DETECTING CYBER-PHYSICAL THREATS AGAINST AUTONOMOUS ROBOTIC SYSTEMS IN ROUTINE MISSIONS

ANATOLIJ BEZEMSKIJ

The thesis submitted in partial fulfilment of the
requirements of the University of Greenwich
for the Degree of Doctor of Philosophy

This research programme was funded and carried out
in collaboration with Defence Science and Technology Laboratory

DOCTOR OF PHILOSOPHY
May 2017

DECLARATION

"I certify that the work contained in this thesis, or any part of it, has not been accepted in substance for any previous degree awarded to me, and is not concurrently being submitted for any degree other than that of Doctor of Philosophy being studied at the University of Greenwich. I also declare that this work is the result of my own investigations, except where otherwise identified by references and that the contents are not the outcome of any form of research misconduct."

STUDENT: ..... DATE: .....

ANATOLIJ BEZEMSKIJ

SUPERVISOR: ..... DATE: .....

DR. RICHARD J. ANTHONY

SUPERVISOR: ..... DATE: .....

DR. GEORGE LOUKAS

SUPERVISOR: ..... DATE: .....

DR. DIANE GAN

Abstract

Autonomous cyber physical systems are increasingly common in a wide variety of application domains, with a correspondingly wide range of functionalities and types of sensing and actuation. At the same time, the variety and frequency of cyber attacks is increasing in correspondence with the increasing popularity and functionality of these systems, from in-vehicle driver assistance to smart city infrastructure and robotics. These technologies rely on a variety of sensors, actuating nodes and control communications. Each sensor adds context by which the autonomous system can better understand its environment, but each sensor also provides opportunities for attack, as has been observed in a variety of attacks on different systems. Cyber-physical threats are increasing significantly because society is increasingly dependent on cyber-physical and Internet of things systems and devices. Cyber-physical attacks are executed by people with different motivations, intentional or not.

A robotic vehicle testbed has been built and used as a testbed to develop a methodology that is capable of identifying possible threats and their causes. The design of the robotic vehicle testbed is documented with explanations in terms of its sensors, actuators and it operates. A key goal has been to develop a methodology that can automatically characterise the behaviour of the robotic testbed and be able to identify cyber-physical threats in a real-world environment. This testbed environment has met all the requirements for the experimental scenarios that we have identified. A model to observe signal characteristics, including noise level patterns on sensor data streams and incorporating this information to characterise normal or abnormal behaviour of a robotic vehicle is in-

troduced. Following a learning phase, where the vehicle is trained in a non-attack state on the values that are considered normal, it is then subjected to a series of different cyber attacks that have physical impact (cyber-physical attacks) and physical attacks that have cyber impact (physical-cyber attacks). The problem has been approached as a binary classification problem as to whether the robot is able to self-detect if and when it is under attack. The experimental results show that the approach is promising for most attacks that the vehicle is subjected to.

ACKNOWLEDGEMENTS

I would like to thank all my supervisors for their support: Dr. Richard J. Anthony, Dr. Diane Gan, Dr. George Loukas. They have showed me the path on how to become a researcher, how to present myself in front of the academic audience, even for their support in the improvement of my writing skills. Also I would like to thank Jelena Pavlova for her moral support.

I particularly would like to express my gratitude to the funding organisation, the UK Ministry of Defence's Defence Science and Technology Laboratory, and especially to Robert Sayers whose exceptional advice has helped ensure that the end product is applicable to a wide variety of real autonomous systems.

DEDICATION

To planet Earth.

LIST OF PUBLICATIONS

- Vuong, T.P., Loukas, G., Gan, D. and Bezemskij, A., 2015, November. Decision tree-based detection of denial of service and command injection attacks on robotic vehicles. In 2015 IEEE International Workshop on Information Forensics and Security (WIFS), IEEE.

  Note: This paper included part of the literature review on intrusion detection for autonomous vehicles.

- Bezemskij, A., Anthony, R.J., Loukas, G. and Gan, D., 2016. Threat evaluation based on automatic sensor signal characterisation and anomaly detection. ICAS 2016, IARIA.

  Note: This paper has presented the technical setup and the initial heuristic approach.

- Bezemskij, A., Loukas, G., Anthony, R.J. and Gan, D., 2016. Behaviour-based anomaly detection of cyber-physical attacks on a robotic vehicle. CSS-2016, IEEE.

  Note: This paper has presented the improved weight-based heuristic approach.

- Bezemskij, A., Loukas, G., Gan, D., Anthony, R.J., 2017. Detecting cyber-physical threats in an autonomous robotic vehicle using Bayesian Networks. CPSCom-2016, IEEE

  Note: This paper has presented the Bayesian Network integration to identify cyber-physical threats.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

As vehicles become increasingly networked, intelligent and autonomous, they also become increasingly attractive targets to cyber attacks. This project has focused on cyber threats against routine-oriented autonomous robotic vehicles and on producing a capability that allows these vehicles to self-detect such threats. The project has developed anomaly detection mechanisms that are able to detect and characterise such threats by monitoring deviations in characteristics belonging to both the cyber and physical data domains. The scope is on routine missions, where the assumption is that considerable deviations across both the cyber and physical domains are indicators of potential cyber-physical attacks. This report describes the methodology followed so that the capability developed can be installed on any robotic vehicle that supports the Generic Vehicle Architecture [1] and carries a variety of sensors. A primary requirement addressed in this project is that the method does not depend on specific sensor types, models or configurations, but is rather sensor agnostic. By sensor agnostic, we mean that the intrusion detection system does not consider the physical context of each sensor (i.e. what is measured), but merely as one more data source. So, it does not need to understand the context or require specific treatment for any data reported by the sensors. This allows the routine-mission oriented system to learn its environment and form its own understanding based on the data collected and generate a behaviour profile. The key advantage is that this increases applicability for a

variety of robotic routine-mission oriented systems or multi-sensor cyber-physical systems. Secondly, it reduces the integration complexity by requiring standard supervised training to learn its behaviour profile and additionally it is capable to re-learn its environment easily if a sensor upgrade has occurred or if there is a necessity to proceed with routine mission due to a defective or partially working sensor. However, due to the rejection of the sensor's contextual information, such a system may potentially show lower performance than an equivalent that would be designed specifically for one type of environment or one type of attack. Also, this work focuses on the detection of cyber-physical attacks on routine-mission oriented robotic systems, therefore is not suitable for non-routine oriented mission robotic applications which is out of scope of this project.

If an autonomous vehicle is composed of several subsystems for monitoring or controlling certain aspects of the whole system, it is expected that these modules will have to communicate in some way to satisfy the needs of the user. Communication between sub-systems can be done using a vast variety of technologies that are currently used for communication within such systems. Whilst we are interested in physical attacks for a background knowledge base, the research focuses specifically on the types of attack in which the cyber aspect is involved, for example the transmission of false commands. An ideal outcome would be that the cyber-physical system can be equipped with the ability to reason that the probability that a command, sensed scene, operational context etc. is genuine and has not been tampered with. Just as a human picks up a sense of "something being wrong' ahead of actually knowing the full details, a suitably equipped autonomous system, such as a drone, may be able to detect an anomaly, perhaps relying on a learnt history of previous events and missions as the basis of the initial arousal of suspicion. This is particularly the case for autonomous systems that perform routine (hence largely predictable) tasks. . So, a cyber defence mechanism needs to be able to operate onboard the autonomous system to allow it to self-detect unpredictable attacks.

1.1 Autonomous robotic vehicles

Society is becoming increasingly dependent on embedded system environments [2]. The technologies that are used today are applied in various areas, from a simple implementation of an embedded system for a specific purpose to fully automated environments based on the concept of distributed embedded systems. Such systems allow flexibility by using a modular approach to design and implementation. Some areas of interest using distributed embedded systems are: Robotics, Military Vehicles and Drones, Consumer Vehicles, Medical Devices and Supervisory Control and Data Acquisition Systems. They can be fully or partially automated.

Unmanned ground vehicles (UGV) are used in various application domains and most of them fulfil unique tasks. It is not possible to identify all vulnerabilities in such systems, as each system is unique based on its hardware, software and the environment. UGVs have general requirements and limitations that can lead to exploitable vulnerabilities in the system. These kinds of vehicles require a certain level of autonomy to operate and most of them are battery powered to enable electronic system operation. It is also possible to supply power from mains, using fuel engines or solar power and many other power sources. Using fuel engines or solar power it is not possible to guarantee the power source stability and the system must be capable of storing this energy in batteries for use as it is needed. It should be noted that these types of machines can be powered from external power sources, but to ensure ability to recover the system may use batteries. Batteries introduce a potential vulnerability that may be exploited by the attacker. For example an attacker may use battery exhaustion attack which will drain the batteries or use a sophisticated attack that may produce voltage fluctuations in the system that will damage vehicle's electronics. Denial of Service [3] is also applicable to general autonomous system domain and such an attack raises an availability threat, making such a system unusable or disrupting the UGV operation. This type of attack can be accomplished using a physical or wireless approach. If an attacker uses the wireless approach they can achieve

this goal by jamming the wireless signals, so the operator will be unable to control the vehicle. In this case, UGV must be able to identify the attack and try to recover from this situation appropriately. The work conducted by Vuong et al. has shown that Denial of Service attacks on the wireless channel can influence a robot's operation [4], significantly affecting its physical movement and even causing minute vibrations, which can be picked up by a purpose-built cyber-physical detection system [5, 6].

A physical attack has certain requirements, such as a physical connection to a system (eg., planting a malicious component through a supply chain attack). This can be achieved by writing random data to the bus potentially activating system components or if communication protocols are known produce a sophisticated targeted attack that has a certain aim. In the case of the Controller Area Network (CAN) bus where arbitration is taking place, it is possible to seize the bus access, by transmitting dominant bits on the bus. This type of attack will starve out other sub-systems connected to the bus and may lead to disruption in the system.

All general systems are vulnerable to data mining attacks, where information is gathered physically or wirelessly without authorisation. It is a great threat to data disclosure, whether it be encrypted packets that contain control, monitoring, positioning or multimedia data that can be decrypted by having certain knowledge and resources. In a worst case scenario physical communication or wireless is unencrypted and by monitoring the data and evaluating the behaviour of such a system it is possible to distinguish the type of data being transferred. Such disclosure could lead to classified data leakage and be used later for a more sophisticated attack that may be driven by specific objectives. One of the examples would be, that the unmanned robotic vehicle was stolen with a sensitive information on-board. An attacker can use reverse engineering techniques to get access to the data or confidential firmwaresoftware packages that are installed on-board. Such attack is out of scope of this project, but we still do consider such attacks possible.

The usage domain of autonomous UGVs is very broad, as these types of systems can be used in different environments such as battlefields, rescue zones and many others.

However, autonomous systems that are designed to work in a battlefield environment can also be used in a different context, such as surveillance and rescue missions. The Big Dog robot by Boston Dynamics [7] is designed to walk through a rough-terrain environment whether it be a building, crash site or a warzone. This robot is mainly used to act as a mule to carry equipment. There are multipurpose robots as well that can be used in a different context, for example the Cobra MK3, which is a small UGV that supports various sensors and actuators that can be changed on demand to transform a bomb disposal robot into a surveillance robot. There are a variety of threats and attack motives for such systems and they depend on the environment and context of use. If such systems are designed to be autonomous, the robot must rely on internal communication and its logic. The autonomous aspect makes it complicated to ensure self-awareness when they are used in different environments and contexts. The reason is that the robotic system such as unmanned aerial vehicle which learnt unique environment (clear sky) and a specific task through training or operation; may start generate incorrect decision because of the noise caused by a stormy weather. The flexibility achieved by modular design raises new threats as each of these modules can be vulnerable or contain malicious code that will allow an attacker to control the vehicle. As robots can be used in different environments, often they have unique requirements, in terms of sensors such as GPS, accelerometer and compass to increase their self-awareness, so that as to be able to identify their position in a real-world environment and achieve their objectives. If there is a requirement for collision detection, common approaches are to use ultrasonic, LIDAR [7] or other sensors that can provide information about the distance between a robot and an object. All these types of sensors are vulnerable to certain physical threats. For example, ultrasonic sensors are very unreliable if the environment is noisy; the compass sensor is vulnerable to materials that can disturb the magnetic field surroundings; LIDAR is often defeated by mirrors or even paddles of water [8]. Each unique set of sensors on a robot will produce unique vulnerabilities that will threaten the mission success in a worst case scenario circumstances. In addition, all these sensors are vulnerable to generic threats such as Sybil attacks, when the neighbouring nodes have been compromised and their reputation

will influence attacker's target node. As the requirements may change over time, there is a need for robot flexibility and this can be achieved by using a standardised approach to robot design that would support different types of sensors that would be compatible with the robot systems.

Beyond ground vehicles, there is growing interest in autonomous vehicles able to operate in air (Unmanned Aerial Vehicles), water (Unmanned Surface Vehicles) or underwater (Unmanned Underwater Vehicles). These vehicles are commonly used within the battlefield environment, but can also be used for recreation, photography, cinematography etc. One of the examples was shown by Cauchard et al. [9] where researchers studied the use of personal drones and how they interact with people. They have mentioned variety of drone applications, such as running with drones, filming or creating flying displays.

1.1.1 Internal vehicular communication

Communication can be carried out using different protocols based on the purpose of the whole system. The main communication requirement in such systems is reliability, because these systems will be used over a long period of time and they must be reliable as certain aspects of the system must be safety-critical. One of the most common types of communication in vehicular networks is the Controller Area Network (CAN), which provides reliability, through arbitration based on a message ID. This type of arbitration will ensure that the message with the highest priority will always be delivered. There are on-top CAN protocols such as the Time-Triggered Protocol or the MilCAN protocol that uses an extended 29-bit CAN identifier to provide a source address, allows prioritisation, reserved fields specifying a function type or data, as well utilising time-triggered protocol features [10]. By utilising time-triggered protocol features MilCAN allows determinism [11]. The difference between MilCAN (Time-Driven) and CAN (Event-Driven) is the deterministic behaviour. This approach resolves the starvation issue i.e. when a faulty module with the highest priority can take over the bus access by constantly sending messages and starve out other Electronic Control Units (ECU) in the system. ECU

is a term used for in-vehicle embedded systems that are responsible for a specific sub-system. The CAN bus itself can support a maximum bandwidth of 1 Mbps and with increasing requirements such as x-by-wire technology the bandwidth requirements are also increased because timing is critical for this type of technology. Due this limitation, the FlexRay [11] protocol was introduced that improved the bandwidth limitation to 10 Mbps. Even though the bandwidth was increased this protocol is only used for safety-critical systems because of its expensive integration. To provide multimedia application support the Media Oriented Systems Transport (MOST) protocol was introduced [12]. This protocol has variable configuration that can provide variable bandwidth up to 150 MBits [13, 14] MOST is not used for ECU communication, as it is more expensive than FlexRay [11, 15] and has more strict requirements such as fibre-optic cable implementation [16]. All these reviewed protocols have their advantages and disadvantages such as the cost of implementation. To make a system more reliable and flexible, the advantages of these protocols can be merged together and therefore to allow compatibility between these protocols different bridges were developed [15]. When all sub-systems are interconnected it is possible to gather information based on the data transmitted by various modules and represent this information using Health & Usage Monitoring System (HUMS) or Vetronics Integrity and Management System (VIMS) [17] such systems would reduce the maintenance cost and user awareness. Systems, such as modern vehicles, unmanned ground vehicles or unmanned aerial vehicles are heavily dependent on communication between ECUs, because, for example, a modern vehicle that provides partial automation to satisfy user needs is using over 70 ECUs on its Internal Network [18, 19]. An ECU is an embedded system which consists of sensors, actuators and software. Developers rarely create their own designs for ECU components i.e. Integrated Circuits (IC), Micro Controller Units (MCU). They assemble ECUs using off-the-shelf hardware and program them so that the ECU would meet their design specifications and requirements. During the design or development stage of such complex systems, developers have to prioritise their needs correctly and, unwisely, security often has low priority. This is partly because cyber attacks in this context are not as frequent as the ones related to traditional computing

environments, and partly because the assumption is that access to a vehicle's internal network and ECUs is practically difficult, an assumption that has been proven wrong in multiple research papers and real-world attacks [18].

1.1.2 Securing Vehicles' Electronics

The flexibility achieved by using a standardised approach to the design of vehicle architectures, allows data to be gathered on the network and evaluated for different purposes, including cyber-security. The Vetronics Research Centre has proposed a few strategies on how to secure autonomous vehicles from cyber attacks [20, 21]. Their approach to securing autonomous vehicles is to ensure the mission success. Firstly they decrease the chance of an unauthorised access to the system by integrating authorisation between nodes and data encryption. They increase the ability of attack detection using Intrusion Detection Systems (IDS) on certain nodes such as gateways and operating nodes and suggest a way to maintain essential services that are mission critical if the system was compromised. However there are no solutions provided on how to implement such a system, only suggestions on strategies and requirements. To ensure structural integrity they have proposed a system that would be gathering data from different Health and Monitoring Systems (HUMS) modules that are responsible for safety-critical, automotive, weapons or other systems [17] to ease maintenance of a vehicle and provide information about the overall vehicle state and vehicle surroundings to the crew using a user-friendly on-board interface. This is also a step towards the security of vehicle electronics as this system takes into account network health, based on that information it is possible to catch specific anomalies on the network that would enable it to make an assumption if the vehicle was compromised.

1.2 Historical examples of cyber-physical attacks against vehicles

There are a wide variety of known attack methods. They can apply generically to any autonomous system or be specific to a particular type of system, and may be associated with a particular application or use-scenario, or a specific sensor or other component. The complexity of threat modelling for robotic system is due to the application variability (entertainment, warfare applications, medical applications) of robotic systems and their operational environment domains such as: air, ground or water surface. Application types and potential threats will be discussed further in Chapter 2

An attack which is worth mentioning is an incident with RQ-170 UAV that was hijacked by Iranian Military Forces. The Cyber warfare unit of Iranian Military Forces claims to have been able to hijack a United States UAV and land it on Iranian territory. Iran has stated that they were using a sophisticated attack, starting with jamming communication between the drone and Command and Control Centre and that using GPS spoofing they were able to land the RQ-170 UAV on their desired territory. If true, this demonstrates that the autonomy of military unmanned aerial vehicles can be influenced by applying sophisticated attacks that require certain knowledge and equipment. As such vehicles are used by military forces, it is expected that this equipment will serve several years before being discontinued or updated. This introduces new threats as enemy forces can examine the technology and can find new vulnerabilities that can be exploited later in a battlefield during a war or other military operation. While this attack has not yet been confirmed by non-Iranian sources, the concept has been proven extensively, but for civilian UAVs and civilian ships (and hence for civilian GPS), by the University of Texas at Austin [22].

Another example of aircraft vulnerability exploitation was when Hugo Teso, a computer security expert asked himself if he was able to hack into an aircraft that are used in a civil aviation. During the hack at the Box Security Conference 2013 he demonstrated how he was able to attack a civil aircraft [23]. His attack was focused on a specific Flight Man-

agement System (FMS) which is installed on a certain aircraft models. Without having advanced knowledge of aircraft communication, procedures and protocols, he started his research around this area and was able to gather all necessary information from public sources. This information included communication protocols, the type of equipment used and the necessary procedures during the takeoff, flying and landing states. As it was not possible to gain access to a real aircraft, his decision was to build a virtual test bed using real equipment that is used on aircraft. He was able to obtain all items that would allow him to proceed with his experiments. During his presentation at the conference he suggested various ways of obtaining such equipment, for example: buying equipment from aviation scrap yards, on the black market and through online shopping. The equipment was obtained using online auctions and shops. He was able to find a flight management control computer for $400 and the item description said that it was an add-on for a Microsoft Flight Simulator, while it actually was real equipment used on aircraft [23]. An aircraft management system was acquired for $10 in an online auction. This shows that it is possible to obtain real equipment externally i.e. not using original vendors, who would possibly ask for licenses or certificates to ensure a buyer's legitimacy. By using a reverse engineering approach he was able to gain knowledge about the system, how it was structured and using this knowledge it was easy to identify the existing weak points of the system and to exploit them. In the end he created a virtual test bed environment and was able to demonstrate that when an aircraft activates the auto-pilot function it is possible to control the aircraft using a desktop PC, laptop or Smart phone [23]. It was an interesting fact that the equipment he was using supported scripting, so there was also the possibility to upload a script into the system which could then be triggered by a specific event, for example GPS coordinates.

The Grand DARPA Challenge in 2007 showed that modern trends in vehicles are going towards full autonomy [24] and participants have made huge progress [25, 26] when a challenge required a fully autonomous vehicle to drive in an urban environment and most of the vehicles were able to successfully finish the challenge. These kinds of challenges show that a lot of research is going on in this area. Modern vehicles nowadays provide

various features such as cruise control, automatic parking and many other features that will allow the driver to relax. By giving control to an automated environment the outcome will be more efficient i.e. decreasing fuel consumption. This kind of functionality is achieved through sensor and control unit communication as described earlier. A group of researchers performed security analysis on modern automobiles and reviewed various types of attacks which compromised the vehicle [18]. Their test bed was a vehicle that was built in 2009 with a high number of autonomous features. As they had physical access to the system they have used different techniques to gather and spoof data. In conclusion they had enough information to turn on and off different parts of the system, by spoofing sensor data. Also it was possible to upload malicious code to the memory which would trigger if certain criteria was met. As an example a script could turn off the brakes and headlights, when it is dark, rainy and the speed is over 80mph. The malicious code would be able to delete itself after a crash. These kind of attacks show that if the person has the motivation, necessary skills and by using a sophisticated approach, it is possible to compromise any embedded system whether it would be a complex system such as an aircraft, vehicle or miniature safety-critical systems such as pacemakers or insulin injectors.

1.2.1 Other Types of Cyber-Physical Attacks

In this section, the embedded systems that are used in various environments such as robotics, subsystem control, medical equipment and many others will be discussed including how these systems can be attacked and what possible motives an attacker can have to infiltrate such system.

Jerome Radcliffe was a security specialist, diagnosed with type 1 diabetes, who, because of his health condition, was interested if it was possible to hack into insulin pumps that maintain the insulin level. He was comparing such a system to an industrial control system's supervisory control and data acquisition (SCADA) system. They both monitor sensor data and control their environment based on this data [27]. During the Black Hat

2011 conference, he was able to demonstrate that it was actually possible to control these devices and increase and decrease the insulin injection remotely from 30 meters range. A year later Barnaby Jack, a computer security expert, made a further step and developed an exploit to increase the range of the insulin pump hack up to 90m, this was demonstrated by Stuart McClure at the RSA Security Conference in February 2012 [28]. In October of 2012, during his presentation at the BreakPoint Security Conference in Melbourne, he was able to reverse engineer a pacemaker transmitter and stated that he was able to deliver a deadly electric shock to the person who was using the device [29]. In 2013, during the Black Hat Conference, he wanted to give a presentation about exploiting these vulnerabilities without the need for knowing identification numbers, but he was unable to do so due to his sudden death (due to unrelated cause) a few days before the conference [30]. The attacks mentioned here were accomplished with the use of wireless technology as both of these devices were using radio frequency communication and it was possible to send spoofed packets to the device and control it using this approach. These computer security experts where driven by curiosity, and were interested in raising concern around the security of embedded devices, as currently there is little or no security at all which they successfully demonstrated.

1.3 Research methodology

Cyber-physical research has primarily focused on vulnerabilities found in autonomous system's components. These can be parts of critical national infrastructure [31], unmanned robotic vehicles subsystems [32]. However, such an approach to this research has limitations because the subsystems may change and the data generated by such systems may not. Additionally, some system vulnerabilities can be patched and need to be constantly maintained and severe vulnerabilities have to be patched. This introduces an issue if the system is at a distant location, it would be complex to maintain [33] because of limited access. By maintaining such systems in a timely manner and performing system patching on the day when they are released, such an approach may perform better. However such a

system will still be vulnerable to zero-day attacks. It would be beneficial to have a cyber-physical attack detection approach that would be capable of learning it's surroundings and to be able to adapt.

Our proposal is to assess such systems assuming that the attacks coming from a cyber or a physical domain, always persist. It is only a matter of time when an attack will take place, therefore they constantly pose a threat to the system. Such a view of vulnerabilities will introduce dynamic detection concept in such a way that vulnerabilities become threats that advance from the cyber-physical domain, ignoring the sensor's contextual information or their unique role in the system. Assessment of such threats will provide context information for an autonomous system such as an attack vector and a domain that possesses the cyber-physical threat. Such information can be used as an input for decision making mechanisms that have higher control of the system and its environment and can also be used as a supplementary state of the art methodology [34, 35, 36] that is capable of raising system self-awareness.

1.3.1 Aim

Autonomic computing is currently being integrated into a variety of everyday systems. These systems are controlled by an assortment of complex components and are applicable to a number of different areas such as: industrial automation, emerging smart home systems, unmanned ground vehicles, drones and many others. However, the increased use of such systems has also created an increase in the potential threats from adversaries who will seize any new attack opportunity. These threats can come from both the cyber and the physical domain, and this also includes cyber attacks with intentional physical damage to the system (cyber-physical attacks [8]). As the threats from this new area are still emerging and rapidly evolving, these systems require smart and adaptive security systems that are capable of detecting new threats. In this context, knowledge-based countermeasures that rely on learning the signatures of already known attacks can be impractical. Hence, there is a need for an anomaly-based (also referred to as behaviour-

based [37]) cyber-physical detection approach.

For this purpose, we have developed a robotic vehicle testbed that uses industry recognised communication standards, based on generic vehicle architecture (GVA) with a number of actuators and sensors on-board. Instrumentation data is used for autonomous learning and calibration and to monitor the behaviour of the robotic vehicle testbed. A main requirement of such a model is applicability across a variety of systems and sensor configurations, and to be able to adapt to a variety of systems and environments autonomously. It also needs to be light-weight, so as not to affect the operational capabilities of the vehicle.

1.3.2 Scope

The scope of this project is within the realm of small autonomous vehicles undertaking routine tasks, such as deployable surveillance systems, observation posts or guard and patrol-oriented vehicles. A key challenge is that such systems are usually deployed in an open environment exposed to a variety of cyber-physical threats that need to be monitored. Usually, such systems are powered by a limited power supply (fuel, batteries) and are constrained in terms of available computational power. Therefore, it is beneficial to develop a power-efficient anomaly detection approach that uses behaviour characterisation by monitoring available instrumentation. The behaviour characterisation is reviewed as a supplementary anomaly detection approach that is capable to classify behaviour and identify the threat domain (cyber or physical) when the testbed is under cyber or physical attack.

Naturally, a large number of cyber attacks can be relatively reliably prevented by using encryption. Of course, encryption can fail in several cases, for example when there is an error in an implementation, a weak process for generating and sharing keys between the parties involved, or simply bypassing it completely physically or using deception against the user. Also, in many cases, encryption is simply impractical because of the extra processing and energy requirements. Hence, it is important to have a mechanism for

detecting attacks that have not been prevented. The focus of this project is the detection of physical-cyber and cyber-physical attacks which require monitoring features from both domains. The assumption made is that the robotic system has no encryption or that encryption has been compromised, as is customary in areas that involved interdependent computer networks [38]. Also, it is a long-established principle in cryptography that "The enemy knows the system being used", which has been reformulated also as "It is pessimistic and hence safe, but in the long run realistic, since one must expect his system to be formed out eventually. Thus, even when an entirely new system is devised, so that the enemy cannot assign any a priori probability to it without discovering it himself, one must still live with the expectation of his eventual knowledge" [39].

The robotic vehicle that has been developed carries a variety of sensors and all experiments that have been conducted have been tested with the sensors and actuators that the robotic system has on-board. Sensor configuration other than the testbed may demonstrate different results to what is then presented in this document. The data on the robotic testbed has been collected on the external ZigBee network as well as on the internal CAN bus. Communication between the nodes is evaluated as a data message consisting of up to 8 bytes maximum. However, with these limitations in its current state the methodology should be potentially transferable to commercial products, especially if compatible with generic vehicle architecture.

1.3.3 Objectives

We have set a series of steps that we are going through during the lifetime of this project that enable the development of a mechanism for improving the cyber security of autonomous robotic vehicles and reducing the risk of a disastrous physical impact from such systems. The key objectives are:

1. Familiarisation with cyber attacks which are applicable to cyber-physical robotic systems.

2. Identification of standard test scenarios that can be used on the robotic testbed.

3. Development of a modular robotic vehicle testbed with current industrially relevant technology that enables the gathering of data during the normal operation and during the attack scenarios.

4. Collection and monitoring of data produced by the robotic platform and identification of common indicators of different types of attack, through experiments using the identified scenarios.

5. Development of a mechanism able to evaluate a cyber attack threat through its impact and to identify it during standard test scenarios.

6. Critical evaluation of the extent to which the mechanism facilitates autonomous detection of attacks in real-time.

1.3.4 Research questions

This project is concerned with the cyber-security of autonomous vehicles. The research society is working towards cyber-security improvement using various paths, whether it would be detection of cyber attacks, resilience while under cyber attack or recovery from a cyber attack. This work has combined all these approaches and developed an autonomic mechanism to detect, in real-time, the occurrence of a cyber attack, to evaluate the impact of the attack and also evaluate the potential threat prior to the attack. The main research questions are:

1. What is required for a cyber attack against an autonomous vehicle to take place?

2. How can a cyber attack be detected?

3. How can a cyber-physical robotic system evaluate a cyber threat?

4. Is it possible for a cyber-physical robotic system to self-detect a potential cyber-threat and the potential impact of a cyber attack?

In Chapter 1 we have discussed the introduction of this project, the necessity of cyber-security concepts adaptation to robotic vehicles, there are many robotic vehicle applications that operate in a variety of environments. A malicious activity towards such systems i.e. cyber or physical attack, may have a negative impact on the real-world environment. Further in Chapter 2 a reader will find the Literature Review where we will discuss the research pertaining to his field. The key focus in this chapter will be on the cyber-physical threats to robotics vehicles and their existing vulnerabilities, which will enable us to determine what kind of features need to be taken into account to detect a cyber or physical attack on the vehicular system. The counter-measures for such attacks will be reviewed considering signature and anomaly based intrusion detection systems. The robotic vehicle testbed which has been developed for the purpose of this project will be discussed in Chapter 3. In Chapter 4 a reader will find details on the approach for anomaly-based detection of cyber-physical threats and the methodology of the approach. As a starting point this chapter will cover the experimental environment and its unique features in regards to behaviour definition. The cyber-physical and physical-cyber attacks that have taken place during the experiments with the robotic vehicle testbed are discussed. This chapter also contains the initial results that allowed the conceptual idea of detecting anomalous behaviour using an anomaly-based approach to evolve. This evolution is demonstrated from bottom up, starting by a discussion of the behaviour formation, leading to the development of heuristic binary classification approach and it being summarised by evaluation of the results. This leads to an improvement where weights are introduced to the heuristic binary classification that show a detection rate improvement. The further discussion in Chapter 4 is on the integration of the real-time heuristic binary classification into the robotic vehicle testbed, firstly through an emulation of the real-time detection where data from the sensors was fed one-by-one (emulating real-time) showed real-time detection feasibility which has led to an integration of a real-time defence mechanism on the robotic vehicle testbed. The end of the chapter demonstrates the integration of the heuristic binary classification approach and the Bayesian networks as a demonstration of the interoperability of the proposed methodology with other available techniques. In

Chapter 5 we will summarise this 3-year project referring to our defined aims and objectives, with a discussion of achievements and the results followed by the suggestions for the future work.

Chapter 2

Literature Review

Cyber security in cyber-physical systems and especially vehicles is a relatively new area of study. Relevant research has focused primarily on proof-of-concept attacks [40] on the integrity of sensing and actuation or the availability of communications. In most cases, the proposed defence is limited to survivability and resilience through redundancy [20] or prevention through authentication and encrypted communication [41]. However, this is an overly optimistic approach, as attacks, especially zero-day attacks, do go through and need to be detected. So, the focus here is on the intrusion detection techniques designed specifically for mobile cyber-physical systems and robotic vehicles. Depending on its architecture and application, a robotic vehicle may be able to benefit from communication with other agents (multi-agent) or may need to rely solely on its own sensing capabilities and monitoring processes (single-agent) [6].

A common approach is the multi-agent architecture approach, which focuses on coordination between different agents such as cooperation between vehicles using vehicle-to-vehicle technology [42] or within the context of a swarm of robots [43]. Communication between agents is an important feature in multi-agent architectures Urdiales et al. [44] have evaluated the performance of robotic position self-deployment algorithms and have studied the mobile wireless sensor network concept that can be used for robotic applications. This is because the external communication is a crucial monitoring feature that can be

used in an intrusion detection system with a multi-agent architecture and may allow to detect one agent's suspicious actions, reports, configuration or location. The detection criteria may include consistency with the laws of physics (e.g., whether velocity values measured are physically possible or the location is consistent with the velocity measured [8]), consistency with the sensor measurements reported by neighbouring agents [45], dynamic voting algorithms for security [46], reputation or consensus based [47]. Most of these approaches and detection criteria become impractical in single-agent systems, such as the single semi-autonomous robotic systems without the opportunity to coordinate with other nodes. This demonstrates that the focus has to shift towards identification of relevant characteristics that can be measured by system's sensing capabilities. Such sensing capabilities may stand as a standalone sensor which report raw values on the bus or it may be a sub-system such as LiDAR which does the processing on-board and reports of an outcome to the system with the results [48].

In some application domains, there are various sub-systems that are specified at governmental level, such as the European Union legislation requiring vehicle manufacturers in the European Union to integrate Tyre-Pressure Monitoring System (TPMS) on vehicles ("Commission regulation (EU) No 523/2012 of 20 June 2012"). Based on the nature of such monitoring sub-systems, such sub-systems normally have to be wireless. This introduces an attack vector that potentially allows attackers to spoof these sub-systems' sensor readings. One of the potential scenarios could be that car thieves can follow an expensive vehicle on the highway during night, spoof TPMS sensor values, and cause the driver to stop to change a tyre as an example. Megalingam et al. [49] have noted that in most cases TPMS sensor communication works on 433 Mhz wireless frequencies and propose other wireless communication technologies to be used with more mature security solutions, such as ZigBee communication. Yun et al. [50] see potential in ZigBee technology and suggest to use ZigBee communication for other sub-systems that will incorporate various in-vehicle sub-systems into wireless sensor network. ZigBee communication protocol is commonly used in wireless sensor networks as it covers the low power requirements of nodes in adverse conditions [51]. This demonstrates the need for the development of re-

source efficient systems, in terms of in-vehicle architecture it will slightly reduce the power consumption of vehicle's sub-system therefore reducing the emission of carbon monoxide in vehicles. For this reason, Bluetooth is also commonly proposed [52]. Regardless of the specific protocol chosen followed, the issue is that they all may feature flaws (in their design or their implementation), which have been demonstrated extensively. This means that when cyber-physical systems or a part of such systems are being developed using technologies that are proven to be vulnerable, developers need to take into account the security by design. This issue has been raised in a number of publications for a variety of cyber-physical systems such as Bonaci et al. [53], who have proposed recommendations to be taken into account when developing tele-operated surgical robots, Koscher et al. [40] who have shown a variety of vulnerabilities in existing modern production cars and Wolf et al. [41], who have provided several recommendations that should be taken during the design stage of vehicles. There are a variety of attacks and attack vectors that can be applicable to specific sensors or sophisticated attacks using a sequence of attacks that can lead to malicious system behaviour.

2.1 Cyber threats to vehicles

An attacker's intentions can massively vary based on their motivation, from financial gain and system destruction, to cyber-warfare or even just curiosity and research. As a result, attacks can also differ considerably in terms of aim, but also in terms of system application and configuration, when the impact is an unmanned military vehicle, a driverless car, a surveillance robot or a search-and-rescue robotic vehicle. Looking at the problem from the perspective of the threat imposed to a system, Holm et al. [54] approach the attacker as an object with certain attributes, such as skills and available time to penetrate the target system, and conclude that the likelihood of a successful attack depends more than anything on time and effort spent, especially in preparatory stages for information gathering about the technical specifications of a system and fingerprinting the network to identify possible attack vectors. Spending indeed over two years of research with multiple

highly skilled researchers, an interdisciplinary research group [40] were able to identify valid message IDs on the CAN bus of a modern car and then a variety of flaws, which allowed access to the CAN bus, so as to utilise the identified IDs. Of course, the attack was possible because cars do not have countermeasures specific to such attacks (which are called fuzzing attacks). For detecting such a fuzzing attack, one could monitor network features, because it affects communication and the impact will be on network utilisation and data/communication integrity (Fuzzing tries to identify the functional ID's by sending random packets on the CAN bus). The large ratio of corrupted data introduced is also highly indicative of the fuzzing attack, as commands are sent to/from non-existing IDs, and data values exceed the expected range of values, exhibiting invalid message formats or lengths etc.

Therefore, there is a need to identify what possible threats can exist for a special type of system, as various resources can be used to destabilise an autonomous system. First thought would be to use existing threat models for such systems, but due to industry specificity very few would be applicable. McCarthy et al. [55] have produced a technical report for the U.S. Department of Transportation with their findings and suggestions on the potential security threats in modern automobiles. They state that the existing threat models are uniquely designed for a specific industry and not always are applicable to automobiles, therefore a hybrid can be used to make threat model more precise and applicable for modern vehicles. To show the need for a threat model that could be applicable to cyber-physical systems, few attack scenarios can be thought of. For instance, consider a robotic system used in a medical environment to move in corridors and dispense medicine to the patients. An attack aim is to cause a system to deliver the wrong medicine to the patient. Another examples could be where an attack against a parcel delivery drone is carried out and could have different aims, e.g. to steal the parcel, hijack the delivery drone or if not plain curiosity motive. So, there are various factors that create different threats, driven by different motives. Hamad et al. [56] have made an attempt to classify attacker profiles in their threat model that are applicable to the vehicles. They emphasise attack profile motivations such as fun or vandalism, unsophisticated attackers that use

pre-existing tools without the back round knowledge or possible outcomes caused by such attack. The outcome may vary as well based on the motivation. Abomhara et al. [57] have done thorough research on intruder motivations and their structures. The work was evaluating cyber security in the Internet of Things domain looking at the variety of vulnerabilities, threats, attacks, intruders and their motivations. They have classified possible attackers into certain groups such as: individuals, organised groups, intelligence agencies. These attacker groups can have different motivations and their own motives for financial gain, reconnaissance, cyber-warfare or even cyber-terrorism. From the literature it is eligible to come to a conclusion that will suggest that cyber threats always persist for the robotic systems, and the probability of a successful attack will increase in time as was suggested by Holm et al. [54].

