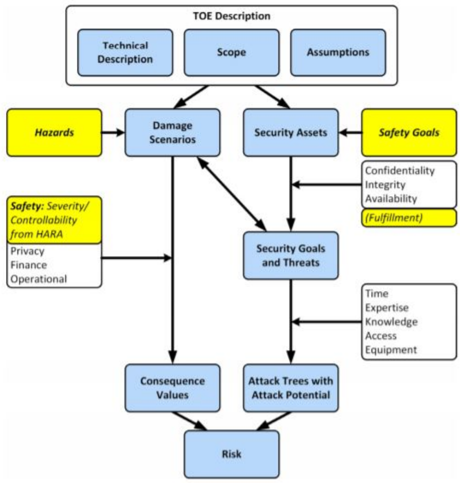


Figure 7 – Forster et al. TARA methodology with integrated safety elements
[42, p.80]

EVITA was an EU project dedicated to establish secure on-board architecture of vehicles through use of hardware and software countermeasures. The project delivered a threat assessment model and hardware security module design which is widely used in industry by ECU designers and on-board hardware and software vendors. As part of the threat assessment model, attack trees are used to visualise security threats and guide security testers on testing efficiency[43].

Vasenov et al. [44] developed a security and privacy threat analysis method for OTA updates in vehicles (Figure 8). The method is novel as it includes the popularly used Microsoft security threat model, STRIDE, with the new, proposed, certification scheme for cybersecurity management systems in vehicles, United Nations Economic Commission for Europe (UNECE) Work Package 29.

	Security	Privacy
Analysis method	STRIDE	LINDDUN
Input	DFD	
	UNECE matrix	
Output	Prioritized security threats	Prioritized privacy threats

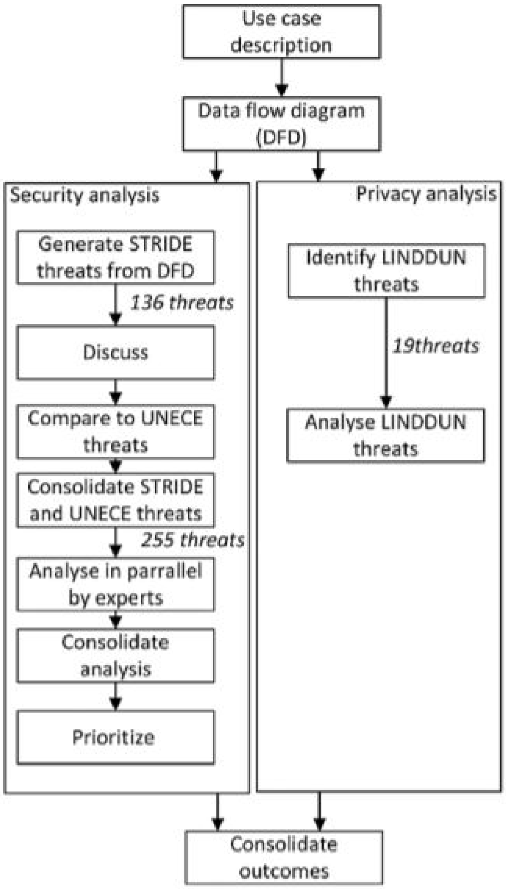


Figure 8 – Security and privacy threat analysis flow

[44, p.3]

The study evaluated the utility of the model in a security assessment of an OTA update of a real car. The found good synergy between the STRIDE threats and the threat catalogue of the UNECE WP29. However, the study noted the limited nature of the security assessment scenario and that further practical evaluation is required to draw more conclusive findings [44, p.6].

The EU SECREDAS (Product Security for Cross Domain Reliable Dependable Automated Systems) project conducted a report of the state-of-the-art for safety, security and privacy analysis and applicability of standards. One of the key products of the report is a survey of the EU automotive industry which details the assessment methodologies in use. Figure 9 demonstrates that the most widely used standards by the automotive industry for security assessment are STRIDE and Common Criteria. OWASP and ISO27009 are also popular due to existing knowledge and expertise of ISO standards and the popular OWASP top 10 for software vulnerabilities[45].

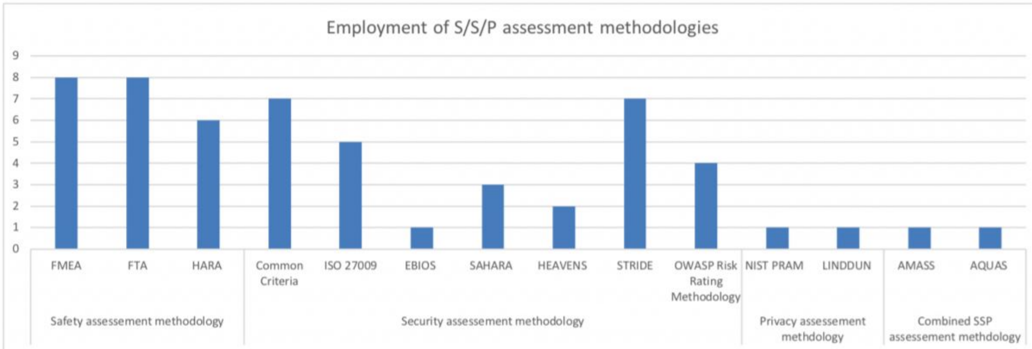


Figure 9 – Usage of Assessment Methodologies - SECREDAS

[45, p.25]

Experimental test beds

Axelsson et al. created a vehicle test bed for security evaluation of cyber physical system. The test bed was based on a small-factor mobile vehicle which was customised to support AUTOSAR, the automotive software standard. The vehicle test bed, developed in 2014, demonstrated that a small-factor device could provide a solution to emulate the protocols and features of a full-factor real-life vehicle. The test bed was not autonomous and relied on remote control by human operator[46].

MIT CSAIL built a low-cost, small-factor autonomous self-driving vehicle for research and development and education. The goal of the MIT CSAIL DuckieBoT vehicle was to build a low-cost option for researchers to evaluate autonomous driving algorithms and explore corner cases. [9].Figure 10 details the features contained in the DuckieBot.

Teams 1st iteration: Features development	Teams 2nd iteration: Behaviors development	Examples of future development
Illumination invariance	Parallel autonomy	Manipulation
LED detector, traffic lights	Object/Vehicle avoidance/following	Inter-bot wireless communication
Odometry calibration from sensor measurements	Traffic light coordination	Model-based control
Lane filtering and control	Stop sign coordination	Vehicle passing
Vehicle detector	Localization and mission planning	Smart infrastructure
AprilTags detection	Robust illumination invariance	Optimal co-design
Local object detection and avoidance		Mobility on demand
Bumpers and shells design and manufacturing	Bumpers and shells design and manufacturing	Safety guarantees

Figure 10 – DuckieBot Features

[9, p.9]

As the aim of the MIT DuckieBot is to provide tangible research contributions to improving autonomy of real-world vehicles, the features and architecture is designed to achieve as close a comparison of real-world autonomous vehicles as possible. This includes the use of ROS which is central to many of the real-world autonomous vehicular architectures. The MIT DuckieBot has never been assessed for cybersecurity testing and research[9].

Tian developed a low-cost autonomous vehicle for research of neural networks. Tian created a code base for a line following car in a low-noise, controlled, test environment. The car was programmed to only follow blue lines and there was no support for curved lane markings. The autonomous vehicle did provide an innovative design in that it overcame the computational resource challenges of the small-factor environment consisting of an onboard computer comprising only a Raspberry pi. Tian's design utilised a Google Coral edge tensor processing unit for accelerated machine learning processing for the object-detection[47].

Zelle et al. built a security test platform for autonomous driving using small-factor autonomous vehicles. The methods used in designing the platform comprised eliciting an attack model of cybersecurity attacks against autonomous vehicles. Based on this attack model the test bed was designed. The test bed is innovative, it includes most of the diverse range of sensors used for perception as well as in-vehicular networks and infotainment systems. Zelle et al. contribution is closest to this work and their paper was released after the development of the test bed contributed in this work. The main differentiation is that this study provides insight into the design of autonomous cognition, evaluation of the test bed for cybersecurity and the driving environment which it is capable of testing[8].

Bhadani et al. created a Cognitive and Autonomous Test (CAT) Vehicle test bed to evaluate autonomous driving. The research problem highlighted in the study was the cost, time and risks of real-world testing and the problems translating test cases from simulators to real-world environments. The study designs and builds a hybrid virtual-physical test bed that incorporates the body physics of a real world vehicle with virtualised sensors and software platforms. ROS is used as the middleware platform. The evaluation of the platform was conducted through an educational program where students used extracted data from the CAT vehicle to improve object detection and tracking[48].

Santos & Schoop developed a framework for cybersecurity testing of autonomous vehicles and evaluated its efficiency through investigation of the survivability of autonomous vehicles after a cyber attack to the vehicles sensors. Their framework consisted of developing test cases and a tool to auto-generate test cases. Their practical evaluation involved the security testing of two sensors; camera and LiDAR. An open-source autonomous driving simulator, CARLA, was used as the experimental testing environment. The authors tool for automatic test case generation only supports CARLA. Their study acknowledges the limitations of this approach, the attack to the sensors was delivered by manual scripts and assumed the attackers could manipulate the sensors perfectly each time. The findings are limited to the CARLA environment and the simulation environment testing couldn't replicate a real-world physical attack or the operational driving domain of the vehicle[49].

2.3 Legal, Ethical and Social Environment for autonomous self-driving Vehicles

The foundations for current nation-state regulation of vehicles is based on the Vienna Convention on Road Traffic 1968. Article 8 of the convention establishes: *"every moving vehicle or combination of vehicles shall have a driver"*[50, p.11]. A driver is defined as: *"Driver" means any person who drives a motor vehicle or other vehicle (including a cycle), or who guides cattle, singly or in herds, or flocks, or draught, pack or saddle animals on a road"*[50, p.6]. The driver is responsible for control of the vehicle and obeying the 'rules of the road'. The rules of the road are defined as the regulation of behaviour for actions such as: position on the carriage way (Article 10), Overtaking and movement of traffic in lines (Article 11), Passing of oncoming traffic (Article 12), Speed and distance between vehicles (Article 13)[50, p.7-15]. The convention contains 55 Articles and 5 Chapters which comprehensively detail every aspect from the positioning of flocks and herds on the road to rules for international driver permits[50].

Estonia acceded to the Convention on Road Traffic on 24 August 1992. The Estonian national legislation for the regulation of vehicles is the Traffic Act [51]. The Traffic Act has undergone numerous updates to accommodate the introduction of connected and autonomous vehicles for logistics and research and development projects. Within the definitions contained in the 4 July 2017 amendments, a self-driving delivery robot must have a user and a controller that is subject to the same regulations as a driver of a traditional vehicle(section 151, sub-section 2)[51]. This ensures continuity of existing laws where full liability for vehicular crashes is assumed by the "human" driver. This important designation of liability also allows semi-autonomous systems such as the Tesla autopilot to operate in Estonian traffic.

The Traffic Act demonstrates an incremental approach to implementation of autonomous systems into real-life traffic environment. Self-driving delivery robots are limited in speed to 6 km/ph and pedestrians and other vehicles are limited to 20 km/ph in their presence and must take special care and observation to not obfuscate their perception and movements[51]. To test self-driving technologies an operator must obtain registration from the Estonian Road Authority. To obtain registration to operate a self-driving vehicle on Estonian roads an operator must demonstrate performance in a series of tests in closed area and traffic scenarios that include:

1. *how the driver is able to control the vehicle manually*
2. *how a person is enabled to take control of the vehicle from automated mode*
3. *how the vehicle is able to operate autonomously*

These tests are consistent with Estonia's perspective of legal challenges of AI. Estonia's National Artificial Intelligence Strategy 2019-21 expresses that Estonia views AI as performing tasks defined by humans and to the express intention of humans[52]. They will not operate independently and therefore the liability still resides with the human operator. This definition of autonomy is consistent with the EU Guidelines for Trustworthy AI which emphasises human supervised and controlled AI[53].

Autonomous self-driving vehicles rely on sensory and perception technologies to create a 3D map of the environment in order to navigate safely and efficiently. They also rely on the imagery captured by high-definition cameras. The recording and storage of this information will include the physical profiles and activities of pedestrians and other drivers, as well as images of private homes and offices[54].

In Europe, autonomous vehicular architectures need to be designed to process and collect data in accordance with the EU General Data Protection Regulation (GDPR). Autonomous vehicle manufacturers need to ensure data subjects have control of the data that is being collected to allow them to exercise their data subject rights. The challenges for manufacturers is building architectures that allow these data subject rights such as the deletion of data, where, in connected and autonomous vehicles, data is shared over multiple platforms and used to inform safer driving decisions. Innovative solutions to this problem include the CarData portal by BMW which allows BMW customers to view the telematics-data which is stored from their vehicle. Blurring of faces and licences plates captured by the high-definition camera would also provide greater privacy protections for pedestrians and other road users[54].

The UN Economic Commission for Europe (UNECE) has a working party on autonomous and connected vehicles. This working party is focused on Work Package 29, Harmonisation of Vehicle Regulations. Work Package 29 aims to update the existing regulatory frameworks to incorporate the technological transformation of vehicular autonomy and AI. Key priorities include: cybersecurity, Event data recorder(EDR)/Data storage for automated driving (DSAD), Validation method for automated driving, advanced driver assistance systems (ADAS) and dynamics(AEB, FCW)[55].

The working group has produced a draft regulation for the UN for implementation of a certification scheme for cybersecurity and cybersecurity management systems for vehicles. The document acknowledges the crucial role of the manufacturer in providing safe and secure systems which are heavily relied on by self-driving and driver-assisted vehicles. The proposed regulations also acknowledge the increasing amount of personally identifiable information (PII) which is retained in modern vehicles. The draft regulations require a vehicle manufacturer to demonstrate that their cybersecurity management system applies to: development, production and post-production phases. The requirements for certification encompass people, process and technology. A vehicle manufacturer must demonstrate to a certification authority the use of cybersecurity controls such as: cryptographic protocols, intrusion detection systems, forensic logging and monitoring systems, penetration testing and threat and risk documentation. The UNECE WP29 also provides a catalogue of threats to vehicles. This catalogue forms the basis for future certification schemes[55]. The threat catalogue is presented in Appendix 3.

Ethics

Ethics and morality are central to human decision-making and therefore inherent in the design of autonomous systems [56]. From review of the related works, the predominant areas of research for ethics in autonomous driving are identified as:

- Dilemma situations

- Human responsibility for AI
- Privacy of personal information

Ethical engineering approaches use philosophical thought experiments termed; dilemma situations. First introduced in 1967, by Foot, the trolley dilemma consists of a scenario in which a person controlling the lever of a trolley must decide whether to stay on a track which would result in the death of five workmen who cannot escape the path of the trolley, or, change to a side-track which would result in the death of one workman [57].

Wächter et. al conducted an experiment on human decision making using the trolley dilemma in driving scenarios. A select group of people from different age ranges were chosen to confront dilemma situations in a driving simulator. The researchers used behavioural observation and data analysis from the simulator for their research conclusions. The results of the experiment found that the majority of participants would; quantitatively minimise harm, adjust decisions based on age of pedestrian, drive on the sidewalk if it minimised harm, and self-sacrifice themselves to avoid pedestrian fatalities. The conclusions of the study established the difficulty in designing an autonomous system for a subjective area such as ethics. For this reason, the design of autonomous systems should require input from ethics experts[57].

Lin's study of autonomous vehicle ethics conformed to the same themes of ethical debate as Wächter et. al. Lin's study reviews the existing literature and theorises questions still left for debate. One question posed by the study; Is programming an autonomous system, in the example of the trolley dilemma, to hit a pedestrian as a calculation of most ethical action, an ethical and legal conflict for countries whose laws promote the right to life and human dignity? Lin also reflects that crash-optimisation, choosing the least cost of human life, can be interpreted as a form of targeting. The conclusion of the study is that the ethics of autonomous systems are imperfect and open for challenge. Societal expectations need to be based on the reality of the limitations of autonomous systems to improve on human decisions and ethical judgement [58].

Ethical design approaches to autonomous driving include Gerdes & Thornton [56] who translated and applied Asimov's three laws of robotics to autonomous systems:

1. An automated vehicle should not collide with a pedestrian or cyclist.
2. An automated vehicle should not collide with another vehicle, except where avoiding such a collision would conflict with the First Law.
3. An automated vehicle should not collide with any other object in the environment,

except where avoiding such a collision would conflict with the First or Second Law.

4. An automated vehicle must obey traffic laws, except where obeying such laws would conflict with the first three laws.[56, p.95].

What do we value? For Gerdes & Thornton this is a fundamental question for ethics in autonomous systems. The design of algorithms relies on assigning priorities or cost to everything that exists in the driving environment. For instance, in a dilemma situation, if the autonomous vehicle has to chose between impacting a motorcyclist with helmet or without one, do we choose the motorcyclist with the helmet because they have a better chance of surviving or the motorcyclist without a helmet, as they broke the road rules, had been given safety warnings and were negligent?

Gerdes & Thornton also explored the ethical question of hybrid control between human and autonomous system. If an autonomous system is ethically engineered why should a human be able to override the decision making? The conclusion of the study is that with the growing use of autonomous systems we will learn to gain trust in the cognition of machines and adjust our expectations.

The EU high-level expert group on AI defined three essential elements of trustworthy AI:

1. lawful - respect for all applicable laws and regulations
2. ethical - respect for ethical principles and values
3. robust - the technical solution should take into account the social environment[53, p.2].

The German Federal Department of Transport and Digital Infrastructure (BMVI) Ethics Commission on Automated and Connected Driving recommended 20 ethical rules. These rules aimed to resolve dilemma situations by embedding adaptive AI solutions in the city infrastructure and in as many points of the driving environment as possible. Applying the logic of German Ethics Commission to the perspective of the trolley dilemma, the importance of the decision-making of the trolley would be mitigated by decisions made by smart infrastructure on the road, road side-units and mobile devices. The responsibility and accountability for ethical decision-making also shifts from the motorist or person at the trolley lever to the manufacturers and operators of smart city technologies and policy makers[59].

Social

Autonomous Self-Driving vehicles must also confront the ethical concern of privacy. A study by Bloom et al. conducted a survey, in five states in the United States, to quantify comfort levels of the public for autonomous vehicle technology. The survey results concluded that the public had the highest level of discomfort for vehicle technologies that can capture and store images of individuals and track and identify individuals and vehicles. Surveyed members of the public were inclined to accept the use of vehicular technologies for these purposes only if it improved safety or to assist in the investigation of a vehicular incident. The survey results found discomfort from members of the public with being in close proximity to autonomous vehicle sensors, such as walking near them or bicycling near them in traffic. The study recommended engagement between commercial autonomous vehicle companies, regulatory authorities and the public [60].

Reig et al. conducted a survey of 32 pedestrians who have interacted with Uber autonomous vehicles. The survey consisted of structured questions about the pedestrians experience of autonomous vehicles. The results of the survey found that pedestrians had little understanding of autonomous vehicular technology and trust was associated with the branding of the autonomous vehicle manufacturer. Pedestrians, when in the presence of an autonomous vehicle with no human driver, felt that they couldn't understand what decisions the vehicle was making in regards to their presence. The study recommended rectification of this issue through utlising audio or visual alerts to indicate the intent of the autonomous vehicle[61].