This led us to a concept where the robotic system is always exposed to a variety of threats in time. We have produced a diagram to visualise the concept and the diagram is shown in Figure 2.1 . The threats are accumulated as the time goes by, potentially increasing the impact of an attack. This might not be a case when the system is well maintained and being patched as new vulnerabilities are published.

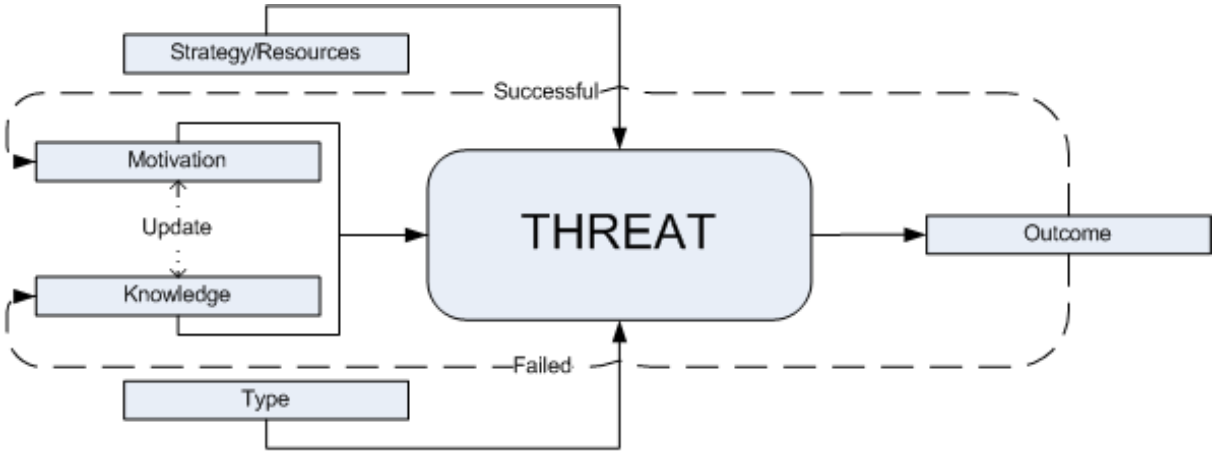

Figure 2.1: Threat model used for threat to impact analysis

Figure 2.1 shows a proposed threat model, where the main weighting factors of a threat are: Attack Motivation, Technology Knowledge, Type of Attack and Resource availability. Threat in this context is seen as an object that has attributes and they have relationships. Initially threat to the system will increase if an attack will be motivated by some means to

compromise the system. The threat will increase as well based on the attacker's knowledge of the subject and his skill set. Motivation and knowledge have two way relationship where the outcome will affect either the motivation or their knowledge about the system. This leads to an assumption that an attacker will attack a system using known vulnerabilities to identify whether the system is vulnerable to that attack or not. This may lead to an outcome which may succeed or fail. Failure updates the attacker's knowledge about the system, which therefore will affect his motivation where he might decide to stop attacking the system or will accept it as a challenge and will perform additional attacks identifying system vulnerabilities.

As was mentioned by Rid et al. [58], we need to take into account the structure of an attacker and their available resources. This means that an attacker can be an individual person with research motivations, as well as an intelligence agency which has reconnaissance mission task during cyber warfare and access to number of cyber security engineers which accumulate knowledge and motivation. We need to take into account the type of an attack, in this model it is used as an attack vector or an attack access point, the importance of this has been mentioned in Hamad et al. [56] work. This can be either access to wired or wireless communication medium, physical access or remote access. One of the examples would be that the malicious node was planted during system development, this opens an opportunity for an attacker to access the communication medium such as CAN bus and execute denial of service attack using single CAN frame disruption as was shown by Cho et al. [59].

This could lead to a sleep-deprivation attack that can be executed using a variety of techniques. For example, Chan et al. [60] propose to trigger modules using communication protocol specifics or malicious software, so as to exhaust the energy available to them. The key indications here are power consumption, network utilisation. An infected module or several modules will transmit unnecessary data on the network and could manipulate actuators to increase power consumption [61]. The power drainage would increase and may show a detectable trend.

The opportunity of planting malicious nodes during system development has higher complexity and is called a supply chain attack as it requires the involvement or exploitation of manufacturers or suppliers of the equipment [62, 63]. Malicious software can be preloaded into the firmware which will activate a logical bomb or trigger the program based on pre-defined circumstances. Mo et al. [64] have analysed the impact of such attacks for smart grid infrastructures. The danger of such an attack is that the equipment is initially installed on an autonomous system and usually it will not disclose itself until a certain point in time, or a specific trigger-event occurs. In terms of autonomous robots the possible indications of such an attack would be a change of behaviour, power consumption or network utilisation. A sudden change in behaviour could lead to identification through the detection of a periodic increase in power consumption and network utilisation when malicious hardware is transmitting data periodically to expose sensitive information. Very little research has been done in this area, however an increase in the level of autonomy in the system or an increase in the number of electronic components, increase the possibility of such a threat [65].

Another form of attack is when the communication link is hijacked without the system realising, called a relay attack. Such attack can be applicable to autonomous vehicles as well as semi-autonomous or remote vehicles [66]. Attacker can intercept communication without system components noticing, some indications can be a sudden change in data transmission timing. If this attack takes place it would be possible to notice a difference in the data transmission arrival time [67].

For cyber-physical systems, a new attack vector is created, called sensor-based data injection [8]. It is a type of physical domain attack where an attacker targets a vehicle's sensors and tries to generate false information. An example is when a robotic vehicle operates in a partially anechoic environment (e.g. bushes), and is relying on ultrasonic sensors. A sensor would generate data that the path is clear when in reality it would not be, for example the ultrasonic pulse transmitted could be absorbed by a path-obstructing object. Also Kho et al. [68] have mentioned that ultrasonic sensors have their disadvantages and can be affected by the weather as well as other sounds can interfere with the

ultrasonic response. In case if a magnetic compass is used for navigation it is possible to attack this sensor using magnetic material, as an example placing a magnet near the sensor, which would cause it to generate false bearing information. More sophisticated approaches have been used as well by Petit et al. [69], who have shown that it is possible to attack a drivereless car's LiDAR system by generating an overwhelming number pulses of the same frequency. This will force the LiDAR system to detect multiple objects that do not exist or miss the ones that do, because it is forced to reach its signal processing capacity (in a form of LIDAR equivalent of the denial of service attack experienced in computer networks). Such attack disrupts main navigation units and is a threat to any type of LIDAR-dependent autonomous system, such as Google's driverless car.

In contrast to sensor-based false data injection attacks, there is considerable work in the area of communication-based false data injection attacks. A variety of approaches can be used, such as node hijack, relay hijacking or introducing a rogue node into the network. In all cases, this type of attack aims to generate falsified data and usually to broadcast it on the network. Based on the approach used there are some common factors that are affected by this attack, as it is one of the common cyber attacks for cyber-physical systems and it can be applied to a variety of applications from autonomous vehicles, smart grids, smart homes or Internet of Things devices [70]. One common countermeasure proposed is to apply reputation based techniques for autonomous robots [71], or detecting false data injection attacks using voting algorithms in wireless sensor networks [72] and at a larger scale in smart grids [73, 74]. The main monitoring features are the network utilisation, data integrity and inconsistency. When this attack takes place there can be a noticeable increase in network traffic, including repetitive data transmissions. The data received from a node will have to be validated against expectations based on the long-term history, as this data might not have a sudden effect (stealthy data injection attack [75]; however in the long-run it might have a major effect such as a misguided path. The most common cyber attack on majority of the system is a Denial of Service attack is used to disrupt communication internally or externally which will have an effect on network availability [76]. Recently it was shown that it can have other indications of an attack rather than

be fully network communication based. Some of the indications are network utilisation [77, 78] or availability [79], CPU load [80], and physical event detection (e.g. halting or signs of chassis vibration) [4].

Communication jamming [81, 82, 83, 84] is an additional attack that is applicable to mobile robotic systems such as autonomous, semi-autonomous or remote unmanned aerial vehicles or unmanned ground vehicles. This attack is used to disrupt communication between one autonomous device and other nodes within the system. The key factors are network availability loss. This attack impacts on the physical domain so it will not be possible to communicate with the autonomous system as communication will be unavailable. To increase survivability of a system it needs to have a mechanism that will cause it to leave the jammed area.

Another attacks are GPS jamming attack [85, 86, 87]. This is highly noticeable as the majority of systems that rely on GPS would be able to identify that a GPS signal has been lost. If an autonomous vehicle is operating indoors or in other GPS-denied environments, it may not be possible to identify whether the signal has been lost due to the environment or due to an attack. However, it is possible to take several factors into consideration, including the GPS signal strength and GPS data inconsistency. A GPS spoofing/meaconing [88, 89] attack is where an attacker has an ability to fake a GPS signal [22], or transmit delayed GPS signals to disrupt navigation of an autonomous vehicle. It is complicated to identify such an attack as they both will qualify as legitimate from a systems point of view. Detection features can be identified by monitoring GPS signal strength, signal noise and data inconsistency. By keeping a history of GPS data it would be possible to analyse the data and identify the expected position. It is noticeable that autonomous systems are exposed to variety of threats that may allow an attacker to hijack a UAV [90] or a yacht, where University of Austin researchers [91] have successfully demonstrated an attack which forced a large yacht to change its course. The existing mitigation techniques for such attacks are discussed in Section 2.3 where we review existing detection and resilience methodologies that are applicable to autonomous and semi-autonomous vehicles. The summary of the attacks is shown in Table 2.1.

Table 2.1: Summary of the attacks

Reference	Attack	Impact
[59, 76, 78, 77, 79, 80, 4]	Communication Denial Of Service	Lack of availability in communication
[81, 82, 83, 84]	Communication Jamming	Lack of availability in communication
[66, 67]	Malicious Relay	Information disclosure
[85, 86, 87]	GPS Jamming	Disruption of autonomous navigation
[88, 89, 22, 90, 91]	GPS Spoofing/Meaconing	Hijacking/Disruption of autonomous navigation
[62, 63]	Supply-Chain	Various, Logic bomb
[60]	Sleep-Deprivation	Battery exhaustion
[61]	Rogue Node	Various, Logic bomb
[70, 71, 72, 77, 74, 75]	False Data Injection	Various, sub-system operation disruption
[8, 69]	Sensory Channel Attack	Various sensor-dependent

2.2 Cyber and physical features

Here, by cyber features, we refer to characteristics relating to a cyber-physical system's networking and computation processes, and by physical features, we refer to characteristics relating to the physical operation of the system, as captured by sensors [8]. Unlike traditional computing environments, where intrusion detection is limited to monitoring cyber features, relating to only data communication and processing, cyber-physical systems present the opportunity of also using features relating to the physical operation of the system. For example, Vuong et al. [5] have shown that while monitoring solely physical features cannot be a reliable method for detecting attacks, if used in conjunction with cyber features, they can noticeably increase detection accuracy and reduce detection latency.

Madan et al. [92] have carried out research on the cyber threats that are applicable to unmanned autonomous systems. They have identified multiple vectors that are under a threat such as Malicious software, Communication jamming, Information theft and multiple other threat vectors. The variability of attack vectors creates a cyber threat domain that allows an attacker to use sophisticated multiple-attack approach to achieve malicious goals.

As an autonomous system which can learn from the data, it is necessary to ensure attack-free environment, because at this stage system is vulnerable to variety of attack that may impact overall operational performance. If an attacker is able to inject false data during

robotic vehicle operation learning it may have delayed impact, even introducing slight variation may have damaging impact on the actuators. The example would be when Stuxnet worm was slightly modifying speed of the turbines which lead to their failure [93], therefore the system may malfunction when placed in a real-world environment if such an attack occurred. The cyber and physical features may demonstrate benefits in terms of an increase of a situational awareness, secondly access to such features may allow intelligent mechanisms to indirectly infer information from multiple sensors, for example in a situation when a compass has failed on the ground robotic vehicle and the robotic vehicle has to return to its start location, assuming that the vehicle knows its mission-start pointing direction, the path can be inferred from the information such as wheel position and motor speed [94].

2.2.1 Cyber features

Naturally, cyber attacks leave cyber traces, e.g. by causing a change in the processing, the network traffic etc. So, it is crucial to monitor various cyber features because they allow for improve situational awareness of the robotic system. In fact, this is the natural approach used in almost all intrusion detection systems, whether for traditional computing systems [3] or for cyber-physical systems [95]. Examples of cyber features include statistics about the network traffic, such as network packet headers or content, processing, disk and memory utilisation, etc. These can be measured against different environmental conditions, the state of a mission and the performance of the vehicle at a given point in time, to uncover evidence of a cyber attack affecting its ability to complete its mission.

2.2.2 Physical features

The physical features such as power consumption, wheel speed, chassis vibration [5, 8], may improve the vehicle's situational threat awareness. Additionally these features may be used to map the decisions made by an operator or a decision making mechanism of

an autonomous vehicle. Monitoring such features over a period of time while robotic vehicle is in operation, it is possible to learn the patterns [96] that may allow to identify particular decisions and can be used in an intrusion detection mechanisms. For mobile robotic systems it is beneficial to monitor their position [45]. The summary of relevant cyber and physical features can be found in the Table 2.2, the researched involved in producing this table is a part of our contribution to Vuong et al. publication [6].

Table 2.2: Intrusion detection approaches and their cyber-physical input features of the robotic cyber-physical systems

Ref.	Type	Comms	Attack Types	Input Features	Detection approach
[45, 37, 46]	*Mobile CPS*	*Wireless*	*Bad Command Injection, Node Hijack*	*Position, Battery Exhaustion Rate Nodes Compromised*	*Dynamic IDS Voting, Positional Discontinuity, Enviroconsistency*
[97]	*Multi-Robot System*	*Wireless*	*Misbehaviour*	*Node Reputation, Behaviour score, Distance Estimation*	*Clustered Monitoring, Voting*
[53]	*Robotic Surgery System*	*Wired*	*Intent Modification, Control Hijack*	*Motor Performance, Network Performance*	*Recommendations for Network Monitoring*
[43]	*Multi-Robot System*	*Wireless*	*Denial Of Service*	*Lack of Connectivity*	*Network Monitoring*
[4]	*Remote-controlled Robot*	*Wired*	*Denial Of Service*	*Motor Performance, Network Peformance*	*Rule-based*
[71, 47]	*Multi-Robot System*	*Wireless*	*Node Failure, Node Misbehaviour*	*Network Performance, Behaviour Score, Node Reputation, Neighbour State, Neighbour Actions, System Configuration, Agent Position*	*Reputation Based, Consensus Based, Set-Valued Consensus*

2.3 Cyber-physical intrusion detection

To prevent attackers from achieving their goals, a variety of intrusion detection techniques have been proposed by researchers. Han et al. [98] have reviewed the challenges that may arise while developing an intrusion detection mechanism, one of the challenges that they mention is the complexity of the system's architecture. Various applications, sensors or actuators may introduce threat to the system, this therefore requires contextual knowledge of the application. If context is being used as a monitoring feature, this reduces the adaptability of the intrusion detection technique due to the system uniqueness and variability of the sensor configuration, therefore intrusion detection techniques commonly are application specific taking into account the contextual environment.

Research in cyber attack resilience for such systems, has focused on detection using a variety of techniques such as anomaly detection based on rule specification [37] where the state is being defined using pre-defined system functionality. The approach by Vuong et al. [4] shows that it is highly beneficial to monitor not only cyber but also physical features to identify cyber attacks [6], for instance to reduce the false positive rate of detection and the detection latency [5]. Various voting algorithms [46] where system nodes are interacting with each other to identify an attack based on behaviour rule specifications have also been proposed. A similar approach has been used by [47], where robotic multi-agents have a reputation based on their observations and try to reach a consensus regarding misbehaving robotic agents. Most researchers agree that a cyber-physical system's security has to be improved at the design stage and for this reason propose the use of more secure communication [99] or the integration of gateway firewalls [41].

Another point of view is to evaluate mission success threats based on the risk of failure. For instance, Orojloo et al. have developed a method for evaluation of the security of cyber-physical systems [100] by measuring the mean time to system security failure with regards to system components and different types of cyber attacks. Majed et al. [101] have proposed a framework for evaluating cyber threat exposure for energy smart-grids by using attack trees and attack-graphs. A variety of reliability [102] and survivability [103] models have also been proposed for cyber threat evaluation.

Yampolskiy et al. proposed a language describing attacks on cyber-physical systems [104]. This language would enable the impact of certain attacks applicable to specific systems to be described. When it comes to threat analysis, there is little research done on quantification of threats. One example from Sandia National Laboratories [105] uses a threat-driven approach for cyber security evaluation of organisations. Some aspects of their findings can be taken into account when a cyber-physical system is evaluated. The majority of these approaches and frameworks take into account an attack based on methods, conditions and impacts.

While very mature for conventional computer systems, the area of intrusion detection is

relatively new in the area of cyber-physical systems, such as vehicles and mobile robots. A relatively common approach is to use a human expert to first specify the safe and unsafe states of the vehicle and determine a large number of rules that cover all potential states, in what is known as behaviour-specification intrusion detection [37]. Rules can also be determined through a more automated learning phase without the involvement of a human expert: The vehicle is subjected to a series of different attacks, observing their impact and training a machine learning system to recognise these. Examples of such supervised learning approaches for the detection of attacks against robotic vehicles can be found in [4, 6, 5], where the rules are formed by a decision tree, which takes into account both cyber and physical features.

When a vehicle does not operate in isolation, but belongs to a team of vehicles, which can make similar observations about their environment and each other, intrusion detection can be based on the identification of misbehaviour of one of the members of the team. There, reputation-based approaches [47] and voting algorithms [46] can prove very useful. For instance, if one vehicle veers off the predefined route or reports very different sensor data, this can be considered as an indication that it may have been compromised..

In most research presented above, researchers have taken into account attack characteristics as input to identify anomalous behaviour. In other words, the type of attack is pre-defined. This limits the practicality and likely effectiveness of a protection mechanism to attacks that have already occurred and are known to the system at hand. Such protection mechanisms are called signature or knowledge-based and are capable of achieving high level accuracy [106] with low amount of false-positives. The benefits of such approach are applicable to systems which are well maintained and the signature knowledge is updated frequently as attacks can evolve and variety of attack vectors can be exploited. However, these assumptions do not hold in most operational environments applicable to cyber-physical systems, such as robotic vehicles, whether autonomous or not. Assuming that a new attack will look like one that has been seen before is reasonable for conventional computer networks, where millions of variations of the same attacks can be seen in the same year. For cyber-physical systems, this is less so, because attacks are less

common and have very different impact depending on the type of system targeted. As a result, knowledge-based approaches, where the vehicle is trained to see specific attacks perform poorly when they encounter new types of attack. At the same time, assuming that a robotic vehicle will belong to a team, where group observation can help spot signs of cyber compromise can be unrealistic in many operational environments

Researchers have experimented with methods for detecting anomalies, but usually only for a particular aspect of a vehicle's operation. An example for aircraft is the detection of false automatic dependent surveillance-broadcast (ADS-B) messages, used by aircraft to broadcast their position to other aircraft and to air traffic control. Strohmeier et al. [107] achieve detection by monitoring statistics regarding the received signal strength (RSS), as it is assumed that false signals would be coming from the ground and thus would have different RSS than signals coming from the aircraft. A similar logic can be followed to protect autonomous vehicles that rely on GPS signals, as, coming from satellites, legitimate GPS signals are naturally much weaker than spoofed signals that would come from a terrestrial source [108].

The authors of [109] have carried out a survey on intrusion detection and have mentioned that the mobile nodes tend to be connected to wireless ad-hoc networks and suggests using anomaly detection IDS on-board rather signature-based that tend to be integrated on wired systems. Mitchell et al. [110] have proposed a specification based intrusion detection approach for unmanned aircraft vehicles, where they transformed rules in to states as features of the unmanned vehicle saying that if the system violates safe-states it will be treated as an abnormal state behaviour. However, specification or rule-based detection mechanisms act as certain limitations of variables such as states, sensor readings or any other contexts. This leads us to a list of generic primitive intrusion detection (which don't take context into account) types which are: signature and anomaly based [111]. To learn the context automatically, a variety of techniques can be used for example using Neural Networks. Alheeti et al.[112] have proposed a hybrid methodology for connected driverless cars that monitor the network and extract key features from the data using proportional overlapping score. This has been tested on the Kyoto benchmark

dataset which was gathered by honeypots and classified using a variety of methodologies. Their findings show the necessity to use only the key features that will be a feed into an Artificial Neural Network, their results showing an intrusion detection accuracy increase by reducing the features that might be unrelated. Ali et al. [113] have followed their work and used a combination of Forward-Feed Neural Networks (FFNN) and Support Vector Machines (SVM) to form a behaviour that can be normal or abnormal, this research is also focused on Vehicle Architecture Network (VANET) monitoring. They propose a response mechanism to a grey hole and rushing attacks that will force vehicle to go in to safe mode which will monitor the traffic on VANET so the vehicle would be able to identify legitimate traffic.

Interestingly, there is also on-going research on intrusion detection system for the in-vehicle network. Waszecki et al. [114] have proposed monitoring internal network traffic using a simple Leaky Bucket approach. In their paper, they have applied this approach to a single CAN bus feature, which is the frame arrival time, demonstrating that a simple methodology not requiring vast amounts of resources is capable of detecting malicious activity on the bus.

Based on the system resource restrictions, other approaches still may apply, such as the work by Kang et al. [115], who have proposed to use Deep Neural Network (DNN) for monitoring the CAN bus network and detect malicious activity on the bus. However, DNNs are computationally heavy as they require a lot of processing power to teach the neurons from the data, on the resource constraint systems this approach would be unlikely be integrated. Nevertheless, Loukas et al. [116] have shown that they can indeed be used if offloaded to a more powerful infrastructure, as long as the network is sufficiently fast. This can both reduce detection latency and perhaps more importantly also reduce energy consumption.

A further step was made by Theissler et al. [117], where multiple methodologies were combined to form a hybrid that would be capable of detecting known and unknown attacks in automotive systems using an ensemble-based anomaly detection approach. They

have used four Two-Class classifiers which are: Mixture of Gaussians (MOG), Naive Bayes, Random Forest and SVM. Additionally they have used four One-Class classifiers which are: Extreme-Value, Mahalanobis, One-Class SVM and a Support Vector Data Description (SVDD). All these classifiers in combination have shown high results for known faults, as well as to the unknown faults. One-Class SVM has found its uses in cyber threat identification of autonomous avionic systems [118], where researchers were able to detect Teardrop, Fuzzing, Port Scan and ARP scan attacks.

2.4 Discussion

It is notable that machine learning is commonly proposed in intrusion detection for vehicles, e.g. driverless vehicles, and also it is important that perception of IDS has gradually shifted towards behaviour classification whether it would be rule-based [77], consensus-based [47] or reputation-based [71]. Knowledge-based detection requires too much information to be known beforehand, and can be particularly weak in the face of zero-day threats. Considering that very few attacks have been recorded and studied for cyber-physical systems, most threats are new threats.

Also notably, most research conducted in the field of cyber-physical intrusion detection is based on mathematical analysis and simulation without experimentation with real systems. Arguably, there is a need for testbeds developed to support experimentation with real attacks on real vehicles, especially autonomous and semi-autonomous vehicles. The challenge there is not only in the design of the proof of concept ideas, but also in the implementation given the severe lack of resources expected on such a system. Another challenge is that monitoring multiple features means that the intrusion detection system needs to account for the variety of value ranges for each feature. For example, infrared sensor reports may be "low, medium, high", while distance sensor reports may be numeric values.

In general, it appears that most detection approaches for vehicles are explicitly or im-

plicitly system-specific. There is a need for an approach that can be applied across a variety of vehicles, and adapt by learning what is normal for that vehicle, so that it is also applicable to unknown/future threats. In the next chapters, we present a testbed for cyber-physical attack experimentation for autonomous vehicles and methods for achieving this dynamic behaviour-based cyber-physical intrusion detection, whose architecture and logic does not require knowledge of the design of the system it protects.

Chapter 3

Robotic System Development

3.1 Robotic Testbed

To facilitate a detailed investigation of autonomous techniques to detect cyber-physical attacks, we have built a richly-instrumented robotic vehicle testbed (Figure 3.1) with a variety of different sensor types which are representative of those used in deployed systems in real-world applications. The control system of the testbed comprises an integrated set of modular embedded systems. It uses a variety of communication protocols that are used in industry robotics, such as CAN [40], RS-485 [119], WiFi [120] and ZigBee [121, 122]. An overall communication architecture of the robotic vehicle can be seen in Figure 3.4 where all components are combined to illustrate overall testbed design.

Table 3.1: Robotic Testbed Installed Equipment

Feature	**Purpose**
CAN bus	Internal communication
ZigBee	External communication
WiFi	Media streaming
Compass Bearing	Navigation correction
Pitch and Roll	Angular momentum
Heatsink Temperature	Electric load heatsink temperature
Robotic Arm Gripper	Picking up objects
DC Motors	Movement
Ultrasonic Rangers	Collision avoidance

System components produce signals and feedback that is used by other system components to change overall system behaviour. Several components that are mentioned in Table 3.1 produce instrumentation data which are used as cyber or physical domain indicators that can help identify a particular behaviour of a system, this will be discussed in detail in Chapter 4. All sensor data are generalised and are treated as data sources. The computational processing in the robotic vehicle testbed is distributed across the variety of embedded processors on the testbed platform.

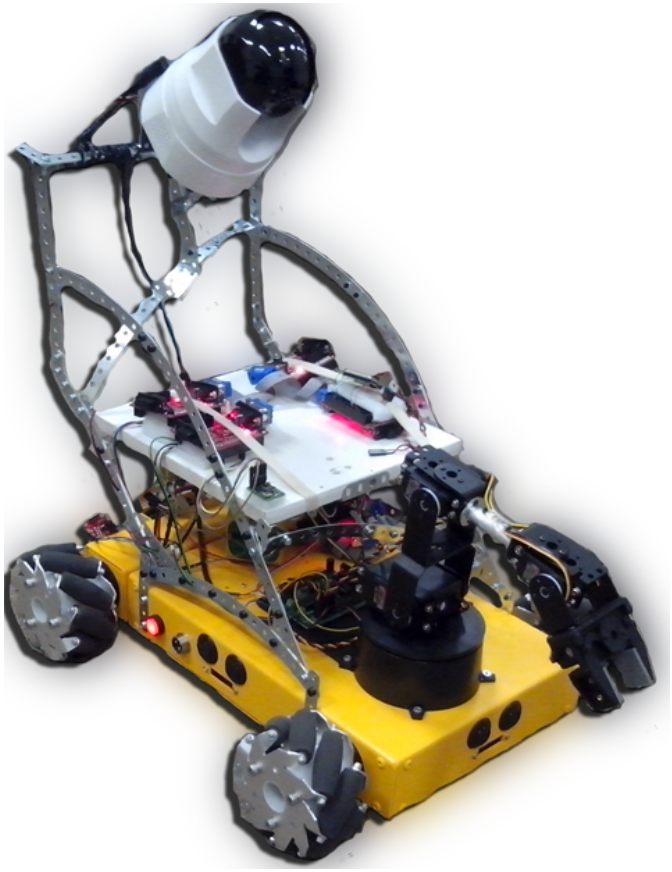

Figure 3.1: Robotic vehicle testbed

The robotic vehicle testbed consists of multiple computing node types, one of them is the AVR-CAN development board with a AT90CAN128 micro controller unit, several of which are used to host specific sensors, additionally a variety of sensing or actuating components share their processing node. For instance, a single node is responsible for processing bearing, pitch and roll sensors, whilst other node is responsible for the analogue to digital conversions. Overall the system contains of six processing nodes, five of which are the

AVR-CAN development boards that are clocked at 16 MHz, the other type of computing node is the STK300 Kanda board powered by Atmel ATmega1281 micro controller clocked at 8 MHz, which is responsible for navigation of the robot.

System components allow the robotic vehicle testbed to undertake a variety of autonomic tasks, such as navigation based on the logical mission layer that represents a sequence of steps given to the testbed. The sensors allow the vehicle to navigate autonomously in an environment using the compass bearing to keep track of the direction, ultrasonic rangers for collision detection and avoidance, and pitch and roll sensors to make direction corrections and inform the system about the environment volatility. Additionally, the system uses an informative meta-data sensor that measures the temperature of the heat sink connected to the on-board voltage regulators which supply power to the camera and the robotic arm. In this way, the system is able to determine if these heavy-current-drawing system components are in use. These sensors and additional meta-data extraction allow automatic characterisation of the behaviour of the robotic testbed vehicle whilst in operation.

To gather the data for off-line analysis, we use an external workstation. The sensor data from the testbed is collected and is stored in a knowledge base on the operator's workstation, however the sensor data which is being transmitted on the internal network is stored on the Raspberry Pi Flash memory. The external communication between the workstation and the robotic testbed vehicle is achieved through using a dedicated ZigBee network. The ZigBee connection also allows us to transmit commands to the testbed (e.g. to initiate missions). The main advantage of ZigBee technology is that it provides a low power communication capabilities and provides various network topologies such as star topology and allows to use peer-to-peer communication, provides data rates up to 250 KB/s [123] and it is commonly used for wireless sensor networks [124] or robotics[121]. The camera is a self-contained unit; its audio and video feed is streamed using a standard WiFi communication protocol. An overview of high-level communication architecture between the workstation and the robotic testbed vehicle can be seen in Figure 3.2.

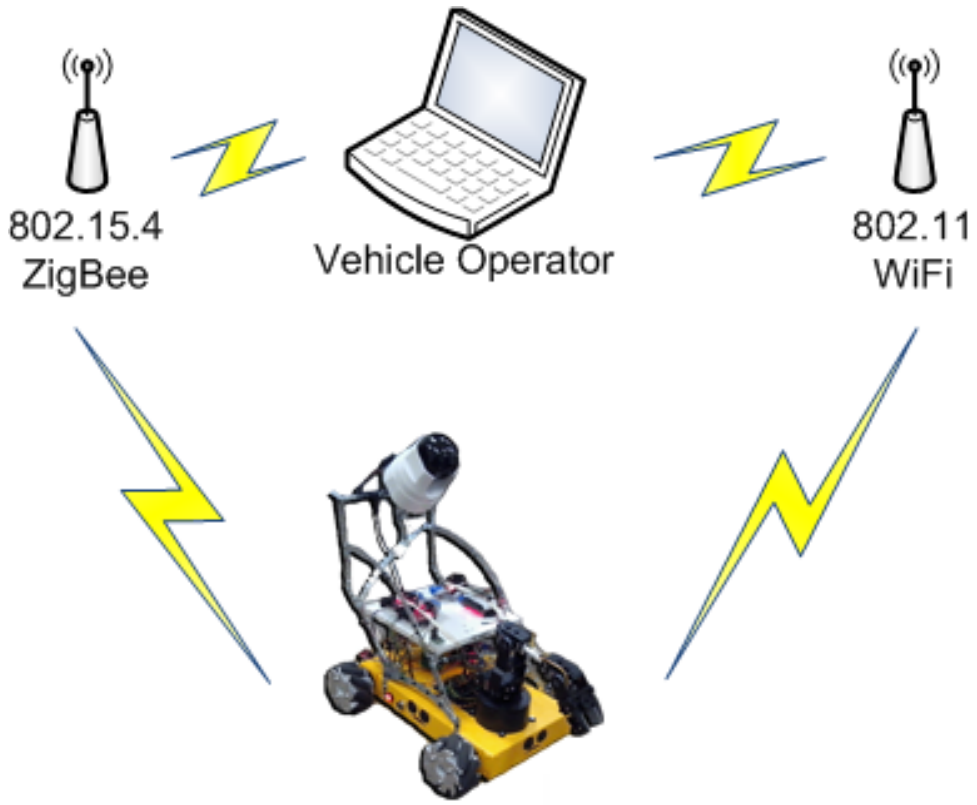

Figure 3.2: High-level communication

A variety of commands can be sent to the robotic vehicle as simple navigation commands, camera or robotic arm control commands. Additionally, the vehicle supports more complex mission task uploads. The command transmission is a one-way communication functionality, commands are only being executed if they are received from verified ZigBee network node and are in a correct format. The robotic vehicle testbed does not send any commands to any external nodes within the ZigBee network. The testbed will periodically report its instrumentation data to a verified connected workstation. The instrumentation report periodicity is one second, due to the low bandwidth ZigBee protocol and unique ZigBee ZE10 module behaviour it is not possible to increase sampling rate with current equipment configuration. Therefore higher-rate sample aggregation is performed on-platform on the sensor hosting nodes and a Raspberry Pi.

For the communication between system components internally, the testbed utilises CAN bus standard. This communication protocol is used commonly as internal communication channel in vehicles [40, 41]. The CAN bus is used to share overall sensor data from all data

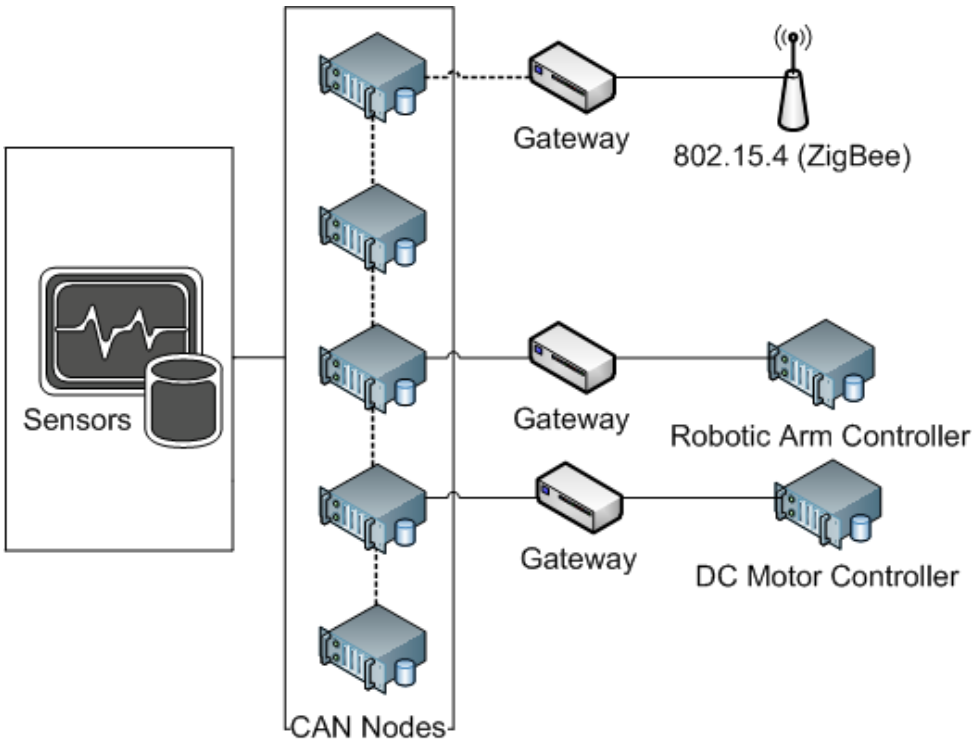

Figure 3.3: Internal Communication: gateways connect different subsystems

sources, including additional meta-data extracted during data analysis by the processing nodes. The internal communication architecture is shown in Figure 3.3. If required the data is re-transmitted to other nodes through gateways and collected at the reporting node which will transmit data to the workstation when appropriate.

The software structure of the robotic vehicle testbed uses a layered architecture, which separates the different levels of reasoning from the lowest physical sensor level, represented by individual embedded nodes performing analogue to digital conversions interpreting signals into an understandable software language. This is shown in the Figure 3.5 and is followed by a detailed description of each layer. The next-higher level is the classification layer where data is analysed using statistical analysis approaches, such as exponential smoothing to determine the trends in the data. A level higher, we have an autonomous module controller layer which controls actuating capabilities based on the data received from the lower layers of the model. The autonomic module controller layer is a set of autonomic controllers that are carrying out their defined tasks, such as robotic arm movement or navigational control. A mission layer then collects knowledge from autonomic

controllers and evaluates if the expected mission goal has been achieved. The layered software approach improves flexibility and maintainability in terms of a robotic vehicle testbed programming, as it separates central and facilitates modular design. These layers are implemented as a set of libraries that can be extended further.

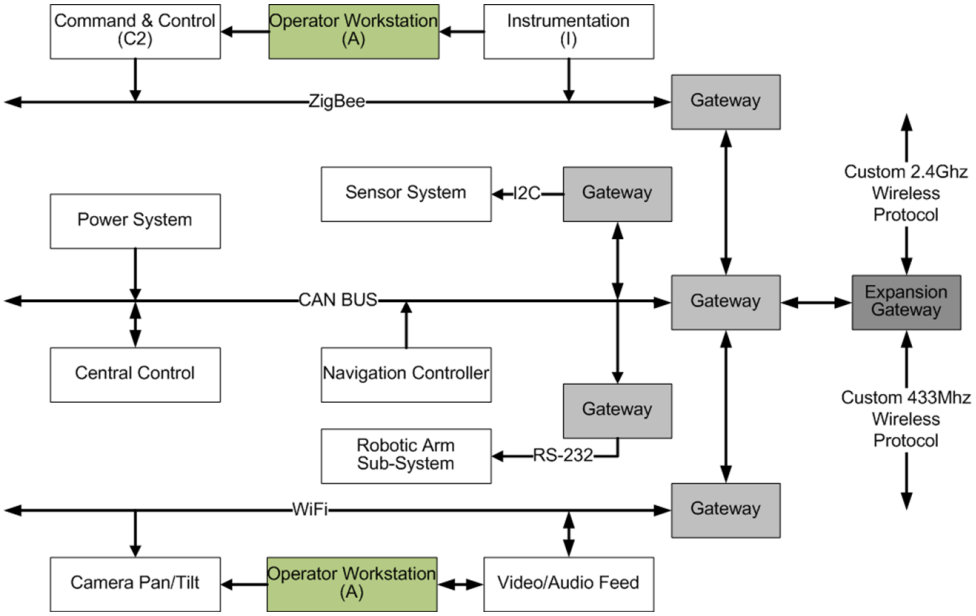

Figure 3.4: Robotic Testbed Vehicle: Communication Architecture

In a real-world environment, there is a variety of physical threats to the system that can be caused by an unknown factors, such as rain which affects the grip with the road, windy weather that can affect vehicle's movement or any other environmental factors that may impact the robotic vehicle. An issue can arise when an attacker will target a specific sensor to disrupt its activity during the learning process, this will affect an overall operation of the vehicle at later operational stages. One of the examples would be to disrupt a compass intentionally during the learning stage such that the robotic vehicle will learn the disrupted pattern as being "normal". We minimise this risk by securing our vehicle from attacks during the learning process.

3.2 Testbed Design: Physical Architecture

The robotic physical architecture consists of the base, four mecannum wheels that allow a robot to perform a navigation in all directions including movement to the side [125]. The base is extended using "Meccano" to increase space for the processing modules, this can be seen in the Figure 3.1. The ultrasonic sensors are embedded in the base of the robot which allow to monitor distance from all sides of the robot, additional sensors were added to the robot as the robot was developed and placed in the appropriate locations. In terms of actuators, the robot has four motors with a 64:1 gearbox, arm gripper and the camera. The camera is controlled over the WiFi communication channel by an operator, other actuators are controlled over the CAN bus or through a set of communication gateways.

The communication between the nodes is carried out using variety of protocols, including RS-232/RS-485 standards, CAN, I2C/TWI. The details of the sub-systems and communication protocols are demonstrated in the Figure 3.7. The communication between various modules is carried out over the CAN bus. In the system each module is responsible for its task based on their mission role or a specific task. For example, the Motor Controller Module would control the speed and direction. Other modules sense the environment for obstacles and measure the environmental humidity and temperature, such context creates their role in the system. The conceptual idea is that the communication within the testbed share their information on the bus and modules that depend or interested in that data are using it.

3.3 Testbed Design: Logical Testbed Architecture

Flexibility and modularity in the testbed is achieved through a development of a conceptual logical architecture that can be distributed throughout the system. The idea is to combine embedded systems and autonomous mechanisms in a layered structure, where each layer represents a logical role in the software stack. The software stack is shown in

the Figure 3.5.

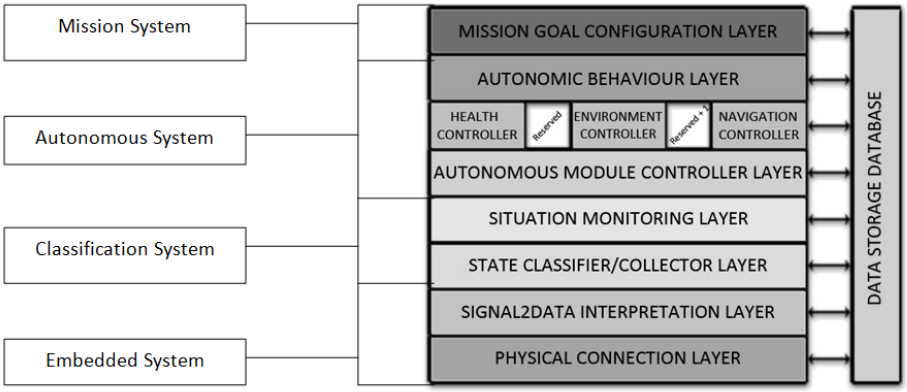

Figure 3.5: Robotic Testbed Vehicle Software Architecture

The **Physical Connection Layer** – is used to show the physical connection between system components whether it would be a wire or a wireless connection. This layer was deployed as it can be a source of an attack for example in Cho et al. [59] work they have demonstrated how an error can be generated by using aggressive interference (i.e. ignoring CAN protocols and standards) on the CAN network that may force ECU to shutdown.

The **Signal2Data Interpretation Layer** – is used to express the embedded system itself. This layer can be a sensor, actuator or a module which consists of multiple systems. The role of such system in terms of this layer is to convert digital or analogue data to meaningful data. This layer also takes into account communication protocols as they are required to be supported by an embedded system to translate the signal into understandable data that can be used further.

The **State Classifier/Collector Layer** – is used to collect the data from all sensors at the specific time to represent a state at "this exact" time. Also this layer covers the periodic state sampling i.e. data samples from the sensors at the configurable moment in time will be stored, the number of the data samples will stored as well as a contextual information about the state which may be used as an indication of an attack or a failure of the node.

The **Situation Monitoring Layer** – is used to identify the current situation of the robotic platform which currently exists i.e. taking into account previous states and iden-

tifying trends for each sensor. This layer calculation would allow to identify trend in the data where it would be needed as this data and the state data would be stored in a database in a system.

The **Autonomous Module Controller (AMC) Layer** – is responsible for making overall decisions based on the current situation, making correction to the mission goal and alter autonomic behaviour of the robot. This layer manages all available controllers, as an example the Navigation Controller is able to make corrections to the vehicle's speed, but only the AMC layer is able to change the direction of the robotic vehicle or make any other significant changes to current system behaviour.

The **Distributed Controller Layer** – is used to approach controllers as a structure consisting of multiple controllers. These include: navigational controller which is responsible for the control of the speed and regulate it based on the encoder feedback, environmental controller is responsible for obstacle detection i.e. will identify obstacles, measure the temperature if required by the mission or monitoring battery discharge and possibly make an independent decision on whether it should proceed with a mission. This information would be shared with other controllers or actuators that could rely on the controller's data. All these controllers operate in parallel because they are distributed throughout the system.

The **Autonomic Behaviour Layer** – is used as a rule-set of limitations and restrictions. As an example during the mission the system should avoid any obstacles within the 50cm range, or when platform identifies that it cannot exceed certain speed. Autonomous Module Controller will make a decision, to set maximum speed restriction on the motor speed, that the platform cannot exceed.

Mission Goal Configuration Layer This layer is used as a mission reference. For example, it holds information on its patrolling mission such as path, speed, particular restrictions for a particular step.

To summarise these layers we will briefly discuss them: an embedded system represents a

physical connection and conversion of analogue or digital signals into meaningful data. A classification system represents data collection from various sensors and actuators and its classification in a specific context. An autonomous system represents a decision making mechanism that takes into account collected data and controls a subject based on its decision, restriction, mission or limitation. A mission system represents the mission goal of a series of steps containing certain restrictions and change of behaviour. All these layers represent their knowledge base of the facts. For example, an embedded system deals with the physical system data, such as signals or raw data values. A classification system classifies the data by adding additional context to the data, such as calculating trends, averages or standard deviation. By performing such operations, the system transforms original data into a meaningful contextual knowledge. This contextual knowledge is passed to the higher levels, as there is no need to transmit the whole data set, this reduces the overhead that needs to be transmitted reducing bandwidth requirements. The data storage database has records of the instrumentation data and is able to provide access to all data samples for all layers that request it, however this will increase an overhead if the data handling is not robust.

The autonomous system deals with data that might be imprecise due to the higher level of context already being available. To achieve higher level of power efficiency by reducing processing requirements, the data needs to be transformed into something meaningful that does not require additional processing. The example of such transformation would be that the node one has access to the data set and the second node requires only the mean value from the data set that is owned by the node one. If node one will perform this calculation and will pass the result to the second node, the second node will have no knowledge about the amount of samples. In this situation, the second node added the context to the data and to reduce the knowledge requirement, the data has gone through prior processing by the node one.

3.4 Testbed Design: Functional testbed design

Here, we have chosen the Generic Vehicle Architecture style representation for our architecture [1]. The "Generic Vehicle Architecture" is the UK Ministry of Defence, Defence Standard 23-09 [126, 127]. This ensures applicability of the approach in a large variety of autonomous and other vehicles, such as those developed and used by the UK Ministry of Defence.

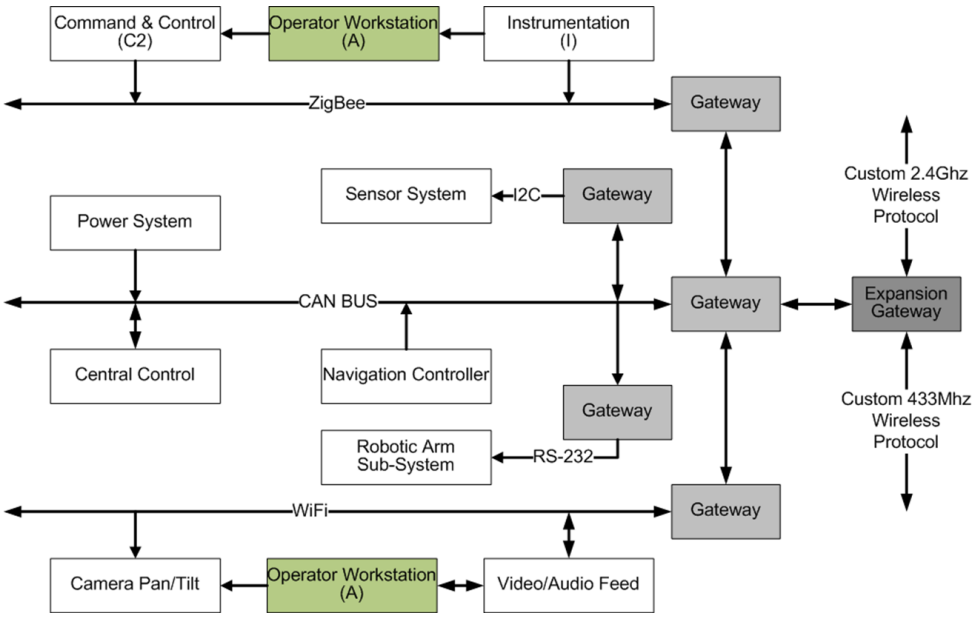

Figure 3.6: Robotic Testbed Vehicle: Communication Architecture

The robotic vehicle testbed consists of three main busses. Internal system communication uses the CAN bus and External communications which is commonly is used in the robotic industry [121, 122, 40]. The CAN bus will interconnect all modules that are responsible for the autonomic behaviour of a system. The power system consists of sensors that monitor the power supply. The sensor system is connected to a bus using different Gateways dependant on the communication technology used by the system. The navigation Controller is used to monitor and control the motors. The actuation system consists of available actuators in the system and all connections to the bus are done using a gateway, as different actuators may require specific communication protocols. The Central Control unit will be responsible for mission execution, diagnostics and logical decisions and this

will be described in the next section. As an improvement for our robotic platform we will implement other wireless technologies that might bring additional attack vectors and opportunities for intelligence gathering to identify the patterns of an attack.

ZigBee communication is used to supply the operator with instrumentation data. That is if operators have to make changes to an existing mission, they can send a command, sequence of commands or upload a mission to the system that is executed on reception. Video and audio are streamed using a camera that is physically attached to the robotic testbed. The communication is carried out using a dedicated channel to stream video and audio feed, received by an operator. The operator is able to use a graphical user interface to change cameras pan and tilt settings. The detailed architecture is illustrated in Figure 3.4 and 3.8. The power system consists of a monitoring sensor which measures the battery

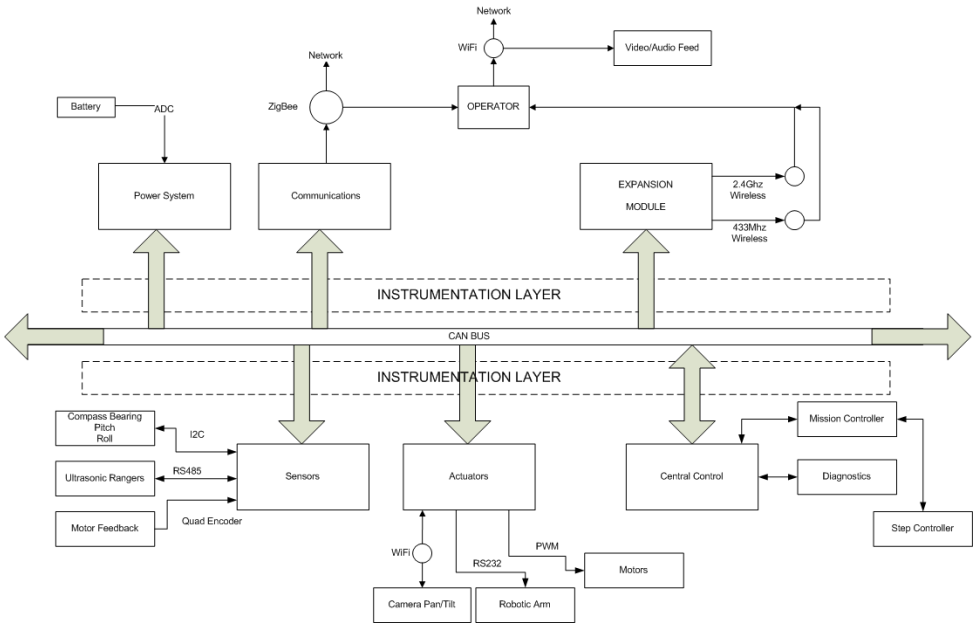

Figure 3.7: Robotic Testbed Vehicle: Subsystem Communication Architecture

power supply levels. This knowledge can be used as a metric for attack identification, as described in Chapter 4. All sensors in the system are distributed, and due to specificity and uniqueness of each sensor or an actuator, they may or may not use the gateway to access the main bus. The actuation system in the testbed supports a robotic arm using the RS-232 communication protocol. The navigation controller is responsible for controlling the vehicle's speed and movement direction.

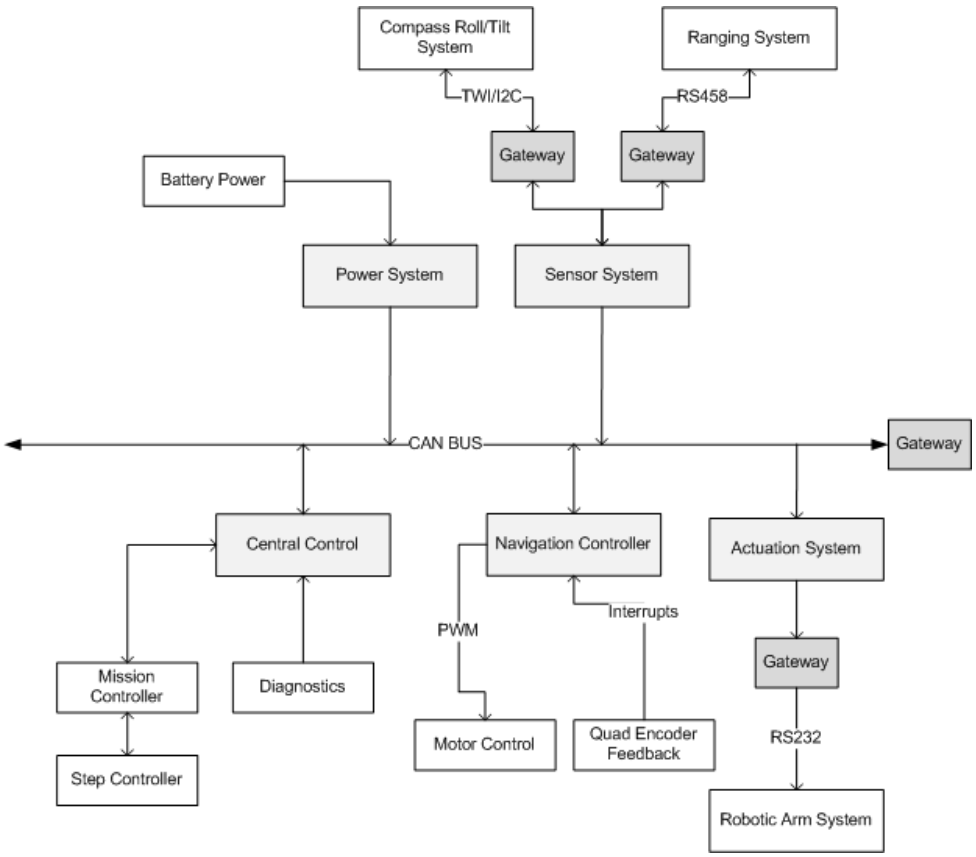

Figure 3.8: Robotic Testbed Vehicle: High-Level Internal Subsystems Overview

3.5 Testbed Design: Behaviour Logic

Mission behaviour from a mission level perspective is illustrated in Figure 3.9. The mission behaviour entities are defined as follows:

- Mission is a series of steps (e.g. move forward, read distance)
- Step is a series of actions (e.g. move forward avoiding obstacle hit)
- Action relies on sensors and their limitations (e.g. distance readings being limited to 2 m)
- Sensors contain general attributes and sensor-specific attributes

The Figure 3.9 demonstrates the higher-level generalised logical mission behaviour of a robotic system, where “Mission” entity has a series of steps, a step may contain series of

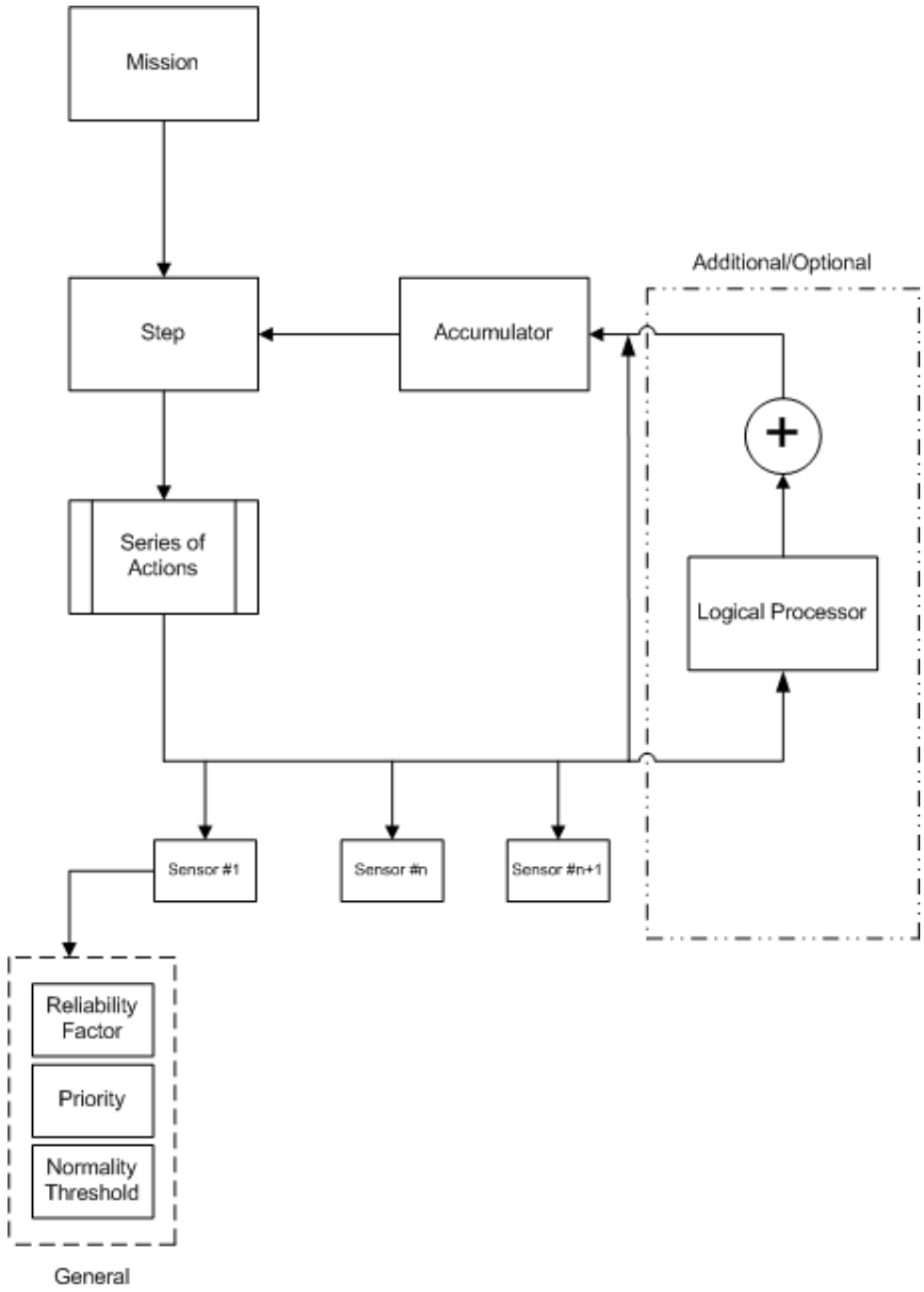

Figure 3.9: Robotic Testbed Vehicle: Generic Mission Logic Behaviour

actions. All actions rely on the sensors that have their own attributes. The sensor specific attributes are: Reliability factor, Priority and Normality threshold. The reliability factor is a measurement which describes the reliability of the sensor and its accuracy. The priority attribute defines the priority of the sensor for the step i.e. the step task is to move forward enabling obstacle detection feature, the ultrasonic sensor will have high priority as the step relies on this specific sensor. The normality threshold is the average

deviation of the sensor.

For example, a basic scenario may be that the robotic vehicle has to scan a two-meter square area for any obstacles, and if it finds any obstacles it notifies an operator that attention is required. The operator at the mission level can set coordinates on the map and enable repetition. So, the mission would be to drive on a square path, each side of a square being 2 m and report if anything is found within a specific range. This can be split into two repetitive steps, moving forward and turning, and within the step execution, it takes into account all related sensors to the mission. This illustrated in Figure 3.10: As a

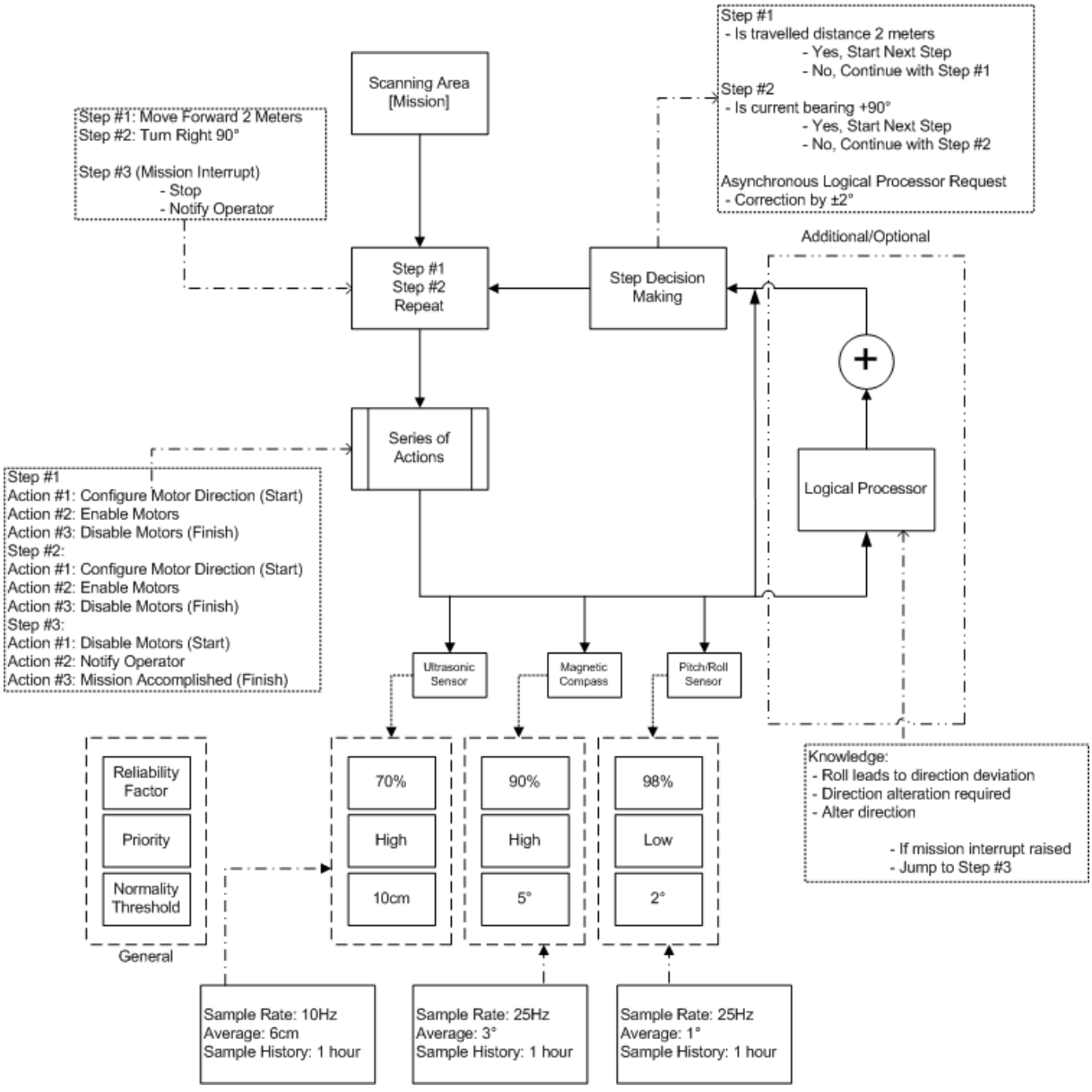

Figure 3.10: Example of Mission Logic Behaviour

step consists of several actions, in context of a current mission these actions would require to sample the range in all directions and actuate the motors controlling speed. From a sensors perspective, it has certain attributes such as the reliability factor and priority. For instance, ultrasonic sensors can be unreliable if a surface is made out of anechoic material or angled (where ultrasonic signal can be deflected), therefore the reliability factor being lower. Also it might be that the sampling rate can influence reliability. Higher sampling rate can notice a change much quicker. The priority of a sensor will be taken into account as well. During the step execution, sensors priority will be taken into consideration, it might be that one of the sensors will fail, but it will not influence a mission. So missions success will be unaffected and this sensor will be ignored. If a logical processor is connected to the system it look for sudden changes, and makes corrections if necessary, as demonstrated in Figure 3.10. This logical processor is a software based set of rules which does the decision making based on the sensor readings. A logical processor has the knowledge from all the sensors available on the system as well as metadata, such as packet arrival rate.

3.6 Robotic Testbed Conclusion

To summarise, the robotic testbed vehicle has been designed to facilitate a variety of experiments targeting different data sources and identifying behavioural abnormalities. Our goal is to develop a methodology that will improve robustness of autonomous vehicles using a sensor-agnostic learning approach where the type of data source does not matter, as the requirement is to learn the "normal" signal characteristics, including noise characteristics, generated by the data sources. This robotic vehicle testbed has been built to conduct experiments for a variety of navigational tasks combined with robotic arm actuation. Additional sensors can be added to extend evaluation of the behavioural model.

Chapter 4

Anomaly-based detection of cyber-physical threats

4.1 Introduction

To provide warnings against cyber attacks on an autonomous system, it is important to establish the impact that different attacks have on the cyber and physical features identified in Chapter 2. Our approach here is data-driven based on experiments run using the testbed described in Chapter 3. This chapter describes the full evolution of the methodologies that are discussed later in this chapter. The chapter starts with a discussion of the experiment environment and the design of the attack scenarios, followed by the initial study. The initial study will discuss our early thoughts, followed by an early hypothesis testing through the experimentation. A reader will find discussions on the decisions we have made and why some of the decisions demonstrated to be inconvenient and required us to re-evaluate our proposed methodology, so a reader will find a mixture of the latest results with an distinctly defined initial results. In this chapter, we discuss different versions of the detection mechanisms which are: off-board and on-board, as well as the offline and real-time detection. The data that has been used for the offline and real-time on-board detection is different, because the data format is different, but at the

same time the experiment environment and attacks were the same.

The initial experiments were based on a single mission step, where the autonomous vehicle is tasked with reaching a predetermined destination relying on its sensing capabilities. This was conducted along a narrow corridor on the fourth floor of the Queen Mary Court building at the University of Greenwich. The experiment was repeated several times, first without the presence of any attack, so as to establish what behaviour can be considered normal for the specific scenario. The assumption here is that a behaviour that differs from what is known to be normal can be considered as suspicious. This is then followed by repeating the experiment several times in the presence of three different types of attacks. These attacks have been chosen based on the literature review, cyber-physical attack impacts that are demonstrated in the Table 2.1 and throughout the discussion with Dstl. To represent an internal cyber-physical attack, we chose a highly damaging supply-chain attack, which included a rogue node located within the internal network of the vehicle. To represent a physical-cyber attack, we chose a sensory channel attack, where a magnet was attached externally to the vehicle by an adversary with the purpose of preventing the testbed from completing it's mission due to an inability to navigate using the compass. As representative of an external cyber-physical attack, we have chosen false data injection:

- Internal Cyber-Physical: Rogue Node launching replay attack (Supply-Chain attack)
- External Cyber-Physical: False sensor data injection through external ZigBee communication channel
- Physical-cyber: Sensory Channel Attack (Magnet affecting compass)

For each attack, we have reviewed the impact on different features (See Section 4.4.4) against the intensity of the particular attack. Note that some attacks disrupt the robot's behaviour, while others may not achieve any perceptible impact. The aim here is to identify those changes in the monitored features, which can help towards the detection of an attack regardless of its outcome.

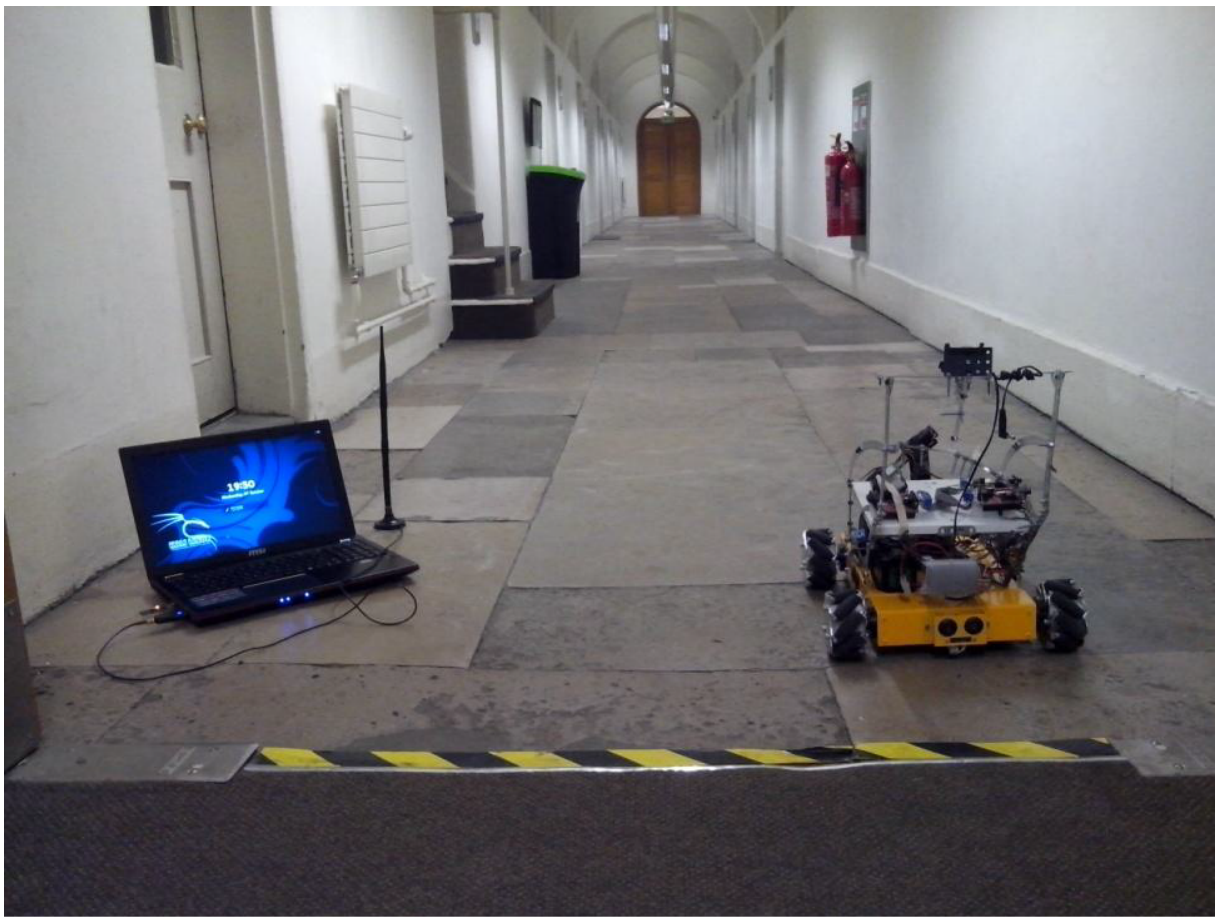

Figure 4.1: The Experiment Environment

4.2 Experiment environment

The experimental scenario selected for this project is a mission in which the robotic vehicle testbed has a simple navigation task to reach the end of a corridor using its own sensing capabilities. The complexity of such a mission is not obvious. The uneven stone flooring of the old corridor has an irregular surface which provides a non-trivial environment, with slopes and bumps, which lead the robotic vehicle to change direction often regardless of the existence or impact of any attack. The flooring has a variety of dents and lumps that affect the testbed movement throughout the experiment and introduce a stochastic randomness that is used to learn normal deviations. The uniqueness of the flooring surface disrupts the direction of the vehicle, forcing it to continuously adapt the speed of its motors and its direction to ensure that it maintains a safe distance from the walls during the mission. The distance between walls is constant. This enables us to identify

the behavioural profile of an environment based on the data source information. The space is a controlled environment that does not change throughout the experiments and was selected as it exercises all the sensor capabilities of the testbed. The reason that we have chosen this equipment setup is discussed in Section 3.1.

The corridor is 28 m long and the distance from wall to wall is 2 m and has a set of inset door openings on either side which allows observation of periodic behaviour in the ultrasonic distance sensor signals as the vehicle passes by. The space used can be seen in Figure 4.1.

The initial experiments were each repeated five times to ensure that the collected data set was representative and these were used for the creation of the initial behavioural profile. Two further experiments were used to evaluate the initial behavioural profile. The behavioural profile was built using patterns in the variation and background noise in the data sources. The focus was on the spikiness of the data variations and the variety of deviations. The experimental environment facilitates repeatability and contains static elements that can be used as guideline features during analysis of collected data, but it also introduces significant stochastic elements which are essential for understanding the normal levels of noise and variability in sensor signals.

The first step involves the learning phase, where over several runs we collect the learning data set that will allow us to create a “normal” behavioural profile. The second step is to evaluate the recognition of a “normal” behaviour profile, as well as the representative normality value of a signature obtained during the learning phase. A signature characteristics which are shown in Table 4.8 are applied to each data source (forming data source profile), a combination of all learnt signatures for each data source will form a “normal” behaviour profile of a system. This “normal” behaviour profile will be used to monitor normality of the system. To reduce computational complexity it is possible to observe system’s normality score without the need to analyse each data source and its behaviour i.e. monitor an overall anomaly index score for the system, if an anomaly index score has exceeded the normality range, the system will examine each data source for anomalous

behaviour and the number of anomalies in its signature characteristics, this will allow to reduce an overall computational power requirement of the system. This will however potentially increase the false negative rate.

4.2.1 Normal scenario: Non-attack

The robotic vehicle's mission is to reach the door at the end of the corridor, while navigating the uneven surface, which forces it to continuously adapt its direction based solely on its sensing capabilities. For the reference a reader can observe the data from several sensors in Figure 4.2 during robotic vehicle's testbed operation.

4.2.2 Cyber-physical attack: Rogue Node

This scenario evaluates a situation where a rogue node has been planted in the system, typically through the supply chain. After a delay, the rogue node is activated, assuming a logical bomb approach. When activated, it starts replaying packets that it can access on the network, as described in Table 4.1. Although based on packet replay, this is effectively an internal denial of service attack, which aims to overwhelm the internal network and impede the exchange of messages/commands between the various internal nodes. The attack was structured such that it would support variability, so a packet amplification approach is used. The duration of an attack is an attribute that allows us to perform attack for certain period of time. This attack was extended further with an additional attribute where we execute attack 2 times i.e. per scenario the rogue node is activated twice for 25 seconds. For the reference a reader may observe the impact in the Figures 4.5, 4.7.

Intensity	**Replay Packet Amplification**	**Attack Duration**	**Attack Repetition per Scenario**
Low	x3.5	25 Seconds 40 Seconds	2 Times
Moderate	x10		
High	x85		

Table 4.1: Rogue node attack settings

4.2.3 Physical-cyber attack: Sensory channel attack

Autonomous robotic vehicles rely heavily on the healthy operation of their onboard sensors and associated communications and analysis systems. An adversary can attempt to exploit the physical weaknesses of the sensing technologies employed to affect the success of an autonomous vehicle's mission. The example used here is to physically attach a magnet on the vehicle while it is moving. This affects the onboard compass and the associated steering capability of the vehicle. The magnet is removed 40 seconds later to measure the differences in the features before, during and after the physical disruption. The impact of such attack is demonstrated in Figures 4.4, 4.8. This attack has been improved further by introducing more structured approach and as an attribute attack we have used distance between the neodymium magnet and the compass sensor. The complication of repeatability of such attack is that we are unable to control the magnetic field accurately, however further study showed that it has a massive impact on the navigation capabilities which is noticeable by the detection mechanism, however in case where robotic vehicle does not rely heavily on the sensor, the attack deviations might not be noticed.