2.4 Discussion

From the related work [46][8][22][23][24][62][14],several key factors emerged for the choice of test beds used for cybersecurity testing:

- **Cost** comprises the implementation cost of the test bed, both components and labour.
- **Complexity** is defined as the complexity in designing, engineering and maintaining the test bed.
- **Reliability** is the accuracy of the results to the real-world operational vehicles.

Table 2 show the comparison of each test bed.

	Simulation	Small Factor Test Bed	Real-World
Cost	low	low	High
Complexity	low	medium	High
Reliability	low	unevaluated	High

Table 2 – Factors influencing choice of test bed

The review of the cyber attacks in the literature concluded that simulators provided unreliable and inaccurate results compared to real-world testing[22]. Table 3 presents the comparison of each test bed. As the small-factor test bed was only used in Quinonez et al. [27] study and as such they are unevaluated for security test cases, an informed opinion is made based on the analysis of the designs of Zelle et al. [8], Axelsson et al. [46], MIT DuckieBot[9] and DeepPi car[47].

cybersecurity Test Case	Simulation	Small Factor Test Bed	Real-World
Hardware & Compute Attacks	Yes	Yes	Yes
Connected Vehicle Attacks	Yes	Yes	Yes
Sensor and Perception Attacks	Yes	Yes	Yes
Physical Access	No	Yes	Yes
Damage Incurring	No	Yes	No
Environmental Perturbations	No	Yes	Yes
Full list of Sensors and Systems	No	Yes	Yes
Real-World Driving Environment	No	No	Yes

Table 3 – Comparison of autonomous self-driving test beds for cybersecurity testing

The review of the cybersecurity testing methodologies established the importance of incorporating the UNECE WP29 threat catalogue with an established security threat model such as STRIDE.UNECE WP29 represents the future for certification of vehicular systems for cybersecurity risk management.

3 Design and Development

3.1 Test Bed Concept

The design and development of the small-factor test bed artifact is a key phase of the DSRM. The research entry-point is the problem-centered approach. The research problem this test bed is focused on solving is; *is it possible for a low-cost, small-factor, autonomous self-driving vehicle test bed to support realistic scenarios for cybersecurity testing?*

The predominant elements required in the test bed artifact to resolve this problem are:

1. Emulation of the features of a real-world operational vehicle within a low-cost, small-factor design.

2. Support for realistic cybersecurity test cases.

3.2 Feasibility of Design

The feasibility analysis of design of a low-cost, small-factor test bed consisted of reviewing the TalTech ISEAUTO and the related works for the state-of-the-art for cybersecurity of autonomous self-driving vehicles. The ISEAUTO is a relevant vehicle as it used in the FINEST project and is the target system for realising the benefits of improved cybersecurity. Figure 11 presents the ISEAUTO hardware diagram which lists the sensors and perception technologies, hardware and compute systems and connected vehicle interfaces.

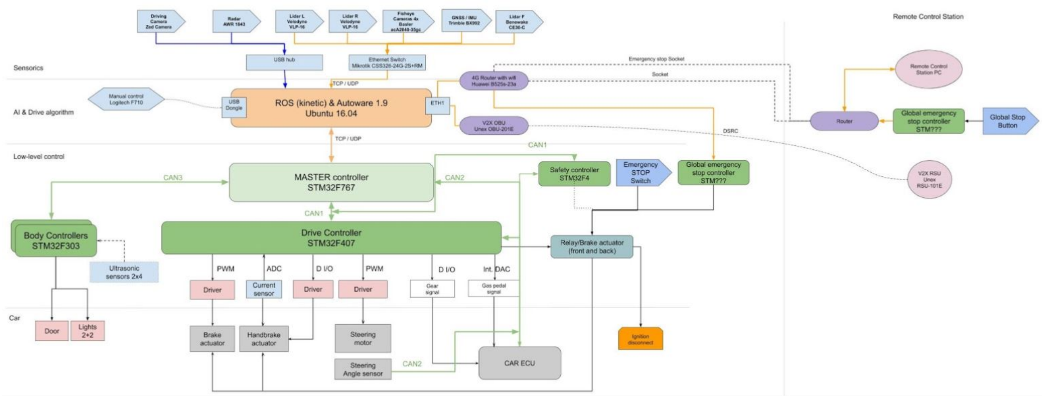


Figure 11 – ISEAUTO Hardware Diagram

[63]

The first consideration for design and development is whether to build from scratch or develop an existing low-cost, small-factor autonomous self-driving vehicle. Based on the related works, the MIT DuckieBot and DeepPi car were chosen to develop as a test bed. The justification for this decision are the comparison of key systems of the ISEAUTO with the MIT DuckieBot and DeepPi car:

- Emulation of key systems:
  - ROS
  - Camera sensor
  - On-board Control PC
  - Network interfaces
  - Actuation (Pulse Width Modulation (PWM))
  - Remote control station PC
- Cost of components under €1000
- Efficient usage of limited computational resources available.

The second consideration is support for realistic scenarios for cybersecurity testing. Based on the related work, Figure 12 lists the cybersecurity testing and research applications the low-cost, small-factor test bed can support.

Security Research Application	MIT DuckieBot	DeepPi Car
Vulnerability Research	Camera Sensor Computer Vision + Image Processing ROS Kinetic Network Interfaces Applications	Camera Sensor Object Detection Python3 WiFi Applications
Impact Analysis	Closed Loop Control Process PID Controller	Closed Loop Control Process PID Controller
Mitigation Research	Secure Middleware Physics Based Anomaly Detection Network Anomaly Detection Application Security	Physics Based Anomaly Detection Network Anomaly Detection Application Security
Metrics	Recovery Time Objectives Recovery Point Objectives Situational Awareness	Recovery Time Objectives Recovery Point Objectives Situational Awareness
Data and Models Development	Security Test Case Evaluation Methods	Security Test Case Evaluation Methods
Security Validation	Test Case Evaluation	Test Case Evaluation
Interoperability	Sensor and Autonomous Drive Cognition Network and Autonomous Drive Platform	Sensor and Autonomous Drive Cognition Network and Autonomous Drive Platform
Digital Forensics	ROS Logs (ROSbags) Video Logs	Video Logs
Operator Training	Remote Operator Console	Remote Operator Console

Figure 12 – Research and testing applications of low-cost, small-factor test bed

[64]

3.3 Low-cost, small-factor test bed for cybersecurity evaluation

3.3.1 Experimental Test Bed Smart City Environment

Duckietown is a man-made environment for autonomous self-driving vehicles created by MIT CSAIL. The Duckietown smart city emulates real-word structures of smart cities by using machine readable road side units (RSU) and road markings. The smart city environment is constructed of two layers; **Floor Layer**, **Signal Layer**.

The **Floor Layer** is where the road markings exist and the road network is mapped. The floor layer is a modular construction consisting of tiles which can be customised to suit different road maps. For the construction of the experimental test bed used in this thesis, 9 tiles were assembled in a 3 x 3 configuration. In the DuckieTown smart city there are three line colours which have their own rules, as per traffic laws; white, yellow, red.

The solid white lines symbolise the road boundaries for which the autonomous self-driving vehicle must remain within. The yellow dashed lines represent the road lanes. Each yellow line piece must be 5cm in length with 2.5cm space between each piece. Red lines are used

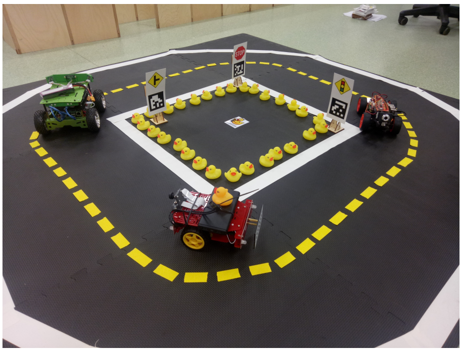


Figure 13 – DuckieTown in TUT Robotics Lab

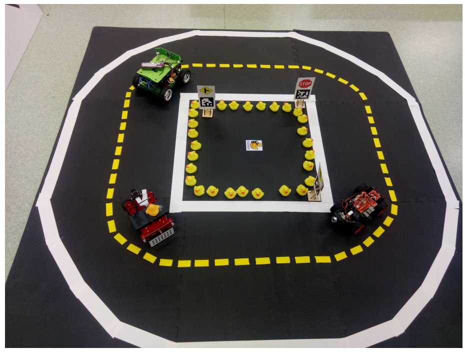


Figure 14 – DuckieTown in TUT Robotics Lab

for stopping a autonomous self-driving vehicle at an intersection.

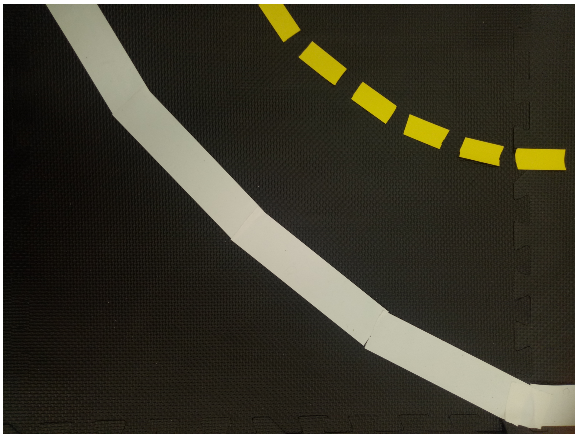


Figure 15 – Floor Tile - DuckieTown

The **Signal Layer** comprises all of the signals that the autonomous self-driving vehicle require for navigation. In the experimental test bed used in this thesis the signals are represented by machine readable RSUs. The RSUs are constructed with a pictorial representation of a road marker used by the image processing of the autonomous self-driving vehicle and a fiducial marker for greater perception.

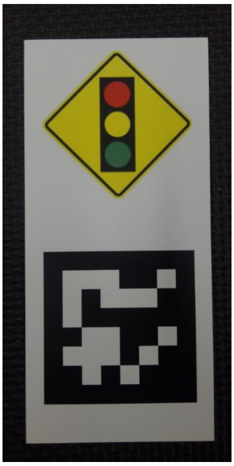


Figure 16 – Traffic Light (Top-Pictoral, Bottom-Fiducial Marker

Figure 17 – Traffic Light - April Tag ID

Wireless networks are used for communication whilst driving in the DuckieTown environment.

3.3.2 Experimental Test Bed Autonomous Self-Driving Vehicles

MIT Duckiebot

The Duckiebot (Figure 18, Figure 19) is a small factor autonomous self-driving vehicle developed by MIT in 2016. The intent of the design of Duckiebot was to create an affordable self-driving platform that researchers and educators could use to teach autonomous systems and evaluate deep learning algorithms for autonomous driving. The cost of the components needed to build the Duckiebot is approximately €250.

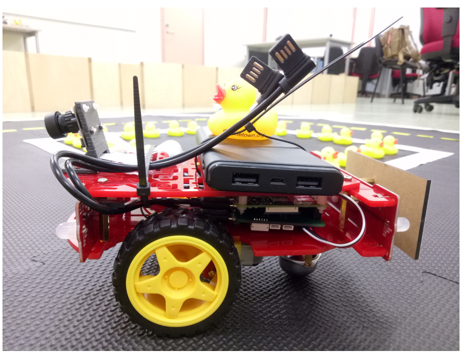


Figure 18 – MIT Duckie Bot - Side View

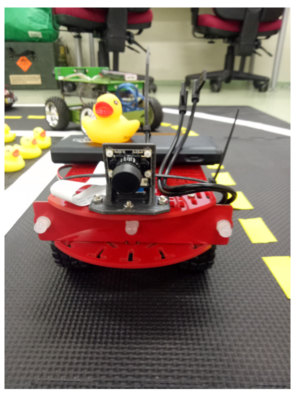


Figure 19 – MIT Duckiebot - Front View

The DuckieBot architecture uses a 5mp pixel raspberry pi camera for sensing. The hardware for the AI and Drive Algorithm is built on Raspberry PI Model 3B hardware. Debian Linux 9 is used for the OS as the Raspberry PI utilises an ARM processor. The software platform is built upon Docker utilising ROS Kinetic. An 32Gb SD is used for local on-board storage and a 100Gb USB drive can be inserted in the Raspberry PI to allow more storage for logging. A 5v, 10400 mAh, battery is used to power the DuckieBot. Actuation is performed by the motor driver which connects to servo motors. The DuckieBot steers in a radial circuit and there is a steel bell underneath that maintains balance.

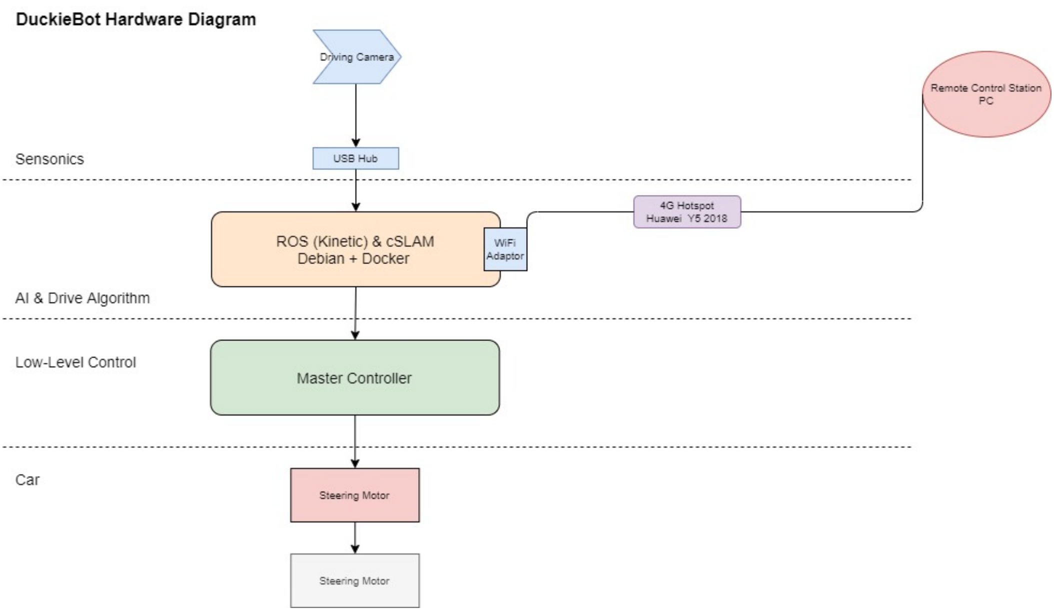


Figure 20 – Duckiebot Hardware Diagram

The code base for autonomous vehicles is highly complex and a commercial autonomous vehicle with a full-sensor profile can reach over 10 millions of lines of code. Autonomous vehicles require numerous operations to be executed in parallel, in the DuckieBot, the lights (led_emitter), autonomous control(joy_node), camera (camera_node), LKAS (line_detector_node) need to all be in simultaneous operation for driving. ROS allows developers to work on code for individual components and operations of the vehicle and centrally manage the execution. Without a centrally managed system it is difficult to troubleshoot, maintain and develop the code base of the autonomous vehicle. In ROS, the ROS Master centrally manages communication between ROS nodes and tracks the messages they are exchanging.The benefits of ROS is efficient code organisation and hardware abstraction.

Figure 21 lists the ROS nodes active during a simple operation, camera footage of the Duckiebot. The rosbridge allows communication of the information from the ROS nodes to be visualised in a dashboard web interface, which in the Duckiebot, is the mission control platform. Figure 22 provides the architecture of the ROS nodes as it would look for another simple operation, stopping the DuckieBot.

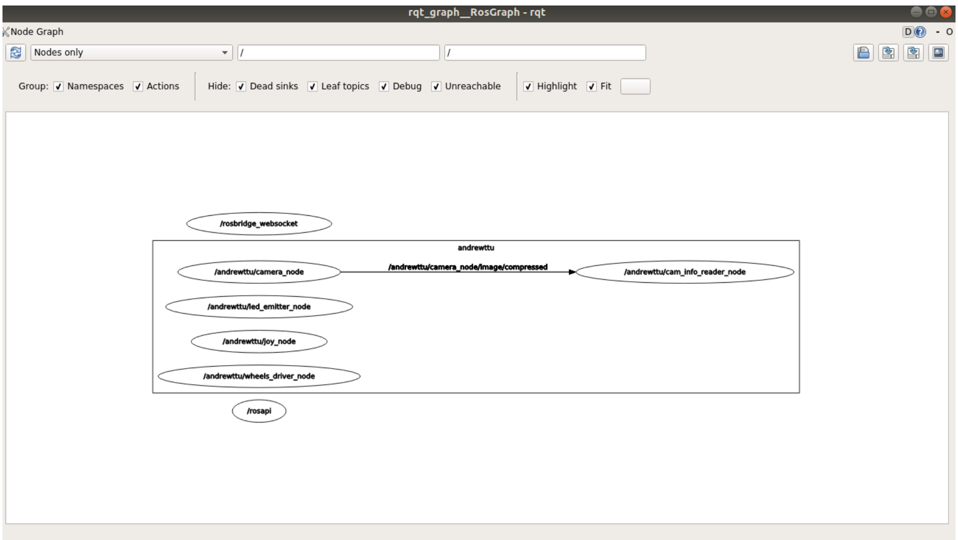


Figure 21 – ROS Nodes for Camera Footage - DuckieBot

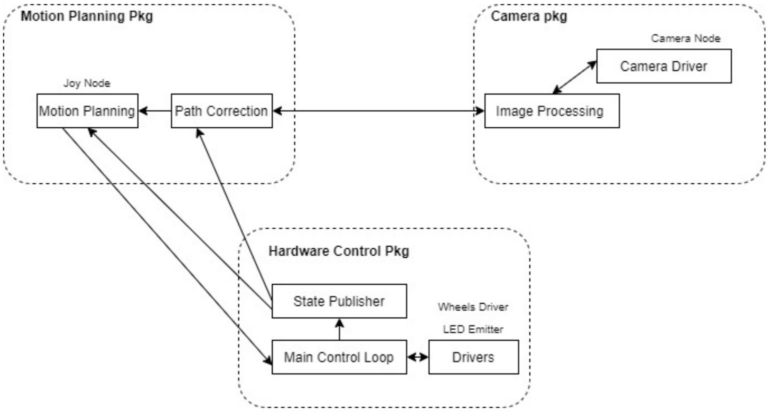
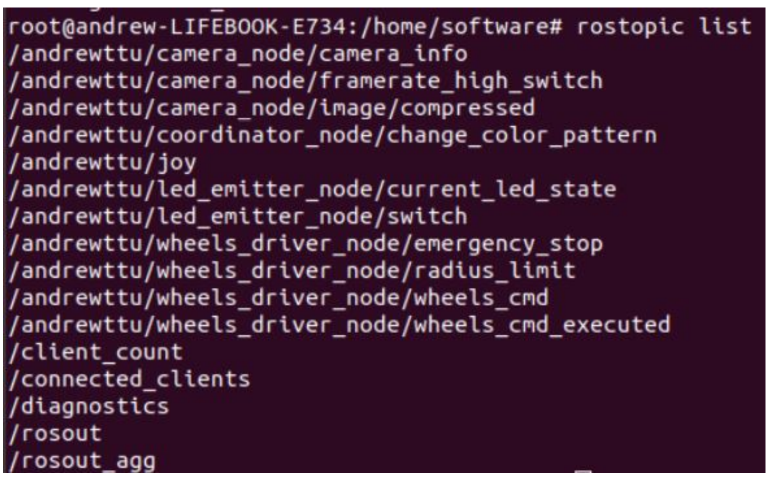