4.2.4 Cyber attack: False data injection

In the external packet injection scenario, we have exploited the existence of a Zigbee link that the legitimate operator uses to communicate with the vehicle. Here, the adversary sends spoofed commands using Killerbee wireless devices [128]. A variety of data types have been used to measure the cyber-physical impact of the attack on the features monitored and the details of these experiments are shown in Table 4.2:

Sensor Data Packet Injection				
Intensity	**Delay between packets**	**Duration**	**Type of Data**	**Recipient**
Low	1 Second	40 Seconds	Sensor data only (collected during robotic vehicle testbed development) (1) (collected during robotic vehicle testbed operation) (2)	Broadcast Message
Moderate	0.5 Second			
High	0.3 Second			
Command Data Packet Injection				
Low	5 Seconds	40 Seconds	Command data only (commands are: turn left/right, move forward/backward, stop)	Unicast Message
High	0.5 Second			

Table 4.2: External packet injection settings

The data behaviour during such attacks can be observed in Figures 4.3, 4.6. Further this attack scenario has been improved to make an attack more realistic, we have captured the packets whilst the robotic vehicle was undergoing the mission, and these packets have been replayed at the exact timing in other scenarios, so we would be able to evaluate if the proposed method is able to detect slight variations in the sensor data.

4.3 Observable experiment results

Using the experimental results, we were able to identify a wide variety of changes across the cyber-physical plane that can be attributed to each attack. In total, we were able to gather real-time measurements for 17 distinct features which are noted in Table 4.3, both physical (sensor-based) and cyber (communication and processing-related). As an initial

Table 4.3: Description of the data sources

Physical Features		Cyber Features	
Name	Abbr.	Name	Abbr.
Battery Voltage	DS2	Packet Arrival Time	DS1
Compass Bearing	DS3	Action Indicator	DS11
Pitch	DS4	Sequence Number	DS12
Roll	DS5	CAN Packet Rate	DS13
Front Distance	DS6		
Back Distance	DS7		
Left Distance	DS8		
Right Distance	DS9		
Temperature	DS10		
Motor #1	DS14		
Motor #2	DS15		
Motor #3	DS16		
Motor #4	DS17		

study we have conducted attack experiments and were observing noticeable results which are described in Table 4.4. The mission impact has been evaluated as visually observed robotic vehicle behaviour and has been classified as None, Low, High and High (Failure).

The classification None describes that there was no observable impact on the robotic vehicle routine-mission task, classification Low defines that there was slight observable

Cyber attack	Intensity	Significant changes across the cyber-physical plane	Mission Impact
Rogue Node (Packet Replay)	Low	Increased internal network packet rate (Figure 4.7) Frequency of spikes Coinstantaneous spoofed sensor data from certain sensors while under attack	None-Low
Rogue Node (Packet Replay)	Moderate		None-Low
Rogue Node (Packet Replay)	High	Increased internal network packet rate Coinstantaneous spoofed sensor data from majority of sensors while under attack (Figure 4.5) Non-Responsiveness	High (Failure)
Sensory Channel using Magnet	N/A	Abrupt change of compass data to a new near-constant value (Figures 4.4, 4.8) Coinstantaneous sensor data after the change Inconsistency with mission expectation from other sensors	High (Failure)
External Packet Injection (Sensor Data)	Low-High	Increased external network packet quantity Coinstantaneous spoofed sensor data from majority of sensors while under attack High frequency of spikes Coinstantaneous sensor data from certain sensors Repetitive sensor data Enviroconsistency from previous mission knowledge (Figures 4.3, 4.6)	None-High
External Packet Injection (Command Data)	Low	Difference between operator's and autonomous vehicle's measurements of external traffic Inconsistency with previous mission knowledge	High (Failure)
External Packet Injection (Command Data)	High	Significant difference between operator's and autonomous vehicle's measurements of external traffic Power consumption increased Abrupt variation of actuator data	High (Failure)

Table 4.4: Cyber-Physical Impact Indications

impact such as delays on communication channels, High classification defines that the robotic vehicle had observable misbehaviour such as disruption of navigation tasks, no-

ticeable delay in communication, however the robotic vehicle was able to recover. The High (Failure) classification defines full robotic vehicle disruption such as actuator failure, unresponsiveness of communication and robotic vehicle was unable to continue its mission and was unable to recover. As a reference Figure 4.2 can be used as it demonstrates the ideal data variation during the normal experiment scenario, where y-axis shows the sensor data and x-axis represents time.

Figure 4.2: Normal scenario run no. 3 sensor data

As can be seen in the Table 4.4, different attacks produce a variety of unique impact features. It is beneficial to monitor these and an example would be that during the sensor packet injection attack the frequency of spikes produced are observed as seen in Figure 4.3. Further analysis of the packet injection attack impact on the signature characteristics

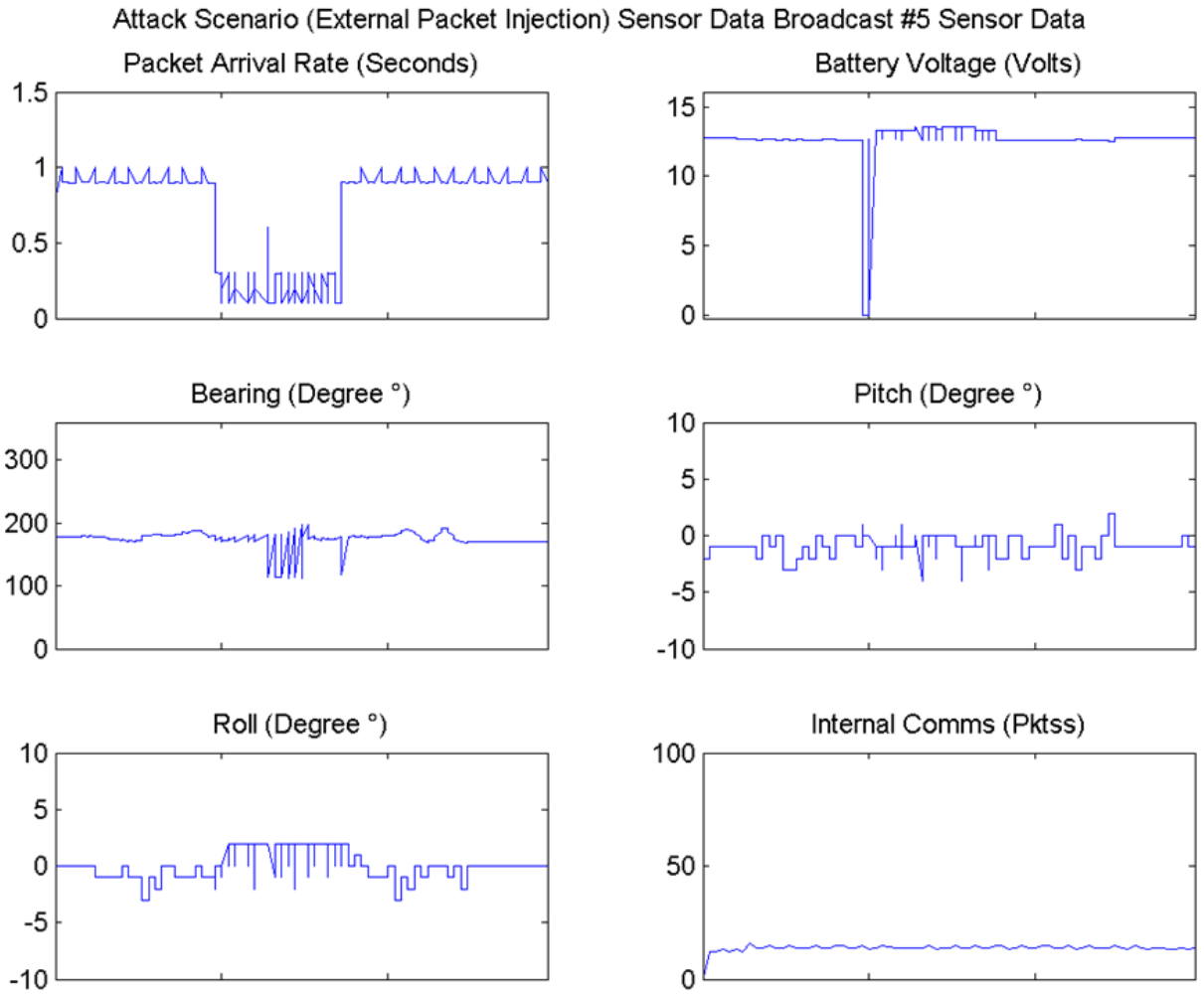

Figure 4.3: Sensor packet injection attack data behaviour

can be found in Table 4.5

In the case of a physical attack we have noticed that the attack may be detected by observing the behaviour of the change of the bearing in a sensory channel attack using a magnet and this can be observed in the Figure 4.4. It is noticeable that during the attack

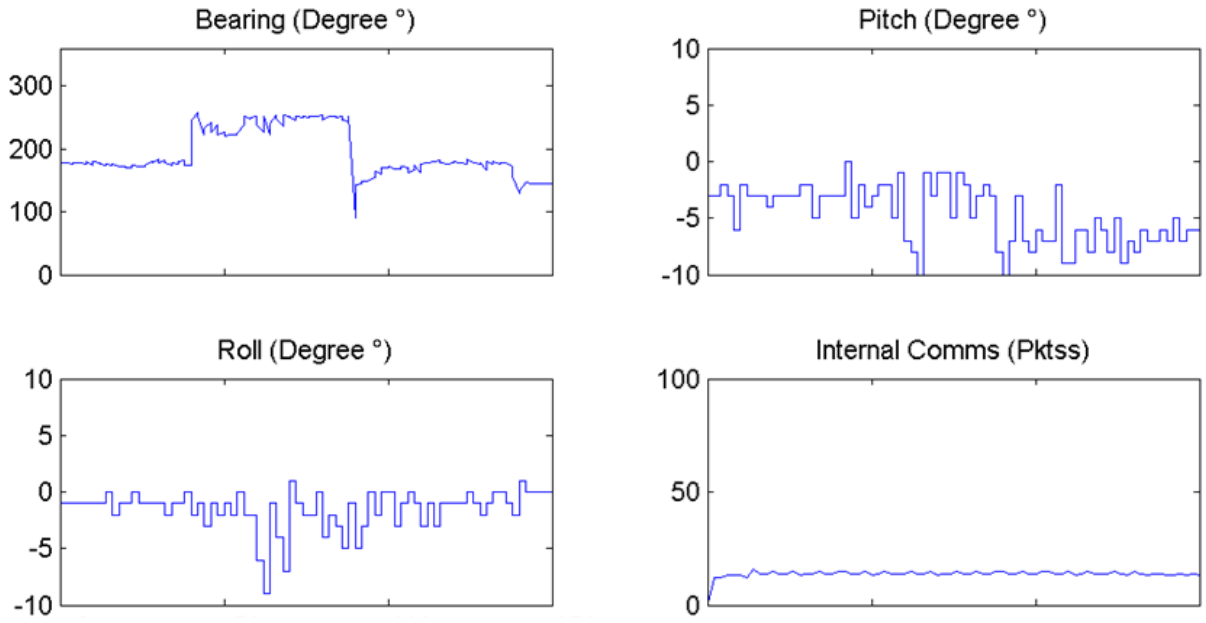

Figure 4.4: Sensory channel using magnet attack data behaviour

the bearing changed drastically followed by some slight changes and it then stabilised having nearly coinstantaneous data behaviour. Further analysis in terms of signature characteristics can be seen in Table 4.6.

Another interesting fact was noticeable by looking at the data from the Rogue node attack which is noted in Table 4.4 as it may have significant changes in the data similar to the sensor packet injection such as frequency of spikes and increased network packet rate (this can be observed in Figure 4.5. However it showed that by repeating packets on

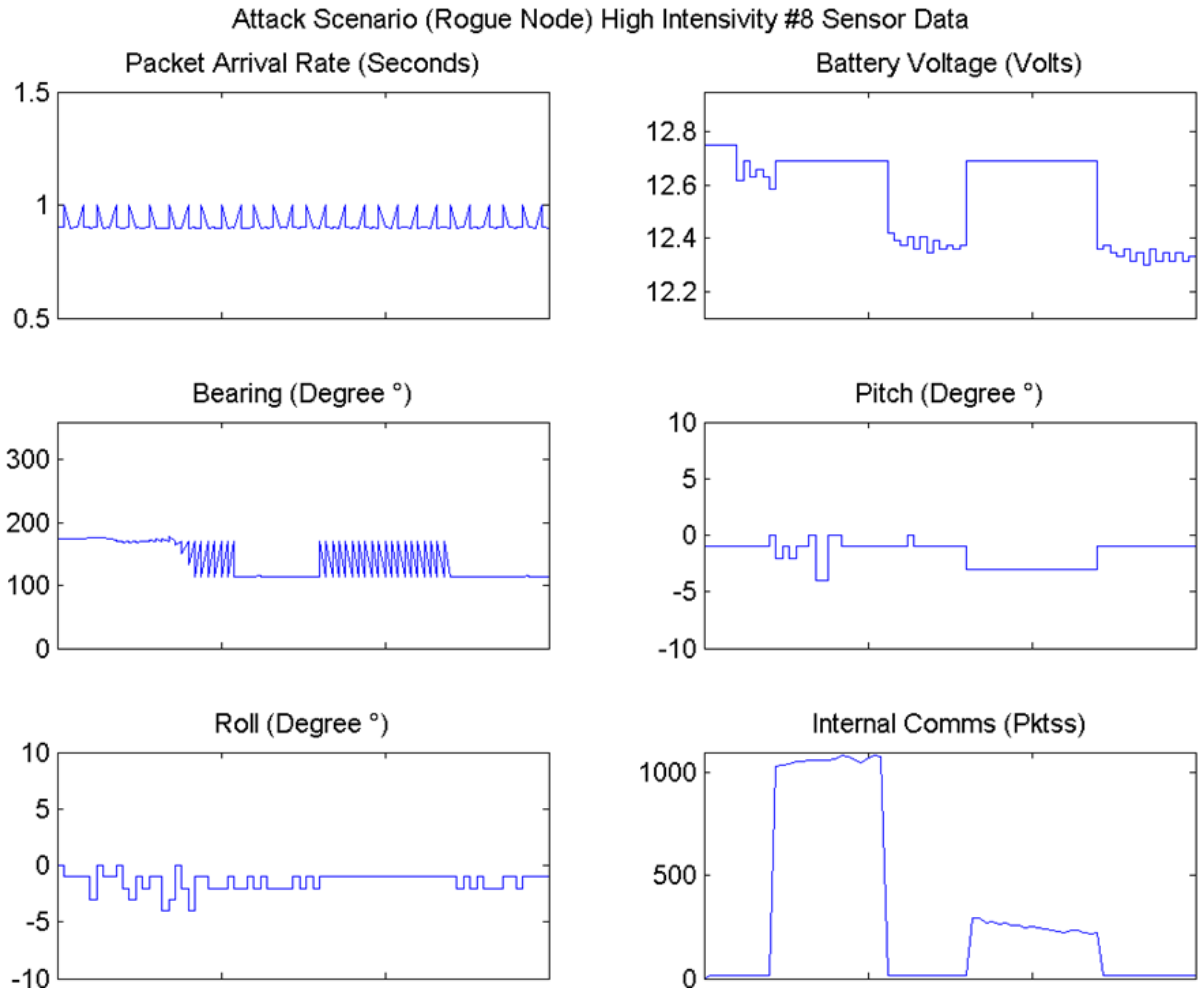

Figure 4.5: Rogue node attack data behaviour

the internal network the attack increases the spikiness as well as coinstantaneous data behaviour on certain data sources such as Battery Voltage, Pitch or Roll. This happened because these data sources have low sampling rates, and because of the high intensity oversampling of the mechanism that transmits the data to the workstation. The detailed impact on the signature characteritics can be found in Table 4.7.

Additionally we identified that the command injection attack had constant failure rates, therefore it was rejected as it has not allowed us to perform multiple repetitive scenarios to

collect data due to a High (Failure) mission impact. We have evaluated the performance of the proposed system using a prototype implemented on our autonomous vehicle, against a number of different incidents, which the robot had not previously been subjected to.

4.3.1 Cyber-physical Attack C1: Replay Packet Injection

In this scenario, data previously collected are replayed with different intensities (1, 2, or 3 packets/s).

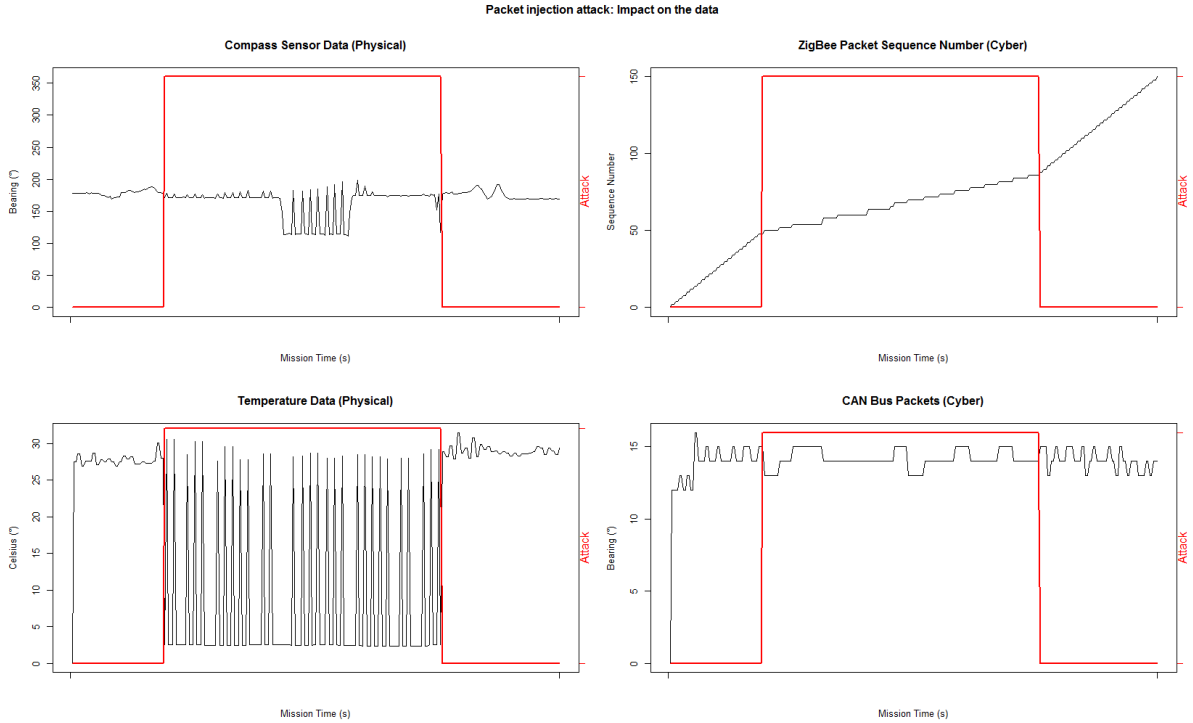

Figure 4.6: Impact on the sensors during Replay Packet Injection attack

The attack lasts for 40 s and starts 40 s after the start of the robotic vehicle's mission. Its impact on the senor data bearing is shown in Figure 4.6, where **x-axis** represents mission time and **y-axis** represents the data measurement. An interesting input feature was identified to detect replay packet injection as in the top-right graph represents the sequence numbers, during normal operation sequence numbers are incremented by 1, however in this case they are repeated multiple times as the mission time goes forward.

4.3.2 Cyber-physical Attack C2: Rogue Node

In this experiment, a rogue node is integrated into the system and begins replaying packets within the internal communication network. The attack has variable intensities and uses amplification of the packet rate, where it replays each packet that is captured multiple times. For attack scenario data set each attack is repeated twice using a number of different amplifications. The experiment (Figure 4.7) includes two iterations of 25 s of normal operation and 25 s of attack with a low intensity of the replay packet amplification.

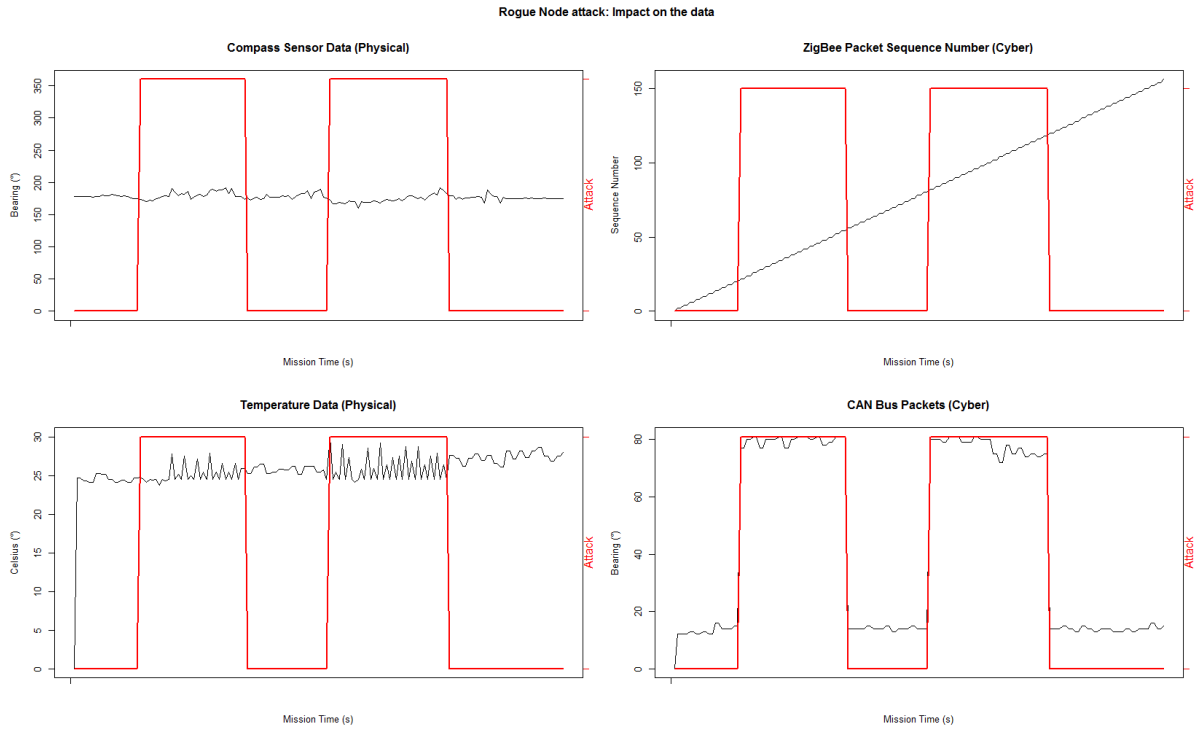

Figure 4.7: Impact on the sensors during Rogue Node attack

This particular attack achieves an impact similar in nature to that of a denial of service attack on a conventional computer system [129], causing a rapid increase in the network utilisation.

4.3.3 Physical-cyber Attack P1: Compass manipulation

In this experiment, a magnet is placed close to the compass sensor while it is in operation. After 40 s of normal operation, the magnet is applied. It is removed after a further 40 s.

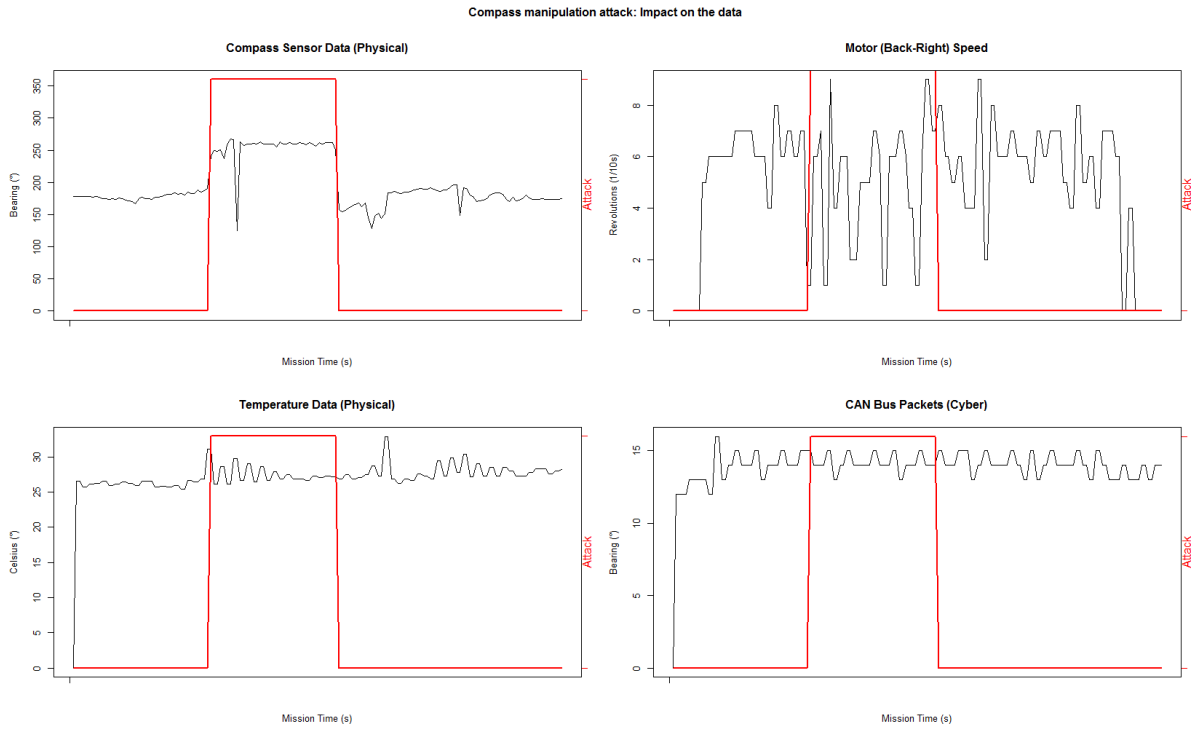

Figure 4.8: Impact on the sensors during sensory-channel attack

The impact on the bearing reported can be seen in Figure 4.8. Such a change dramatically affects the vehicle's orientation ability. For the other scenarios different distance was applied between the compass sensor and the magnet. Also in the Figure we can notice normal variation of the internal communication traffic (bottom-right).

4.3.4 Normal Failure F1: Broken wheel

This experiment occurred unexpectedly, as based on the learnt knowledge there was an unexpected high number of false-positive alarms during an experiment. Further investigation showed that a mechanical fault had occurred, as the mecanum wheel holder screws had worked loose due to the vibrations caused by the previous experiments. Figure 4.9

demonstrates the increased deviation in the bearing during the experiment, as well as the abrupt changes in the motor feedback. This occurrence was noticed when scenarios were evaluated based on the learnt knowledge of previous learning scenarios and the fault was identified during the observation of the robotic testbed vehicle. These runs have been

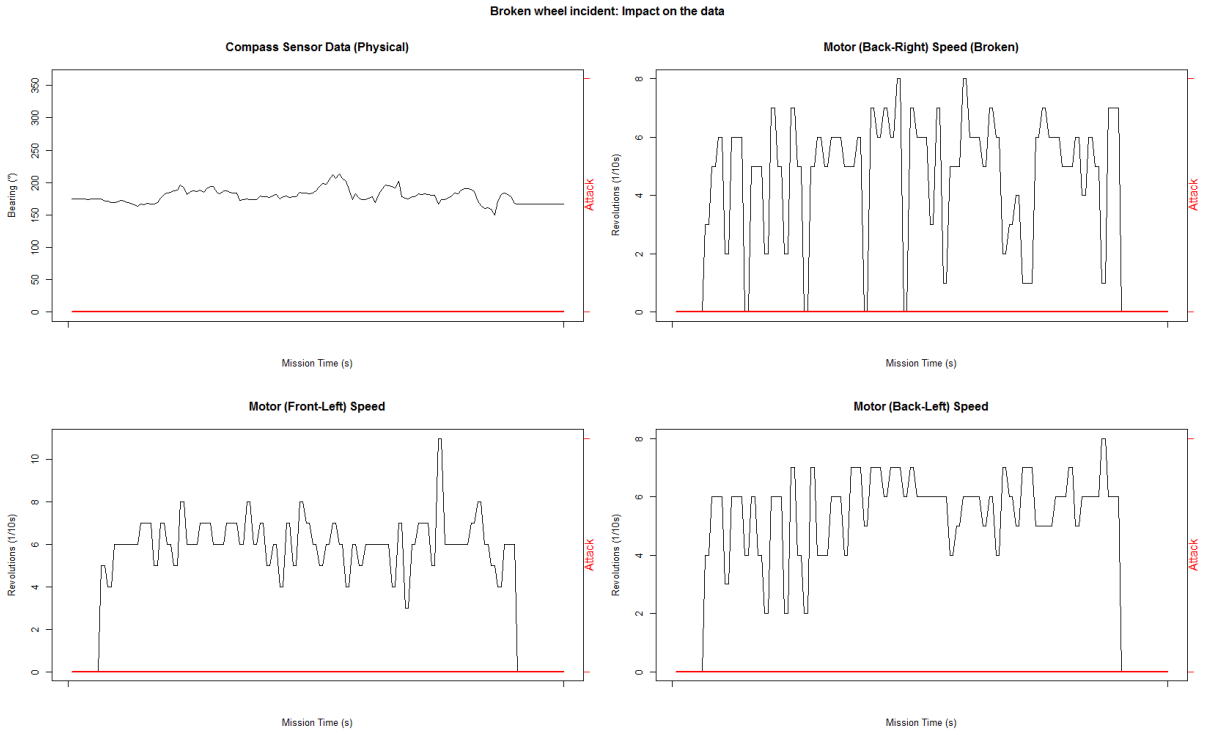

Figure 4.9: Impact during broken wheel incident

included in our analysis to explain the methodology and also to demonstrate the ability of this model not only to identify an attack, but also to identify mechanical faults as well. The fact that our signature method revealed this unexpected issue, validates the sensor-agnostic approach.

4.4 Forming signature pattern

In this section we will discuss the methodology for normal behaviour identification and further analysis of the anomaly detection leading to a methodology to achieve further improvements. One of the key hypotheses tested by the approach followed in this project is that cyber-physical security for an autonomous vehicle can be achieved by simultaneously

monitoring digital and physical information regarding its state. Towards this goal, this report details a set of experiments performed with an autonomous vehicle undertaking a relatively simple task of navigation in a corridor under normal operation, as well as in the presence of a variety of cyber-physical attacks against it. These include internal attacks involving a rogue node implanted through the supply chain, a sensory channel physical-cyber attack, as well as external cyber-physical attacks based on command injection, both sensor data and communication based. In our experiments, the robotic vehicle measures 17 different features in real-time. The experimental results indicate several interesting changes to the real-time values of these features, which in some cases clearly indicate a specific type of attack.

The methodology used to identify normal behaviour is focused on measuring the variability of the data sources to enable us to form a reasoned quantifiable value that will represent normal behaviour. Such an approach allows a robotic system to evaluate its past and present behaviour to determine if the system is under attack by measuring anomalies that are coming from the various data sources. It will be demonstrated how an experimental behaviour model methodology is validated through a set of experimental results that were gathered in a controlled environment. The identified normal behaviour definition will be used as a reference to demonstrate how a behaviour pattern model can be used in cooperation with anomaly detection techniques to improve cyber attack identification. Anomaly detectors are attached to data sources that are available on the system to increase the accuracy of anomaly detection within the system. The discussion here will cover the early approach towards integration of a defence mechanism on the existing system and on its operation. In the appendices a reader will find the results from data sources that were used in development of a pattern model and validate the approach towards the security of autonomous vehicles using behavioural and anomaly detection approach.

4.4.1 Attack Detection Methodology

In this section we will evaluate our behaviour model with the set of data that has been gathered through our experiments and we demonstrate that the behaviour model is capable of identifying abnormal behaviour.

Attack detection in autonomous systems is a challenging task, as the system is exposed to a real-world environment where a lot of unknown factors can exist that cannot be predicted during a mission. So a robotic vehicle has to learn its operational environment to be able to refer the operational behaviour to normal behaviour. By detecting anomalies it will decide whether it is an attack or the vehicle's behaviour is within normal condition bounds. For autonomous vehicles it is a challenging task to identify an attack, as a vehicle heavily relies on its sensing capabilities, monitored data and possible threats from a real-world environment.

Our aim is to use an experimental approach and develop a mechanism that is able to identify an attack with a high precision rate using anomaly detection techniques and methodologies. We assume that the system is fault-tolerant, and the only threats are raised from the cyber-physical domain. Our findings have demonstrated that it is possible to identify abnormal behaviour by monitoring deviations in real time.

The attack detection approach consists of several stages. The first stage requires the robotic testbed vehicle to learn its surroundings to identify normal deviations that can be caused by a real-world environment, gained knowledge is used as a definition of the robotic vehicle normal behaviour. The next stage required additional validation testing to allow the system to evaluate itself and identify any anomalies in the signature characteristics that may arise during normal operation. The number of these anomalies (throughout all signature characteristics) for a single data source are assigned as an individual "anomaly allowance" for a specific data source. To improve intrusion detection capabilities of the IDS, a weight scheme is generated as described in Section 4.6.2. The weights are generated based on the frequency that anomalies occurr in a certain signature characteristic. For

example, if the anomaly occurrence is frequent for a specific characteristic, its weight is reduced. This reduces the impact on the further behaviour formation. The next stage requires the system to monitor data in real-time and perform statistical analysis to identify anomalies that can be caused by the environment. If the system recognises a higher number of anomalies than were identified during the learning stage, the autonomous vehicle system evaluates monitored data from other data sources. Multiple data sources are evaluated to identify whether the vehicle operates within normal levels. If multiple data sources detect greater number of anomalies, such behaviour is classified as abnormal.

4.4.2 Identification of Normal Behaviour

Normal behaviour can be defined in various ways and can have a variety of attributes included which identifies such behaviour. However with every combination of attributes the definition of normal behaviour will require additional resources.

To define this normal behaviour we have used an approach that would allow us to quantify the robotic testbed operation. The focus of our experiments was to have data from multiple runs of the same scenario, in the controlled environment, where each run can be unique as the robotic testbed vehicle often changes direction which it then needs to correct. To do this we have developed an experimental behaviour model based on patterns that measure deviations (as shown in Figure 4.10) and produce a score for each data source. This approach will be discussed in more detail in Section 4.4.4. Normal behaviour was identified by executing the mission steps several times to gather data that would represent our defined normal behaviour. The aim of these runs was to identify noise levels that are then used for normal behaviour forming. In real-world environments, autonomous vehicles can encounter sources of noise that might disrupt the their sensing capabilities, such as motors disrupting the electromagnetic field that influences the compass readings.

4.4.3 Behaviour Initial Analysis

At the stage when data is received from data sources, the robotic testbed vehicle has to learn the environment through the noise levels from the data sources. Data sources have their own levels of unique operational noise which can be used to identify anomalies, a system then evaluates trends and normal variability in the data. Figure 4.10 shows how the data from the data sources is processed to identify normality. The Robotic vehicle has to reason based on the raw data from the sensors and collect valuable information that it can use for normal behaviour definition. For the data analysis, we use the following features:

- Short-term trend analysis
- Long-term trend analysis
- Exponential Smoothing
- Variability
- Deviations
- Spike energy (Area that exceeds defined normality range)

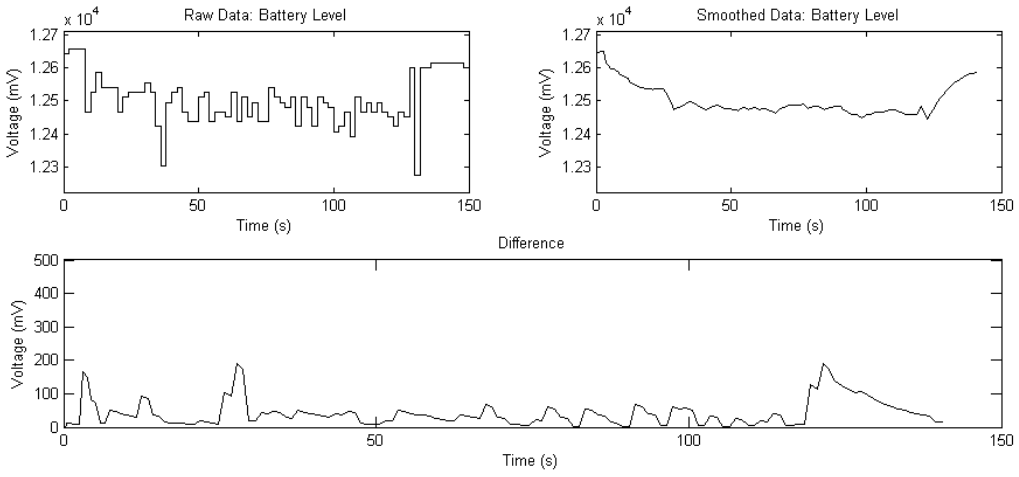

Figure 4.10: Exponential Smoothing and Variability

Every data source is analysed as a separate entity. As mentioned earlier, the data sources can have their own unique features which in some cases are better indications of an attack than others. Here we measure the quality of indicators of various data sources as an indication of an attack. For example, for temperature it would be common to have an increasing trend but high variability would be unusual. We aim to classify these unique features of data sources to form behaviour, that would be analysed in real-time. Heat sink temperature that we use on our robotic testbed can provide information on the system's functionality i.e. on our testbed system, the external heat sink has two voltage regulators attached that power up the wireless camera and a robotic arm. If these functionalities are used, the heat sink temperature will increase by dissipating resistance heat because of the load applied on the voltage regulators. So, in the future, relations can be created between data sources to identify anomalies in the system.

By analysing the data, we have learnt that the robotic testbed vehicle when exposed to a real-world environment running multiple experiments, each experiment may produce abnormal data at a specific moment in time. This leads to the introduction of a concept where a certain amount of anomalies will be accepted and these anomalies will be used as an indication of an anomalous behaviour. For example a robotic vehicle knows that during a mission it may see two abrupt changes within a specified time frame but if the robotic vehicle encountered more anomalies it will trigger an alarm. One option that we have used for the analysis was to monitor data deviations using multiple thresholds i.e. as a baseline we calculate the mean value of the sensor data readings, thereafter we calculate the standard deviation. If the data deviation exceeds the region between 0.5 of standard deviation and 1 standard deviation this will be identified as a spike in this region. If the sensor reading deviates more than 1 standard deviation it will be counted as a spike in another region, these regions are not limited to a specific number and can be identified autonomously from the data. This allows us to measure the anomalies within the specific region between the lowest and highest thresholds. Additionally the contextual information can be extracted if an anomaly exceeds a higher threshold, as we can measure spike severity level i.e. a glitch in the data might appear which will exceed the region

and will only appear in 1 data sample, this is counted as low severity level. However if the data readings exceed the region for the following 10 data samples, then the severity level will increase. This would allow the system to evaluate the spike and decide whether it poses a threat to the system. Normal bearing variation from an evaluation scenario is being shown in Figure 4.11, the spike severity level can be seen from a sensory channel attack scenario bearing deviations shown in Figure 4.12.

These thresholds can be used as a warning of a threat and for threat classification. It would be beneficial to identify regions automatically and these would be able to be tuned whilst the robotic vehicle operates. A reinforced learning approach is more feasible for such a task as it would be possible to access the knowledge base that will be developed through a learning stage, and then through the robotic vehicle operation the data sources would evolve. We have mentioned earlier that during normal operation it would be arduous

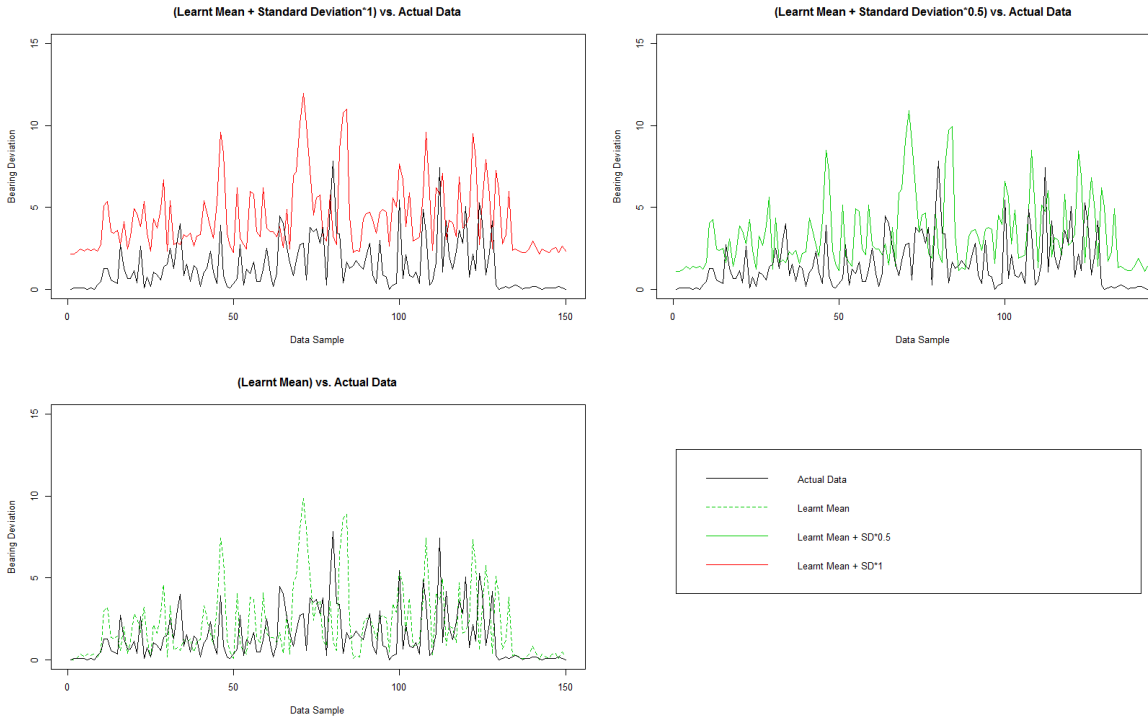

Figure 4.11: Normal Behaviour (Bearing Variation)

to avoid glitches in the data that can be caused by a real-world environment. We have mentioned that measuring the area of the spike would allow the model to identify the severity of the deviation. By tuning the data source knowledge base, it is possible to

take such glitches into account and compensate their severity as they are expected to happen, thus increasing the precision of the identification of the anomalies when in the presence of a threat. In Figure 4.12, one can observe that the energy of the spike (area of the spike over the highest threshold) exceeds the spike energy that is found during the normal evaluation scenario experiment. This demonstrates that the described approach has the potential to be able to detect an anomaly. Such an approach can be used to

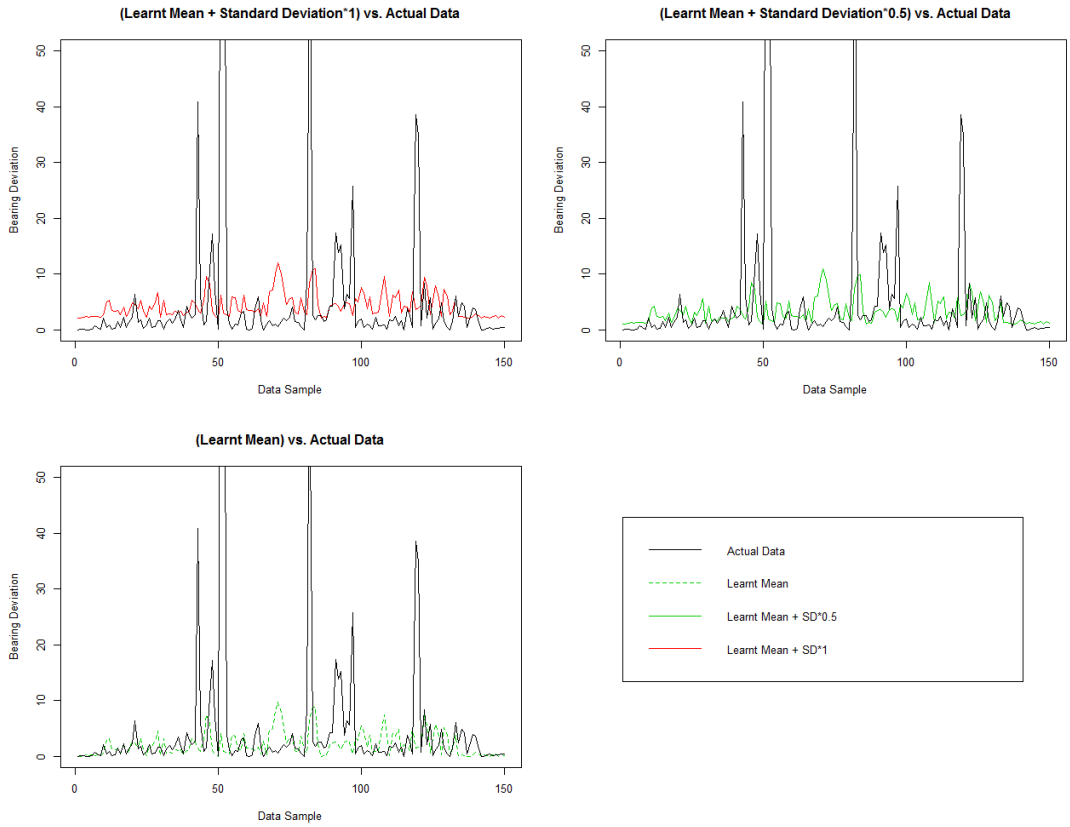

Figure 4.12: Sensory Channel Attack Behaviour (Bearing Variation)

perform the initial analysis allowing us to extract reasoned meta-data that can be used for instantaneous anomaly detection. All these features that we collect are described in this section and will be used to identify normal behaviour patterns. The raw data from the data sources can be found in the appendices to demonstrate to the reader the kind of observable impact different attacks may have.

4.4.4 Forming Behaviour Pattern

In the previous section, we have described the features that will be used for a pattern definition. The experimental pattern model is capable of quantifying our robotic testbed activity by measuring the features described earlier and producing an activity score. This score is represented by the deviation coefficients of each reporting node of the system, this enables a sensor agnostic approach through generalisation of all reporting nodes such as sensors, sub-systems or received commands. In the initial study all data source anomalies were treated equally, further when the weighting mechanism was introduced, that has emphasised anomalies that never been seen, and reduced the value of anomalies that tend to occur frequently. This has been done as a proof of concept for this approach. The key feature of the pattern is that it allows performing a calculation on the system, which has processing power, to determine that the system behaves within the defined normality. The data source behaviour is represented by the signature which consists of multiple signature characteristics and is shown in the Table 4.8.

Raw data thresholds are gathered as they demonstrate the minimum and the maximum values of the data source while the vehicle is undertaking an activity. The combination of features can produce other features that will enable the identifying the differences. The difference between raw data and the exponential mean can determine the changes that can be expected while the robotic vehicle is in operation. We are also interested in measuring the standard deviation in real-time that will allow the system to develop an expectation threshold when the defence mechanism is integrated onto the system. The standard deviation is a key feature that would have a higher weight, as our pattern model demonstrated that it influences the overall score significantly and is one of the main features used for anomaly identification. Earlier we have mentioned that we cannot avoid glitches in the data on an autonomous vehicle as we can't predict all characteristics of their operating environment at the design stage. If we increase thresholds we might start losing key indications of an attack which could lead to a cyber attack going unnoticed. The pattern model will exploit them as noise in the data source behaviour and use the

spike indicators for a behaviour definition and in evaluating spike severity.

These indicators will be merged together with the same scenario experiments to help us identify the key indicators of specific attacks that can then be used in this project. We will review physical and cyber attacks to make our model more precise. The behaviour pattern model demonstrated its ability to identify cyber and physical attacks and the approach that we are using by attaching anomaly detectors to each data source enables identification of the malfunctioning module, which raises a threat to the mission or functionality. To demonstrate that this model works, we have run multiple activities that allowed us to gather enough data for analysis. Figure 4.13 shows how normal behaviour was represented initially using our pattern model, where the bottom part with numbers represents the mean value for each signature characteristic, on the right the mean of all signature characteristics is shown.

We have extracted signature characteristics from all sensor readings that our system had. Initially we have used an average value of all available sensors to see if it is possible to quantify our overall scenarios. The mean values that were produced were used as an initial score, the figure demonstrates every run used in the experiment, average values of each characteristic and the sum of the averages that represents an experiment score. These values were not normalised at the time of the experiment. In our initial study to define normal behaviour we have taken five test scenarios under normal conditions to evaluate our methodology. We have applied the pattern model for the data we have gathered and produced a score to define normal behaviour.

Run	Raw Min	Raw Max	Exponential Min	Exponential Max	Exponential Diff Min	Exponential Diff Max	Standard Deviation	Variance	Spike > 50%	Spike > 100%	Spike > 150%	Spike >200%	Mean
1	Normal Behaviour Learning Set												355.3237
2													396.4444
3													329.5507
4													341.2976
5													361.3073
	655.6731	919.3697	834.1205	901.7642	0.0000	47.8274	21.1888	1007.1354	73.4706	11.0118	1.5412	1.2706	372.8575
1	Normal Behaviour Testing Set												370.6181
2													343.4539
	630.7290	702.7870	636.6596	690.6204	0.0000	29.8883	17.4959	826.7181	58.1462	7.0546	0.7206	0.2647	357.2669

Figure 4.13: [**Initial Study**] Evaluation of Pattern Model

When we got the score for our normal behaviour learning set, we have run additional test

scenarios to evaluate our model. The expectation was that the score will be similar to other normal behaviour scenarios; this is shown in Figure 4.13 and it has validated our normal behaviour approach. This table shows the specifics of a robotic vehicle run and its activity status in a numerical representation including the signature characteristics' averages used for informative purposes. From a learning set we can learn the thresholds that we can expect if we enable anomaly detection in the system. As mentioned previously reinforcement learning would allow the dynamically adjustment of data sources through a life cycle of a robotic vehicle. Normal behaviour validation allowed us to proceed to the next step and set up a variety of thresholds that would allow us to identify malicious data sources due to a physical or a cyber attack. Our testing set allowed us to learn the number of normal anomalies that are considered as normal behaviour within the system.

Run	Raw Min	Raw Max	Exponential Min	Exponential Max	Exponential Diff Min	Exponential Diff Max	Standard Deviation	Variance	Spike > 50%	Spike > 100%	Spike > 150%	Spike >200%	Summary
Packet Arrival Rate													
1	0	0	0	0	0	0	0	0	0	0	0	0	0.0000
2	0	0	0	0	0	0	0	0	0	0	0	0	0.0000
Battery													
1	1	1	1	1	0	0	0	0	0	0	0	0	4.0000
2	0	1	1	1	0	0	0	0	0	0	0	0	3.0000
Bearing													
1	0	0	0	0	0	0	0	0	0	0	0	0	0.0000
2	0	0	0	1	0	0	0	0	0	0	0	0	1.0000

Figure 4.14: [**Initial Study**] Data Anomaly Analysis (Normal Data Set)

The anomaly detection itself uses the actual data and referring to a learnt signature of data deviation averages. For identification we use the average deviation and the exponential mean to create a limitation or a spike region. Then we compare the actual data source to identified deviations and if the actual data exceeds specified limit, they are marked as anomalies.

We have used our testing set as a learning set, to identify the number of anomalies that the system will consider a normal condition. As we can see from the Figure 4.14 the battery signature shows four anomalies in one run, which affects the minimums and the maximums of the signature characteristics and is due to battery discharge i.e. the battery voltage is lower than it was during learning data set. The quantity of these anomalies are treated as an anomaly allowance for this individual data source during the robotic system

normal operation.

The number of the anomaly allowance is identified automatically from the data and is assigned to an individual sensor or the data source. In this current case, the battery voltage feature will have an allowance of four anomalies within the signature space, if the number of anomalies exceed this limit (four anomalies), the data source will be classified as having abnormal or malicious behaviour. The anomaly allowance is identified during the testing stage and is assigned to data sources individually, because different data sources may have different behaviour. As this behaviour is identified from the data directly, the anomaly allowance for each data source may vary based on the robotic system operational environment or sensor type. The same approach is visualised for a different sensor and is demonstrated in the Figure 4.16.

The system can validate other sensors and if the defence mechanism identifies more than one malicious data source it can be treated as an attack. Such an approach allows to identify a malicious data source. When a malicious data source was identified it is possible to validate the overall system stability and act appropriately by applying resilience techniques or ignoring the data source as it has no impact on the mission success. However, this attack mitigation technique has to be aware that if there is a malicious node in the system that will falsely inject packets pretending to be another node, this may lead to a problem where an attacker can force the system to ignore a correctly operating node. Such a sophisticated attack is out of scope of this project, however it is necessary to take into account such possibilities. In that case additional hardware or software solutions will have to be installed to resolve such issues. Such as, adding additional diagnostic solutions to evaluate the data source health or hardware solutions that will manage the power supply of data source. When the initial normal behaviours were identified in our initial study, based on the score which is represented by a sum of all deviations in the system, we performed pattern model validation using the attack scenarios data set. The resulting scores can be seen in Figure 4.15. It is demonstrated that the pattern model allowed us to evaluate the attack scenarios and with a high precision rate we can identify a cyber attack using our pattern model approach. During the normal behaviour evaluation,

Run	For the reference: Sum of All Deviations (average) in Normal Scenario cases were 384 (Learning Set) and 355 (Testing Set)		Sum of All Deviations
1	Sensor Injection Attack Behaviour		27728
2			52346
3			27713
4			28312
5			27838
		Average Score	35520
1	Rogue Node Attack Behaviour		357
2			394
3			388
4			1673
5			851
		Average Score	794
1	Sensory Channel Attack		1059
2			1110
3			1267
4			1005
		Average Score	1203

Figure 4.15: [**Initial Study**]Evaluation of a Pattern Model (Attack Data Set)

we have seen the score of all system deviations which potentially allows to use it as a reference, additional parameters can be added to this model to tune the sensitivity which will introduce the range of normality.

Figure 4.15 demonstrates the score of all deviations in the system during the attack experiments. The scores are much higher than the learnt normal behaviour and the normality validation experiments. However it was not able to classify abnormal behaviour in several cases where a rogue node attack was used. This was due to the low impact on other systems. However it was noticeable that the proof of concept has potential and was able to classify the experiment runs. The concept is not perfect and there are cases where it will fail to classify an attack experiment. For example, the compass sensor reports no deviations from the learnt normality but there is a temperature increase of 10 degrees (acting abnormally). This will raise the score insignificantly allowing abnormal behaviour of the system to go unnoticed. For this purpose deviation coefficients were used to normalise the data deviations. The rest of attack scenarios have a significant increase in the score in respect to the learnt normality score. Additionally from normal experiments we have identified that at any point in time during the experiments it is possible to notice multiple data source false-positive classifications and as was mentioned earlier that this methodology takes into account anomalies that act as monitoring features for detection of an actual threat. In combination these two approaches, which are to monitor deviations in the data of a data source and to monitor the number of anomalous data sources will allow it to evaluate the system behaviour and a potential threat to the system. From our experiments a variety of attacks showed an increase in multiple features that we were

collecting, such as variance in sensor data injection attack or the number of spikes in all attack scenarios. By using both methods which are anomaly detection, based on the deviations and pattern model, we can tolerate glitch reactions that are caused by the environment or are randomly generated by the data sources.

4.4.5 Experiment Impact Analysis

In this section, we discuss attack impacts, impact factors and how various data sources can be used in combination to identify an attack. We had discussed earlier how we evaluated the normal behaviour of our testbed vehicle using the collected test scenario data set to identify the normal anomaly levels to identify an attack.

4.4.5.1 False Data Injection Attack

The first attack is a Sensor Data Injection Attack using the external network. In this scenario an attacker was able to capture packets when the testbed vehicle was going through a testing stage, while under development. The data that an attacker was able to capture was:

- Battery Level (Units: mV Range: 0-15000)
- Bearing (Units: Degrees Range: 0-3600)
- Pitch (Units: Degrees Range: -90 - +90)
- Roll (Units: Degrees Range: -90 - +90)
- Ultrasonic Range Meters (Units: cm Range: -1(NaN) - 250)
- Temperature Units: (Degrees Celsius)

While a robotic vehicle is carrying out a mission step, the attacker uses a packet injection replay attack on the communication channel between the robotic vehicle testbed and the

operator. Using our pattern model approach for the detection of anomalies, we are able to identify data sources that are affected, as seen in Figures 4.5, 4.6 and 4.7. Table 4.5

	Packet Arrival Rate	Battery	Bearing	Pitch	Roll	Ultrasonic Ranger Meters	Temperature	Sequence Numbers	Internal Comms. Rate	Motors
Raw Data Minimum		X	X				X			
Raw Data Maximum		X	X							
Exponential Data Minimum		X	X	X			X			X
Exponential Data Maximum		X	X						X	
Exponential Deviation Minimum										
Exponential Deviation Maximum	X	X	X		X	X	X	X		
Spike Region (0.5*Std >x <1.0*Std)	X	X	X	X		X	X			
Spike Region (1.0*Std >x <1.5*Std)			X	X		X	X			
Spike Region (1.5*Std >x <2.0*Std)			X	X		X	X	X		X
Spike Region (x >2.0*Std)				X		X	X	X		X

Table 4.5: Sensor Based External Injection Attack Impact

shows that while the robotic vehicle is under a false data injection attack using sensor data, additional data sources are affected as well. It is observable that the deviations in some data sources exceed two standard deviations from the learnt data. There is no major observable impact on the system itself. Even though the mission impact is unnoticeable, it is clearly shown that the system is capable of detecting anomalies. Such a detection system can be distributed over the robotic vehicle and an operator's machine to increase its ability to detect an attack and react accordingly to a raised mission threat. The key indications of such an attack would be an increased variance in the data and the spikiness levels in a manner that is distinctly noticeable in Figure 4.6.

4.4.5.2 Sensory Channel Attack

During the sensory channel attack, the attacker generates invalid data by influencing the magnetic field that is used by the compass module. This is done using a magnet that is

strong enough to disrupt the magnetic field.

	Packet Arrival Rate	Battery	Bearing	Pitch	Roll	Ultrasonic Ranger Meters	Temperature	Sequence Numbers	Internal Comms. Rate	Motors
Raw Data Minimum			X	X						
Raw Data Maximum			X	X		X				X
Exponential Minimum										X
Exponential Maximum			X			X				X
Exponential Deviation Minimum			X							
Exponential Deviation Maximum			X							X
Standard Deviation (Std)		X	X	X		X	X			X
Spike Region (0.5*Std >x <1.0*Std)			X	X		X				X
Spike Region (1.0*Std >x <1.5*Std)				X						
Spike Region (1.5*Std >x <2.0*Std)		X		X						
Spike Region (x >2.0*Std)										

Table 4.6: Sensory Channel Attack Impact

When the results of a sensory channel attack were evaluated we were able to determine the affect of an attack on the system. By injecting false data, multiple data sources where affected allowing us to determine if the vehicle was under attack. All impact indications for the sensory channel attack experiments can be seen in Table 4.6. A major impact can be seen on the actuating capabilities of the robotic vehicle due to its autonomy i.e. magnetic readings were falsely injected through the physical domain and the robotic vehicle went into recovery mode and was trying to correct itself to follow the specified path. However due to an attack the vehicle was unable to recover its position correctly. Sensing redundancy could be used for specific recovery states that are unknown to an attacker to improve a system's limitation and resilience to such attacks. One of the improvement that can be done towards the identification of a compass manipulation attack is thorough analysis of the deviations that show abrupt changes. An example of such deviations is shown in Figure 4.12 where the energy of a spike can be analysed to improve anomaly detection.

4.4.5.3 Rogue Node Replay Attack

To evaluate the rogue node replay attack we have used a number of different intensities. The impact of an attack can be seen in Table 4.7. Low intensity replay attacks might have an unnoticeable observable effect; however it can have a long-term impact on the system itself. Intensity specification can be found in Section 4.2.2. The Stuxnet worm which infected Iranian SCADA systems was slightly modifying rotational frequencies of centrifuges which led to the failure of the actuators.

(L) Low/Moderate Intensity (H) High Intensity incl. (L)	Packet Arrival Rate	Battery	Bearing	Pitch	Roll	Ultrasonic Ranger Meters	Temperature	Sequence Numbers	Internal Comms. Rate	Motors
Raw Data Minimum				H					H	
Raw Data Maximum									L	L
Exponential Minimum				H			L		H	L
Exponential Maximum									L	L
Exponential Deviation Minimum										H
Exponential Deviation Maximum						L			L	L
Standard Deviation (Std)				H			L		L	L
Spike Region (0.5*Std >x <1.0*Std)				H		L	L		L	H
Spike Region (1.0*Std >x <1.5*Std)				L		L	L			H
Spike Region (1.5*Std >x <2.0*Std)				L		H				H
Spike Region (x >2.0*Std)				L		H				H

Table 4.7: Rogue node replay attack (mixed intensity) impact

A low intensity attack can have a long-term effect on the autonomy logic when various machine learning techniques are used. From our experiments we have also noticed that an attack has an effect on the actuators of the robotic testbed vehicle, which could lead to their failure as they are wearing out the actuators resource. In the case of a high intensity attack, the robotic vehicle testbed had failed. In this attack scenario we can notice fewer indications of an attack itself, to improve identification it would be possible to measure the quantity of allowed anomalies, and if they exceed the normal behaviour

could be classified as abnormal and treated appropriately.

4.4.6 Detection Methodology Analysis

In this document we have discussed our methodology towards the security of autonomous vehicles from cyber attacks. We have shown how we approached our experiments, which types of attack were used to disrupt the autonomous vehicle during its task and in some cases we have noticed that lower severity impacts sometimes have no influence on the execution of a mission. However using our approach towards the anomaly detection will warrant using it for early warning attack identification. Initially the idea was to test the methodology at an early stage and identify whether the methodology is able to distinguish normality using our hypothesis of monitoring deviations and changes in the data source signature characteristics. All the following changes affect the score representation and the applicability for integration with existing systems.A pattern model treats all indications that are included, using fixed weights attached. One of our goals was to make the pattern model more dynamic and adaptable in a way that it will be capable of learning the environment once. The key indications that we are focusing on, will be monitoring deviations and the frequency of deviations. In the next section we will discuss the early stage of the defence mechanism integration.

4.5 Heuristic Binary Classification (HBC)

Detection of cyber threats is an expanding area of study in the embedded systems domain. The need for cyber security has increased significantly and there are many researchers currently working towards cyber-physical security of such systems, such as the decision tree-based approach in [4] using decision trees for anomaly detection, and the behaviour rule specification in [46]. In this section, we evaluate our robotic testbed system behaviour by monitoring components with instrumentation installed on the system.

Several different attack vectors can apply to cyber-physical systems. We divide these into cyber-physical and physical-cyber. Cyber-physical attacks are attacks in cyberspace that adversely affect the physical space. For instance, an attacker can target the communication between the system and the operator to disrupt normal system operation. In an autonomous system, a system's own autonomy can be used against it to take control over the autonomous system. Conversely, physical-cyber attacks are the ones performed in physical space to adversely affect cyberspace [8]. A trivial example would be physical damage that would make the network unavailable. A non-trivial example would be an attack consisting of custom laser beams targeting an autonomous vehicle's LiDAR [32], or externally generated noise targeting the ultrasonic sensors so as to confuse the vehicle's spatial awareness. Such attacks that manipulate the input to sensor systems with the purpose to affect the operation of a system that depends on them are often referred to as sensory channel attacks.

Previously, there had been little or no consideration for cyber security during the design of safety-critical systems, but this is changing since the practical cyber-physical attacks against vehicles were showcased for the first time a few years ago [18][41]. Ten years ago, the threat level was significantly lower, but now with the availability of electronic devices such as Arduino kits and a variety of sensors that can be used for educational purposes, consumer products and industrial applications are wide spread. With increasing knowledge in this area the threat to such systems increases. An attacker may not necessarily have the intention of disrupting the system. Motives can vary and the outcomes can range from small value fluctuations to possible lethal injuries [37]. This shows that there is a need to secure cyber-physical systems.

4.5.1 Normal Behaviour Definition

Our behavioural model uses a sensor-independent approach, in the sense that the sensor-signal characterisation is performed without any additional contextual information to indicate the type of the sensor. Each different type of sensor has its unique output, but

we are not interested in determining the type of the sensor, but instead we are interested in learning the signal characteristics under normal operating conditions and thus being able to automatically determine when an anomalous condition occurs by monitoring data source signature.

A compass sensor provides a valuable example: due to the limited speed at which the vehicle can turn, there is a corresponding limit to the rate at which the compass bearing can be expected to change. The compass bearing will also contain a certain amount of noise as the vehicle travels over non-perfect surfaces and does not track in a perfect straight line (there is a detectable "wobble" of typically one to two degrees). These characteristics can be learnt by examining the signal over a series of test missions, without having to explicitly know that the sensor is a compass. For simplicity, we demonstrate the impact on the compass of a cyber-physical disruption using a magnet-based sensory channel attack. By placing a magnet in the vicinity of the sensor, we cause a variable disruption of the vehicle's navigational ability. By so doing, the data stream from the sensor is affected in two detectable ways. Firstly the sharp change in bearing when the magnet is applied (or removed), and secondly, in the reduced noise levels since the magnet causes the sensor to read near-static values (which are anomalous because they are suspiciously "clean"). The proposed approach enables attack detection without prior knowledge of the attack type. The compass example is a part of the experimental set used in our evaluation.

We represent the characteristics of sensor signals in a signature format that can be used to compare expected and actual behaviour in order to detect anomalous events. The signature comprises a number of features whose values are learnt during the mission experiments described earlier. The features describe characteristics such as the signal-to-noise levels, maximum and minimum sensor readings detected, size and frequency of spike values and rate of change of sensor values. The signature approach facilitates evaluation of the enviroconsistency [45] of a particular trace. For example, the system may learn that a particular data source generates data values distributed in the range 100 to 400 with a mean of 200 during normal operation. The new trace can be compared against the expected behaviour based on these specific characteristics. There is no need to compare

the raw data directly. The model will determine whether a particular trace represents normal or abnormal behaviour based on the distance between the trace characteristics and the corresponding values in the signature.

Table 4.8: Signature Characteristics

Value Type	**Characteristic**	**Abbreviation**
Raw	Minimum	SC1
	Maximum	SC2
Exponential Smoothing	Minimum	SC3
	Maximum	SC4
	Lowest Difference	SC5
	Highest Difference	SC6
Deviation	Standard Deviation (Std)	SC7
Spike Regions	0.5*Std - 1.0*Std	SC8
	1.0*Std - 1.5*Std	SC9
	1.5*Std - 2.0*Std	SC10
	Over 2.0*Std	SC11

Our signature format contains various characteristics, as shown in Table 4.8. Values are exponentially smoothed to provide a basis for comparing instantaneous values with the recent trend, thus detecting noise levels and abrupt changes in values which are short lived are categorised as spikes. A similar concept has been used in a dynamic system in [130].

4.5.2 Identification of anomalous signals and behaviour

The signatures are constructed during the learning stage to define normal behaviour on a per-sensor signal stream basis. To capture the range of normal behaviour the experiments were repeated five times to identify the domain of values where the data sources operate and their normal deviations. Using such an approach it is possible to classify normality when the system operates within the normal experiment environment. Currently, we evaluate the results of test runs off-line after each run, however the learnt-signature based approach has the potential to be used in real-time, for self-protection of the autonomous vehicle. In this publication, we review multiple results from seven experiments and different scenarios which are: learning stage, evaluation stage, and physical-cyber compass

attack scenario.

We use exponential smoothing because it enables dynamic smoothing to be more reactive or passive by changing the α value. It is a simple and efficient means by which to follow an unfolding trend in sample values. The equation for exponential smoothing is $E_t = E_{t-1}(1-\alpha) + \alpha x_t$, where E is exponentially smoothed value, t represents the index in time, x represent value from the sensor and α is the weight assigned to the current value and $1-\alpha$ is the weight assigned to the previous exponentially smoothed value. The technique is efficient in regards to memory and processing and so is well suited for use in embedded systems as it does not require to store full data set of values and can be used to analyse rate of change of the data [130]. Each element of a signature comprises of characteristics that may indicate an anomaly. An operational signature is applied to a learnt "normal" signature, and this facilitates observation of a data source anomalous behaviour and reasoning about component behaviour at the higher layers of our software stack and evaluate the deviations from normality. By observing deviation coefficients (from the learnt normal characteristics), we form a dynamic behaviour score for the data source. The issue that we have encountered is that these deviations had to be normalised in some way, because of the nature of this project the aim is to focus on the routine-mission robotic systems which have routine tasks, the decision was made that during the "Learning" stage, the data from the sensors is normalised by using the mean such that deviation coefficients ($\mathbf{d}^l$) during "Learning" stage are provided by $d^l_{i,j} = \frac{x_{i,t} - \bar{x_i}}{\bar{x_i}}$, where i represents a data source and j represents a signature characteristic, $x_{i,t}$ is the sensor value for a particular i at time t, and $\bar{x_i}$ is the mean of the known set of values. This allows us to offload the "Learning" stage from the robot as it requires to recalculate the mean during each iteration, but such computation can be done on-board. In real-time, the classifier has knowledge of the learnt mean (the last calculated known mean) ${\bar{x_i}}^0$ and calculates the new (real-time) deviation coefficients ($\mathbf{d}^r$) by using ${\bar{x_i}}^0$ such that $d^r_{i,j} = \frac{x_{i,t} - {\bar{x_i}}^0}{{\bar{x_i}}^0}$.

The behavioural score (from the signature) can be used to evaluate the level of threat to the system i.e. the higher the deviation from the learnt normality, the higher the likelihood of an attack. In the Table 4.9 we can see an improved version of what was discussed in the

Section 4.4.2, in current case we have used deviation coefficients to normalise data, the table shows an example analysis of data from a compass bearing sensor. By comparing signature elements we identify those elements which indicate that an attack might be present. The deviation extents are weighted and combined to determine the likelihood of an actual attack, i.e. co-deviation on multiple elements reinforces the attack risk.

Table 4.9: Compass Bearing Behaviour Signature Data

		Deviation Coefficient		
Characteristic	Value	Learnt	Test	Attack
Minimum	167.8	0.0292	0.0148	0.4388(A)
Maximum	194.7	0.0151	0.0128	0.3180(A)
Exponential Minimum	170.9	0.0145	0.0035	0.0995(A)
Exponential Maximum	188.9	0.0083	0.0171(A)	0.3269(A)
Exponential Deviation Minimum	0.0	0.0000	0.0000	0.0000
Exponential Deviation Maximum	10.1	0.2362	0.1289	>1.0000(A)
Standard Deviation (Std)	5.3	0.1094	0.0582	>1.0000(A)
Spike Region (0.5*Std >x <1.0*Std)	55	0.2435	0.3043(A)	0.9348(A)
Spike Region (1.0*Std >x <1.5*Std)	25	0.5632	0.4079	0.8684(A)
Spike Region (1.5*Std >x <2.0*Std)	10	0.5576	0.5455	0.8485(A)
Spike Region (x <2.0*Std)	5	0.4462	0.2308	0.6154(A)
Threat	**Summary**	2.2231	1.7237	17.921

Table 4.9 shows the deviation comparison from a variety of data sets which are a **Learning scenario**, **Test scenario** and **Compass bearing attack scenario**. By learning we mean that the vehicle is operated in a series of known missions which exercise the sensor signals across their normal value ranges. We have mentioned earlier that the learning happens essentially in two stages which are: "learning" and "testing". Through learning we identify the data source behaviour characteristics such as minimum and the maximum range or how the data tends to change by monitoring spike regions. The **Threat Summary Score** is identified when the system has gone through the "testing" stage and identified number of anomalies and the normal deviation coefficient values, in case of the data source shown in Table 4.9 for presentation purposes the "testing" scenario was taken so that the difference in the "learnt" case and the "test" case was the highest. In its primitive case this defines the range of normality which in a current case would be the $Normality = Abs(2.2231 - 1.7237) = 0.4994$, therefore normality range becomes

$1.7237 \leq Normality \leq 2.7225$. This might increase the number of false-positives in more stochastic environments, however such approach can be tuned further through the scaling factor and be configured specifically for the system. The scope of this project is to develop a detection mechanism that will be applicable to routine oriented robotic tasks with certain amount of randomness in the environment. Following the earlier discussion concerning the compass sensor, the vehicle can be operated moving over various types of surfaces to determine the level of noise in the compass sensor signal and also can be made to turn at various angular rates-of-change in compass sensor values.

By test scenario, we mean that the vehicle is operated (post learning) in a variety of normal scenarios, with the objective of testing the vehicle's ability to detect the abnormalities solely on the basis of finding anomalous conditions where the sensor signals do not conform to expected learnt behaviour.

By attack scenario, we mean that the vehicle is operated using the same conditions as in the learning and test scenarios, but during the experiment we place the magnet near the compass sensor for forty seconds. This is to validate the behavioural profile approach and determine if the vehicle is able to identify anomalous conditions. To demonstrate this ability to identify anomalous behaviour the data set of one of the "Attack" scenarios is used, as can be seen in Figure 4.16.

To calculate the deviation coefficient $d^l_{i,j}$ for each signature characteristic i and data source j, a set of learning scenarios is used. Each scenario n provides a different set of values for the deviation coefficients, which we denote here as $d^{l,n}_{i,j}$. After we run all scenarios, we choose for each {i,j} pair the deviation coefficient value for the particular scenario n where it was the highest among all scenarios. So, ${d^l}_{i,j} = \max_n(d^{l,n}_{i,j})$. These maximum values are stored and used as reference for anomaly detection.