Figure 22 – ROS Architecture for Stopping Operation - DuckieBot

There are two ROS communication types: Topics and Services. A ROS topic is a named communication bus which nodes exchange messages. A node can be a publisher or a subscriber. A publisher shares information with another node, a subscriber receives information from a node. The relationship between nodes is many-to-many and a publisher shares a topic without knowing which node will subscribe to it. Similarly a subscriber will subscribe to a topic without knowing which node published it. Figure 23 presents the ROS topics in a stopping operation on the duckiebot. In this communication, the joy_node is conducting a message exchange with the wheels_driver_node, the topic emergency_stop will initiate an operation of the servo motor to stop the duckiebot.



```
root@andrew-LIFEBOOK-E734:/home/software# rostopic list
/andrewttu/camera_node/camera_info
/andrewttu/camera_node/framerate_high_switch
/andrewttu/camera_node/image/compressed
/andrewttu/coordinator_node/change_color_pattern
/andrewttu/joy
/andrewttu/led_emitter_node/current_led_state
/andrewttu/led_emitter_node/switch
/andrewttu/wheels_driver_node/emergency_stop
/andrewttu/wheels_driver_node/radius_limit
/andrewttu/wheels_driver_node/wheels_cmd
/andrewttu/wheels_driver_node/wheels_cmd_executed
/client_count
/connected_clients
/diagnostics
/rosout
/rosout_agg
```

Figure 23 – ROS Topics - DuckieBot

ROS services are like topics, except they support one-to-one communication between nodes. A service is a request-response type remote procedure call (RPC). In a service communication a node requests from another node a service and the providing node replies back. Services also have unique named communication like topics. Listed below are some of the services for which the colour filter node is communicating to the other nodes.

```
/andrewttu/vehicle_filter_node/get_loggers
/andrewttu/ground_projection/get_image_coordinate
/andrewttu/camera_node/get_loggers
/andrewttu/lane_pose_visualizer_node/set_logger_level
/andrewttu/image_transformer_node/get_loggers
/andrewttu/joy_node/get_loggers
/andrewttu_to_map/get_loggers
/andrewttu/decoder_node/get_loggers
/andrewttu/ground_projection/set_logger_level
/andrewttu/vehicle_avoidance_control_node/set_logger_level
/rosapi/get_param_names
/rosapi/service_host
/andrewttu/led_emitter_node/get_loggers
/andrewttu/inverse_kinematics_node/set_baseline
```

ROS uses rosbags for logging. Figure 24 shows a rosbag logging session. The rosbags collect the publisher and subscriber information, the nodes and the topics being exchanged. This information is valuable for forensics, fault diagnostics and cyber adversaries as it depicts the operations of the vehicle.

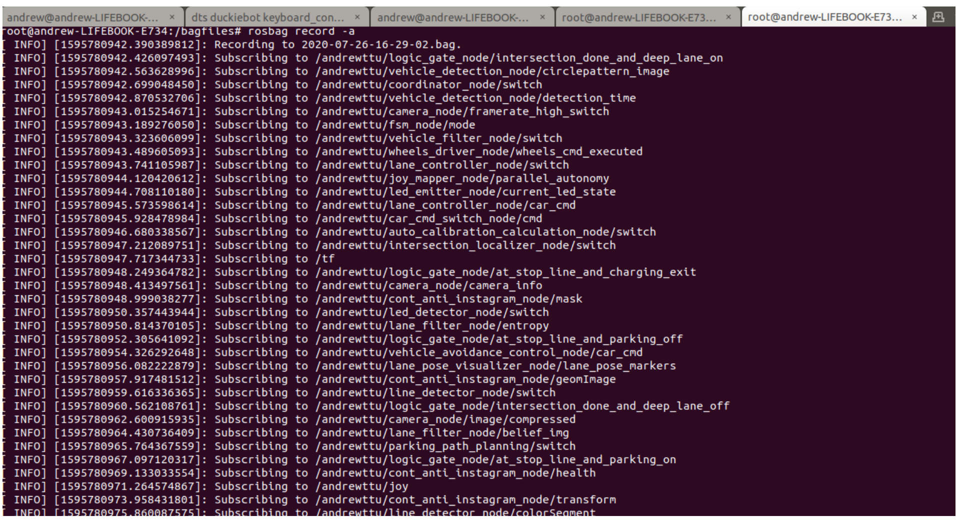


Figure 24 – ROSBAG - Logging Publisher information

Docker is used to manage the DuckieBot environment. As DuckieBot is constantly evolving due to it’s use as an educational and research product, Docker provides an efficient means to implement new images/programs and enhance the use of the limited resources of the Raspberry Pi based system. Figure 25 show the list of running containers in docker.

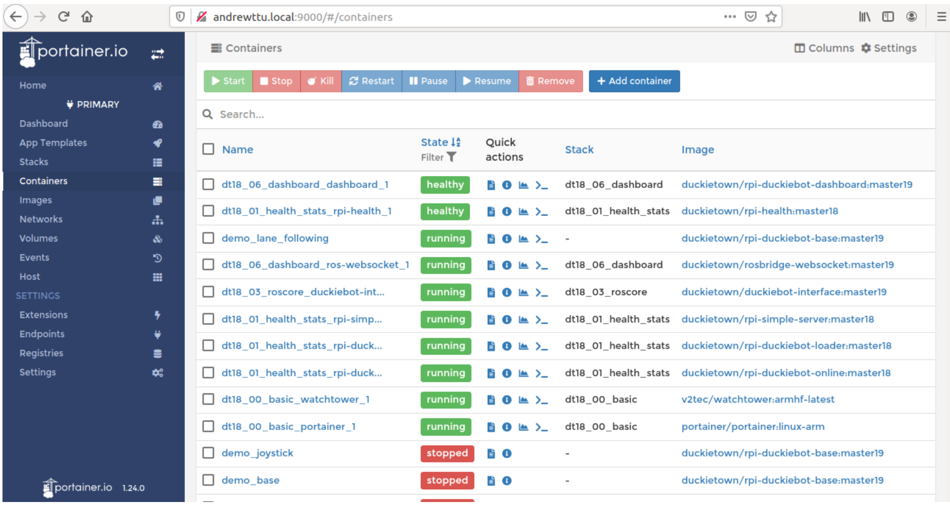


Figure 25 – Docker- Containers

Autonomous Driving Cognition

Duckiebot uses computer vision and image processing for autonomous driving decisions. There are two key aspects to ensure accurate driving of the duckiebot: integrity of the camera sensor, accuracy of the algorithms used for image processing.

Firstly, the camera sensor requires calibration to ensure integrity of the computer vision to enable algorithms to be applied. The DuckieBot camera is calibrated using a specially designed checkerboard panel comprised of black and white squares, each 31mm (Figure 26). This is intrinsic calibration, it's purpose is to resolve discrepancies that can come with camera's parameters straight from the manufacturer. The checkerboard acts as a predetermined patter.

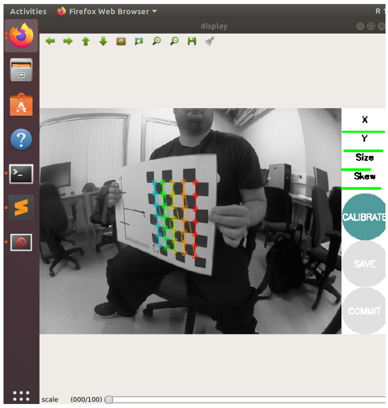


Figure 26 – Intrinsic Camera Calibration

Secondly, the extrinsic calibration aims to use the data of the pictures correctly without error. One object is confirmed in different pictures so that equal pixels can be found. Extrinsic calibration establishes the orientation between the camera and object that the picture is taken from (Figure 27).

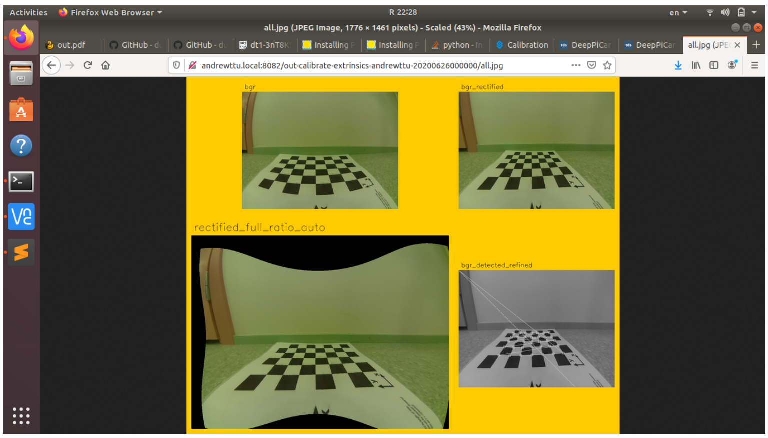


Figure 27 – Extrinsic Camera Calibration

The aim of image processing for autonomous vehicles is to detect road markings (lanes, boundaries) within the driving environment and to filter out disturbances or potential manipulation. The driving environment, from a computer vision perspective, is noisy. The DuckieBot uses colour recognition to find the yellow lane lines, white boundary lines and red stop lines.

As depicted in the image in Figure 29, the environment can generate noise which can be interpreted incorrectly based on the colour. To ensure this doesn't happen the Duckiebot applies two image processing algorithms; Canny edge detection and the Hough transform.

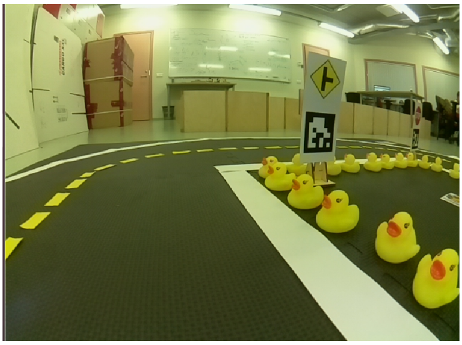


Figure 28 – DuckieBot - Camera Filter

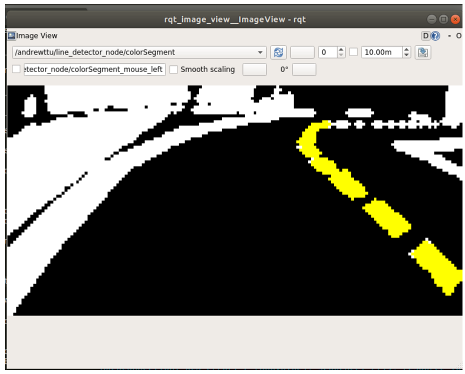


Figure 29 – DuckieBot - Colour Recognition Filter

The intended aim of these techniques is to apply an edge filter and reduce noise by applying a Gaussian blur to isolate the shape of the yellow lane marking and white border lines and make a hypothesis of the best lane position of the DuckieBot. The code for the DuckieBot image processing is provided in Appendix 4.

Figure 30 – DuckieBot - Edge Filter

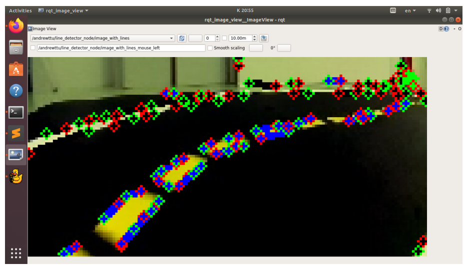


Figure 31 – DuckieBot - Line Detector

Remote Control Operations

Duckiebot mission control platform is a graphical user interface that allows a human operator control of the Duckiebot. The human operator is able to visualise the speed of the vehicle, steering angle of the vehicle and the on-board camera vision. The operator is able to toggle between autonomous mode and manual control. The operator GUI relies on network connection to the same network of the DuckieBot, this is configured in Docker.

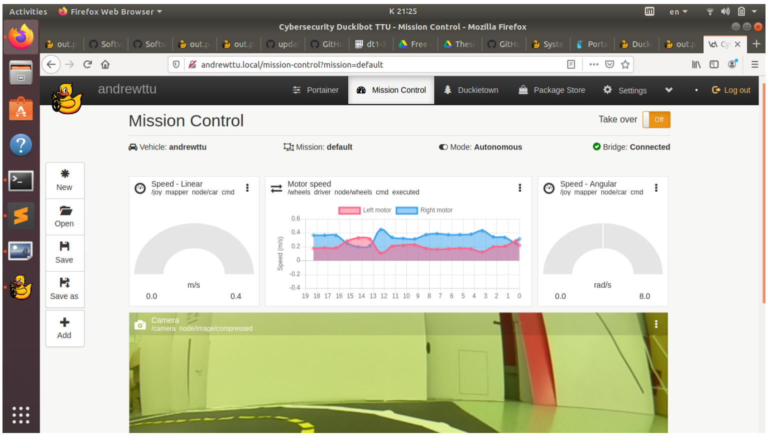


Figure 32 – DuckieBot Mission Control Platform

3.3.3 DeepPi Car

The DeepPiCar is a reference architecture for simulation of autonomous self-driving vehicles developed by David Tian, a software-engineer at Google.

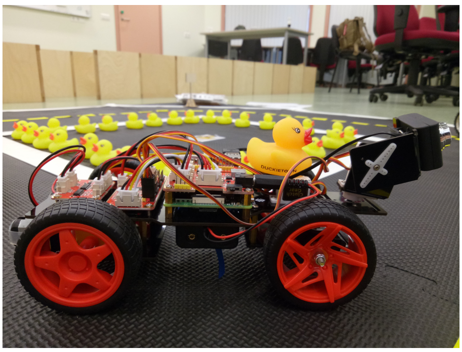


Figure 33 – DeepPi Car - Side View

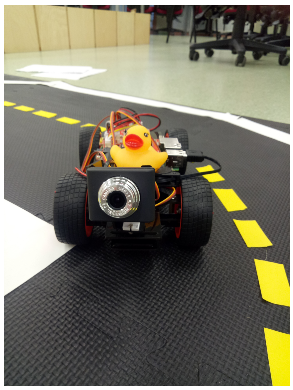


Figure 34 – DeepPi Car - Front View

The DeepPi car uses a Raspberry Pi model 4 for the on-board computer. The operating system is Raspbian buster, an operating system made for arm processors. There is a 32 Gb SD card for internal storage. Sensing is performed by a camera sensor, originally a 2mp camera, later upgraded to 5 mp. Connection with the remote control terminal is via the wireless network interface. Actuation is performed by the motor driver which connects to servo motors. The DeepPi car, unlike DuckieBot, uses mechanical steering, the body physics is more representative of a real-world vehicle. Power is provided by 2 x 18650 3.7v lithium ion batteries.

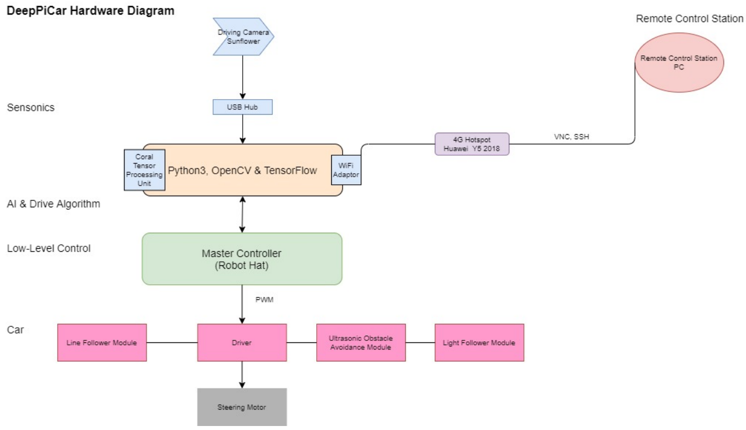


Figure 35 – DeepPi Car Hardware Diagram

Python3 is used for the code base. Unlike DuckieBot, the DeepPi doesn't use ROS and the operation of the car is executed by a main module which makes calls to other python modules. Figure 36 shows the python modules in the DeepPi car. Similar to the ROS packages, each module is program for either a hardware or software component of the vehicle.

```
andrew@andrew-LIFEBOOK-E734 ~/DeepPicar-v2                              [10:43:34]
> $ ls                                                                 [±master ✓]
actuator-adafruit_hat.py   LICENSE                    scripts
actuator-arduino.py        local_common.py            sync-video.py
actuator-drv8835.py        maxperf.sh                 take.sh
actuator-mc33926.py        model-3conv_1pool.py       test
actuator-null.py           model-5conv_3fc.py         test-model2.py
camera-null.py             model-5conv_4fc.py         test-model3.py
camera-webcam.py           model.py                   test-model4.py
create-epoch.sh            params.py                  test-model.py
data_ordered.py            picar-mini-kbd-common.py   train.py
data_shuffled.py           preprocess.py              view-video.py
images                     README.md                  visualize.py
input_kbd.py               run.py
```

Figure 36 – DeepPi Car Python Modules

Autonomous Driving Cognition

The camera sensor and OpenCV (Computer Vision) is for image processing. Google Tensorflow is used for machine learning. Due to the restricted computing resources available in the raspberry pi, a Google Coral edge tensor processing unit (TPU) is used for high-speed machine learning inferencing. The Coral TPU allows for 4 trillion operations (inferences) to be performed per second using 2w of power.

The process for training the deep learning of the DeepPi Car involved the following:

- Installation of OpenCV for computer vision and image processing.
- Installation of Tensorflow and test object detection capability. To do this the COCO

(Common Object in COntext) object detection model was run.

- Build LKAS into the detection model by training lane detection

Without accurate training of the object detection, the results can be inaccurate and lead to over detection or inaccurate detection.

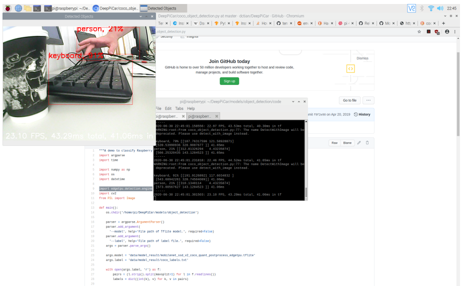


Figure 37 – COCO Object Detection

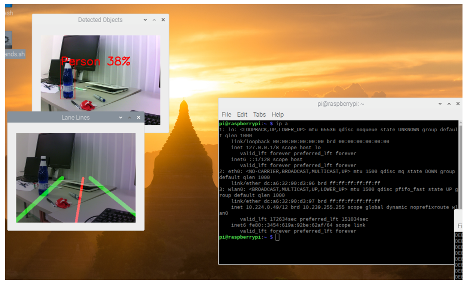


Figure 38 – COCO Object Detection Misclassification

Like the DuckieBot, Canny Edge Detection and the Hough transform was used to build the LKAS.

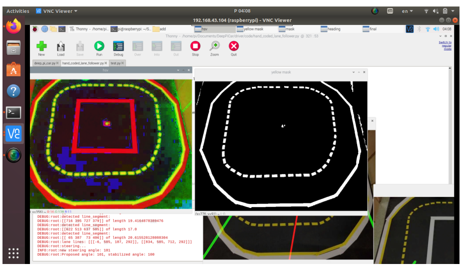


Figure 39 – HSV and Canny Edge Detection

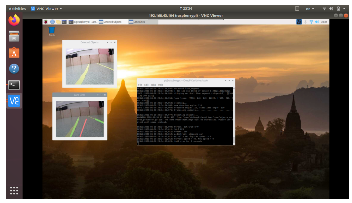


Figure 40 – Line Keeping Assistance System Calibration

The original code base was written by David Tian for blue lines and as a line follower not a LKAS. For this thesis, the author rewrote the code as a LKAS for yellow lines. The code is available on this link: `https://gitlab.com/Self-DrivingRoberts/experimental-testbed-autonc` `-/tree/master/public` .

In the development of the DeepPi car problems were encountered due to the limited compute resources of the Raspberry Pi, the sensitivity of the mechanical components and the lack of centralised efficient code management due to not using ROS. Early in the development the object detection was encountering issues due to the poor definition of the 2mp camera. The camera was upgraded to 5mp, however, the increased size impacted the mechanical movement brackets. The camera was stripped and reconfigured on the DeepPi car, which enabled correct maneuvering. The LKAS was shown to have worked, however, due to the increased computational resources required, the frame rate of the camera is not consistent and therefore the DeepPi car loses sensing of the road after 30 seconds.

Remote Control Operations

The DeepPi remote control operations has limited functionality. The operator can login to a server which provides access to a GUI that allows functionality such as viewing the on-board camera and manual control of the vehicle. The operator is unable to toggle between autonomous mode and human control. When human manual override is initiated autonomy is lost until the vehicle is rebooted.

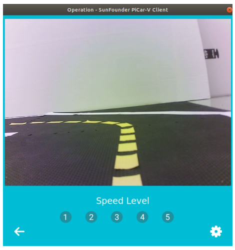


Figure 41 – Remote Control

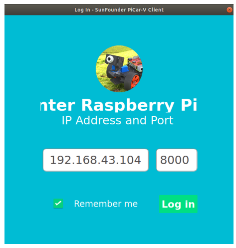


Figure 42 – Remote Control Server

3.4 Demonstration

The autonomous self-driving vehicle test bed has been demonstrated to Starship robotics, ISEAUTO and ZF. The test bed is also available for viewing on a YouTube channel: `https://www.youtube.com/channel/UC7cXB9DSG6UCQAYHw4vkrSQ/videos`. The videos on this YouTube channel were created by the author.

The test bed is also available to be used as an open source lab. Each of the vehicles support remote connection. A researcher interested in conducting security tests can remote into the vehicle using VNC or SSH and run their tests.

4 Evaluation

4.1 Method

Test cases are used to evaluate the autonomous self-driving vehicle test bed. The practical security threat analysis method by Vasenev et al. [44] was customised to generate the test cases to evaluate the test bed. The method established by Vasenev et al. is for internal security testing and assumes privileged information access such as data flow diagrams. A customised method was used in this thesis as it is tailored for an adversarial approach with no prior knowledge of the autonomous vehicle.

	Tool/Model
Analysis Method	STRIDE
Input	Observation of the test bed
	UNECE WP29 Matrix
Output	Prioritised Security Threats

Table 4 – Analysis Method

Table 4 details that STRIDE is used as the security analysis method as it is the most widely used for automotive[45]. The inputs to the STRIDE analysis come from expert opinion. Firstly, experts observe the driving behaviour and on-board systems of the Duckiebot and Deep Pi.Based on their observational analysis they provide their opinion as to what they think are realistic threats to the vehicle based on their experience and testing processes in their organisations. Secondly, these identified threats are compared to those listed in the UNECE WP29 threat matrix. The reasoning for this is the cybersecurity certification scheme from UNECE WP29 represents the future for automotive cybersecurity certification and the inclusion of UNECE WP29 provides real-world relevance to the testing. The consolidated list from the UNECE WP29 analysis is then presented to the experts for consideration of what threats should be prioritised for testing. The output of the STRIDE analysis are prioritised threats threats.

Expert opinion is used for the identification of security threats for the STRIDE analysis and as the means to select the prioritised security threats. Figure 43 shows the analysis flow to generate test case for experimental testing. The analysis flow recommended by Vasenev et al. has been tailored to include their contributions.

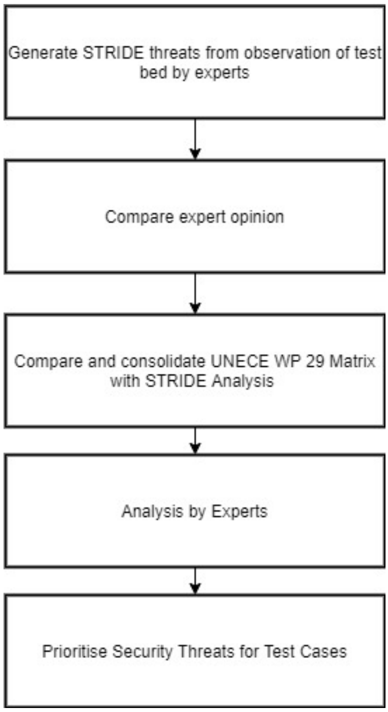


Figure 43 – Flow of Test Case Generation

4.2 Expert Interviews

The method for inclusion of expert opinion followed the Technological Delphi method as outlined by Bayona-Ore et al[65]. The Technological Delphi method consists of four characteristics of what required for the inclusion of expert opinion in research:

1. Use of experts who are in a specific field or have technical knowledge and are part of the expert panel.
2. Iterative process to allow experts to provide more than one opportunity to provide an opinion.
3. Opportunity for feedback should feedback at the end of the experiment.
4. Each interviewee should not know each others answers to ensure the integrity of the opinion and avoid biases

The Technological Delphi method utilises technological means for facilitation of an expert opinion feedback loop. Technology used for communication of the test bed and feedback with experts comprised of email, Skype, YouTube and workshops in the TTU Robotics laboratory.

Expert interviews were conducted with ISEAUTO, Starship Robotics, and ZF. The interviewees met the criterion of experts as their roles consisted of; autonomous driving security engineer, senior security engineer, director for safe driving, autonomous driving algorithm designer and security architect. Each of their companies are considered leaders in the automotive industry, autonomous logistics, and autonomous vehicle education research.

Each interviewee, as per the method in figure 43, observed the test bed. Starship and ISEAUTO viewed the test bed at the TTU Robotics Laboratory and ZF viewed the test bed on the YouTube channel. The experts contributed threats based on their understanding of real-world scenarios and how they test in their own organisations. The consolidated list of threats, which combined all three expert opinions and those identified in the UNECE WP 29 Matrix, were reviewed by the experts and they prioritised the threats to evaluate the test bed, based on real-world cyber threats experienced by their autonomous vehicles.

The experts provided feedback on the results of the test case experiments.

To ensure this work is published in an open forum and to protect each interviewee from revealing the tactics, techniques and procedures used in cybersecurity testing in their organisation, their opinions have been summarised to allow inclusion in this thesis. Their names, roles and discussion with this author will not be published.

4.3 Security Test Cases

4.3.1 STRIDE Analysis

STRIDE THREAT ANALYSIS			
	Threat	**Property Violated**	**Threat Definition**
S	Spoofing	Authenticity	T1 – A malicious attacker spoofs the roadside units to manipulate the drive logic to veer the vehicle off the road
			T2 – A malicious attacker spoofs the road markings to manipulate the drive logic to veer the vehicle off the road
T	Tampering	Integrity	T3 – A malicious attacker tampers with the road markings to manipulate the drive logic to veer the vehicle off the road
			T4 – A malicious attacker tampers with the camera sensor using a laser pointer to blind or shield its perception to manipulate the drive logic to veer the vehicle off the road
			T5 – An innocent maintenance engineer executes a malicious cryptocurrency or ransomware malware hiding as a firmware update for a vehicle system created by an angry mechanic/insider
			T6 – An angry mechanic/insider inserts malicious ROS package to execute processes to alter the vehicle driving behaviour
R	Repudiation	Non-Repudiation	T7 – An angry mechanic/insider changes the access credentials to the vehicle control and logs so the vehicle controller cannot access data about their vehicle
I	Information Disclosure	Confidentiality	T8 – A malicious attacker eavesdrops on the ROS vehicular messaging system for information gathering.
			T9 – An angry mechanic/insider unauthorised accesses the autonomous vehicle logs to extract data to sell to the competition
D	Denial of Service	Availability	T10 – A malicious attacker conducts a denial of service of the short-range wireless network of the autonomous self-driving vehicle
E	Elevation of Privilege	Authorisation	T11 – An angry mechanic/insider elevates their privileges to super user to be able to change in-vehicle messages

Figure 44 – STRIDE Threat Analysis

4.3.2 UNECE WP 29 Matrix

STRIDE REF	High level and sub-level descriptions of vulnerability threat			Attack Method	
T6 T11	4.3.1 Threats regarding back-end servers	1	Back-end servers used as a means to attack a vehicle or extract data	1.1	Abuse of privileges by staff (insider attack)
T11				1.2	Unauthorised internet access to the server (enabled for example by backdoors, unpatched system software vulnerabilities, SQL attacks or other means)
T9				1.3	Unauthorised physical access to the server (conducted by for example USB sticks or other media connecting to the server)
T7		2	Services from back-end server being disrupted, affecting the operation of a vehicle	2.1	Attack on back-end server stops it functioning, for example it prevents it from interacting with vehicles and providing services they rely on
T1 T2		4	Spoofing of messages or data received by the vehicle	4.1	Spoofing of messages by impersonation (e.g. 802.11p V2X during platooning, GNSS messages, etc.)
T4	4.3.2 Threats to vehicles regarding their communication channels	5	Communication channels used to conduct unauthorized manipulation, deletion or other amendments to vehicle held code/data	5.5	Communications channels permit manipulation of vehicle held data/code
T3		6	Communication channels permit untrusted/unreliable messages to be accepted or are vulnerable to session hijacking/replay attacks	6.1	Accepting information from an unreliable or untrusted source
T8		7	Information can be readily disclosed. For example through eavesdropping on communications or through allowing unauthorized access to sensitive files or folders	7.1	Interception of information / interfering radiations / monitoring communications
T5	4.3.4 Threats to vehicles regarding unintended human actions	15	Legitimate actors are able to take actions that would unwittingly facilitate a cyberattack	15.1	Innocent victim (e.g. owner, operator or maintenance engineer) being tricked into taking an action to unintentionally load malware or enable an attack
T10	4.3.5 Threats to vehicles regarding their external connectivity and connections	16	Manipulation of the connectivity of vehicle functions enables a cyberattack, this can include telematics; systems that permit remote operations; and systems using short range wireless communications	16.3	Interference with short range wireless systems or sensors

Figure 45 – UNECE WP29 Consolidated Matrix

4.3.3 Expert Analysis

Threats to the vehicle communication channels, their sensors and perception were rated as high priority by a majority of expert opinion. The justification for this is that it offers a low-cost, low-skill attack that can be as successful as a complex software or network attack. Experts expected adversarial machine learning attacks, sensor spoofing and blinding and manipulation of the variables in the driving environment to be a realistic and common attack surface that will be seen on the streets of Tallinn and Helsinki. One expert thought the inclusion of environmental perturbations of sensors such as fog, rain, smoke would be interesting to replicate in the small-factor environment as this forms part of the combined process for security and safety testing of their autonomous vehicle.

Threats to vehicle systems from malware was another highly rated concern. Realistic scenarios include an angry mechanic or engineer manipulating an update script to install a malicious ransomware or cryptocurrency malware. The experts saw insider threats as one of the more likely scenarios as internal knowledge about update procedures and in-vehicular components and networks were crucial for a successful attack. They opined the likelihood of success of external adversarial attacks were reduced due to technical controls such as code signing and secure communication between components.

A majority of expert opinion accentuated the importance of threats to the external connectivity and connections. The justification for prioritising network attacks is that, in their opinion, most urban mobility transport operators operate multiple autonomous vehicles and a cyber attack that impacts the availability of the network or the confidentiality of the network could lead to multiple vehicles being taken control of by the attacker or taken offline from the remote operator console.

4.3.4 Prioritised Security Threats

Test Case	Threat Definition
Test Case 1	A malicious attacker spoofs the road markings to manipulate the drive logic to veer the vehicle off the road
Test Case 2	A malicious attacker tampers with the road markings to manipulate the drive logic to veer the vehicle off the road
Test Case 3	A malicious attacker tampers with the camera sensor using a laser pointer to blind or shield its perception to manipulate the drive logic to veer the vehicle off the road
Test Case 4	A malicious attacker spoofs the roadside units to manipulate the drive logic to veer the vehicle off the road
Test Case 5	An innocent maintenance engineer executes a malicious cryptocurrency or ransomware malware hiding as a firmware update for a vehicle system created by an angry mechanic/insider
Test Case 6	A malicious attacker eavesdrops on the ROS vehicular messaging system for information gathering.
Test Case 7	A malicious attacker conducts a denial of service of the short-range wireless network of the autonomous self-driving vehicle
Test Case 8	Smoke from fire obscures the driving environment causing vehicle to take adverse driving behaviour.**

** Test Case 8 was conducted on the express wish of one of the experts. They combine safety and security testing in their processes and they wanted to see the capacity of the small-factor vehicle to conduct this experiment.

Figure 46 – STRIDE Threat Analysis

4.4 Security Test Case Evaluation

4.4.1 Sensor and Perception Security Test Cases

Test Case 1: A malicious attacker spoofs the road markings to manipulate the drive logic to veer the vehicle off the road.

Experiment Setup: The autonomous self-driving vehicle is set on autonomous mode for 5 minutes allowing the vehicle to navigate traffic.

1. Attacker observes the autonomous self-driving vehicle to understand how the autonomous drive cognition makes decisions.

2. Attacker crafts an image for projection on the driving environment. Figure 47 and 48 demonstrate images chosen for projection.

3. Attacker positions the projector in proximity to the vehicle and uses a remote control to initiate the projection attack.

Figure 47 – Malicious Projection Image

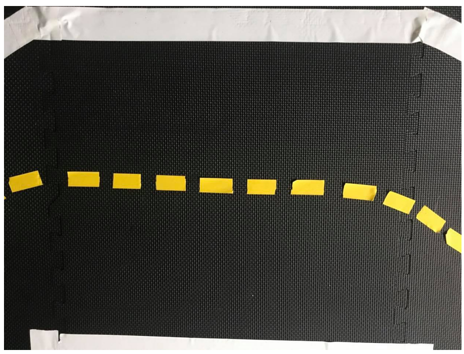


Figure 48 – Malicious Projection Image

Experiment Recording: `https://www.youtube.com/watch?v=TYszVeblKEo` **Experiment Results:**The phantom attacks were unable to alter the driving actions of the duckiebot. Figure 49 shows the faint image of the phantom spoofed yellow line which is barely visible due to the bright profile of the driving environment. Figure 50 visibly shows the phantom spoofed line, due to a larger spoofed image being projected by the attacker. The figure 50 image, from the Duckiebot camera shows that the autonomous drive cognition is detecting the edges and texture of the yellow lines and white boundaries but is not detecting the phantom image. This is due to the lack of edges, texture and geometry of the phantom image.

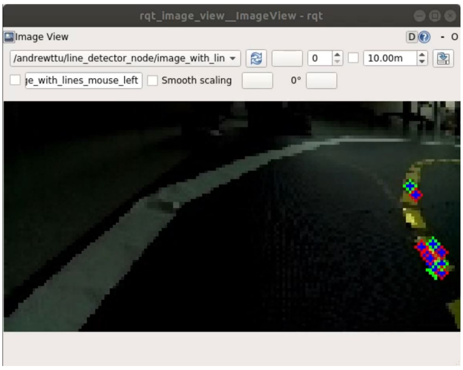


Figure 49 – Projector Attack 1

Figure 50 – Projector Attack 2

For Attack 4 (51) and 5 (52), the attacker uses larger and greater definition spoofed images and includes yellow and white lines in order to spoof both lane markings and boundaries. The attack is still unsuccessful as the autonomous drive cognition does not detect any edges, texture or geometry of the phantom image. The attacker, pictured in figure 52, is only 20 cm away from the road surface. To provide a clear phantom image the projector had to be close to the target surface.

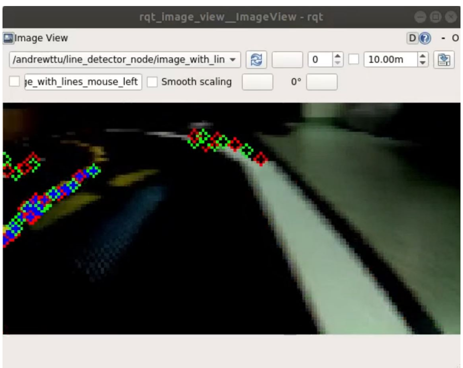


Figure 51 – Projector Attack 4

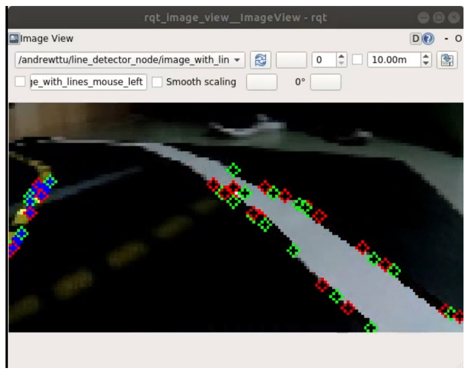


Figure 52 – Projector Attack 5

All of the variables in the Nassi et al. experiments were recreated with the Duckiebot. The Phantom images were left projecting on the road surface for 10 minutes, the size of the images were increased, the definition of the images increased, projection on different sections of the floor and different environmental light. The DuckieBot was resilient to the phantom attack and the autonomous drive cognition was not spoofed by the phantom images.

Conclusion: Whilst a spoofing attack using projection is a novel and interesting method to manipulate an autonomous vehicle it is unlikely or has low probability of success. The projection must contend with natural light, which means the attack must be undertaken at night and it is not too difficult to update the object detection algorithm to filter out these attacks.

Test Case 2: A malicious attacker tampers with the road markings to manipulate the drive logic to veer the vehicle off the road.

Experiment Setup: The autonomous self-driving vehicle is set on autonomous mode for 5 minutes allowing the vehicle to navigate traffic.

1. Attacker observes the autonomous self-driving vehicle to understand how the autonomous driving cognition makes decisions.
2. Attacker, using the understanding of the drive control algorithm, perturbs the road markings in the duckietown environment. The attacker can choose a discreet or noisy attack. The discreet attack will be harder for the human operator with the remote control pc to see the perturbation of the road marking.