We detect an anomaly in terms of sensor signal values as a situation where the signal deviates significantly from expected (learnt) mean behaviour, as held in the particular sensor's signature. We investigate the automated detection of anomalies, based on our signature approach using, initially a single sensor. The corresponding data values in Figure

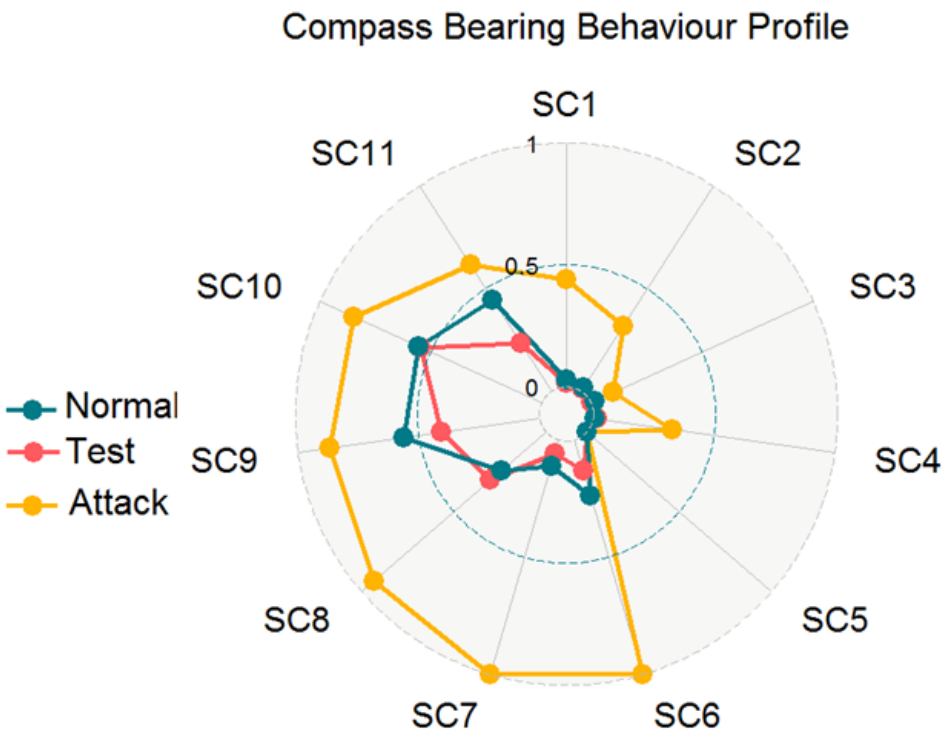

Figure 4.16: Compass bearing behaviour analysis

4.16 are shown in Table 4.9 for one data source using two different scenarios. In Figure 4.16 the deviations are illustrated in a form of radar chart, SC_i represents individual signature characteristic, where SC stands for signature characteristic and $_i$ represents individual characteristic, abbreviation description can be found in the Table 4.8. The learnt behaviour signature is based on running the identical normal scenario experiment five times. The absolute value of the registered maximums of all learning data sets is used as the anomaly limit. To identify the number of acceptable anomalies we have used our test scenario experiment runs. The data set from these runs is evaluated in regards to the "Learnt" knowledge to identify the amount of anomalies that exceed the learnt limit and adapt defence mechanism. This procedure has shown that two anomalies have been observed in the "Test" experiment and the threat summary score has been adjusted accordingly to define the range of normality which was discussed earlier.

The number of anomalies identifiable from an attack scenario data are shown (A) in the Table 4.9 as well as is illustrated in Figure 4.16. The summary score (behavioural score) is a sum of all deviation coefficients of signature characteristics, and is used as an indicator

of a threat to the system. Such score explains the deviation of the data source saying; the higher the score, the higher the deviation, therefore the higher threat risk to the system. This is also shown graphically in the Figure 4.16.

Initially all signature characteristics have equal weight and due to the weighting scheme used, signature elements are significantly out of line with expected values when an attack occurs and so the robot is able to autonomously identify an attack based on the detection of anomalies in the sensor data using our methodology. At a level higher, if we will take into account other data sources we can form a system behavioural profile and identify if the system is exhibiting normal or abnormal behaviour.

4.5.3 Evaluation of Heuristic Binary Classification

Earlier, we have demonstrated how a single data source is analysed producing the behavioural score that can be used at the level higher for surface analysis of the data source. If the behavioural score is exceeding the normality score, the system will investigate the lower layer and will identify what is the cause of such a high score. All system data source signatures operate in the same data domain allowing the system to produce a behavioural score by combining these signatures together.

In this section, we have reviewed a single data source from multiple experiment scenarios. At the level higher observation, the system uses the overall behavioural score produced by all available data sources. Such an approach can decrease the computational power requirement, however potentially a situation can arise where multiple signature characteristic readings are abnormal, but cancel each other out. Leading to a threat summary score which does not indicate a threat. This could mask an actual threat. This can be avoided by an occasional low-level analysis and by generating an interrupt-based procedures when an anomaly has been identified within the signature.

As for the system's final evaluation, it combines the threat summary scores of all data sources to classify the behaviour profile of a vehicle. The system has access to 17 in-

strumentation channels from the internal components, which include physical and cyber features, such as internal communication utilisation, or sensing the physical environment such as a compass bearing. For each data source a signature is automatically generated. These signatures can be used in isolation or in combination to determine the presence of anomalies and thus determine the level of threat.

To summarise our experimental setup, we combine signatures together to form a behavioural profile score of the system which can represent the level of threat to the system during a mission. All data sources have equal weights when combined together producing a sum of all available data source signature scores. Taking into account the primitive normality behaviour definition it is defined for each data source as the most severe difference taken from the "learnt" score and the "testing" scenario score, which for the discussed scenario is $46.323 \leq \textbf{Normality} \leq 62.152$, we assume that the "testing" scenario is more realistic and are using test scenario as a base for a normality range. For the current experimental setup, we have learnt that the overall average behavioural score from the learning scenarios was **46.323**. This score has been produced by a combination of signature summary scores from all available data sources during the learning stage. In this section, the key aspect was made to demonstrate the conceptual idea of a data source signature approach. The test scenario produced a score of **54.237**, we can notice that the score for the test scenario is higher than the "Learnt" behavioural score which was learnt using the learning scenarios, through a thorough investigation of the results we have identified that the number of allowed anomalies has not been exceeded and the overall higher behavioural score was produced by accumulated anomalies that were classified as allowed, resulting in a higher behavioural score. The attack scenario has produced a score of **113.6568**, which is considerably higher than the score produced by the learnt and test scenarios. This shows that the deviations from normality were greatly exceeding the threshold allowance on multiple data source signatures. The threat score is not limited to any value and would increase with either more abrupt environment or with the amount of data sources.

Further improvements have to be made to increase robustness of the described method-

ology. Currently, we are using sensor characteristic weights that are equally distributed, thus affecting an overall threat score of a sensor and system itself. Also to make the system more robust it would be necessary to investigate how correlation affects an overall behaviour score, as some data sources may have dependencies and these dependencies would result in an anomalous accumulative behaviour score that was described earlier. The weighting system has to be enhanced on a data source and signature characteristic level. One of the solutions is the examination of the spikiness level that can be used to implement dynamic weighting. One of the examples would be that the values from a sensor that are continually volatile (e.g., an accelerometer reading when travelling over a bumpy surface). In such cases, a lower weight would be assigned to the particular sensor characteristic or a signature characteristic. In this way the system can adapt to changing environmental contexts. It is less sensitive to noise or spikes when the ambient noise level or spike frequency is higher.

The work presented here forms part of a wider project to develop techniques for autonomous systems to self-detect attacks. In this section we have presented a sensor-agnostic learning technique in which a set of sensor-signal characteristics are collectively represented in a signature for each particular sensor. In terms of detecting attacks, the system need not know the type of the sensor, but instead looks at characteristics such as the typical noise levels, the range of data values, the rate of change of data values, the occurrence of spike values, etc.

The initial signatures are generated in experimental mission scenarios but in the absence of attacks, the data signals from sensors are therefore realistic in terms of data values, noise levels, etc. An attack is subsequently detected by observing significant deviations in one or more signature elements for a specific sensor or across several sensors. This approach lends itself to dynamic adaptation which enables the anomaly detection thresholds to be adjusted in line with the environmental volatility, although to date, we have only addressed this step at the concept level (using static signatures for detection).

The main strengths of our approach are that it can be applied universally across a wide

range of sensor types without needing manual configuration and that multiple signatures can be used to enhance the attack discrimination accuracy (facilitated by the standardised signature representation). In addition it has the potential to operate in a continuous learning mode in which it will adapt to its environmental conditions over short to medium time spans, but it will always be sensitive to abrupt changes.

We have developed a custom testbed vehicle in order to evaluate the approach. The experimental method and some initial results are presented above and illustrate how the vehicle was able to successfully identify anomalous events which were part of an attack. Our current findings are very encouraging. Due to the weighting scheme we use, the effects of significant differences between expected and actual sensor data are amplified and thus we have achieved a high true-positive rate and simultaneously a low false-positive rate.

The current implementation requires a training phase, during which it builds up behavioural signatures based on the sensed data signals. These signatures then form the basis on which reasoning is performed at several layers in our software stack. The first layer is concerned with anomaly detection at the level of a sensor, whereas at higher levels it is possible to gain a picture of the attack status across the entire vehicle.

Further work includes dynamic adjustment of the anomaly threshold, as discussed above with the intention of removing the need to retrain the vehicle for use in different environments, as well as further evaluation on the training algorithm itself to understand the optimal level of training and to avoid over-training or under-training issues.

Our cyber security approach is to consider the robotic system from the perspective that the system initially has no knowledge about itself, i.e. a box-in-a-box concept where the perception of the robotic system is stored in a box with several doors and the outer box represents the operating environment. The robotic system only observes values coming in or out and is not able to directly observe the true outside environment. In such a case, the robotic system's perception has to make sense to itself, without knowing what is outside and is entirely based on sensor data and patterns within. In such a scenario the autonomous system is sensitive to manipulated sensor data and therefore the signature

based approach has been devised specifically to facilitate discrimination between normal and abnormal situations, using a combination of learnt mean behaviour, current data signals and trends in data signals.

4.6 Weighted Heuristic Binary Classification (WHBC)

Vehicular cyber security has traditionally focused on passive attacks and especially on protecting the confidentiality of communications between vehicles or vehicles and smart infrastructures. However, over the last few years, autonomous vehicles have become a routine target for experimental cyber attacks, as demonstrated as early as 2009 by the University of Washington [18, 131] and in numerous blackhat conferences since then. As a result, there is a need for protection systems appropriate for active attacks against an autonomous vehicle's integrity or availability, and the corresponding impact on its actuation. Assuming that some attacks do get through regardless of the preventive measures, one needs to equip a vehicle with a mechanism to detect when this happens and potentially alert an operator or trigger some automated countermeasure. The focus of this work is on the real-time detection of the existence of an attack against a robot. We address both cyber-physical attacks, which are security breaches in cyber space that have an adverse effect in physical space, and physical-cyber attacks which are the reverse [8]. For this, we have developed an autonomous robotic vehicle with a variety of sensor and communication technologies typically found in the industry.

To ensure that any solutions developed are highly practical, we have set the following requirements:

- Detection should be real-time, so as to be able to support rapid and effective countermeasures.
- Detection should be carried out by the vehicle itself, so as to be applicable to autonomous vehicles with limited or no communication with their human operators.

- Detection should not rely on the availability of knowledge of previous attacks, so as to be applicable to unknown attacks too.

We do not rely on attacks on cyber-physical systems being frequent enough to allow for the gathering of a realistic body of knowledge on their impact. To meet these requirements, detection should be behaviour-based rather than knowledge-based. To address the above requirements, we have produced an onboard mechanism that monitors data related to cyber (communication and computation) and physical (actuation and sensing) features of the robot in real-time. During the training phase, the robot learns the normal range for the values of each feature monitored. In actual operation, it tracks the cyber and physical features that are in an abnormal state (beyond their learnt range) and accordingly reasons on whether a vehicle is in an attack state or not. The overall emphasis of the mechanism towards more tolerance for false positives or more tolerance for false negatives is configured by a sensitivity index, which determines the length of the normal range considered by the robot. We further improve on the detection accuracy achieved with this approach by also utilising individual weights for each feature, which are finetuned in a dedicated configuration phase.

In the previous section, we have presented a first attempt to provide completely sensor-agnostic and onboard intrusion detection that is applicable to unknown threats and takes into account both cyber and physical sources. Here, we extend this work considerably by providing a method to quantify the degree to which a vehicle is likely to be under attack without relying on a learning phase, and further improve it with a mechanism that assigns weights to the different data sources. We validate this approach with real-world experiments involving a variety of normal and attack conditions.

4.6.1 Signature of normal behavioural profile

In [34], we have described how signatures are formed and can be used for anomaly or threat identification. This learning phase is shown as “L1” in Figure 4.17. It is based on

an initial signature generation to establish the normal behaviour profile of the sensors on the system. The data from the sensors is transformed into a generic data source format that allows the system to reason about them identically. The learning phase forms a normal behaviour profile based on signature characteristics of each data source and forms normal behaviour variation that is used during the validation phase.

After the learning phase, dynamically detected values are compared with learnt normal behavioural profile signatures. The term "anomaly" is used to denote that a signal characteristic has been measured to be outside its expected normal range. The signature is formed of 11 characteristics which facilitate learning the normal value range limits, as shown in Table 4.8. Differences and deviations from the standard deviation are called spikes and these characteristics can be seen in Table 4.8. The validation phase, shown as "V" in Figure 4.17, classified behaviour based on an overall anomaly index represented by a number of anomalies in the system.

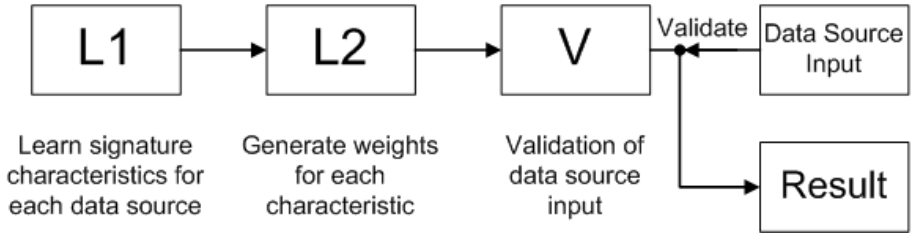

Figure 4.17: Methodology work flow

Each deviation of a signature characteristic is counted to represent an outgoing level of threat from the data source. These deviations are summarised to produce an anomaly index for the data source, this index represents the deviation level.

4.6.2 Anomaly Weighting and Indicator Confidence

To strengthen the detection performance, we have introduced an additional learning phase, shown as "L2" in Figure 4.17, which tunes the system by assigning weights to sources according to their likelihood of appearing anomalous in some normal scenarios too. The

focus of this phase is to learn the number of individual signature characteristic anomalies that may be encountered in a non-attack condition, arising due to environmental noise.

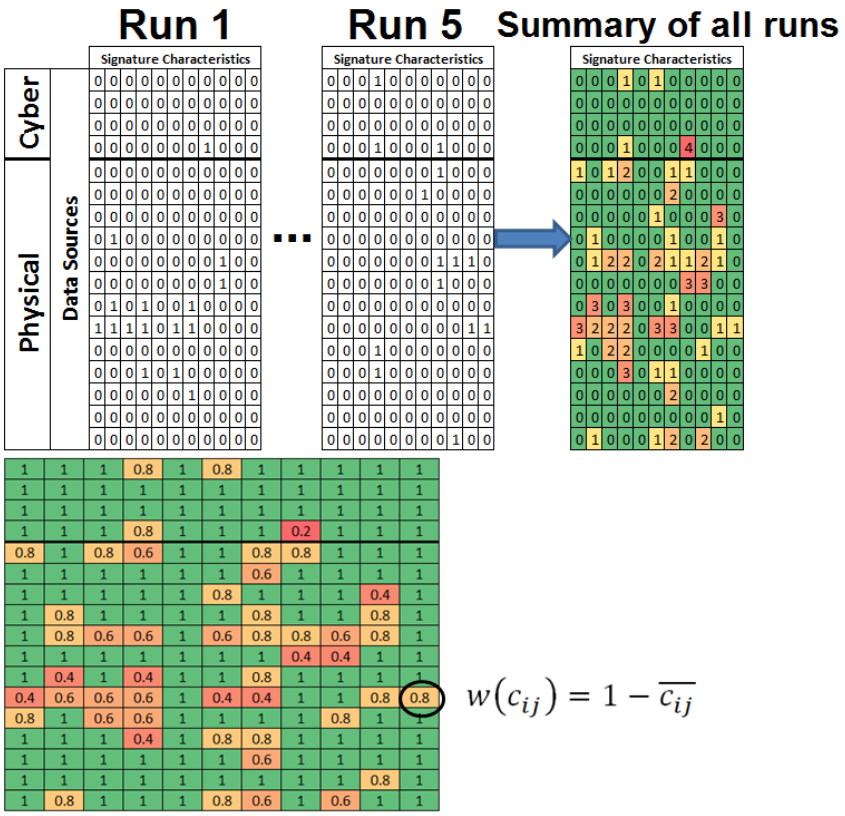

Figure 4.18: [**Initial Study**]Matrix of the anomalies identified. Each row corresponds to a data source and each column to a signal characteristic measured for each source. The colour coding indicates anomalies

In Figure 4.18, we demonstrate how we summarise anomalies by taking the system data source signatures from five non-attack situations. Only two experimental run matrices are shown here for presentation purposes. To reduce the importance of anomalies that tend to occur in a non-attack environment we calculate the weight of each signature characteristic anomaly sample $w(c_{ij})$ in the following way: for a number of n scenarios S_ℓ, $1 \leq \ell \leq n$, we take the complement of the mean of each signature characteristic anomaly sample $c_{ij(\ell)}$, where again i represents a data source and j represents a signature characteristic:

$$w\left(c_{ij}\right) = 1 - \overline{c_{i,j}} = 1 - \left(\frac{\sum_{\ell=1}^{n} c_{ij(\ell)}}{n}\right)$$

which produces the weight of an anomaly sample for the signature characteristic. This

allows the system to derive a more precise score taking into account the anomalies that tend to be less indicative of an attack as they persist in a non-attack conditions.

The calculated value represents the weight of a signature characteristic. If the system learns that a particular signature characteristic has a high probability of anomaly occurrence in a non-attack mission scenario then the importance of such anomaly is reduced. This generates a lower anomaly index for the data source's signature. The sum of all weighted anomalies generates an overall anomaly index that is used as a reference in the intrusion detection mechanism. To improve the methodology further we introduce a dynamic variable that acts as a controller of the "normality" threshold. The "normality" variation is formed during the "L2" phase. The overall anomaly index generated from the non-attack experiments is used as a mean reference and the dynamic variable controls the variation. This allows the detection mechanism to identify anomalous behaviour in two cases: when multiple anomalies are detected generating a high overall anomaly index, as well as when anomalies are not detected, therefore generating a low overall anomaly index. An overview of a work flow of the intrusion detection mechanism can be seen in Figure 4.19.

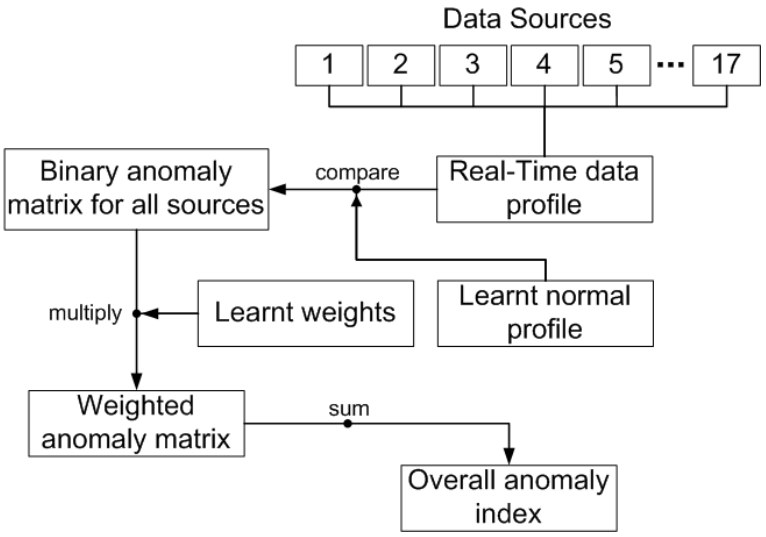

Figure 4.19: Intrusion detection mechanism of the robotic vehicle testbed

When the learnt weights scheme is applied to detected characteristic anomalies in several attack and mechanical failure experiments, the anomalies that have higher weight are

accentuated and the importance of anomalies that tend to occur during a non-attack mission scenario are reduced. This reduces the anomaly score of the system in a non-attack scenario and ensures the score increases when abnormal circumstances occur on characteristics that should not otherwise change during non-attack experiments.

	Compass Attack										Rogue Node Attack									
Packet Arrival Rate	0	1	0	1	0	1	1	0	0	0	0	1	0	1	0	0	1	0	0	0
Action Indicator	0	0	0	0	0	0	1	0	1	2	0	0	0	1	0	0	0	1	0	1
Sequence Number	0	1	0	1	0	1	1	0	0	0	0	1	0	1	0	0	1	0	0	0
Internal Network	0	0	0	0	0	0	0	0	0	0	0	4	0	3	0	4	4	0	4	3	0
Roll	2	0	2	0	0	1	2	0	0	1	0	0	0	0	0	0	0	0	0	2
Front Range	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Back Range	0	0	0	0	0	2	0	0	0	1	0	1	0	1	0	0	1	1	0	1
Left Range	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Right Range	0	3	2	2	0	2	3	1	1	2	1	0	0	1	0	0	0	0	0	0
Temperature	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0
Battery	2	0	3	0	0	0	2	0	1	0	1	0	0	0	0	0	0	0	2	0
Bearing	2	2	2	2	0	2	2	0	0	0	0	0	0	0	0	0	0	1	0	1
Pitch	3	0	2	0	0	1	3	1	1	0	0	1	1	1	0	0	0	1	2	0
Motor #1	0	1	0	1	0	0	0	1	1	2	0	1	0	0	2	1	0	1	0	0
Motor #2	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
Motor #3	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
Motor #4	0	1	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
Anomaly Score	**90.8**										**58.8**									

	Packet Injection Attack										Broken Wheel									
Packet Arrival Rate	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Action Indicator	0	0	0	1	0	0	1	0	0	2	0	0	0	0	0	0	0	0	0	0
Sequence Number	0	0	0	3	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0
Internal Network	0	0	0	3	0	0	0	1	4	2	1	0	0	0	0	0	0	0	1	0	0
Roll	1	0	0	0	0	0	3	0	0	0	0	0	0	2	0	0	1	0	1	0
Front Range	0	0	0	0	0	0	0	4	4	1	0	0	2	0	1	0	2	2	1	0
Back Range	0	0	0	0	0	3	4	4	3	2	0	0	0	0	0	0	1	0	0	0
Left Range	0	0	0	0	0	0	2	4	1	1	0	0	0	0	0	0	0	0	2	1
Right Range	0	1	2	1	0	2	1	0	1	2	2	0	0	1	0	0	0	0	0	0
Temperature	0	0	0	0	0	0	4	2	2	3	0	0	0	0	0	0	1	0	1	0
Battery	4	2	4	2	0	4	3	0	0	0	4	0	2	0	2	0	0	0	1	0
Bearing	2	2	2	1	0	2	2	0	0	1	3	1	2	1	2	0	1	2	0	1
Pitch	0	0	0	1	0	0	0	2	2	0	2	1	0	2	0	0	0	0	0	1
Motor #1	0	0	0	1	0	1	0	1	2	1	0	0	0	0	0	0	0	0	1	1
Motor #2	0	0	0	1	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0
Motor #3	0	1	0	2	0	0	0	2	0	1	0	0	0	0	0	0	0	0	1	0
Motor #4	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1	0
Anomaly Score	**142.2**										**48.4**									

Figure 4.20: Matrices of anomalies spotted for each of the incidents in the experiments (compass manipulation, rogue node, replay packet injection, wheel failure)

In Figure 4.20, we observe 4 different scenarios when the robotic vehicle testbed is in operation and is under cyber-physical attacks (Replay packet injection and rogue node), a physical-cyber attack (compass manipulation) or during an unexpected mechanical failure. The result in each cell is rounded to the nearest whole number for presentation purposes. The overall anomaly index is derived from a sum of all signature characteristic anomaly results when the weighting scheme is applied. This index is used by the intrusion detection system to reason the behaviour of the robotic testbed vehicle at the level of an overall anomaly index observation, reducing the need to analyse each signature characteristic as they are updated, thereby reducing computational requirements.

4.6.3 Performance evaluation of the WHBC

Figures 4.20, 4.6, 4.7 and 4.8 show example experimental runs for different cyber and physical incidents and their impact on the certain cyber and physical data sources used by the detection mechanism. Green corresponds to a normal state with no noticeable

impact from an attack. The colour shift to red indicates a higher anomaly occurrence detection in the set of attack scenarios as well as during the accidental mechanical failure. For evaluation of the proposed binary classifier in discussed detection mechanism, we have used widely used performance evaluation technique for machine learning classifiers and anomaly detection mechanisms[132] or in the cyber security domain for example was used in performance evaluation of the model for forecasting cyber security incidents [133] or a model for cybercriminal mining and profiling in social networks[134]. The receiver operating characteristics (ROC) curves are able to represent false-positive detections against the true positive, as well as providing contextual information about the sensitivity of the model. It is possible to identify the optimal sensitivity metric to achieve a certain detection based on the application and the requirements, additionally using an area under curve (AUC) score it is possible to describe overall detection performance.

An Initial study has shown that classifiers had high detection rates for "Compass Manipulation" and "Packet Injection" attacks, the AUC scores are demonstrated in the Table 4.10. This has demonstrated that the attacks which were designed didn't have controlled variability e.g. it was not possible to control the distance between the compass module and the magnet. Additionally we have not reviewed the scenario where an attacker may sniff the communication packets in an in-field environment and then replay them. This has led us to improve our attack scenarios and allowed us to represent more realistic scenarios by introducing intensification in the attacks.

Table 4.10: [**Initial Study**] Overall performance based on ROC Area Under Curve for Compass Manipulation, Packet Injection and Rogue Node attacks

	CM	**PI**	**RN**
Signature + Weights	1.0	1.0	0.875
Signature	0.938	1.0	0.521

When the attack scenarios were improved, an additional decision was made to drastically increase the number of experiment scenarios to reevaluate our findings more accurately. This has increased the number of "Learning" scenario runs to fifty, the number of "Test" scenarios which are used to identify the weights were increased to seven. The number

of "Normal" scenarios that will be evaluated as false-positive detection was increased to fourteen. The number of each attack scenario was increased to ten giving realistic attack approaches, summing the attack data set to thirty scenarios. The results are shown in Figures 4.24, 4.22, 4.23 and the overall attack detection performance in Figure 4.21.

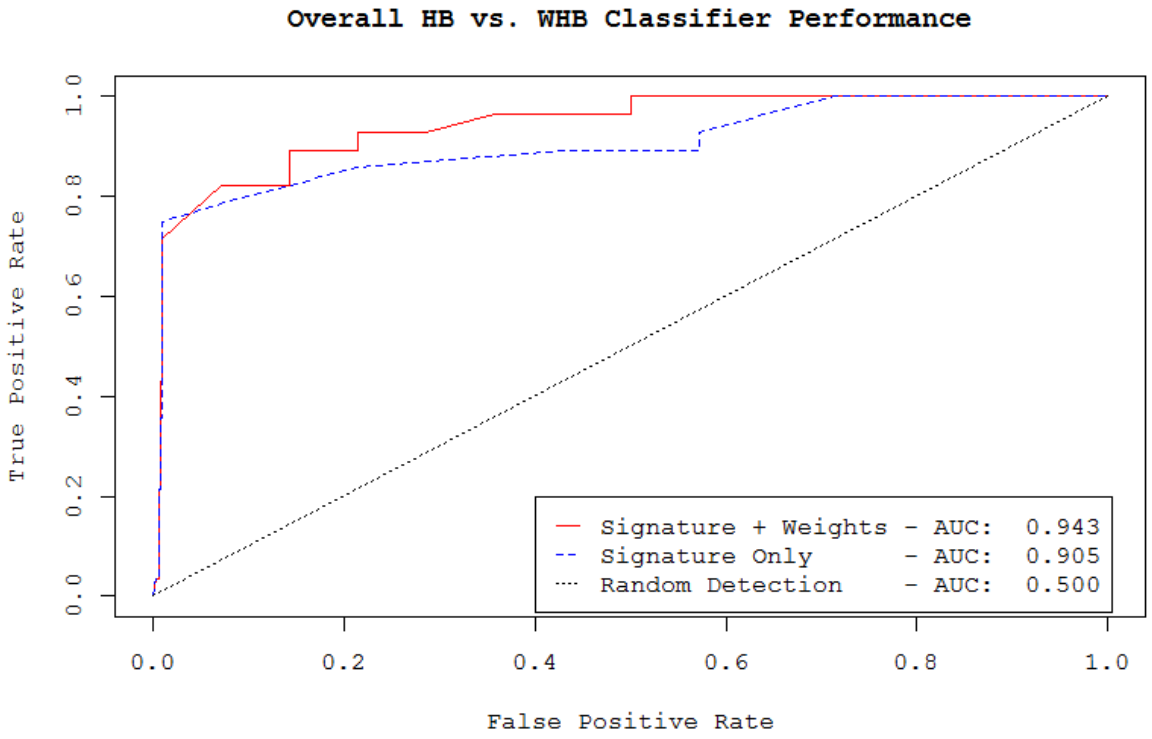

Figure 4.21: Rogue Node Attack ROC Performance

In Figure 4.21 it is shown that the "Signature Only" classifier has a reasonable overall detection performance and it is able to identify three cyber-physical attacks producing reasonably low false-positive rate with an AUC score of 0.905. This however was improved by introducing weights into the model and this is shown as "Signature + Weights" classifier result, but there is a slight reducution in the detection of normal scenarios with a low false-positive rate. This is due to the fact that the frequency of anomalous events are being treated in a such a way that if a sensor is unreliable or produces too much noise on a specific characteristic, this characteristic will be given little weight as this often occurs frequently.

Detection accuracy for the physical compass manipulation attack is shown in Figure 4.22,

demonstrating that the impact of this attack was high and affected multiple data sources, as they have started showing sporadic behaviour in the sensor readings producing high deviations from normality and therefore producing a high score that exceeded the allowed normality threshold score.

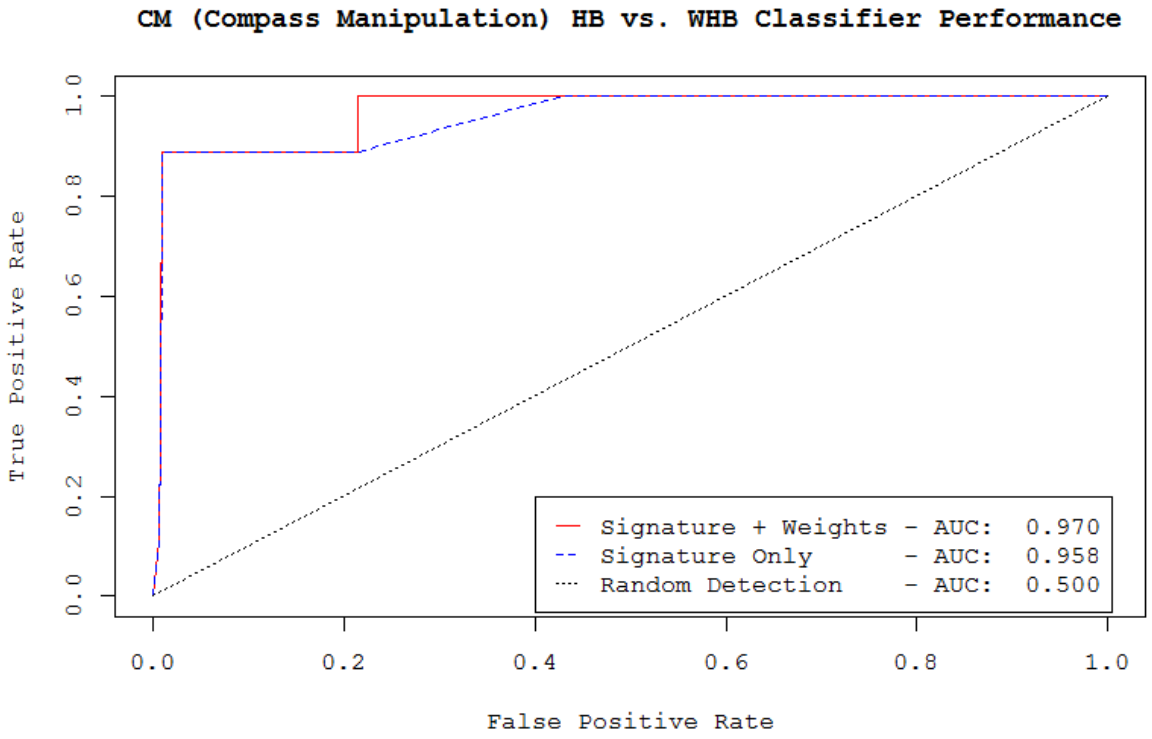

Figure 4.22: Compass Manipulation Attack ROC Performance

In a "Replay Packet Injection" attack experiment the physical impact on the vehicle was low, as the external communication was under attack, and the vehicle relied only on the internal network communication data. For the testbed vehicle it was an issue to detect the attack, as it does not monitor external communication. This issue may be addressed in future work as our project is more concerned about the internal vehicle, as this part of the system allows a testbed vehicle to be autonomic and enables autonomicity in the robotic vehicle. As was mentioned earlier in this document the detection can be done distributively, such that the detection mechanism can be planted on the robotic system itself as well as on the operator's machine. This will increase the overall detection performance of the mechanism. The detection performance with weighing mechanism has

shown an increase in the detection performance rate of the more realistic scenarios, such as when an attacker has remote access to the communication medium and replays packets on the network. The detection performance for "Replay Packet Injection" attack can be seen in Figure 4.23.

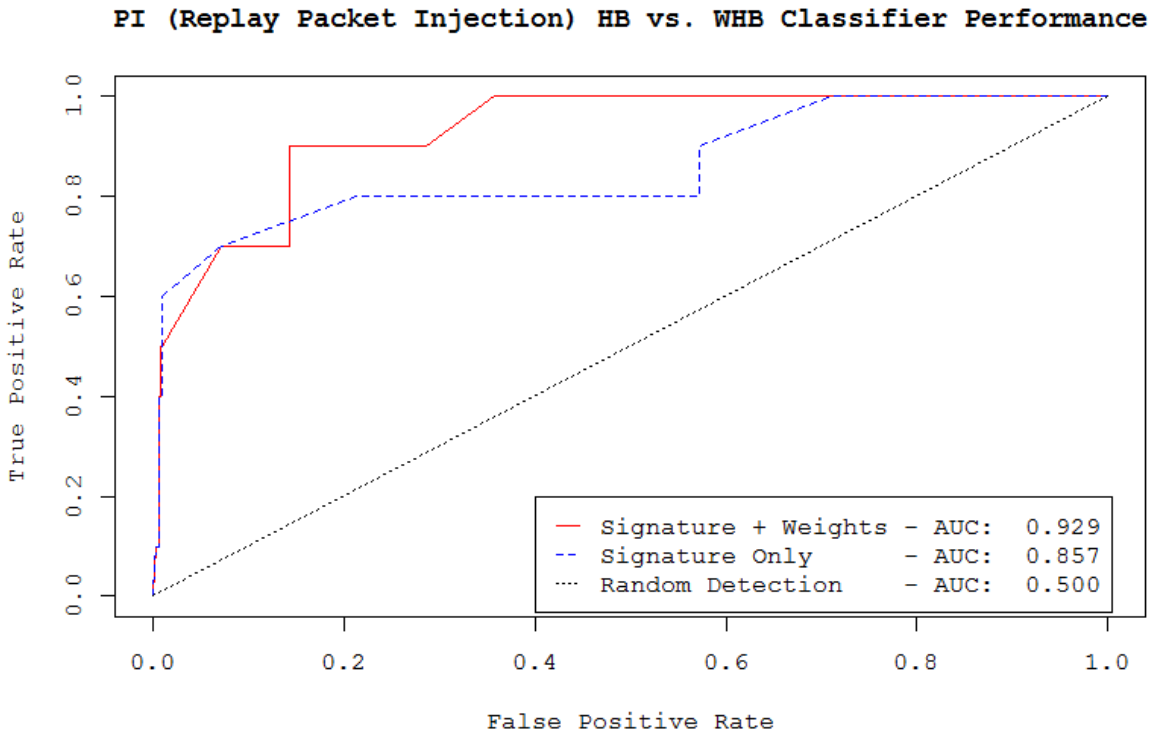

Figure 4.23: Packet Injection Attack ROC Performance

Using the sensor-agnostic approach an issue can arise when deviations are detected in the data source without any impact on the robotic vehicle testbed, increasing the false-negative detection rate. Introducing weights in to the detection technique, improved the detection rate for the rogue node attack as shown in Figure 4.24. This technique emphasises the anomalies that are not frequently seen in non-attack scenarios.

Referring to our initial study experiments it is noticeable that with an increase of scenarios that were used for reevaluation of the methodology, the detection performance has increased as well. Through out the analysis it has been discovered that the cause of that was due to the fact that the repeated packets using high intensity were affecting the robotic vehicle testbed systems and as a result were spoofing sensor readings on the

internal systems. While low intensity packet amplification has shown low impact on the internal systems. The detection performance result can be seen in Figure 4.24.