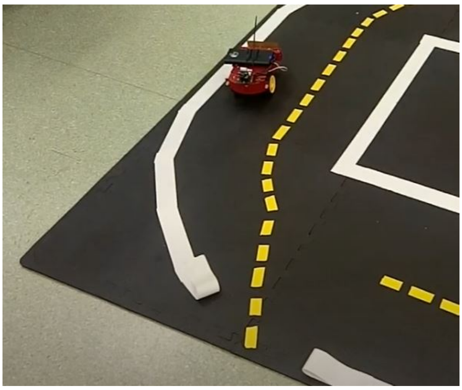


Figure 53 – Tile manipulation - discreet

Figure 54 – Tile manipulation - noisy

Experiment Recording: `https://www.youtube.com/channel/UC7cXB9DSG6UCQAYHw4vkrSQ/videos`

Experiment Results: The experiments used five attacks, LKAS1 to 5. The Results confirmed the findings of Sato et al. . Perturbation of a road marking can manipulate the drive algorithm to cause the autonomous self-driving vehicle to veer off the intended path of travel.

In LKAS Attack 1, the attacker tampered with the yellow lane markers to manipulate the autonomous self-driving vehicle to drive off the road. The curve road part was changed to a straight trajectory and the angle of the lane borders (white lines) were reduced to lessen the width of the road. As Figure 56 demonstrates, the change to the road markings is demonstrable in the DuckieBot camera sensor footage, from the expected road markings exhibited in Figure 55. LKAS 1 was successful in manipulating the autonomous drive cognition of the DuckieBot, however, the DuckieBot's autonomy is programmed to firstly respect the lane boundaries. The DuckieBot followed the tampered yellow line until it detected the lane boundary and then adjusted it's travel path to the correct route.

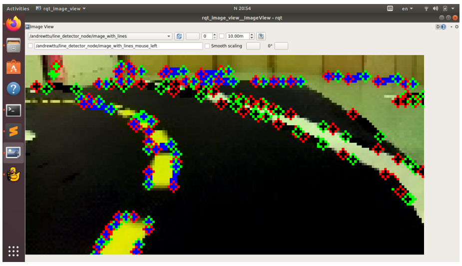


Figure 55 – Normal Traffic Lane Markings

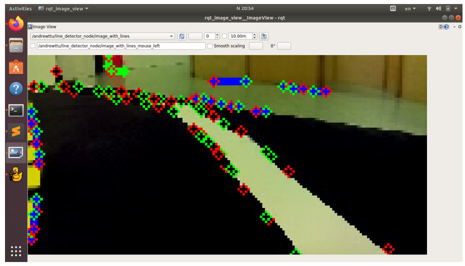


Figure 56 – Spoofed Lane Markings - Discrete

In LKAS 2 and 3 the attacker extended the yellow lane markings further into the lane boundaries. The DuckieBot still respected the boundaries and corrected the path of travel.

LKAS 4 the attacker removed the lane boundaries and extended the yellow lane markings, as shown in Figure 57 . The attack was successful and the DuckieBot veered off the DuckieTown environment and was unable to recover.

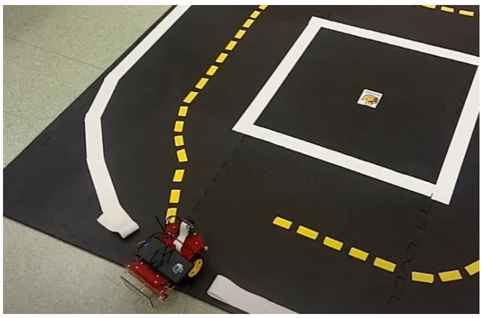


Figure 57 – LKAS5 - Successful Manipulation of Duckiebot

In LKAS 5, a more noisy profile of manipulated lane markings was used by the attacker. The DuckieBot experienced limited manipulation of driving due to the DuckieBot sensing yellow markings in the distance and calculated an accurate route of travel.

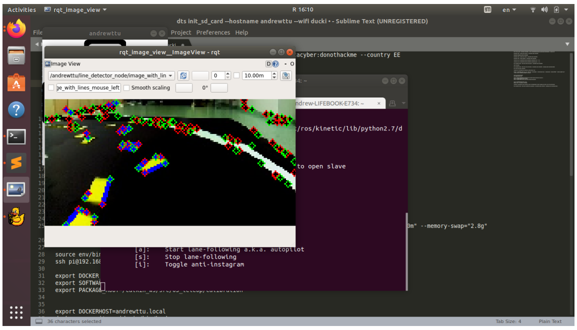


Figure 58 – Spoofed Lane Markings - Noisy

Conclusion: Although this threat seems simplistic in the experimental test bed environment, the implications for a real-world operational vehicle are stark. As Sato et al. demonstrated, an attacker can use a 3D printer to print a tampered road patch and place it on the road markings of a highway. If this test had occurred on an autonomous vehicle travelling at 40 mph the results of the impact analysis would show the extent of damage to which sensor and perception attacks can cause.

Test Case 3: A malicious attacker tampers with the camera sensor using a laser pointer to blind or shield its perception to manipulate the drive logic to veer the vehicle off the road.

Experiment Setup: The autonomous self-driving vehicle is set on autonomous mode for 5 minutes allowing the vehicle to navigate traffic.

1. Attacker observes the autonomous self-driving vehicle to understand how the drive control makes decisions.
2. Attacker, using the understanding of the drive control algorithm, sets up a bosch industrial laser at the side of the road.
3. Attacker turns on the laser to beam a red line across the road surface, spoofing the red stop line programmed into the autonomous self-driving vehicle.

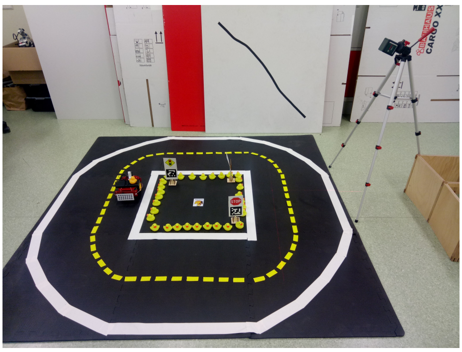


Figure 59 – Bosch Laser spoof attack

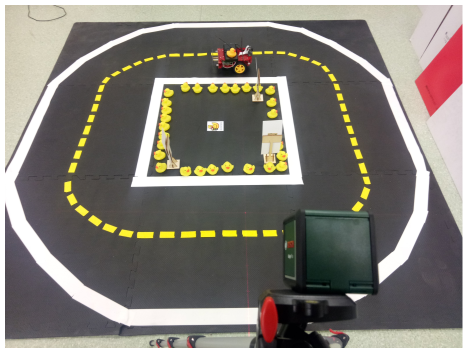


Figure 60 – Bosch Laser spoof attack

Experiment Recording:`https://www.youtube.com/channel/UC7cXB9DSG6UCQAYHw4vkrSQ/videos`

Experiment Results: The results of the experiment were that the laser was successful in tampering with the camera sensor which resulted in the autonomous driving cognition altering the course of the vehicle to proceed off the road.

The laser must be held steady and focused on the camera lens long enough to disturb the computer vision. Figure 61 demonstrates the DuckieBot veering off the road from the laser perturbation.

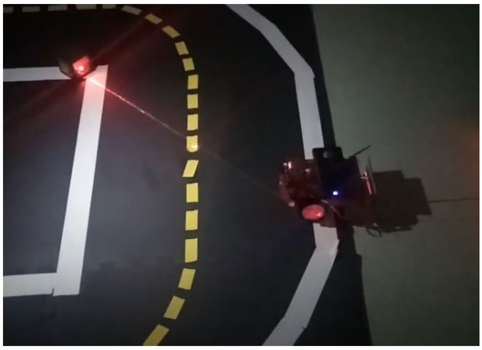


Figure 61 – Laser Attack - Crash 3

A concerning aspect of the attack was the lack of detection of the laser from the camera. Figure 62 shows a laser perturbation from a spot laser beam. The only recognition of the computer vision is the solid green line at the top left of the screen. This is the autonomous driving cognition mistaking the red, of the laser beam, with the pre-programmed rules of a red line for the stop condition.

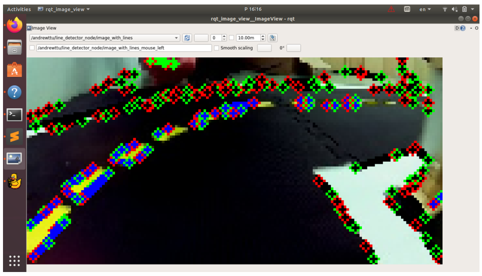


Figure 62 – Laser Attack - Crash 7

The laser attack test case was conducted over 10 times. Only on three occasions was it successful due to the requirement for correct placement on the camera lens.

Conclusion: The laser attack presents a real-world threat to operational autonomous self-driving vehicles. The attack is inexpensive and can be conducted by an unskilled attacker. The camera sensors of a real-world vehicle are much larger and present an easier target for adversaries. Defensive mechanism that can be implemented to mitigate against this attack include improving the algorithm to filter out disturbances from lasers.

Test Case 4: A malicious attacker spoofs the roadside units to manipulate the drive logic to veer the vehicle off the road.

Experiment Setup: The DuckieBot is set on autonomous mode for 5 minutes allowing the DuckieBot to navigate traffic.

1. Attacker observes the autonomous self-driving vehicle to understand the how the drive control algorithm makes decisions.
2. Attacker, using the understanding of the drive control algorithm, tampers with the stop sign . The attacker uses yellow dashed lines and white border lines to cover the stop sign with the intent of getting the DuckieBot to proceed through the stop sign.

Experiment Results: Due to the problems encountered with the object detection the experiment was unable to be conducted. The object detection in the both the DuckieBot and the DeepPi is unable to function correctly as there is too much delay in the frame rate of the camera. Due to this the vehicles cannot detect objects in the environment consistently whilst driving. Using the object detection whilst the DuckieBot is static the manipulated road sign is inaccurately detected as a lane marker. It can be seen that this attack would be successful in manipulating the object detection of a working vehicle.

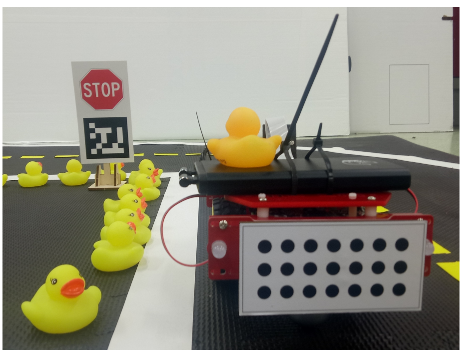


Figure 63 – Correct Stop Sign

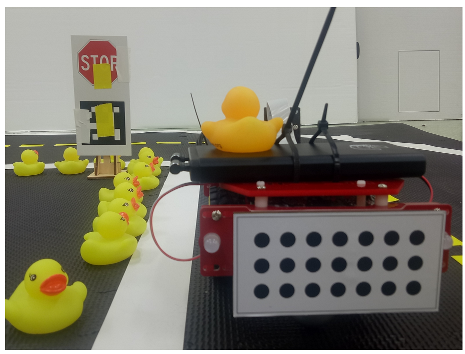


Figure 64 – Adversarial Machine Learning Rogue Sign

4.4.2 Hardware & Compute Test Cases

Test Case 5: An innocent maintenance engineer executes a malicious cryptocurrency or ransomware malware hiding as a firmware update for a vehicle system created by an angry mechanic/insider.

Experiment Setup:

1. Angry Mechanic uploads malware script (Linux.MulDrop.14) from dark web
2. Malware script is packaged as a bash script that is labelled "update".
3. Maintenance engineer initiate "update" script with intention update vehicle firmware.

Experiment Results: The "update" firmware (Figure 65) was executed by the innocent maintenance engineer working on the DeepPi car.

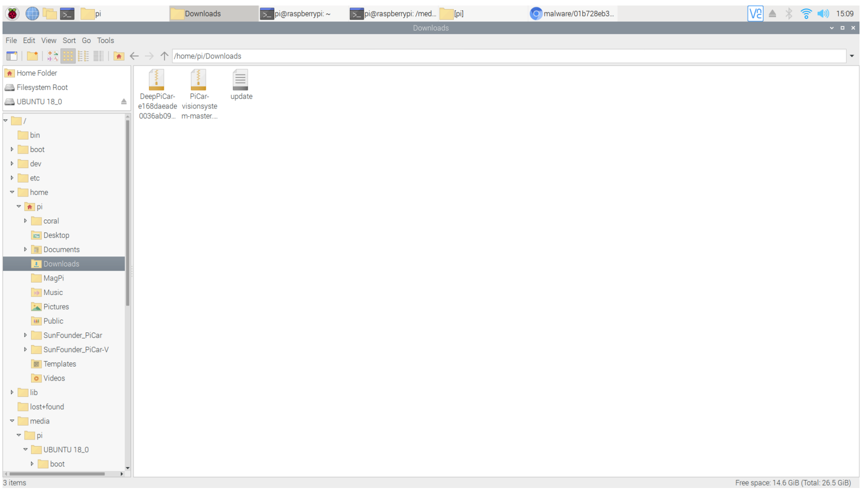


Figure 65 – Update File

The update firmware contained the Linux.MulDrop.14 script.Linux.MulDrop.14 is a bash script containing a cryptomining program. Once infected on a host computer the Linux.MulDrop.14 installs several libraries and processes for it's operation and then tries to install zmap (network scanner) and ssh pass (utility for establishing ssh connections). It uses zmap, in an infinite loop, to discover the network and find raspberry pi's and other embedded devices with port 22 (ssh) open. If these are found, it connects to the device using ssh with default passwords.It then changes the configuration settings of the device to allow a connection to a command and control node used for cryptomining.

On the DeepPi car, the malware installed it's libraries and zmap and ssh pass and began a zmap scan of the network. The DeepPi was on a private 4G network that also had the DuckieBot connected. As these devices do not use default passwords it was unable to establish a connection to them. The DuckieBot is also managed through docker environment which adds another layer of protection. The zmap scans only marginally impacted the performance of the network of the DeepPi car. As figure 66 shows the zmap scan was sending 50,000 packets to the target device, but these are only looking for open port 22.

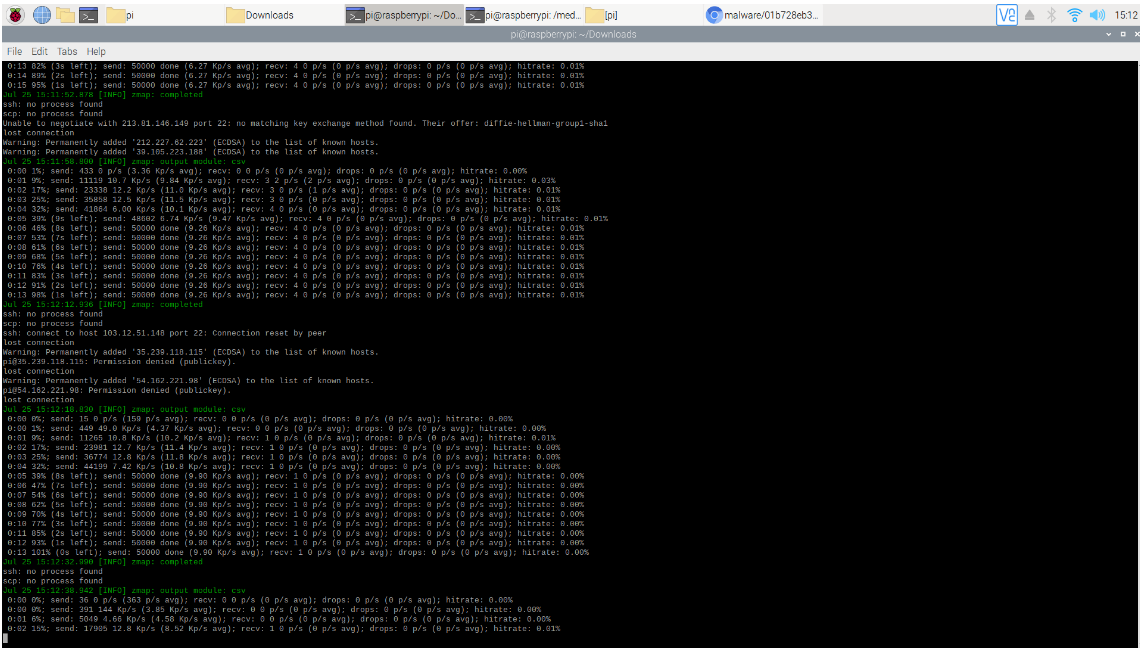


Figure 66 – ZMAP Scan

An interesting event happened during the experiment. The 4G cellular private network lost connection during the malware execution and the DeepPi Car switched over to the TalTech wireless network. The zmap process then started to scan the TalTech network for open Raspberry pi and embedded devices.TalTech IT Security incident response team saw the DDoS traffic and removed the DeepPi car from the TalTech Wireless network within 10 minutes.

Conclusion: The implant of the malware on the DeepPi on-board computer was easy and required low-skill. The experiment demonstrated the importance of basic IT security controls in vehicles such as not using default passwords and regular patching. The malware leakage to the TalTech network provided an interesting observation: an autonomous vehicle could lose access to a secure network and instead connect to a more vulnerable network which would allow malware to propagate more extensively. This highlights the importance security controls on the car and on the edge servers which the autonomous car sends and receives data from.

Test Case 6: A malicious attacker eavesdrops on the ROS vehicular messaging system for information gathering.

Experiment Setup: For this attack, the attacker needs to be on the same network as the vehicle.

1. Attacker scans the network and identifies the vehicle
2. Attacker eavesdrops on the ROS communication by spoofing the ROS Master

Experiment Results: Figure 67 shows the commands required for spoofing the ROS Master in the attacker environment. Port 11311 is the default port for the ROS Master.

```
root@andrew-LIFEBOOK-E734:/home/software# ROS_MASTER_URI=http://192.168.43.240:11311/
root@andrew-LIFEBOOK-E734:/home/software# ROS_HOSTNAME=andrewttu
root@andrew-LIFEBOOK-E734:/home/software# rqt_graph
```

Figure 67 – ROS Eavesdropping

The attacker proceeds to use the rqt_graph command to print the ROS node and topic activity of the operational vehicle.

Figure 68 – ROS Graph

Figure 68 shows communications of the ROS Master that the attacker is eavesdropping. The attacker can use this to learn of the operations of the vehicle and then use the same spoofing of the ROS Master to then initiate malicious processes or stop critical safety processes.

Conclusion: ROS is highly insecure. The version that the DuckieBot is running is the same as the vehicles used in the FinEst project. There is no authentication and secure communication of the ROS Master. The ROS Master also uses HTTP so it is vulnerable to a number of other malicious web application attacks.

4.4.3 Connected Vehicle Security Test Cases

Test Case 7: A malicious attacker conducts a denial of service of the short-range wireless network of the autonomous self-driving vehicle.

Experiment Setup: The autonomous self-driving vehicle is set on autonomous mode for 5 minutes allowing the vehicle to navigate traffic.

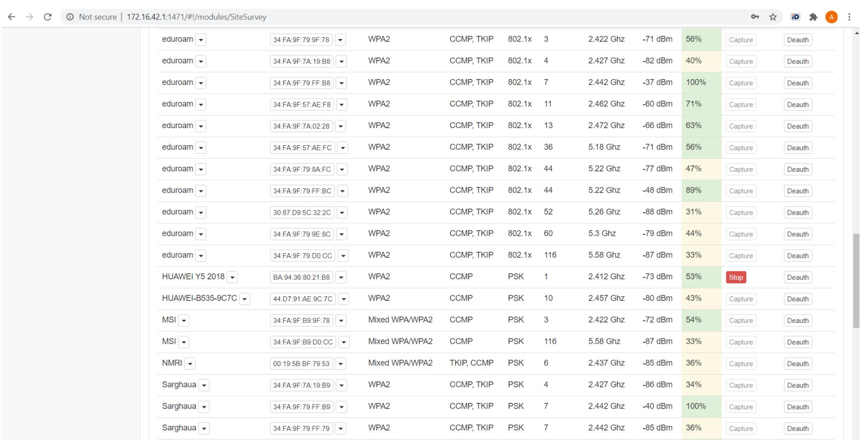
1. Attacker scans wireless and cellular networks of the vehicle using WiFi Pineapple or a PC with network scanning software such as nmap or airmagnet.

2. Attacker finds the WiFi access point connecting to the human operator console and autonomous self-driving vehicle.

3. Attacker De-authenticates the devices connected to the WiFi access point.

Experiment Recording: `https://www.youtube.com/watch?v=YWg_tpIIpP0`

Experiment Results: A scan of all wireless networks was conducted using the Hak5 WiFi pineapple device. The WiFi pineapple can be considered a malicious access point that acts as a man-in-the-middle between the wireless network and the client device. It can scan, capture traffic and execute a number of attacks such as capturing passwords of insecure network protocols. Figure 69 presents the outcomes of the wireless network scan. The HUAWEI Y5 2018 network is identified as the vehicle network from analysing the signal strength and capturing the traffic. Figure **??** demonstrates the attacker selected the network to conduct the deauthentication attack.



eduroam	34:FA:9F:79:9F:78	WPA2	CCMP, TKIP	802.1x	3	2.422 Ghz	-71 dBm	56%	Capture	Deauth
eduroam	34:FA:9F:7A:19:B8	WPA2	CCMP, TKIP	802.1x	4	2.427 Ghz	-82 dBm	40%	Capture	Deauth
eduroam	34:FA:9F:79:FF:B8	WPA2	CCMP, TKIP	802.1x	7	2.442 Ghz	-37 dBm	100%	Capture	Deauth
eduroam	34:FA:9F:57:AE:F8	WPA2	CCMP, TKIP	802.1x	11	2.462 Ghz	-60 dBm	71%	Capture	Deauth
eduroam	34:FA:9F:7A:02:28	WPA2	CCMP, TKIP	802.1x	13	2.472 Ghz	-66 dBm	63%	Capture	Deauth
eduroam	34:FA:9F:57:AE:FC	WPA2	CCMP, TKIP	802.1x	36	5.18 Ghz	-71 dBm	56%	Capture	Deauth
eduroam	34:FA:9F:79:8A:FC	WPA2	CCMP, TKIP	802.1x	44	5.22 Ghz	-77 dBm	47%	Capture	Deauth
eduroam	34:FA:9F:79:FF:BC	WPA2	CCMP, TKIP	802.1x	44	5.22 Ghz	-48 dBm	89%	Capture	Deauth
eduroam	30:87:D9:5C:32:2C	WPA2	CCMP, TKIP	802.1x	52	5.26 Ghz	-88 dBm	31%	Capture	Deauth
eduroam	34:FA:9F:79:9E:8C	WPA2	CCMP, TKIP	802.1x	60	5.3 Ghz	-79 dBm	44%	Capture	Deauth
eduroam	34:FA:9F:79:D0:CC	WPA2	CCMP, TKIP	802.1x	116	5.58 Ghz	-87 dBm	33%	Capture	Deauth
HUAWEI Y5 2018	BA:94:36:80:21:B8	WPA2	CCMP	PSK	1	2.412 Ghz	-73 dBm	53%	Stop	Deauth
HUAWEI-B535-9C7C	44:D7:91:AE:9C:7C	WPA2	CCMP	PSK	10	2.457 Ghz	-80 dBm	43%	Capture	Deauth
MSI	34:FA:9F:B9:9F:78	Mixed WPA/WPA2	CCMP	PSK	3	2.422 Ghz	-72 dBm	54%	Capture	Deauth
MSI	34:FA:9F:B9:D0:CC	Mixed WPA/WPA2	CCMP	PSK	116	5.58 Ghz	-87 dBm	33%	Capture	Deauth
NMRI	00:19:5B:BF:79:53	Mixed WPA/WPA2	TKIP, CCMP	PSK	6	2.437 Ghz	-85 dBm	36%	Capture	Deauth
Sarghaua	34:FA:9F:7A:19:B9	WPA2	CCMP, TKIP	PSK	4	2.427 Ghz	-86 dBm	34%	Capture	Deauth
Sarghaua	34:FA:9F:79:FF:B9	WPA2	CCMP, TKIP	PSK	7	2.442 Ghz	-40 dBm	100%	Capture	Deauth
Sarghaua	34:FA:9F:79:FF:79	WPA2	CCMP, TKIP	PSK	7	2.442 Ghz	-85 dBm	36%	Capture	Deauth

Figure 69 – Scan of Wireless Networks

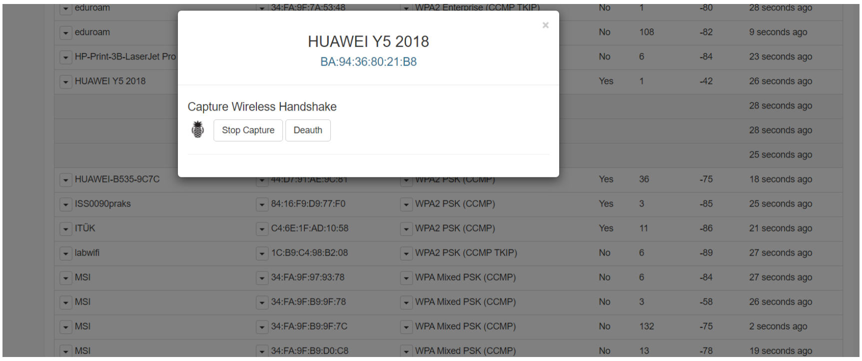


Figure 70 – De-Authentication of Vehicle WiFi Network

Figure 71 shows the workflow of the deauthentication attack. The attacker connects to the vehicle network, monitors the traffic and then deauthenticates the client, which in this case is the DuckieBot.

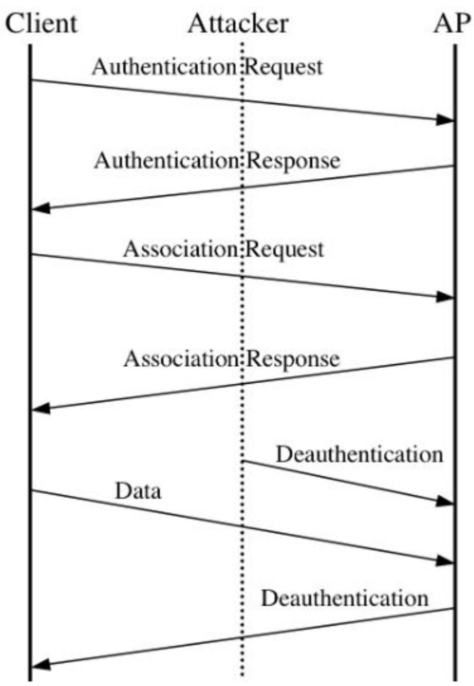


Figure 71 – Deauthentication workflow

[66, p.108]

The deauthentication attack was attempted twice. Both attempts were successful. Figure 72 shows the human remote operator console after it loses access to the network connection with the DuckieBoT and the DuckieBot accelerates off the road. Figure 73 shows the DuckieBot impacting the wall when it loses connectivity. The DuckieBot continues to accelerate on hitting the wall.

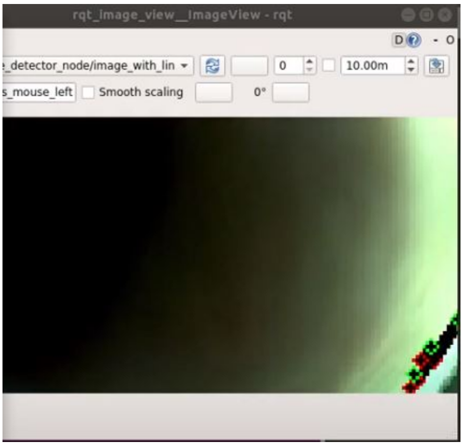


Figure 72 – Human Remote Operator Console View - WiFi Crash

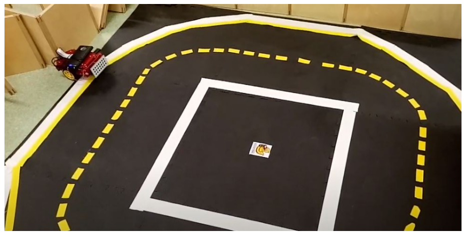


Figure 73 – WiFi Crash

Conclusion: The DDoS attack had the most impact due to lost of control of the human operator to safely stop the vehicle. Only with manual intervention to turn off the battery at the DuckieBot was the vehicle stopped. This demonstrates the catastrophic scenario, in a hybrid control mode, if the human control is lost, there is little that can be done to ensure the safety of the vehicle and it's occupants.

4.4.4 Environmental Perturbations

Test Case 8: Smoke from fire obscures the driving environment causing vehicle to take adverse driving behaviour.

Experiment Setup:

1. A 400w smoke machine is placed next to the environment. The smoke machine is filled with special liquid and then activated using the command controller. Smoke envelops the driving environment.

Note: This experiment was conducted with a fire extinguisher close by in case of fire.

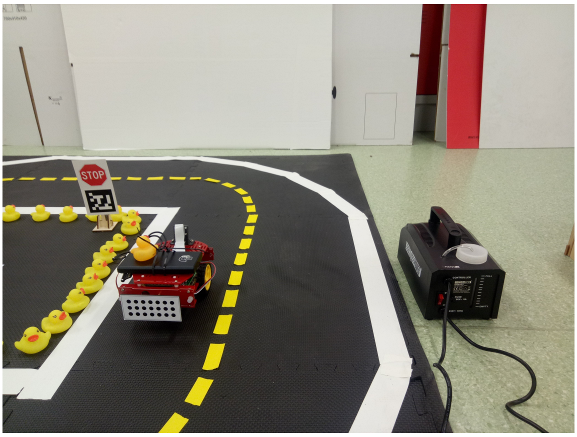


Figure 74 – Environmental Setup - Smoke Machine and DuckieTown

Experiment Recording: `https://www.youtube.com/watch?v=yLjuV5sMnwo`

Experiment Results: The experiments were conducted under three different lighting conditions: controlled lights, natural light, controlled dark lighting. In all lighting conditions the smoke was able to perturb the camera sensor to alter the driving path of the DuckieBot to crash out of the road environment.

The initial experimental tests, which were unsuccessful in altering the DuckieBot path, showed that the most important variables were the denseness of the smoke and the ability of the smoke to linger in the air to envelop the camera. The first three smoke experimental tests demonstrated the autonomous driving cognition being lost due to the smoke hazard, however, as the smoke stream was momentary, the detection of the lane markings were recovered in time to navigate accurately. Figure 75 shows the smoke perturbing the object detection of the lane markings and figure 76 displays how the object detection was recovered.

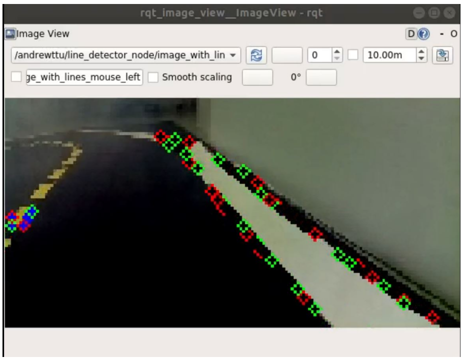


Figure 75 – Smoke - Test 5 External View

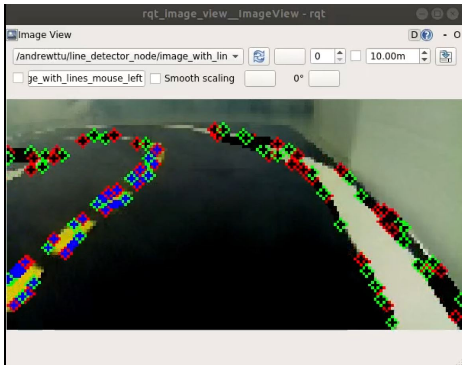


Figure 76 – Smoke - Test 5 DuckieBot Object Detection

Figure 77 and figure 78 shows the smoke affecting the autonomous driving cognition to the point were the DuckieBot is unable to recognise the lane markings.

Figure 77 – Smoke - Test 5 External View

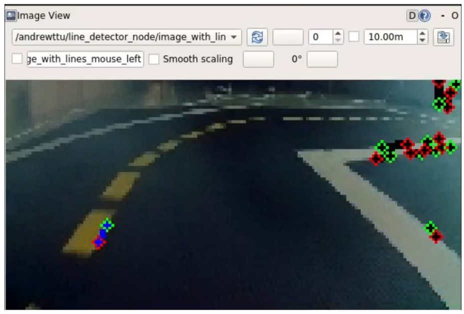


Figure 78 – Smoke - Test 5 DuckieBot Computer vision

Conclusion: The test case demonstrated the utility of the small-factor test environment in being able to simulate diverse environment conditions. Based on the results of the test case it may be possible to include safety testing in the scope of the test bed.

4.5 Test Case Feedback

The expert interviewees commented that the small-factor autonomous test bed was an innovative and creative solution for cybersecurity testing. The feedback of the DuckieBoT and DeepPi were that they were useful for test cases involving ROS and the drive algorithm and discovering edge cases for cybersecurity testing. To increase relevance of the small-factor test bed for operational vehicles, the small-factor concept needs to be extended to include embedded components such as ECUs and in-vehicular networks. Also, the multi-sensor fusion framework should be included in the architecture of the vehicle so sensor redundancy can be evaluated. Limitations of the small-factor test bed identified by the experts were the limited ability to simulate real-world environmental conditions such as snow storms and the speed of a real-world operational vehicle.

4.6 Discussion

4.6.1 How can a low-cost, small-factor, autonomous self-driving test bed be used for cybersecurity testing?

The test bed supported test cases provided by expert opinion and generated from a STRIDE analysis which included threats from the UNECE WP 29 threat catalogue. The test cases demonstrated that the test bed can allow for cybersecurity testing of the sensors and perception, computer & hardware and connected vehicle.

The small-factor test bed demonstrated it's use in validating the viability of proof-of-concept attacks such as that of the projector attack. Based on the results of the testing, it was able to be shown that the projector attack was very difficult to accomplish and had a low probability of success in the real-world.

The WiFi test case provided insights into possibilities for interoperability and human operator research. The vulnerabilities of the network interface, exploited in the cybersecurity test case, impacted the vehicle behaviour and human control.

4.6.2 How can a low-cost, small-factor autonomous self-driving vehicle and driving environment be designed?

Two autonomous self-driving vehicle were created for less than €300. The characteristics they shared with real world operational vehicles included the software systems, network interfaces and algorithmic control of driving behaviour. Small-factor autonomous self-driving vehicles.

In the design of small-factor vehicles physical properties are an important consideration. The DeepPi car's mechanical steering mechanism provides a more realistic comparison to real-world vehicles, whilst, the DuckieBot is able to use it's LEDs to drive in dark and low-lighting environments.

Adding additional hardware in the small-factor vehicle requires multiple upgrades to the architecture, such as; batteries, re-wiring, re-assembly of parts, cooling systems, data storage and memory. During the design, the configuration of the DuckieBot had to be changed as the components melted due to excessive heat. During the course of the design and experiments it took weeks of effort to reconfigure the DuckieBot and DeepPi car to replace components with upgraded versions. This effort, however, pales in comparison to the required effort to upgrade or change the design of a real-world operational vehicle.

4.6.3 How can cybersecurity testing of a small-factor autonomous self-driving vehicle test bed used to improve cybersecurity of the FinEst autonomous self-driving vehicles?

Control of the small-factor environment allowed greater diversity of cybersecurity testing with lower cost and less resources required. A fundamental proof of this is the test LKAS manipulation. In a real-world environment this would require repainting a road, the vehicle must be clear of obstacles and pedestrians and any damage to the vehicle would end the experiment. In the small-factor environment the experiment could be executed as many times as possible and the effort to achieve the setup of the testing scenario and repair any damage was minimal.

The modular nature of the small-factor environment allows features to be added as designs and technology of autonomous vehicles change. This is also true of the software systems. For autonomous vehicular projects of a research and development nature such as those used in the FinEst project, the small-factor test bed allows for agility in testing system design changes.

4.6.4 What are the limitations of test beds for autonomous self-driving vehicle cybersecurity testing?

The small-factor testbed cannot exactly replicate the architecture of a full-factor autonomous vehicle. Key differences are the diversity of embedded components and the limited computational resources of the small-factor vehicles. In the architecture of a full-factor autonomous vehicle the neural network will use resources locally, such as the NVIDIA Drive platform will be on-board the vehicle. This is opposed to the small-factor environment, which, due to it's limited computation resources must access resources in a cloud environment such as Google Colab.

4.6.5 Can automation and sensor failures caused by cyber attacks be identified using an experimental test bed?

As aforementioned, there is an increase in accidents of autonomous self-driving vehicle due to failures of object-detection and sensor and perception technology. The related work demonstrated how a cyber adversary could construct the same manipulations using adversarial tactics. One of the fundamental values of the small-factor environment for security testing demonstrated in the test case evaluation is that it can evaluate sensors and perception against a wide range of adversarial cyber threats and include damage incurring test cases.

5 Conclusion

5.1 Conclusion

This thesis sought to solve the problem of whether small-factor test beds could provide a viable option for the testing for cybersecurity of real-world operational autonomous vehicles such as those used on the streets of Tallinn to Helsinki. This was successfully proven with the development and evaluation of a test bed consisting of two small-factor autonomous self-driving vehicles and a driving environment. The design established that a small-factor autonomous self-driving test bed could be created, at low-cost, under €300, and resemble systems used on operational vehicles such as; ROS, network interfaces and drive control functionality.

The evaluation of the test bed using realistic test cases provided by experts proved that cybersecurity testing in the small-factor environment was viable and valuable in performing a variety of tests on sensors and perception, communication channels and hardware and compute. The results of the test cases demonstrated that vulnerabilities could be found in the small-factor environment that had relevance to the real-world environment. These findings can be used to improve the security of the vehicle to cyber attacks by implementation of defensive controls as well as increasing the awareness of automotive engineers and algorithm designers of the vulnerabilities of their systems.

Limitations of the test bed environment were that it couldn't fully replicate the diversity of electrical components, speed and environmental conditions of a real-world operational vehicle. Another major limitation in the use of small-factor autonomous vehicles is the limited computational resources available on-board. For robust, trained object-detection, the small-factor autonomous vehicle needs to utilise resources from the cloud for operation of the object-detection algorithm, storing of training data and to alleviate resource usage locally on the small-factor vehicle. As this is one of the first such studies into small-factor test beds, the development and innovation of small-factor autonomous vehicles may bridge this gap.