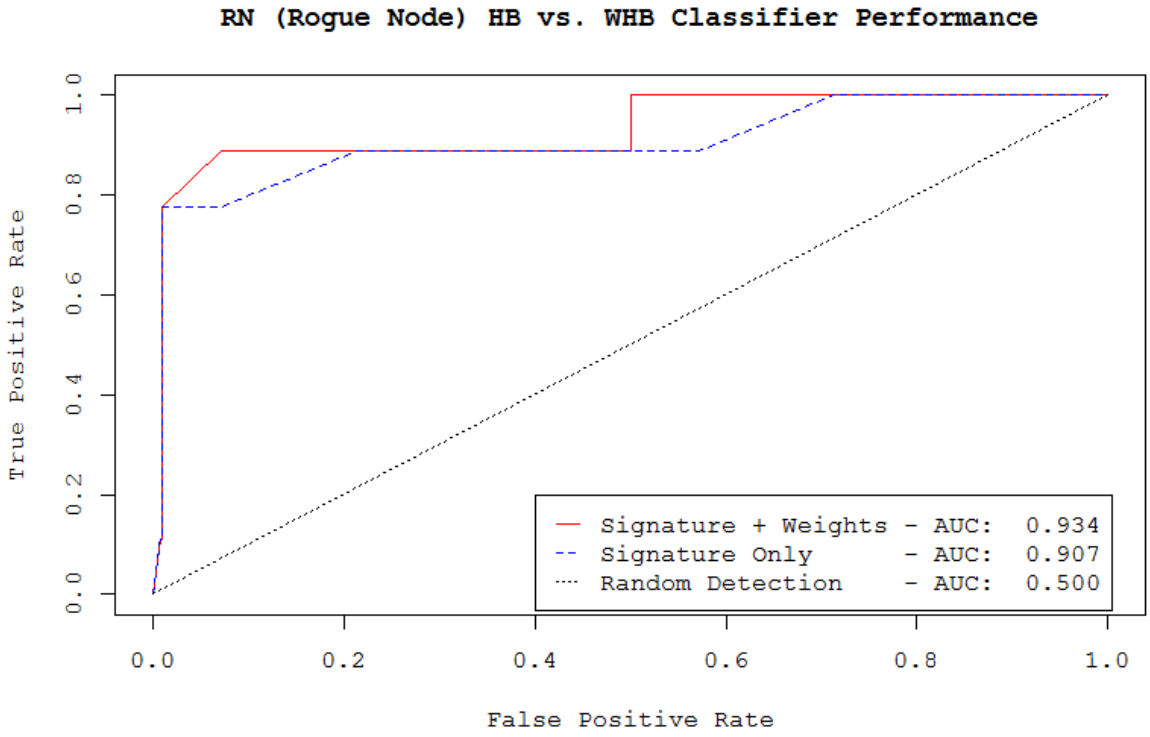

Figure 4.24: Rogue Node Attack ROC Performance

In this section we have presented a sensor-agnostic methodology that is demonstrably able to run on-board a resource-constrained autonomous vehicle and detect in real-time attacks, which it has not been exposed to before. We achieve this by learning the normal ranges for a wide variety of cyber and physical data sources and applying weights to fine tune their importance for detecting anomalous behavioural profiles. The approach has shown to be promising for the variety of different (unknown to the vehicle) attacks that we experimented with, including an unplanned physical failure.

4.7 Real-Time Heuristic Binary Classification

In previous sections we have described our approach to anomaly identification. In this section we will discuss the issues and the reasons for switching from an offline analysis to

real-time domain and what kind of issues may arise.

4.7.1 Real-Time Anomaly Identification (Offline)

To simulate the real-time anomaly classifier we have used the same data set we have collected from an operators workstation. This data set was used in our previous work. The methodology that we have approached in real-time approach which is different from out offline detection. The learning phase is done in the same way as it was done in our previous work. The methodology collects the whole data set without any manipulations as it comes from an operators workstation and is not using the weights improvement to keep the data as a discreet object. Based on the user specification the data is collected and

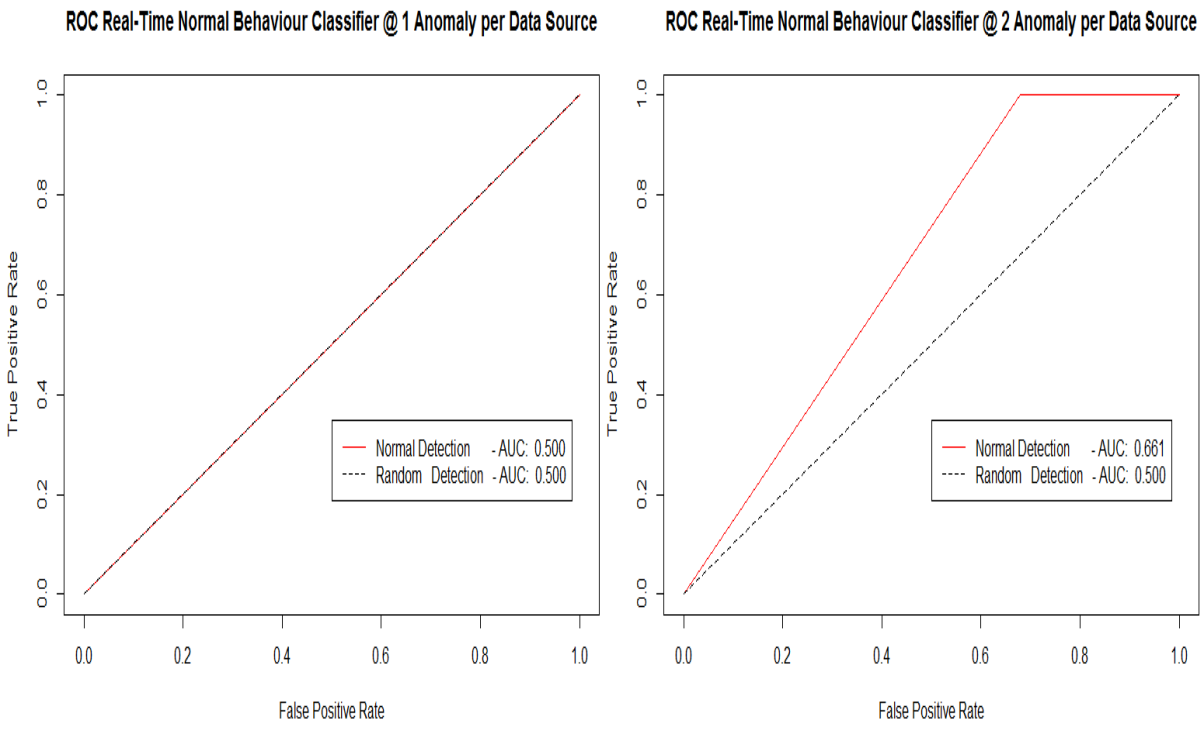

Figure 4.25: Real-Time Normal Behaviour Identification ROC Performance (1 or 2 data source anomalies are used)

when it has reached the learning limit, it will calculate the signature characteristics for behaviour characterisation. For the testing phase the methodology will start the testing phase in the same way as it was done previously, it will evaluate the data severeness in

terms of deviations and will identify the amount of anomalies that will be used as an indication of an anomalous behaviour.

When the model has been trained, then the data set samples are fed to the model, with each sample the signature characteristics are being reevaluated to be tested by the model. There are several issues that arise in such case is that the signature characteristics will raise anomalies. This is due to a problem of data starvation, in that the new model that is being calculated does not have enough samples to generate the model more accurately the model that will be ompared. Another limitation that this approach will have is that the false positive rate is increasing, this is due to methodologies used in the heuristic binary classification signatures are used. One of the examples would be in calculating the standard deviation. The reason is that if we want to calculate standard deviation it is preferable to have the whole data set to calculate more precise standard deviation. If a signature characteristic such as identifying a minimum value limit can be identified through the sampling of one single variable, in case of standard deviation we are not able to do it in a precise manner due to the requirement of an array of samples.

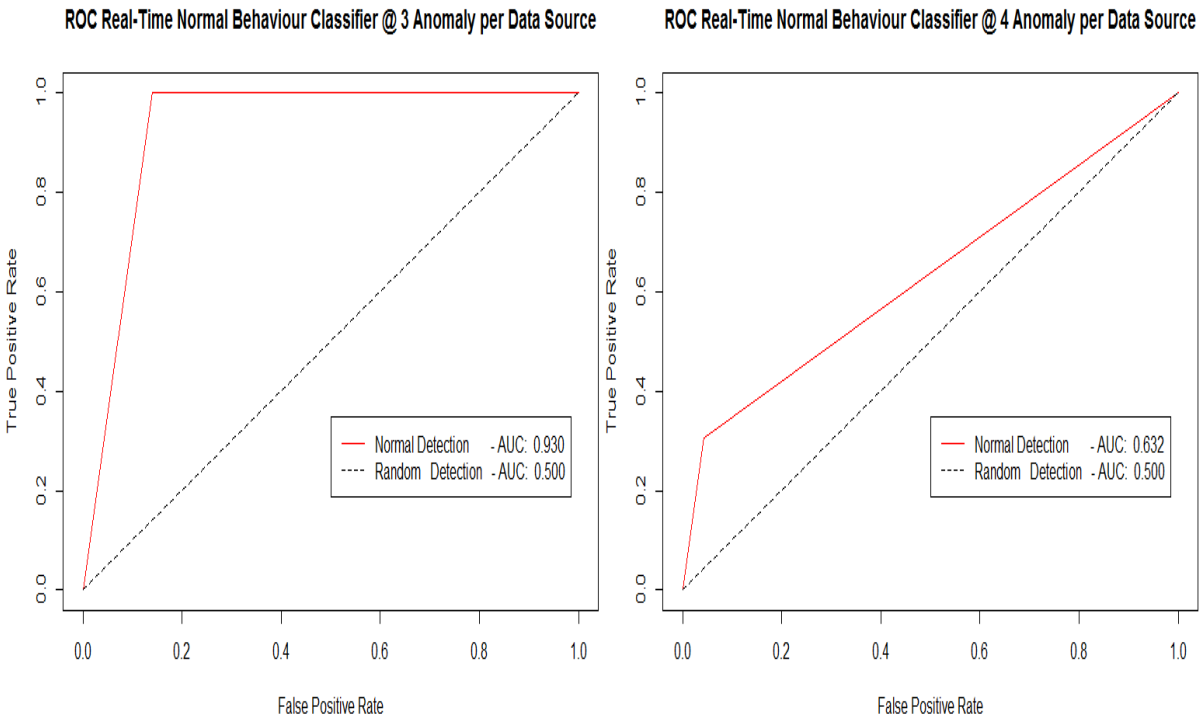

Figure 4.26: Real-Time Normal Behaviour Identification ROC Performance (3 or 4 data source anomalies are used)

The samples are then fed into the model which will be compared to a learnt model. The idea is to look at the incoming values taking into account all the previous samples, thus generating more precise signature for each of the data sources. This can be called as a warm up process which is applicable to any system that has to initialize itself. This heat up activity will influence overall performance score when evaluated. To evaluate the performance we have use ROC as can be seen in Figures 4.25, 4.26 and 4.27. In Figure 4.25 we are able to observe absolute random normal behaviour identification in real-time given that the model has been tuned as being over-sensitive, and will treat any anomaly as an abnormal behaviour. However slightly reducing the sensitivity of the model by increasing allowance of of acceptable anomalies the performance starts to increase. Figure 4.26 shows a drastic increase producing an AUC score of **0.930**, this was the highest score we were able to observe from our dataset. Increasing the allowance further on, demonstrated that the normal behaviour detection started to decrease to a point where there is full detection uncertainty as demonstrated in the Figure 4.27.

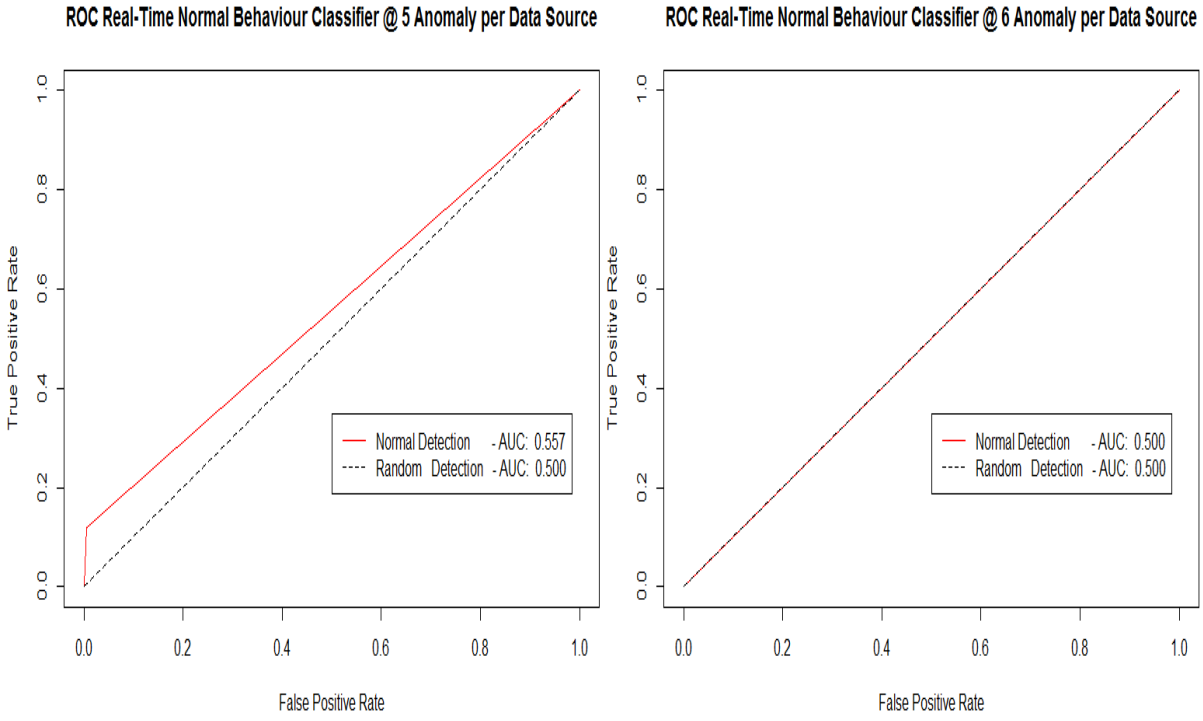

Figure 4.27: Real-Time Normal Behaviour Identification ROC Performance (5 or 6 data source anomalies are used)

This method allowed to identify the balanced number of anomalies that provide best

detection rate. This parameter will be used further in this research as it provides better performance over the others.

4.7.2 Defence Mechanism integration

The first stage of the defence mechanism integration is to gather data on the operator's control machine. This is done so as to be able to access additional processing resources that are not available on a robotic vehicle as it is built with power efficiency in mind. Such approach would allow trying out experimental approaches in anomaly detection. Following stage is to integrate defence mechanism on the robotic vehicle testbed, which will allow cooperating with operator's machine and improving detectability of attacks. The goal is to enable cooperation between the operator's system and the robotic system to improve anomaly detection and allow autonomous system to analyse data independently. The key steps of the defence mechanism implementation on the operator's machine is to support following functions:

- Learn the operating environment of a robotic testbed vehicle based on received sensor data
- Collect and store all incoming sensor data in a format to ease analysis
- Support functions such as: Activate/Deactivate Defence Mechanism, Learn

One of the reasons for involvement of an operator's machine was that during defence mechanism development we will be able to test experimental approaches, and we will be able to store knowledge bases for different environments. The defence mechanism was integrated in to Graphical User Interface. Graphical user interface can be seen in Figure 4.28 where blue light represents learning state.

Graphical user interface gathers all information that we have described earlier in this document, and will form normal behaviour of the robotic vehicle during learning state. When anomaly detection mechanism will be activated, application will use two colour

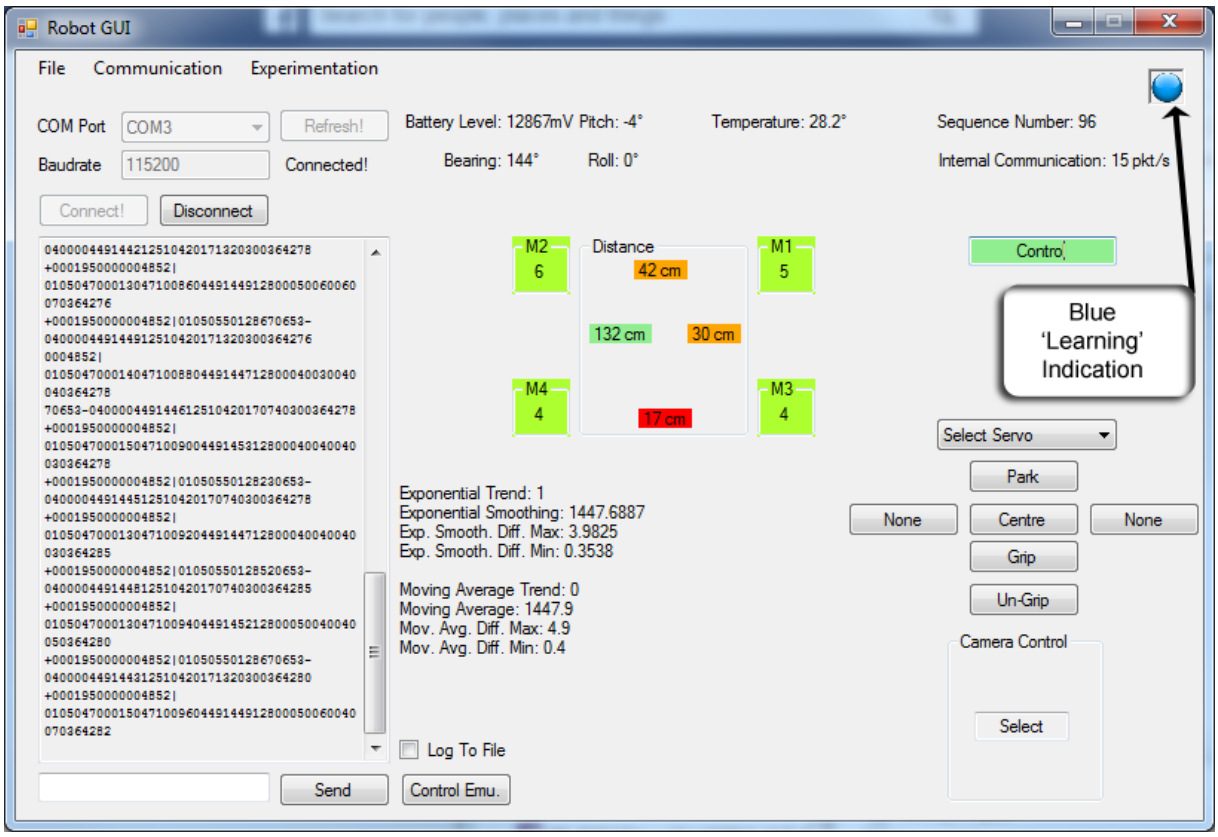

Figure 4.28: Robotic Testbed User Interface

codes that represent identified anomaly (red) or if no anomalies were found will represent it using green colour These are shown in the same position as blue 'learning' indication, which is shown in figure 4.29.

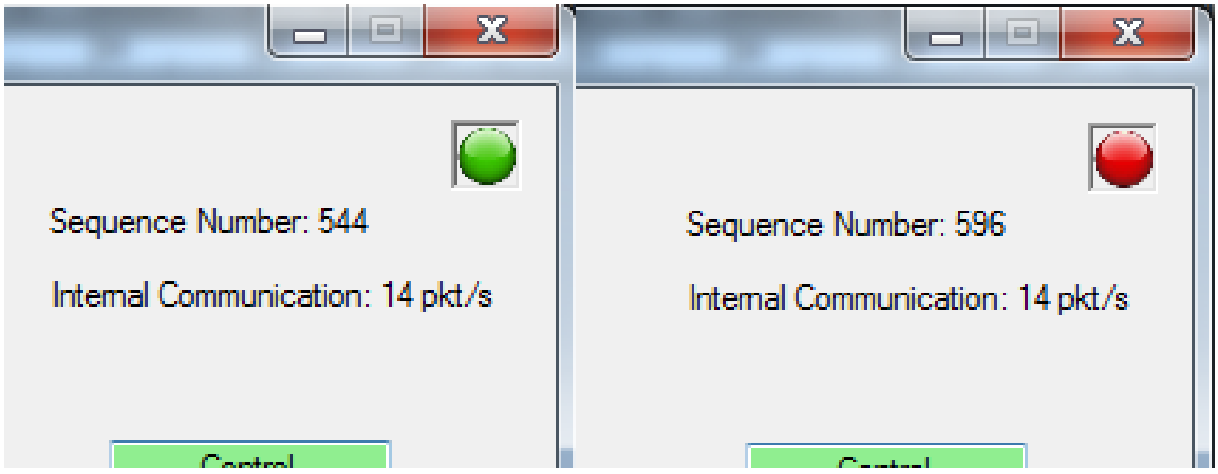

Figure 4.29: Normal and Abnormal Behaviour Colour Codes

Software is being developed as a library that will allow identification of anomalies and will introduce an early warning system on the operator's workstation. Additionally in the

vehicle's control software it is possible to monitor data source readings and observe the anomalies in real-time, example of one of the motors behaving anomalous can bee seen in Figure 4.30.

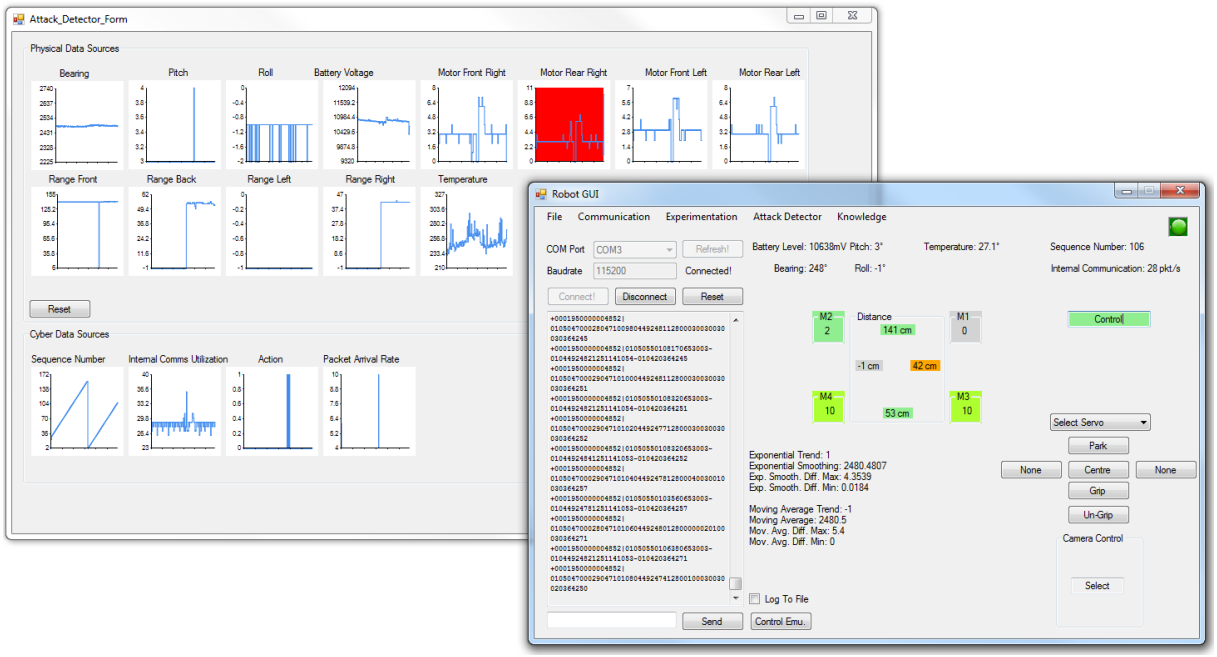

Figure 4.30: Anomaly identified in data source monitoring window

Detection mechanism on an robotic vehicle testbed was developed and integrated on the Raspberry Pi 1 Model B. This embedded system has been chosen as it provides C libraries is resource efficient and the code developed in C can be ported on an Atmel ATMega micro controller, however porting the code to a different platform is out of the scope of this project is to develop a prototype that will be able to detect cyber-physical threats on an embedded system which will be on-board of the vehicle. The embedded version works such that it feeds the data from the CAN bus directly to the classifier which goes through three stages: Learning using specified number of samples from the CAN bus, afterwards it evaluates the data using specified number of data samples to learn the amount of anomalies that will be treated as noise or the environmental glitch, then the mechanism will proceed to detection.

From the Figure 4.31 it is observable how this defence mechanism operates, for presentation purposes to run the mechanism (which can provide the output in a user readable format) we have used a bash shell script which calls a **init.sh** script and a **run.sh** and

Figure 4.31: Defence mechanism operation on the Raspberry Pi 1

reinitialise Ethernet and wireless interfaces due to an issue where an initialisation of CAN interface would corrupt the configuration of these communication interfaces, code listings can be found in Figure 4.32.

```
pi@RoboPi ~/filter_project $ cat start_blind.sh
#/bin/bash
/home/pi/can-test_pi2/init.sh
/home/pi/filter_project/run.sh
ip link eth0
ip wlan0 down
ip wlan0 up
pi@RoboPi ~/filter_project $ cat /home/pi/can-test_pi2/init.sh
#/bin/bash

/sbin/ip link set can0 up type can bitrate 125000
pi@RoboPi ~/filter_project $ cat /home/pi/filter_project/run.sh
#/bin/bash

/home/pi/filter_project/filter "../can-test_pi2/candump can0"
```

Figure 4.32: Code listings for scripts: start_blind.sh, init.sh and run.sh

The defence mechanism stores ID of a message and treats it as a generic data source, however certain protocol specific features need to be provided for defence mechanism to operate such as if there are different sensors sharing the same ID and message then they

are categorised as different types. In our prototype version we use defined types or if the protocol specifics were not defined it will use 32-bits as a data source value. Anomalies that are mentioned in the Figure 4.31 are in a binary format representing signature characteristics which are mentioned in the Table 4.8. Then the defence mechanism calculates total of anomalies from the incoming data and compares to a number that was learnt during the testing stage and makes a decision whether data source is anomalous or not. The final decision is made based on the learnt knowledge of the system. During the testing stage the defence mechanism identified amount of allowed glitches in the data source, as well as the number of anomalous data sources have been seen at the same time, which in the described case 2 anomalous data sources are counted as an allowed false alarm.

4.7.2.1 Performance Analysis

The performance of the heuristic binary classification has been measured programmatically. We have used a time stamp provided by an operating system (Raspbian, Linux Kernel 4.4.27+). The method is that the program issues a timestamp from an operating system, does the anomaly detection on 500 data samples that are used to enable accurate calculation of the time taken for a single measurement, thereafter issues a newer time stamp and calculates the difference between these time stamps. The result is at the operating system time precision, for measuring performance of the classifier we were monitoring execution time in nanoseconds. These functions are shown in Listing 4.1.

Listing 4.1: Timestamp functions

```
/* These functions were used to measure execution time */
struct timespec timer_start(){
    struct timespec start_time;
    clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &start_time);
    return start_time;
}

long timer_end(struct timespec start_time){
    struct timespec end_time;
    clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &end_time);
```

```
    long diffInNanos = end_time.tv_nsec - start_time.tv_nsec;
    return diffInNanos;
}
/* example was found on Stackoverflow: http://stackoverflow.com/questions/22579157/kernel
    -mode-clock-gettime */
```

Performance analysis has shown that the execution time of the heuristic binary classification has a linear trend as the methodology was designed to be modular and generalize any sensor data that is fed to the model. This allows the model to be predictable in the sense of an execution time and can be used as an indication in the future. To calculate the trend we have used a linear regression approach. The averaged classifier execution times are presented in Figure 4.33, in the graph we demonstrate how long it takes to perform anomaly detection on a variable number of data sources. The measurement process was repeated five times, each time sampling the data source 500 times to overcome the operating system's limited time-measurement granularity, as discussed above.

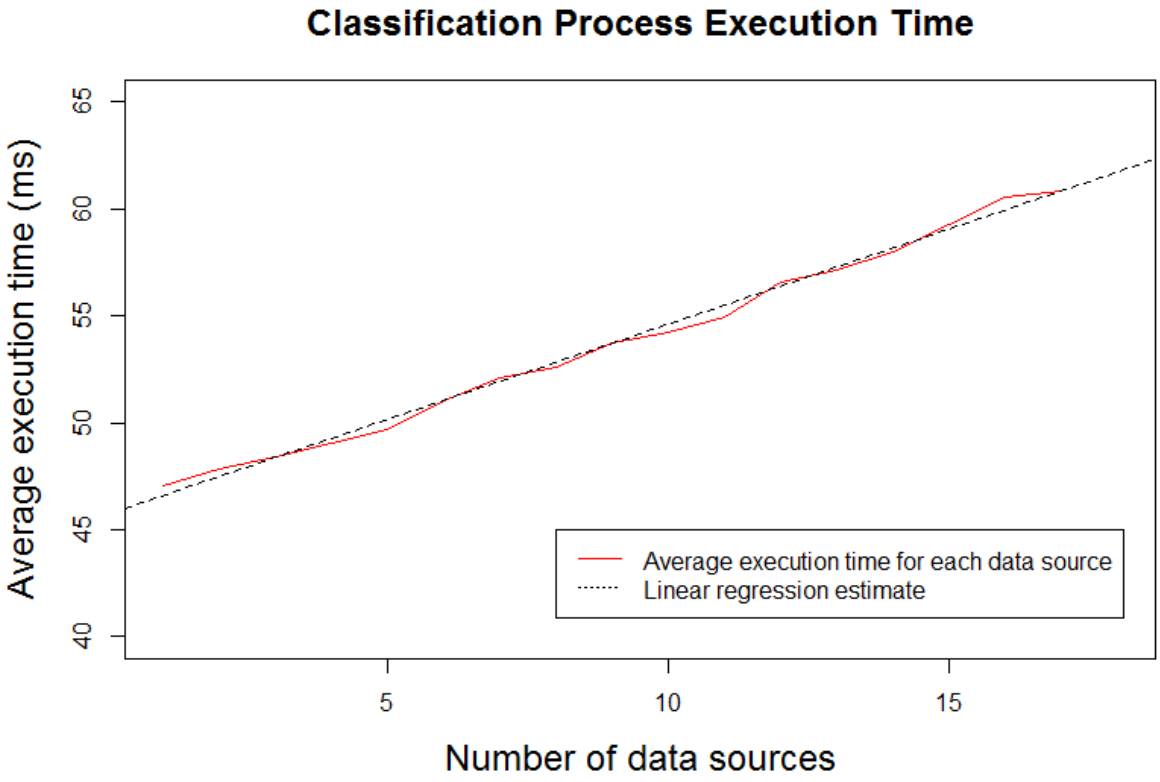

Figure 4.33: Classification process execution time

From the data we have identified that the single data source in the detection phase requires

a certain amount of execution time to provide the anomaly detection result. We have to note that the experiment was conducted on a Raspberry Pi 1 (Raspbian) clocked at 800 Mhz which is provided by the "modest" overclock setting. To evaluate the computational performance of the Raspberry Pi 1 with "modest" configuration, we use commonly used floating points operations per second (FLOPS) metric [135] to measure computational performance. To perform such benchmark we use LINPACK benchmark tool which is used as a standard for evaluation of computer system's computational performance [136]. The score produced for the Raspberry Pi 1 configuration in the idle state was 49.52 MFLOPS using the "Unrolled Double Precision" setting. Our analysis of detection mechanism showed that one data source requires **0.8298** ms of execution time to process five hundred of samples, this leads us to the conclusion that one data source requires **1.6596** μs to process one sample, taking into account that the computational performance of a Raspberry Pi 1 is 49.52 MFLOPS and the execution time we can extract the cost in terms of floating points operations, that are required by the detection mechanism to process 1 data sample for 1 data source. We do this the following way, knowing that the processor is able to perform 49520000 FLOP a second, we know that to process 1 data source and 1 sample it took $1.6596E-06$ s, therefore the cost in FLOP's would be $49520000 * 1.6596E-06 = 82.18339$ FLOP. The FLOP metric can be used for an evaluation of the detection mechanism's performance on other computer systems that are able to perform floating point operations. In regards to the CPU utilisation of Raspberry Pi which can be seen in Figure 4.34, while running the detection mechanism.

We have also measured the memory usage in a variety of robotic vehicle testbed states, where the robotic testbed vehicle has been executing specific tasks such as: moving, idling or undergoing defined mission. Overall we have had 1230 samples of process memory usage reported by the operating system. We have focused on the virtual memory and the resident size memory as it demonstrates an overall memory requirement and the amount of memory required for classification processing. The result of this can be seen in Figure 4.35, which shows that the memory usage is very low, and the variations are so small that it can be considered as effectively constant.

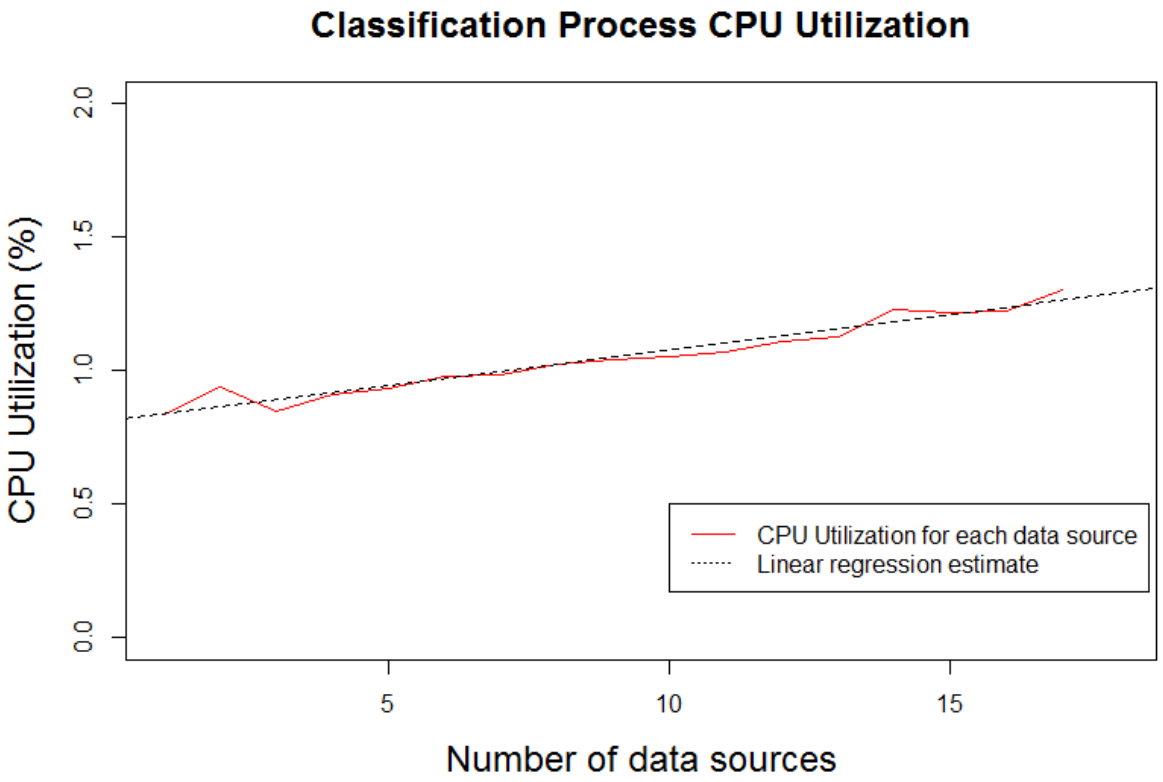

Figure 4.34: Classification process CPU utilization

To conclude the computational requirements we can state that it is possible to integrate this model onto an embedded system. Based on the knowledge we got it seems that **theoretically** it should be possible to integrate this model into our current Atmel ATMega micro controllers which are installed on the robotic testbed. This is based on the fact that the memory requirements are met. The main concern would still exist as the ATMega1281 is clocked at 16 MHz and in the mean time the Raspberry Pi is clocked at 800 Mhz. This shows that the Atmel micro controller is running 50 times than the lower clock speed. If we take this into account we can rescale our execution time findings based on the difference in the clock between two systems. This would lead us to **82.98** microseconds execution time for one data source and one sample and would require **41.49** milliseconds of execution time for one data source and five hundred data samples.

For some systems it may be acceptable, however it may raise an issue for safety-critical real-time systems. This can be avoided by integrating the detection model into a system which has higher computational power. Theoretically it is possible and based on our

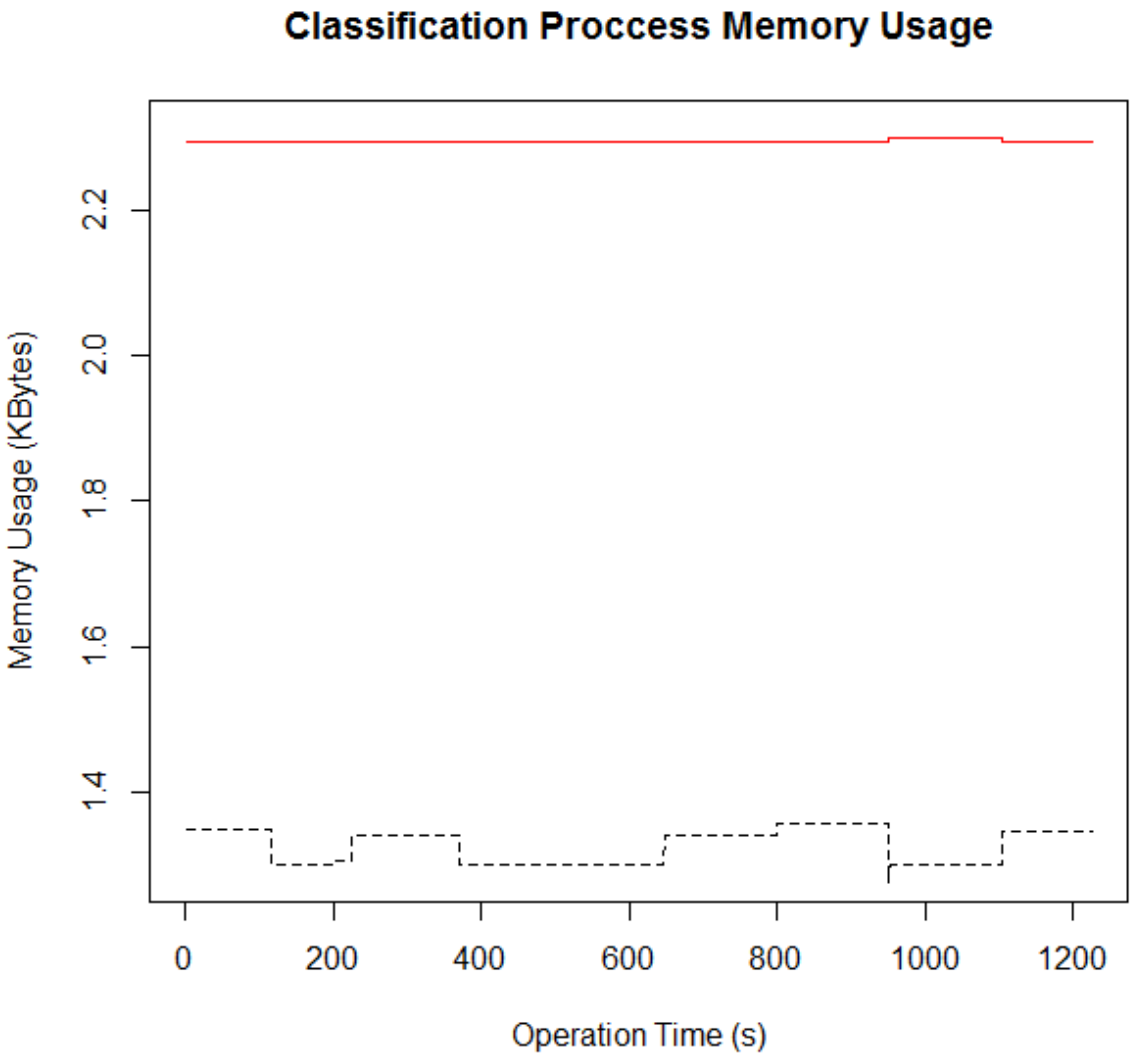

Figure 4.35: Memory consumption during operation

conclusions in the worst case scenario, the detection latency would be $E_t * (n_s - 1)$, where E_t is the execution time taken for one data source analysing one sample and n_s is the number of data sources available in the system and used for anomaly detection. Therefore theoretically on an ATMega1281 micro controller with 100 data sources we get the worst case detection latency of $82.98 * (100 - 1) = 8215.02(\mu s) = \ 8,215(ms)(Theoretically)$, the result for a Raspberry Pi would be $1.6594 * (100 - 1) = 165.94(\mu s)$ which is acceptable to a broad range of systems. As the model has been designed to be generic and ignore any environmental context thus ignoring the relationships, the detection latency can be reduced further by implementing an algorithm on multi-core CPU's. This would allow

to run the algorithm in parallel thus reducing the execution time. Another improvement that would improve detection latency is to enable multi-threading support. The current implementation of the algorithm has been done on a single core CPU and detection is achieved as a single thread on the system.