As identified in the case of the Tesla crash in Florida and the Uber crash in Arizona, integrity of sensors and the autonomous driving algorithm is of predominant importance for safety and security of the autonomous vehicle and it's passengers. The evaluation of the test bed demonstrated that cyber attacks that impact the sensors and perception of an autonomous vehicle could be replicated in a small-factor environment. The contribution of the small-factor test bed artifact and the methods outlined in the test cases provide a tangible contribution that autonomous system designers can use to validate vulnerabilities in sensors and perception to prevent events such as the aforementioned occurring in real-world traffic.

The code for the DeepPi vehicle has been published under an open source license and can be found in the following link: `https://gitlab.com/Self-DrivingRoberts/experimental-testbed/-/tree/master/public/DeepPiCar`. The video demonstrations for the cybersecurity test cases is publically demonstrated on YouTube and can be found in the following link: `https://www.youtube.com/channel/UC7cXB9DSG6UCQAYHw4vkrSQ/videos`

5.2 Future Work

As the contribution of this thesis had a practical objective of improving the cybersecurity of vehicles in the FINEST Twins Center of Excellence project, the next phase of this work will be to build a small-factor version of the TalTech ISEAUTO autonomous vehicle. The next phase will attempt to emulate the full sensor profile of the ISEAUTO, in-vehicular networks such as CAN and embedded components. The new small-factor test bed environment will also be tested to support new cybersecurity testing process methodologies within the working of the International Alliance for Mobility Testing and Standardisation (IAMTS) Working Group 4 - Cybersecurity.

The DuckieTown test bed environment will also be extended to include v2v, v2i and v2x network interfaces. The aim will be to increase the functionality of the test bed and conduct research of; digital forensics and human operator cybersecurity awareness.

References

[1] Anton Tammsaare. *Tõde ja õigus*. 1932.

[2] Praveena Penmetsa, Emmanuel Kofi Adanu, Dustin Wood, Teng Wang, and Steven L. Jones. Perceptions and expectations of autonomous vehicles – a snapshot of vulnerable road user opinion. *Technological Forecasting and Social Change*, 143:9 – 13, 2019.

[3] R. Mariani. An overview of autonomous vehicles safety. In *2018 IEEE International Reliability Physics Symposium (IRPS)*, pages 6A.1–1–6A.1–6, March 2018.

[4] A.Davies (2019, May.). Tesla's Latest Autopilot Death Looks Just Like a Prior Crash, WIRED Magazine. Accessed 20 March, 2020. [Online]. Available:`https://www.wired.com/story/teslas-latest-autopilot-death-looks-like-prior-crash/`.

[5] A.Marshall (2018, Mar.). Uber's Self-Driving Car Just Killed Somebody. Now What?, WIRED Magazine. Accessed 20 May, 2020. [Online]. Available:`https://www.wired.com/story/uber-self-driving-car-crash-arizona-pedestrian/`.

[6] C.Miller & C.Valasek (2015, Aug.). Remote Exploitation of an Unaltered Passenger Vehicle. Accessed 20 June, 2020. [Online]. Available:`http://illmatics.com/Remote%20Car%20Hacking.pdf`.

[7] Ralf-Martin Soe. Finest twins: Platform for cross-border smart city solutions. In *Proceedings of the 18th Annual International Conference on Digital Government Research*, dg.o '17, page 352–357, New York, NY, USA, 2017. Association for Computing Machinery.

[8] D. Zelle, R. Rieke, C. Plappert, C. Krauß, D. Levshun, and A. Chechulin. Sepad – security evaluation platform for autonomous driving. In *2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 413–420, 2020.

[9] Jacopo Tani, Liam Paull, Maria T. Zuber, Daniela Rus, Jonathan How, John Leonard, and Andrea Censi. Duckietown: An innovative way to teach autonomy. In Dimitris Alimisis, Michele Moro, and Emanuele Menegatti, editors, *Educational Robotics in the Makers Era*, pages 104–121, Cham, 2017. Springer International Publishing.

[10] C. Herley and P. C. Van Oorschot. Sok: Science, security and the elusive goal of security as a scientific pursuit. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 99–120, 2017.

[11] EU Horizon 2020 Research and Innovation Programme. Future Automated Bus Urban Level Operation System, (May 20, 2020). ://fabulos.eu/.

[12] S.Cantrill (2013, Jun.). Speaking Frankly: The allure of Pasteur's quadrant. Accessed 20 March, 2020. [Online]. Available:http://blogs.nature.com/thescepticalchymist/2013/06/speaking-frankly-the-allure-of-pasteurs-quadrant.html.

[13] Ken Peffers, Tuure Tuunanen, Marcus Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *J. Manage. Inf. Syst.*, 24(3):45–77, December 2007.

[14] Jonathan Petit and Steven Shladover. Potential cyberattacks on automated vehicles. *Intelligent Transportation Systems, IEEE Transactions on*, PP:1–11, 09 2014.

[15] Abasi-Amefon O. Affia, Raimundas Matulevicius, and Alexander Nolte. Security risk management in cooperative intelligent transportation systems: A systematic literature review. In Hervé Panetto, Christophe Debruyne, Martin Hepp, Dave Lewis, Claudio Agostino Ardagna, and Robert Meersman, editors, *On the Move to Meaningful Internet Systems: OTM 2019 Conferences - Confederated International Conferences: CoopIS, ODBASE, C&TC 2019, Rhodes, Greece, October 21-25, 2019, Proceedings*, volume 11877 of *Lecture Notes in Computer Science*, pages 282–300. Springer, 2019.

[16] S. Parkinson, P. Ward, K. Wilson, and J. Miller. Cyber threats facing autonomous and connected vehicles: Future challenges. *IEEE Transactions on Intelligent Transportation Systems*, 18(11):2898–2915, Nov 2017.

[17] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, page 6, USA, 2011. USENIX Association.

[18] V. L. L. Thing and J. Wu. Autonomous vehicle security: A taxonomy of attacks and defences. In *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 164–170, Dec 2016.

[19] Sghiri Meryem and Tomader Mazri. Security study and challenges of connected autonomous vehicles. In *Proceedings of the 4th International Conference on Smart City Applications*, SCA '19, New York, NY, USA, 2019. Association for Computing Machinery.

[20] K. Ren, Q. Wang, C. Wang, Z. Qin, and X. Lin. The security of autonomous driving: Threats, defenses, and future directions. *Proceedings of the IEEE*, 108(2):357–372, 2020.

[21] European Union Agency for Cybersecurity (2019, November.). ENISA good practices for security of Smart Cars. Accessed 20 March, 2020. [Online]. Available:`https://www.enisa.europa.eu/publications/smart-cars`.

[22] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on machine learning models. 07 2017.

[23] Ben Nassi, Dudi Nassi, Raz Ben-Netanel, Yisroel Mirsky, Oleg Drokin, and Yuval Elovici. Phantom of the adas: Phantom attacks on driver-assistance systems. Cryptology ePrint Archive, Report 2020/085, 2020. `https://eprint.iacr.org/2020/085`.

[24] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z. Morley Mao. Adversarial sensor attack on lidar-based perception in autonomous driving. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 2267–2281, New York, NY, USA, 2019. Association for Computing Machinery.

[25] Takami Sato, Junjie Shen, Ningfei Wang, Yunhan Jia, Xue Lin, and Qi Alfred Chen. Security of deep learning based lane keeping system under physical-world adversarial attack, 03 2020.

[26] Drew Davidson, Hao Wu, Rob Jellinek, Vikas Singh, and Thomas Ristenpart. Controlling uavs with sensor input spoofing attacks. In *10th USENIX Workshop on Offensive Technologies (WOOT 16)*, Austin, TX, August 2016. USENIX Association.

[27] SAVIOR: Securing autonomous vehicles with robust physical invariants. In *29th USENIX Security Symposium (USENIX Security 20)*, Boston, MA, August 2020. USENIX Association.

[28] Se-Yeon Jeong, I-Ju Choi, Yeong-Jin Kim, Yong-Min Shin, Jeong-Hun Han, Goo-Hong Jung, and Kyoung-Gon Kim. A study on ros vulnerabilities and countermeasure. In *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '17, page 147–148, New York, NY, USA, 2017. Association for Computing Machinery.

[29] Nils Weiss, Markus Schrötter, and Rudolf Hackenberg. On threat analysis and risk estimation of automotive ransomware. In *ACM Computer Science in Cars Symposium*, CSCS '19, New York, NY, USA, 2019. Association for Computing Machinery.

[30] Ishtiaq Rouf, Rob Miller, Hossen Mustafa, Travis Taylor, Sangho Oh, Wenyuan Xu, Marco Gruteser, Wade Trappe, and Ivan Seskar. Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, page 21, USA, 2010. USENIX Association.

[31] S. Tbatou, A. Ramrami, and Y. Tabii. Security of communications in connected cars modeling and safety assessment. In *Proceedings of the 2nd International Conference on Big Data, Cloud and Applications*, BDCA'17, New York, NY, USA, 2017. Association for Computing Machinery.

[32] Road vehicles - Safety of the intended functionality. ISO/PAS 21448:2019. Accessed: 20 June, 2020. [Online]. Available:. `https://www.iso.org/standard/70939.html`.

[33] Road Vehicles - Functional Safety. ISO 26262. Accessed: 20 June, 2020. [Online]. Available:. `https://www.iso.org/standard/68383.html`.

[34] Road vehicles — Cybersecurity engineering. ISO/SAE DIS 21434. Accessed: 20 June, 2020. [Online]. Available:. `https://www.iso.org/standard/70918.html`.

[35] Information technology — Security techniques — Evaluation criteria for IT security (Common Criteria). ISO/IEC 15408-1:2009. Accessed: 20 June, 2020. [Online]. Available:. `https://www.iso.org/standard/50341.html`.

[36] Hardware Protected Security for Ground Vehicles. SAE J3061. Accessed: 20 June, 2020. [Online]. Available:. `https://www.sae.org/standards/content/j3061_201601/`.

[37] Cybersecurity Guidebook for Cyber-Physical Vehicle Systems. SAE J3061. Accessed: 20 June, 2020. [Online]. Available:. `https://www.sae.org/standards/content/j3061/`.

[38] Intelligent Transport Systems; Security. ETSI TS 102 940 - 102 943. Accessed: 20 June, 2020. [Online]. Available:. `https://www.etsi.org/deliver/etsi_ts/102900_102999/102943/01.01.01_60/ts_102943v010101p.pdf`.

[39] Automotive Cybersecurity Management System Audit. VDA-QMC AK ACSMS. Accessed: 20 June, 2020. [Online]. Available:. `https://vda-qmc.de/fileadmin/redakteur/Publikationen/Gelbdrucke/VDA_Yellow_Volume_ACSMS_EN_1_edition_2020.pdf`.

[40] The fundamental principles of automotive cyber security. BSI PAS 1885:2018.

[41] Connected automotive ecosystems. Impact of security on safety.Code of practice. BSI PAS 11281:2018.

[42] David Förster, Claudia Loderhose, Thomas Bruckschlögl, and Franziska Wiemer. Safety goals in vehicle security analyses. In *17th escar Europe : embedded security in cars (Konferenzveröffentlichung)*, a method to assess malicious attacks with safety impact. 2019.

[43] Alastair Ruddle, David Ward, Benjamin Weyl, Sabir Idrees, Yves Roudier, Michael Friedewald, Timo Leimbach, Andreas Fuchs, Sigrid Gürgens, Olaf Henniger, Roland Rieke, Matthias Ritscher, Henrik Broberg, Ludovic Apvrille, Renaud Pacalet, and Gabriel Pedroza. Security requirements for automotive on-board networks based on dark-side scenarios, 01 2009.

[44] Alexandr Vasenev., Florian Stahl., Hayk Hamazaryan., Zhendong Ma., Lijun Shan., Joerg Kemmerich., and Claire Loiseaux. Practical security and privacy threat analysis in the automotive domain: Long term support scenario for over-the-air updates. In *Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems - Volume 1: VEHITS,*, pages 550–555. INSTICC, SciTePress, 2019.

[45] L.Shan (2019, April.). D10: State-of-the-art Analysis and Applicability of Standards". Accessed 20 July, 2020. [Online]. Available:`https://secredas-project.eu/wp-content/uploads/2017/01/SECREDAS-D10-2.pdf`.

[46] J. Axelsson, A. Kobetski, Z. Ni, S. Zhang, and E. Johansson. Moped: A mobile open platform for experimental design of cyber-physical systems. In *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 423–430, 2014.

[47] D.Tian (2019, April.). Deep Learning, Self Driving Robotic Car on a Shoestring Budget. Accessed 20 March, 2020. [Online]. Available:`https://towardsdatascience.com/deeppicar-part-1-102e03c83f2c`.

[48] Rahul Bhadani, Jonathan Sprinkle, and Matthew Bunting. The CAT Vehicle Testbed: A Simulator with Hardware in the Loop for Autonomous Vehicle Applications. In *Proceedings 2nd International Workshop on Safe Control of Autonomous Vehicles (SCAV 2018), Porto, Portugal, Electronic Proceedings in Theoretical Computer Science*, volume 269, 04/2018 2018.

[49] Eduardo Sens dos Santos. Towards a simulationbased framework for the security testing of autonomous vehicles. 2018.

[50] Convention on Road Traffic, Vienna, 8 November 1968. Available:. https://www.unece.org/fileadmin/DAM/trans/conventn/crt1968e.pdf. Online; Accessed 02 March, 2020.

[51] Traffic Act, RTI, 07.05.2020.30. Available:. https://www.riigiteataja.ee/en/compare_original?id=519052020004. Online; Accessed 02 July, 2020.

[52] Republic of Estonia: GCIO Office (2019, May.). Artificial Intelligence for Estonia. Accessed 20 March, 2020. [Online]. Available:https://www.kratid.ee/in-english.

[53] Independent High-Level Group on Artificial Intelligence (2019, April.). Ethics Guidelines for Trustworthy AI. Accessed 20 March, 2020. [Online]. Available:https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai.

[54] T.Wessing (2019, Apr.). Who is in the driver's seat? Data protection challenges in the connected car landscape", lexology. Accessed 22 May, 2020. [Online]. Available:https://www.lexology.com/library/detail.aspx?g=b77a4401-e042-4de7-b416-ab7ddd9fdc8b.

[55] Draft Recommendation on Cyber Security of the Task Force on Cyber Security and Over-the-air issues of UNECE WP.29 GRVA, Geneva, 20 September 2018. Available:. https://www.unece.org/fileadmin/DAM/trans/doc/2018/wp29grva/GRVA-01-17.pdf. Online; Accessed 02 May, 2020.

[56] J. Gerdes and Sarah Thornton. *Implementable Ethics for Autonomous Vehicles*, pages 87–102. 05 2016.

[57] Maximilian Wächter, Anja Faulhaber, Felix Blind, Silja Timm, Anke Dittmer, Leon Sütfeld, Achim Stephan, Gordon Pipa, and Peter König. Human decisions in moral dilemmas are largely described by utilitarianism: virtual car driving study provides guidelines for advs. *Science and Engineering Ethics*, 25, 06 2017.

[58] Patrick Lin. *Why Ethics Matters for Autonomous Cars*, pages 69–85. 05 2016.

[59] Federal Ministry of Transportation and Digital Infrastructure. " ETHICS COMMISSION AUTOMATED AND CONNECTED DRIVING", (2017, June). https://www.bmvi.de/SharedDocs/EN/publications/report-ethics-commission.pdf?__blob=publicationFile. Online; Accessed 02 March, 2020.

[60] Cara Bloom, Joshua Tan, Javed Ramjohn, and Lujo Bauer. Self-driving cars and data collection: Privacy perceptions of networked autonomous vehicles. In *Thirteenth*

Symposium on Usable Privacy and Security (SOUPS 2017), pages 357–375, Santa Clara, CA, July 2017. USENIX Association.

[61] Samantha Reig, Selena Norman, Cecilia G. Morales, Samadrita Das, Aaron Steinfeld, and Jodi Forlizzi. A field study of pedestrians and autonomous vehicles. In *Proceedings of the 10th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, AutomotiveUI '18, page 198–209, New York, NY, USA, 2018. Association for Computing Machinery.

[62] Ningfei Wang Yunhan Jack Jia Xue Lin Takami Sato, Junjie Shen and Qi Alfred Chen. Poster: Security of deep learning based lane keeping assistance system under physical-world adversarial attack. In *The Network and Distributed System Security Symposium 2020*, NDSS '20, 2020.

[63] R.Sell. *ISEAUTO Hardware Diagram V2*. 2019.

[64] A. Hahn, A. Ashok, S. Sridhar, and M. Govindarasu. Cyber-physical security testbeds: Architecture, application, and evaluation for smart grid. *IEEE Transactions on Smart Grid*, 4(2):847–855, 2013.

[65] Luz Bayona-Ore, Ronald Fernández Zavala, and María Luyo Cruz. Expert opinion process: Applications in education. In *Proceedings of the 10th International Conference on Education Technology and Computers*, ICETC '18, page 172–176, New York, NY, USA, 2018. Association for Computing Machinery.

[66] Haitham Ameen, Mohd Shahidan, and Haydar Mohammed. An automated approach to detect deauthentication and disassociation dos attacks on wireless 802.11 networks. 06 2015.

[67] Jonathan Petit and Steven Shladover. Potential cyberattacks on automated vehicles. *Intelligent Transportation Systems, IEEE Transactions on*, PP:1–11, 09 2014.

Appendix 1 - Attack Surfaces in Autonomous Automated Vehicles

TABLE I
ATTACK SURFACES IN AUTONOMOUS AUTOMATED VEHICLE

Target	Means	Feasibility of the attack	Physical access	Ease of detection by driver	Ease of detection by system	Probability of success	Direct consequence(s)	Hazard created	Mitigation technique
Infrastructure sign	change sign (fake, irrelevant)	low	n/a	high	low	low-medium	false reaction	traffic disturbance	harden infrastructure sign change; map database of sign in-vehicle; driver reporting
	alter (change speed), make it unreadable	high	n/a	high	low	low-medium	false/no reaction	traffic disturbance	harden infrastructure sign change; map database; driver reporting
	remove (e.g. stop sign)	high	n/a	high	low	low-medium	no reaction	traffic disturbance	harden infrastructure sign change; map database; driver reporting
Machine vision	blind (only source of information)	high	no	medium	high	high	degraded mode	driver disturbance	multiple cameras with different angle
	blind (other source of information available)	high	no	medium	high	high	turn off the camera	none	n/a
	fake picture/emergency brake light (only source of information)	low	no	medium	low	medium	false reaction	driver disturbance	other source of data
	fake picture/emergency brake light (other source of information available)	low	no	medium	low	medium	false reaction	driver disturbance	n/a
GPS	spoofing	high	no	low	medium	high	wrong positioning	traffic disturbance or crash hazard	authentication
	jamming	high	no	low	medium to high	high	no accurate positioning information available	need to stop vehicle unless other location info sources available	Anti-Jam GPS techniques, high-quality IMU
In-vehicle devices	inject malware	medium	yes for USB, no for others	low	medium	medium	depends on malware's capability	depends on malware's capability	Separation infotainment/safety buses; Intrusion Detection System/Anti-virus/Firewall
	head unit attack	medium	yes	high*	medium	medium	display unexpected information	driver disturbance	Protection of display of safety status information
Acoustic sensor	interference (electromagnetic, loud sound, inaudible)	medium	no	low to medium	low	low	turn off the sensor	n/a	filter; spectrum analysis
	fake crash sound	high	no	low to medium	low	low	false reaction	traffic disturbance	other source of data (e.g. radar)
	fake ultrasonic reflection	medium	no	low	low	low	false positive or false negative obstacle detection	traffic disturbance or low-speed crash	other source of data (e.g. lidar)
Radar	chaff	medium	no	medium	high	medium	degraded mode	traffic disturbance	filter; other source of data
	smart material (non reflective surface, invisible object)	low	no	medium	low	medium	no detection of surroundings	collision	other source of data
	jamming (saturation with noise)	high	no	low	high	medium	turn off radar/degraded mode	traffic disturbance	filter; other source of data
	ghost vehicle (signal repeater)	high	no	medium*	medium	medium	false detection	traffic disturbance	filter; other source of data
Lidar	jamming	high	no	low	high	medium	turn off lidar/degraded mode	loss of situation awareness by vehicle	filter; other source of data
	smart material (absorbent, reflective)	high	no	medium*	medium	medium	false detection (e.g. fake delineation)	traffic disturbance	filter; other source of data
Road	modify delineation	low	n/a	medium	low	low	false detection	traffic disturbance	driver reporting
	hack smart lane LEDs	low	n/a	low	low	low	false detection	traffic disturbance	
in-vehicle sensors	eavesdropping (tire pressure, bluetooth)	high	no	low	low	medium	privacy leak	none	in-vehicle security
	eavesdropping CAN bus	high	yes	medium	low	medium	reverse engineering	none	in-vehicle security
	inject CAN messages	medium	yes	medium	high	medium	false message from internal sensors	driver/traffic disturbance	in-vehicle security
Odometric sensors	magnetic attack	high	yes	low	low	medium	wrong position/navigation	traffic disturbance	other source of data
	thermal attack of gyroscope	medium	yes	low	low	low	wrong position/navigation	traffic disturbance	casing; other source of data

Electronic device(s)	EMP	low	no	low	high	medium	temporary to permanent damage to electronic components	disabling vehicle automation	EMP protection
Maps	Map poisoning	low	no	low	medium	medium	wrong maneuver	traffic disturbance, accident	authentication of maps server

[67]

Appendix 2 - ENISA Smart Car Attack Scenarios

Table 1: Smart cars attack scenarios

ATTACK SCENARIOS	SEVERITY[44]
1. **Vulnerability exploit in a communication stack**: exploitation of a vulnerability in a communication stack of an in-vehicle network (e.g. no protection mechanism against replay attacks, lack of authentication, etc.) can lead to severe issues such as critical ECU reprogramming and taking control the vehicle over the Controller Area Network (CAN bus).	High
2. **Mobile car application[45] being hacked/attacked allowing access to the car**: by hacking the mobile application, an attacker could order a car to drive him somewhere although he is not allowed to do so.	High
3. **Attack on remote servers to influence car behaviours**: several attack scenarios exist regarding remote servers. For instance, an attacker could compromise map data with the aim to affect plausibility checks, or even alter data on traffic conditions to change the current car itinerary resulting in an inefficient service.	High
4. **Fake communication unit to compromise telematics unit and deploy rogue firmware:** use of malicious communication unit, such as Base Transceiver Station (BTS), Wi-Fi router, RSU, with the objective to spread a malware or just disrupting the infrastructure communications.	High
5. **Large scale deployment of rogue firmware after hacking OEM back-end servers:** penetration of OEM back-end servers with the aim to initiate malicious firmware updates could lead to devastating results as this kind of attacks is highly-scalable.	High
6. **Hacking an RSU with the aim to spread wrong traffic and safety messages:** as RSUs constitute an important part of the autonomous vehicles' ecosystem, they could be the target of hackers in order to create traffic jams or other kind of disruptions.	High – Medium
7. **Rogue vehicle sending wrong information through V2V interfaces:** vehicles unknown from the infrastructure (e.g. counterfeit cars) that are deployed to decrease the safety level by sending wrong information about traffic conditions and other functionalities (i.e. fake information with the aim to update map data).	Medium

8. Sensor fooling by adversarial perturbation: attack scenarios to disrupt the sensors' proper functioning by different means depending on the targeted sensor (e.g. flash the camera, relay the light waves from the LiDAR).	Medium – Low
9. Communication jamming: producing radio interferences to disrupt wireless networks so the vehicles cannot emit or receive V2X messages.	Low
10. GNSS spoofing: by replacing GNSS signals, an attacker can fool a third-party service into thinking that the vehicle is elsewhere in either time or location. This can lead to accident or vehicle theft.	Medium
11. Blocking critical messages at automation level 4: an attacker can block critical messages, such as Denial of a Service (DoS) attack, and prevent the semi-autonomous vehicle (or driver) from reacting appropriately to the situation (e.g. apply the brakes, warn the driver that he needs to take control of the vehicle, etc.).	High

[21]

Appendix 3 - UNECE Threat Catalogue

High level and sub-level descriptions of vulnerability/ threat			**Example of vulnerability or attack method**	
4.3.1 Threats regarding back-end servers	1	Back-end servers used as a means to attack a vehicle or extract data	1.1	Abuse of privileges by staff **(insider attack)**
			1.2	**Unauthorised internet access** to the server (enabled for example by backdoors, unpatched system software vulnerabilities, SQL attacks or other means)
			1.3	**Unauthorised physical access** to the server (conducted by for example USB sticks or other media connecting to the server)

High level and sub-level descriptions of vulnerability/ threat			Example of vulnerability or attack method	
	2	Services from back-end server being disrupted, affecting the operation of a vehicle	2.1	**Attack on back-end server stops it functioning**, for example it prevents it from interacting with vehicles and providing services they rely on
	3	Data held on back-end servers being lost or compromised ("data breach")	3.1	Abuse of privileges by staff (**insider attack**)
			3.2	**Loss of information in the cloud**. Sensitive data may be lost due to attacks or accidents when data is stored by third-party cloud service providers
			3.3	**Unauthorised internet access to the server** (enabled for example by backdoors, unpatched system software vulnerabilities, SQL attacks or other means)
			3.4	**Unauthorised physical access to the server** (conducted for example by USB sticks or other media connecting to the server)
			3.5	**Information breach** by unintended sharing of data (e.g. admin errors, storing data in servers in garages)
4.3.2 Threats to vehicles regarding their communication channels	4	Spoofing of messages or data received by the vehicle	4.1	**Spoofing of messages** by impersonation (e.g. 802.11p V2X during platooning, GNSS messages, etc.)
			4.2	**Sybil attack** (in order to spoof other vehicles as if there are many vehicles on the road)
	5	Communication channels used to conduct unauthorized manipulation, deletion or other amendments to vehicle held code/data	5.1	Communications channels permit **code injection**, for example tampered software binary might be injected into the communication stream
			5.2	Communications channels permit **manipulate** of vehicle held data/code
			5.3	Communications channels permit **overwrite** of vehicle held data/code
			5.4	Communications channels permit **erasure** of vehicle held data/code
			5.5	Communications channels permit introduction of data/code to the vehicle (write data code)
	6	Communication channels permit untrusted/unreliable messages to be accepted or are vulnerable to session hijacking/replay attacks	6.1	Accepting information from an **unreliable or untrusted source**
			6.2	**Man in the middle** attack/ session hijacking
			6.3	**Replay attack**, for example an attack against a communication gateway allows the attacker to downgrade software of an ECU or firmware of the gateway
	7	Information can be readily disclosed. For example through	7.1	**Interception of information** / interfering radiations / monitoring communications

High level and sub-level descriptions of vulnerability/ threat			Example of vulnerability or attack method	
		eavesdropping on communications or through allowing unauthorized access to sensitive files or folders	7.2	Gaining **unauthorised access** to files or data
	8	Denial of service attacks via communication channels to disrupt vehicle functions	8.1	**Sending** a large number of garbage **data** to vehicle information system, **so that it is unable to provide services** in the normal manner
			8.2	**Black hole attack**, in order to disrupt communication between vehicles the attacker is able to block messages between the vehicles
	9	An unprivileged user is able to gain privileged access to vehicle systems	9.1	An unprivileged user is able to **gain privileged access**, for example root access
	10	Viruses embedded in communication media are able to infect vehicle systems	10.1	**Virus** embedded in communication media infects vehicle systems
	11	Messages received by the vehicle (for example X2V or diagnostic messages), or transmitted within it, contain malicious content	11.1	Malicious **internal** (e.g. CAN) **messages**
			11.2	Malicious **V2X messages,** e.g. infrastructure to vehicle or vehicle-vehicle messages (e.g. CAM, DENM)
			11.3	Malicious diagnostic messages
			11.4	Malicious **proprietary messages** (e.g. those normally sent from OEM or component/system/function supplier)
4.3.3. Threats to vehicles regarding their update procedures	12	Misuse or compromise of update procedures	12.1	Compromise of **over the air software update procedures**, This includes fabricating system update program or firmware
			12.2	Compromise of **local/physical software update procedures**. This includes fabricating system update program or firmware
			12.3	The **software** is **manipulated before the update process** (and is therefore corrupted), although the update process is intact
			12.4	**Compromise** of cryptographic keys of the software provider **to allow invalid update**
	13	It is possible to deny legitimate updates	13.1	Denial of Service attack against update server or network to **prevent rollout of critical software updates** and/or unlock of customer specific features
4.3.4 Threats to vehicles regarding unintended human	14	Misconfiguration of equipment or systems by legitimate actor, e.g. owner or maintenance	14.1	**Misconfiguration of equipment** by maintenance community or owner during installation/repair/use causing unintended consequence

High level and sub-level descriptions of vulnerability/ threat			Example of vulnerability or attack method	
actions		community	14.2	**Erroneous use** or administration of devices and systems (incl. OTA updates)
	15	Legitimate actors are able to take actions that would unwittingly facilitate a cyber-attack	15.1	Innocent victim (e.g. owner, operator or maintenance engineer) being **tricked into taking an action** to unintentionally load malware or enable an attack
			15.2	**Defined security procedures** are not followed
4.3.5 Threats to vehicles regarding their external connectivity and connections	16	Manipulation of the connectivity of vehicle functions enables a cyber-attack, this can include telematics; systems that permit remote operations; and systems using short range wireless communications	16.1	Manipulation of **functions designed to remotely operate systems**, such as remote key, immobiliser, and charging pile
			16.2	**Manipulation of vehicle telematics** (e.g. manipulate temperature measurement of sensitive goods, remotely unlock cargo doors)
			16.3	Interference with **short range wireless systems** or sensors
	17	Hosted 3rd party software, e.g. entertainment applications, used as a means to attack vehicle systems	17.1	**Corrupted applications**, or those with poor software security, used as a method to attack vehicle systems
	18	Devices connected to external interfaces e.g. USB ports, OBD port, used as a means to attack vehicle systems	18.1	**External interfaces** such as USB or other ports used as a point of attack, for example through code injection
			18.2	Media infected with a **virus** connected to a vehicle system
			18.3	**Diagnostic access (e.g. dongles in OBD port)** used to facilitate an attack, e.g. manipulate vehicle parameters (directly or indirectly)
4.3.6 Potential targets of, or motivations for, an attack	19	Extraction of vehicle data/code	19.1	Extraction of copyright or proprietary software from vehicle systems (product **piracy**)
			19.2	Unauthorized access to the **owner's privacy information** such as personal identity, payment account information, address book information, location information, vehicle's electronic ID, etc.
			19.3	Extraction of cryptographic keys
	20	Manipulation of vehicle data/code	20.1	Illegal/unauthorised changes to **vehicle's electronic ID**
			20.2	**Identity fraud.** For example if a user wants to display another identity when communicating with toll systems, manufacturer backend
			20.3	Action to **circumvent monitoring systems** (e.g. hacking/ tampering/ blocking of messages such as ODR Tracker data, or number of runs)

High level and sub-level descriptions of vulnerability/ threat			Example of vulnerability or attack method	
			20.4	Data manipulation to **falsify vehicle's driving data** (e.g. mileage, driving speed, driving directions, etc.)
			20.5	Unauthorised changes to **system diagnostic data**
	21	Erasure of data/code	21.1	Unauthorized deletion/manipulation of **system event logs**
	22	Introduction of malware	22.2	Introduce **malicious software** or malicious software activity
	23	Introduction of new software or overwrite existing software	23.1	**Fabrication of software** of the vehicle control system or information system
	24	Disruption of systems or operations	24.1	**Denial of service**, for example this may be triggered on the internal network by flooding a CAN bus, or by provoking faults on an ECU via a high rate of messaging
	25	Manipulation of vehicle parameters	25.1	Unauthorized access of **falsify the configuration parameters** of vehicle's key functions, such as brake data, airbag deployed threshold, etc.
			25.2	Unauthorized access of **falsify the charging parameters**, such as charging voltage, charging power, battery temperature, etc.
4.3.7 Potential vulnerabilities that could be exploited if not sufficiently protected or hardened	26	Cryptographic technologies can be compromised or are insufficiently applied	26.1	Combination of short **encryption keys** and long period of validity enables attacker to break encryption
			26.2	Insufficient use of cryptographic algorithms to protect sensitive systems
			26.3	Using already or soon to be deprecated **cryptographic algorithms**
	27	Parts or supplies could be compromised to permit vehicles to be attacked	27.1	**Hardware or software, engineered to enable an attack** or fails to meet design criteria to stop an attack
	28	Software or hardware development permits vulnerabilities	28.1	**Software bugs**. The presence of software bugs can be a basis for potential exploitable vulnerabilities. This is particularly true if software has not been tested to verify that known bad code/bugs is not present and reduce the risk of unknown bad code/bugs being present.
			28.2	**Using remainders** from development (e.g. debug ports, JTAG ports, microprocessors, development certificates, developer passwords, …) can permit access to ECUs or permit attackers to gain higher privileges
	29	Network design introduces vulnerabilities	29.1	**Superfluous internet ports left open**, providing access to network systems

High level and sub-level descriptions of vulnerability/ threat			**Example of vulnerability or attack method**	
			29.2	Circumvent **network separation** to gain control. Specific example is the use of unprotected gateways, or access points (such as truck-trailer gateways), to circumvent protections and gain access to other network segments to perform malicious acts, such as sending arbitrary CAN bus messages
	30	Physical loss of data can occur	30.1	**Damage** caused by a third party. Sensitive data may be lost or compromised due to physical damages in cases of traffic accident or theft
			30.2	Loss from **DRM** (digital right management) conflicts. User data may be deleted due to DRM issues
			30.3	The (integrity of) sensitive data may be lost due to IT **components wear and tear**, causing potential cascading issues (in case of key alteration, for example)
	31	Unintended transfer of data can occur	31.1	Information breach. Private or sensitive data may be leaked when the **car changes user** (e.g. is sold or is used as hire vehicle with new hirers)
	32	Physical manipulation of systems can enable an attack	32.1	**Manipulation of OEM hardware**, e.g. unauthorised hardware added to a vehicle to enable "man-in-the-middle" attack

[55]

Appendix 4 - DuckieBot LKAS Code

This code was created by the DukieTown[9] project and is not a contribution of the author.

```
import numpy as np
import cv2

from .line_detector_interface import Detections, LineDetectorInterface
import duckietown_utils as dtu


class LineDetector2Dense(dtu.Configurable, LineDetectorInterface):
    def __init__(self, configuration):
        # Images to be processed
        self.bgr = np.empty(0)
        self.hsv = np.empty(0)
        self.edges = np.empty(0)

        param_names = [

            'hsv_white1',
            'hsv_white2',
            'hsv_yellow1',
            'hsv_yellow2',
            'hsv_red1',
            'hsv_red2',
            'hsv_red3',
            'hsv_red4',

            'dilation_kernel_size',
            'canny_thresholds',
            'sobel_threshold',
        ]

        dtu.Configurable.__init__(self, param_names, configuration)

    def _colorFilter(self, color):
        # threshold colors in HSV space
        if color == 'white':
```

```
            bw = cv2.inRange(self.hsv, self.hsv_white1, self.hsv_white2)
        elif color == 'yellow':
            bw = cv2.inRange(self.hsv, self.hsv_yellow1, self.hsv_yellow
        elif color == 'red':
            bw1 = cv2.inRange(self.hsv, self.hsv_red1, self.hsv_red2)
            bw2 = cv2.inRange(self.hsv, self.hsv_red3, self.hsv_red4)
            bw = cv2.bitwise_or(bw1, bw2)
        else:
            raise Exception('Error: Undefined color strings...')

        # binary dilation
        kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(self.dilat

        # refine edge for certain color
        edge_color = cv2.bitwise_and(cv2.dilate(bw, kernel), self.edges)

        return bw, edge_color

    def _lineFilter(self, bw, edge_color):
        # find gradient of the bw image
        grad_x = -cv2.Sobel(bw/255, cv2.CV_32F, 1, 0, ksize=5)
        grad_y = -cv2.Sobel(bw/255, cv2.CV_32F, 0, 1, ksize=5)
        grad_x *= (edge_color == 255)
        grad_y *= (edge_color == 255)

        # compute gradient and thresholding
        grad = np.sqrt(grad_x**2 + grad_y**2)
        roi = (grad>self.sobel_threshold)

        #print np.unique(grad)
        #print np.sum(roi)

        # turn into a list of points and normals
        roi_y, roi_x = np.nonzero(roi)
        centers = np.vstack((roi_x, roi_y)).transpose()
        normals = np.vstack((grad_x[roi], grad_y[roi])).transpose()
        normals /= np.sqrt(np.sum(normals**2, axis=1, keepdims=True))
```

```
        lines = self._synthesizeLines(centers, normals)

        return lines, normals, centers

    def _findEdge(self, gray):
        edges = cv2.Canny(gray, self.canny_thresholds[0], self.canny_thr
        return edges

    def _checkBounds(self, val, bound):
        val[val<0]=0
        val[val>=bound]=bound-1
        return val

    def _synthesizeLines(self, centers, normals):
        lines = []
        if len(centers)>0:
            x1 = (centers[:,0:1] + normals[:, 1:2] * 6.).astype('int')
            y1 = (centers[:,1:2] - normals[:, 0:1] * 6.).astype('int')
            x2 = (centers[:,0:1] - normals[:, 1:2] * 6.).astype('int')
            y2 = (centers[:,1:2] + normals[:, 0:1] * 6.).astype('int')
            x1 = self._checkBounds(x1, self.bgr.shape[1])
            y1 = self._checkBounds(y1, self.bgr.shape[0])
            x2 = self._checkBounds(x2, self.bgr.shape[1])
            y2 = self._checkBounds(y2, self.bgr.shape[0])
            lines = np.hstack([x1, y1, x2, y2])
        return lines

    def detectLines(self, color):
        bw, edge_color = self._colorFilter(color)
        lines, normals, centers = self._lineFilter(bw, edge_color)
        return Detections(lines=lines, normals=normals, area=bw, centers

    def setImage(self, bgr):
        self.bgr = np.copy(bgr)
        self.hsv = cv2.cvtColor(bgr, cv2.COLOR_BGR2HSV)
        self.edges = self._findEdge(self.bgr)

    def getImage(self):
```

```
return self.bgr
```