Throughout this project we were hypothesising whether it would be possible to implement an anomaly detection technique which would be capable of showing a reasonable detection performance and being able to run on the low powered micro controller. The results show that it is possible and the way the model does the detection it does not require the contextual information about the environment. However in its current state it is limited to the CAN bus and expects the raw values from the sensors which is 1 byte. If the data is in the range of 2 to 8 bytes it needs a descriptor which will convert the raw sensor value into a numeric form. The structures of information are not supported, however it is possible to describe the structure of the model and feed the data to the model. One of the examples would be that if the model received encrypted data it would not be able to operate accurately thus producing unreliable results. If the system meets the model requirements then additional testing would be required, as our experiments demonstrated.

The heuristic binary classification model supports a number of tuning parameters providing flexibility for a variety of systems such as a behavioural threshold, anomaly index thresholds, learning range, evaluation range and others. This has to be explored further as to how these tuning parameters will affect detection accuracy on other systems. At the moment this model has been tested experimentally on two robotic vehicles of which one is our testbed and is documented here. A second robotic vehicle has showed detection results based on the ROC (Area Under Curve), scoring around 0.85. The other data set that has been used for evaluation was from a commercial research project which was focused on identifying the behaviour using smart wearables and was able to identify the abnormal events in behaviour. Due to the experimental nature we have not included the performance from this data set. This shows that the model is applicable at least for robotic vehicles that have routine mission tasks. However it is not limited to robotic vehicles and can be integrated into other systems such as smart environments, potentially

in smart homes and industrial automation environments identifying the degradation of actuators or being used in similar tasks.

4.8 Using Bayesian networks to identify an attack's domain of origin

The methodology described in the previous section has focused on being sensor-agnostic. This principle allows to integrate a mechanism on a variety of robotic systems and achieve reasonably high detection rates in a restricted resource environment, as well as being flexible in terms of being used as a feedback to other intrusion detection mechanisms that work with the data in a numeric form or discreet. However, it does not have the ability to tell anything more about the nature of a threat beyond its existence. To address this, we have also followed an alternative approach using Bayesian Networks, which are frequently used in various industries such as medical[137], business[138] or various cyber security domains such as evaluating cyber-threats in smart grids[139] or evaluating cyber-security risks in nuclear instrumentation and control systems[140]. Bayesian networks can be used in statistical analysis providing a probability of events based on certain evidence. We have already described that the heuristic binary classification method is capable of identifying normal behaviour using sensor agnostic approach i.e. not taking into account the sensor context information. The methodology uses a self-learning approach to generalise the sensor data into signatures and uses these signatures as a data source's unique description that demonstrates a sensor's specifics. We have used a Bayesian network based approach because it is capable of working with discreet data, that can be in any generic form. This provides multiple paths on how to use such a model. As it can learn and create relationships between the nodes, no expert knowledge is required to identify the conditional probabilities of various events. Here, we have used the statistical analysis environment **R**, which is an open-source programming language that has access to publicly available libraries for Bayesian networks [141], neural networks[142] and many other useful tools.

Bayesian network support was provided by the **R** library **bnlearn**.

Here, we use again the same data as in Section 4.7.1. The first step is to identify the relationships between entities in the data. Researchers have published a variety of algorithms for identification of relationships between entities for Bayesian Networks which have their strengths and weaknesses. We have used the Hill-Climbing algorithm to construct a Direct Acyclic Graph (DAG). This creates all connections so that the graph does not have cycles or disconnected entities in the end. Other provided algorithms were used, including Incremental Association Markov Blanket, Max-Min Parents & Children, Tabu Search and other algorithms that are provided by the **bnlearn** library packages. The Hill-Climbing algorithm demonstrated that it is capable of generating a closed DAG taking into account all entities that are given to the Bayesian Network. The weakness of Bayesian network is that it uses Supervised learning approach, that is the data set needs to have data for the events that it is being queried. In our cases, we use a data set with **Normal**,**Cyber Attack** and **Physical Attack**. These are the events that will be queried given the evidence which are the data source heuristic binary classification output. When we train the Bayesian network, we use a 70/30 ratio for data used for learning and testing respectively.

Taking into account our learnt best detection rate of acceptable anomalies, the evaluation of Bayesian network approach has started by evaluating the best detection scenario with a later description of the weak results using different parameters. Figure 4.36 contains variety of events that the Bayesian network is being queried. The evaluated events are the detection of Normal Behaviour, Cyber Threat Behaviour and a Physical threat behaviour accompanied with a random (0,0)(1,1) coordinate line representing uncertainty.

From the ROC graph, we can observe that the Cyber Threat detection has a high AUC score that is used for evaluation, following by the Normal Behaviour detection and with a poorer detection of physical threats. This is due to a stochastic mission behaviour which is producing high amount of noise that is cancelling out the attacks themselves. However, it is still performing well and can produce high probability identification of a threat domain from the learnt data set. We have also experimented with a variety of data

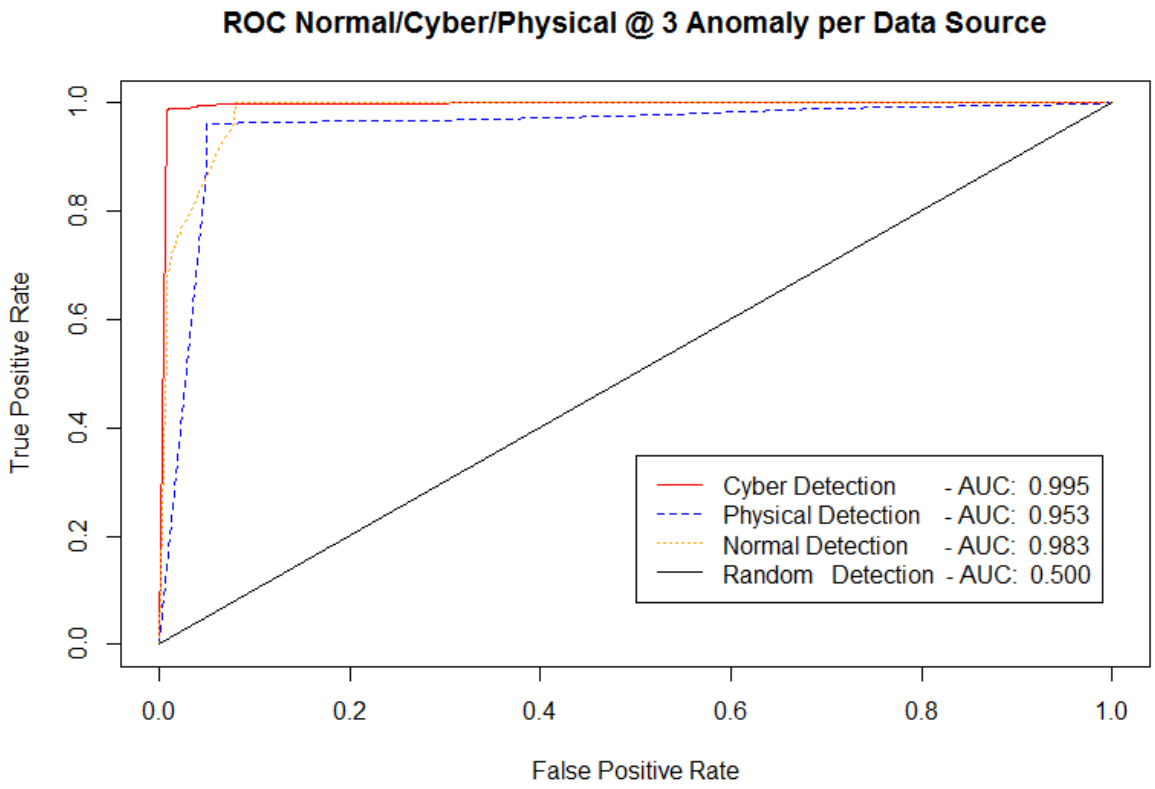

Figure 4.36: Bayesian network performance identifying cyber-physical domain threat

sources looking only at the cyber features and only physical features.

Figure 4.37 demonstrates how the threat detection is performing. It is noticeable that the performance of a cyber threat identification has not changed, however all other domain detection has decreased. This is reasonable as cyber attack vector is exposed to an attacker and an attacker will take over control of the system by manipulating cyber features that are capable to produce an impact. Detection of a normal behaviour was decreased, this shows the necessity of the monitoring the cyber features of the system when decisions are made by the system to evaluate it's behaviour. Physical threat detection show promising results as it is capable to identify physical threat with a lower confidence level based only on the cyber features meaning that it necessary to monitor the physical features as they are affected by the malicious activity of the system.

Figure 4.38 demonstrates the capability of this methodology to produce accurate probabilities by only monitoring physical features. The detection rate produces reasonably

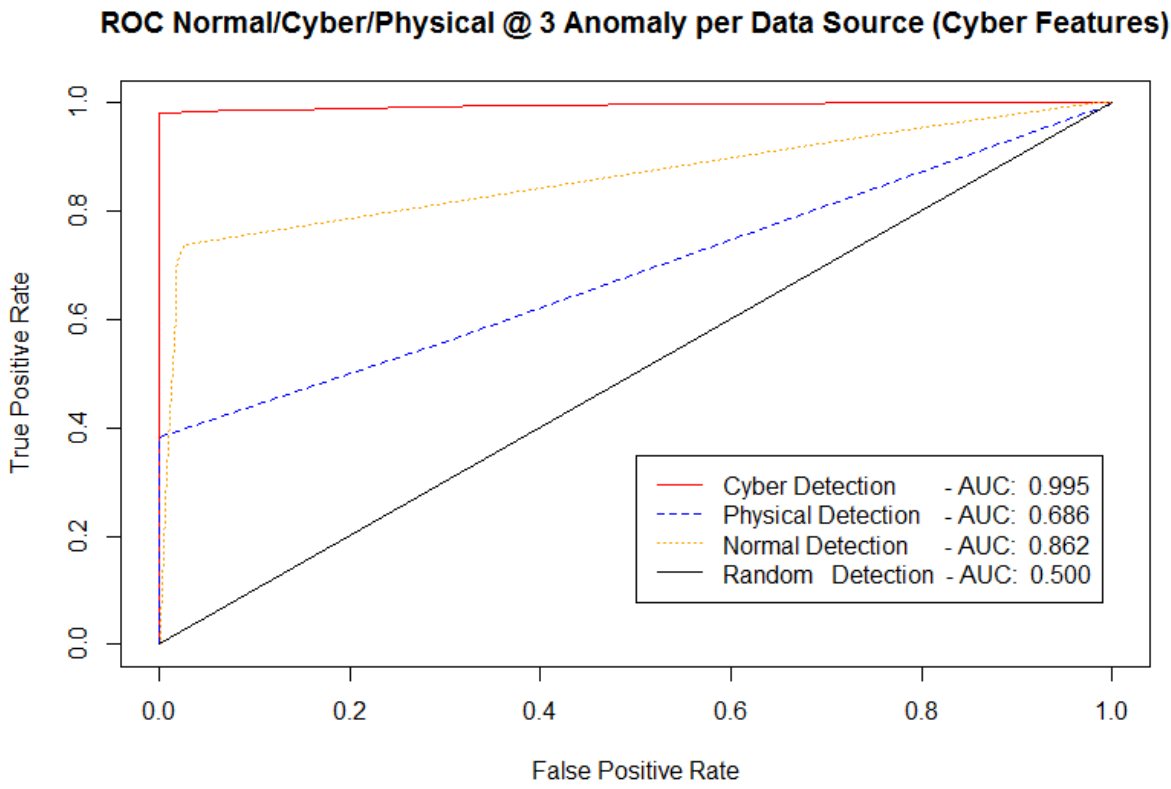

Figure 4.37: Bayesian network performance identifying cyber-physical domain threat using only cyber features

high performance for a cyber-physical domain threat detection. However, we observe an interesting fact if we compare the discussed Figure 4.38 with previous Figure 4.36 and focus on the Physical domain detection. The AUC performance score is slightly lower than the produced in the Figure 4.38. This shows that there is a potential situation when the combinations of various domain features may act as noisy evidence when probability is calculated for a specific event whether it would be probing cyber of physical threat domains. However the difference is so insignificant that the advantages of using all cyber and physical features allow to improve the identification of a normal behaviour as well as the cyber domain threats.

The key advantage of using Bayesian network is that it can accept variable range of evidence. Providing capability of producing probabilities based on several data sources with the rest keeping as unknown variables. However, the weakness of such an approach is that if less evidence is provided to Bayesian network it will require more processing

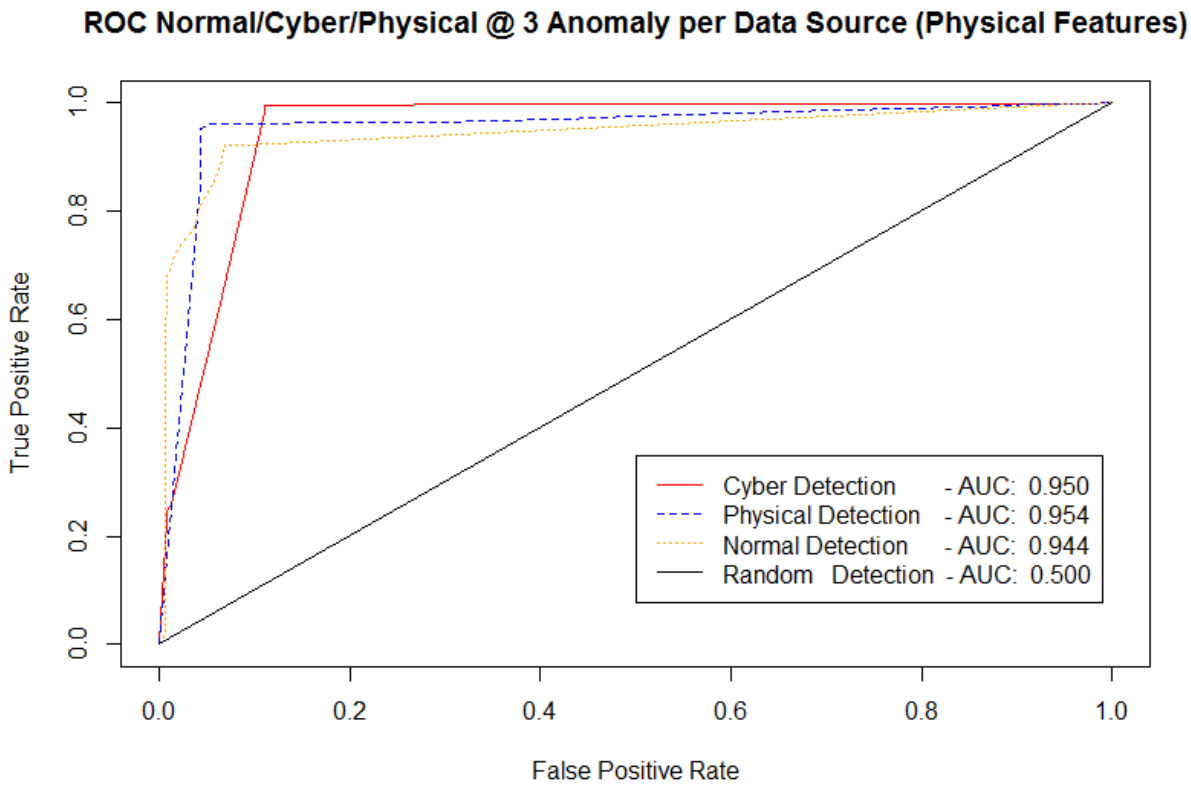

Figure 4.38: Bayesian network performance identifying cyber-physical domain threat using only physical features

power as it has to calculate the probabilities of the unknown variables. This trade off needs to be taken into account when model parameters are tuned for the system. The data set which is discussed in this section is limited to one second sampling rate taking into account latest values from the data sources. If the data will be kept in its original format this will be limited by the data reception order, as usually the data does not come in order. One example would be that the system has several components connected to a CAN bus using different IDs, this will restrict communication transmission to one data source transmission because of the CAN bus specifics as it is incapable transmitting data in parallel and data will be sent based on CAN bus prioritisation of the message ID. This will lead to a situation when the data is received in order of the data source transmission, therefore becoming event driven. The issue may arise where single data sources may have abrupt behaviour which is not reliable in probability calculation, thus producing high rate of false positive detection relying on a single data source. It is recommended to collect

multiple data source samples and make a query of Bayesian network to provide threat probability using multiple data sources.

Chapter 5

Conclusion

Cyber attacks against cyber-physical systems have become common place, whether intentional or not, and their impact can be disproportionately damaging, both in cyber and in physical space. The robotic vehicle used for this research is richly equipped, making use of a variety of common off-the-shelf technologies for sensing, control and communication, used in the industry. In addition, its design is modular, based on the generic vehicle architecture principles, so that the results of this research can be applicable and generalised to a variety of different vehicles.

We have introduced a model for observing signal characteristics, including noise level patterns on sensor data streams and have incorporated this information to characterise normal or abnormal behaviour of a robotic vehicle. The approach is behaviour-based, in the sense that the robotic vehicle first needs to learn what is normal about itself, i.e. trained in a non-attack state, and is then asked to detect different types and degrees of deviations when exposed to actual cyber attacks performed in a laboratory environment. From a data analysis perspective, this is a binary classification problem, as to whether a robot is able to detect attacks when they are ongoing and that it is in a normal state when they are not. We have addressed the problem first with a heuristic classification approach looking at a variety of features monitored in real-time on the vehicle, as well as with an approach based on Bayesian Networks. The model was designed in a way that is

sensor-agnostic, and based on our experiments it can work with a variety of sensor data. Potentially, this model supports retrofitability, but as this aspect was out of the scope of this project, this work can be explored as future work.

5.1 Key contributions

- Designed and implemented an autonomous and modular robotic vehicle testbed for cyber-physical security experimentation.
- Designed, implemented and experimentally evaluated a heuristic intrusion detection approach that allows an autonomous robotic vehicle to self-detect the existence of attacks against it in both the cyber and physical domain without prior knowledge of these attacks.
- Designed, implemented and experimentally evaluated a Bayesian network based approach that allows an autonomous robotic vehicle to self-detect attacks against it and additionally tell whether they originate from the cyber or physical domain.

5.2 Brief overall evaluation

One of this project's strongest points is that it benefited from genuine support, feedback and recommendations on real-world applicability by Dstl, the industrial funding organisation. As a result, it was driven by industrial requirements for modularity and applicability in a large variety of real-world applications. This led to the adoption of popular and commercially-relevant sensing, actuation, computation and communication technologies, of the Generic Vehicle Architecture [1], as well as of routine missions that are realistic in military, surveillance, patrol and other common applications of unmanned ground vehicles. We believe that this is a satisfactory solution to the inherent problem in cyber-physical systems, where their design (and hence the cyber and physical threats that correspond to it) is highly specific to specific types of application.

5.3 Future Work

The research carried out in this project and its outcomes have generated a number of interesting challenges that can be explored as areas of future work. Below, we include some of them:

5.3.1 Cyber-physical intrusion detection in complex missions

The scope of this project was limited to routine missions, where what is normal remains largely the same during the whole of the mission. However, in many cases, an autonomous robotic vehicle may be involved in much more complex missions, where the different component steps can differ enormously between them. In such situations, it would be advisable to introduce a dynamic state identification element to the intrusion detection system. Take the very simple example of a UAV and only three possible states: takes off or cruises. The take-off state involves several stochastic characteristics due to the effect of wind gusts, ground surface and high acceleration causing increased vibration or deviation from the expected path. In contrast, in the cruising phase, the conditions may be much calmer, hence normal values may be very different to normal values during the take-off phase. State tuning based on dynamic identification of the state would involve changing what is considered normal (e.g. the threshold values) according to the state identified by the vehicle itself.

5.3.2 Optimising Bayesian Network-based attack domain identification

In this project, we have studied the integration of the heuristic model with Bayesian Networks to identify the threat-domain causing anomalous behaviour of the system. The motivation was Bayesian Networks provide multiple classifiers for a specific purpose given certain amount of evidences. However, in some cases it is possible to end up in a situation

where certain data sources do not have any relationships with the other data sources, thus having little or no influence on the overall cyber-physical threat identification value. This increases the computation and hence power requirements as they still have to be taken into account. A potential direction here is to develop or utilise optimisation algorithms that would adapt according to the vehicle's condition and mission, still abiding by the original philosophy of being sensor-agnostic and applicable to a wide variety of cases.

5.4 Parting thoughts

The overall experience throughout the project was highly engaging, not only for the variety of practical and theoretical work involved and range of skills developed, but also because this is an area of continuously and rapidly increasing importance. When we started, there was very little prior work of relevance, but this is now certainly not the case. Several research groups globally are looking into novel approaches for detection, prevention and mitigation of cyber-physical threats. This is a result of the increasing importance of both the cyber security and cyber-physical system fields, and this research sits where the two meet. Also, and perhaps more importantly, the potential impact of cyber-physical attacks has increased dramatically in a very short period of time. In the past, where reliance of embedded computation and communication was less, the impact would have been minimal, but technological evolution is leading to a situation, where attacks against traffic lights, autonomous trains, driverless vehicles, water and power pants put people's lives in danger.

There is no doubt that we are currently experiencing a shift in thinking in terms of the security of cyber-physical systems. Security is an important requirement, while in the past in was not. However, the reality is that the vast majority of cyber-physical systems used today, from unmanned military vehicles and light railways to nuclear facilities, were designed and installed long before this shift in thinking. Effectively, they were designed at a time when there were fewer or different cyber threats. Today, integrating security by

design into a system is a standard recommendation, and this is the approach we used in this project. However, there is still a place for approaches that are able to be retrofitted into legacy cyber-physical systems.

In either case, whether following a security by design approach or aiming to retrofit security, threats evolve continuously. For example, it is now considered certain that any political or military conflict between enemy states will involve some form of cyber attack (from cyber espionage to kinetic cyber-physical attacks). In the past, the focus would be on physical attacks. Lately, the focus is on cyber attacks. This project looked at threats in general, whether they originate in cyber space and have an impact in physical space, or they originate in physical space and have an impact in cyber space. From the perspective of a cyber-physical system that exhibits a degree of autonomy, a threat can originate in either domain. This project has contributed towards reducing the impact of such threats by allowing an autonomous cyber-physical system to self-detect them and potentially identify their domain of origin.

References

[1] Flavio Bergamaschi, Dave Conway-Jones, and Nicholas Peach. Generic vehicle architecture for the integration and sharing of in-vehicle and extra-vehicle sensors. In *SPIE Defense, Security, and Sensing*, pages 76940B–76940B. International Society for Optics and Photonics, 2010.

[2] Alexander M Wyglinski, Xinming Huang, Taskin Padir, Lifeng Lai, Thomas R Eisenbarth, and Krishna Venkatasubramanian. Security of autonomous systems employing embedded computing and sensors. *IEEE Micro*, 33(1):80–86, 2013.

[3] Georgios Loukas and Gulay Oke. Protection against denial of service attacks: A survey. *The Computer Journal*, 2010.

[4] Tuan Vuong, Avgoustinos Filippoupolitis, George Loukas, and Diane Gan. Physical indicators of cyber attacks against a rescue robot. In *IEEE International Conference on Pervasive Computing and Communications*, pages 338–343. IEEE, 2014.

[5] Tuan Phan Vuong, George Loukas, and Diane Gan. Performance evaluation of cyber-physical intrusion detection on a robotic vehicle. In *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/-DASC/PICOM), 2015 IEEE International Conference on*, pages 2106–2113. IEEE, 2015.

[6] Tuan Phan Vuong, George Loukas, Diane Gan, and Anatolij Bezemskij. Decision tree-based detection of denial of service and command injection attacks on robotic

vehicles. In *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*, pages 1–6. IEEE, 2015.

[7] David Wooden, Matthew Malchano, Kevin Blankespoor, Andrew Howardy, Alfred A Rizzi, and Marc Raibert. Autonomous navigation for bigdog. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4736–4741. IEEE, 2010.

[8] George Loukas. *Cyber-Physical Attacks: A Growing Invisible Threat*. Butterworth-Heinemann, 2015.

[9] Jessica R Cauchard, Kevin Y Zhai, James A Landay, et al. Drone & me: an exploration into natural human-drone interaction. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 361–365. ACM, 2015.

[10] Periklis Charchalakis, George Valsamakis, Bob Connor, and Elias Stipidis. Milcan and ethernet. In *9 th International CAN Conference*, 2003.

[11] Dan Summers, E Stipidis, FH Ali, and P Charchalakis. Flexray–milcan bridging. In *2006 IEEE Vehicle Power and Propulsion Conference*, pages 1–4. IEEE, 2006.

[12] Ch Thiel and R König. Media oriented systems transport (most [r]) standard for multimedia networking in vehicle environment. *VDI BERICHTE*, 1415:819–834, 1998.

[13] Otto Strobel, Ridha Rejeb, and Jan Lubkoll. Communication in automotive systems: Principles, limits and new trends for vehicles, airplanes and vessels. In *Transparent Optical Networks (ICTON), 2010 12th International Conference on*, pages 1–6. IEEE, 2010.

[14] Andrzej Sumorek and Marcin Buczaj. New elements in vehicle communication media oriented systems transport protocol. *Teka Komisji Motoryzacji i Energetyki Rolnictwa*, 12(1), 2012.

[15] Suk-Hyun Seo, Jin-Ho Kim, Sung-Ho Hwang, Key Ho Kwon, and Jae Wook Jeon. A reliable gateway for in-vehicle networks based on lin, can, and flexray. *ACM Transactions on Embedded Computing Systems (TECS)*, 11(1):7, 2012.

[16] Thomas Nolte, Hans Hansson, and Lucia Lo Bello. Automotive communications-past, current and future. In *2005 IEEE Conference on Emerging Technologies and Factory Automation*, volume 1, pages 8–pp. IEEE, 2005.

[17] D Abdulmasih, PI Oikonomidis, R Annis, E Stipidis, and P Charchalakis. Operational integrity monitoring for military vehicle's integrated vetronics architecture. In *System Safety, 2011 6th IET International Conference on*, pages 1–6. IET, 2011.

[18] Karl Koscher et al. Experimental security analysis of a modern automobile. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 447–462. IEEE, 2010.

[19] Alberto Sangiovanni-Vincentelli and Marco Di Natale. Embedded system design for automotive applications. *IEEE Computer*, 40(10):42–51, 2007.

[20] Aditya Deshpande, Obowoware Obi, Elias Stipidis, and Periklis Charchalakis. Integrated vetronics survivability: Requirements for vetronics survivability strategies. In *System Safety, 2011 6th IET International Conference on*, pages 1–6. IET, 2011.

[21] A Deshpande, O Obi, E Stipidis, and P Charchalakis. Security in integrated vetronics: Applying elliptic curve digital signature algorithm to a safety-critical network protocol-ttp/c. In *System Safety, incorporating the Cyber Security Conference 2012, 7th IET International Conference on*, pages 1–5. IET, 2012.

[22] Jahshan Bhatti and T Humphreys. Hostile control of ships via false gps signals: Demonstration and detection. *Navigation*, 2016.

[23] Hugo Teso. Aircraft hacking: Practical aero series. *Hack In The Box*, 2013.

[24] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA urban challenge: autonomous vehicles in city traffic*, volume 56. springer, 2009.

[25] Jonathan Bohren, Tully Foote, Jim Keller, Alex Kushleyev, Daniel Lee, Alex Stewart, Paul Vernaza, Jason Derenick, John Spletzer, and Brian Satterfield. Little ben: the ben franklin racing team's entry in the 2007 darpa urban challenge. *Journal of Field Robotics*, 25(9):598–614, 2008.

[26] Sören Kammel, Julius Ziegler, Benjamin Pitzer, Moritz Werling, Tobias Gindele, Daniel Jagzent, Joachim Schröder, Michael Thuy, Matthias Goebl, Felix von Hundelshausen, et al. Team annieway's autonomous system for the 2007 darpa urban challenge. *Journal of Field Robotics*, 25(9):615–639, 2008.

[27] Jerome Radcliffe. Hacking medical devices for fun and insulin: Breaking the human scada system. In *Black Hat Conference presentation slides*, volume 2011, 2011.

[28] John Viega and Hugh Thompson. The state of embedded-device security (spoiler alert: It's bad). *IEEE Security & Privacy*, 10(5):68–70, 2012.

[29] Jack Barnaby. "broken hearts": How plausible was the homeland pacemaker hack?, Feb 2013.

[30] Peter Foster. Barnaby jack, one of world's best known computer hackers, dies suddenly at 35, Jul 2013.

[31] Bill Miller and Dale Rowe. A survey scada of and critical infrastructure incidents. In *Proceedings of the 1st Annual conference on Research in information technology*, pages 51–56. ACM, 2012.

[32] Jonathan Petit and Steven E Shladover. Potential cyberattacks on automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):546–556, 2015.

[33] Moez Jerbi, Sidi-Mohammed Senouci, Tinku Rasheed, and Yacine Ghamri-Doudane. An infrastructure-free traffic information system for vehicular networks. In *Vehicular technology conference, 2007. VTC-2007 fall. 2007 IEEE 66th*, pages 2086–2090. IEEE, 2007.

[34] Anatolij Bezemskij, Richard John Anthony, Diane Gan, and George Loukas. Threat evaluation based on automatic sensor signal characterisation and anomaly detection. In *Proceedings of The Twelfth International Conference on Autonomic and Autonomous Systems*, pages 1–7. IARIA, 2016.

[35] Anatolij Bezemskij, George Loukas, Richard J Anthony, and Diane Gan. Behaviour-based anomaly detection of cyber-physical attacks on a robotic vehicle. In *Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS), International Conference on*, pages 61–68. IEEE, 2016.

[36] Anatolij Bezemskij, George Loukas, Diane Gan, Richard Anthony, et al. Detecting cyber-physical threats in an autonomous robotic vehicle using bayesian networks. In *The 10th IEEE International Conference on Cyber, Physical and Social Computing*. IEEE, 2017.

[37] Robert Mitchell and Ing-Ray Chen. Behavior rule specification-based intrusion detection for safety critical medical cyber physical systems. *Dependable and Secure Computing, IEEE Transactions on*, 12(1):16–30, 2015.

[38] Rui Zhang and Quanyan Zhu. Attack-aware cyber insurance of interdependent computer networks. In *The 16th Annual Workshop on the Economics of Information Security*, pages 1–54, 2016.

[39] Claude E Shannon. Communication theory of secrecy systems. *Bell Labs Technical Journal*, 28(4):656–715, 1949.

[40] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. Experimental security analysis of a modern automobile. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 447–462. IEEE, 2010.

[41] Marko Wolf, André Weimerskirch, and Christof Paar. Secure in-vehicle communication. In *Embedded Security in Cars*, pages 95–109. Springer, 2006.

[42] John Harding, Gregory Powell, Rebecca Yoon, Joshua Fikentscher, Charlene Doyle, Dana Sade, Mike Lukuc, Jim Simons, and Jing Wang. Vehicle-to-vehicle communications: Readiness of v2v technology for application. Technical report, National Highway Traffic Safety Administration, 2014.

[43] Sachin Shetty, Tayo Adedokun, and Lee-Hyun Keel. Cyberphyseclab: A testbed for modeling, detecting and responding to security attacks on cyber physical systems. In *The Third ASE International Conference on Cyber Security*. Academy of Science and Engineering (ASE), USA, ASE 2014, 2014.

[44] Cristina Urdiales, Francisco Aguilera, Eva González-Parada, Jose Cano-García, and Francisco Sandoval. Rule-based vs. behavior-based self-deployment for mobile wireless sensor networks. *Sensors*, 16(7):1047, 2016.

[45] Robert Mitchell and Ing-Ray Chen. A hierarchical performance model for intrusion detection in cyber-physical systems. In *Wireless Communications and Networking Conference (WCNC), 2011 IEEE*, pages 2095–2100. IEEE, 2011.

[46] R Mitchell and I.-R. Chen. Adaptive Intrusion Detection of Malicious Unmanned Air Vehicles Using Behavior Rule Specifications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, PP(99):1, 2013.

[47] Adriano Fagiolini, Marco Pellinacci, Gianni Valenti, Gianluca Dini, and Antonio Bicchi. Consensus-based distributed intrusion detection for multi-robot systems. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 120–127. IEEE, 2008.

[48] Andrew T Hudak, Nicholas L Crookston, Jeffrey S Evans, David E Hall, and Michael J Falkowski. Nearest neighbor imputation of species-level, plot-scale forest structure attributes from lidar data. *Remote Sensing of Environment*, 112(5):2232–2245, 2008.

[49] Rajesh Kannan Megalingam, C Jayakrishnan, Sriraj Nambiar, Rudit Mathews, Vishnu Das, and Pramesh Rao. Automatic pressure maintenance system for tyres in

automobiles to reduce accidents. In *VLSI Systems, Architectures, Technology and Applications (VLSI-SATA), 2016 International Conference on*, pages 1–6. IEEE, 2016.

[50] Doo Seop Yun, Young-Jin Kwon, Seung-Jun Lee, et al. Development of the base-station platform for in-vehicle wireless sensor network system. In *Information and Communication Technology Convergence (ICTC), 2015 International Conference on*, pages 1180–1182. IEEE, 2015.

[51] Junguo Zhang, Wenbin Li, Ning Han, and Jiangming Kan. Forest fire detection system based on a zigbee wireless sensor network. *Frontiers of Forestry in China*, 3(3):369–374, 2008.

[52] A Vasanthara and K Krishnamoorthy. Tire pressure monitoring system using soc and low power design. *Circuits and Systems*, 7(13):4085, 2016.

[53] Tamara Bonaci, Jeffrey Herron, Tariq Yusuf, Junjie Yan, Tadayoshi Kohno, and Howard Jay Chizeck. To make a robot secure: An experimental analysis of cyber security threats against teleoperated surgical robots. *arXiv preprint arXiv:1504.04339*, 2015.

[54] Hannes Holm, Khurram Shahzad, Markus Buschle, and Mathias Ekstedt. P2 cysemol: Predictive, probabilistic cyber security modeling language. *IEEE Transactions on Dependable and Secure Computing*, 12(6):626–639, 2015.

[55] Charlie McCarthy, Kevin Harnett, and Art Carter. Characterization of potential security threats in modern automobiles: A composite modeling approach. Technical report, Volpe National Transportation Systems Center, 2014.

[56] Mohammad Hamad, Marcus Nolte, and Vassilis Prevelakis. Towards comprehensive threat modeling for vehicles. In *the 1st Workshop on Security and Dependability of Critical Embedded Real-Time Systems*, page 31, 2016.

[57] Mohamed Abomhara and GM Kien. Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks. *Journal of Cyber Security*, 4:65–88, 2015.

[58] Thomas Rid and Ben Buchanan. Attributing cyber attacks. *Journal of Strategic Studies*, 38(1-2):4–37, 2015.

[59] Kyong-Tak Cho and Kang G Shin. Error handling of in-vehicle networks makes them vulnerable. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1044–1055. ACM, 2016.

[60] Haowen Chan and Adrian Perrig. Security and privacy in sensor networks. *computer*, 36(10):103–105, 2003.

[61] Sathish Alampalayam Kumar, Tyler Vealey, and Harshit Srivastava. Security in internet of things: Challenges, solutions and future directions. In *System Sciences (HICSS), 2016 49th Hawaii International Conference on*, pages 5772–5781. IEEE, 2016.

[62] Sandor Boyson. Cyber supply chain risk management: Revolutionizing the strategic control of critical it systems. *Technovation*, 34(7):342–353, 2014.

[63] Colin Williams. Security in the cyber supply chain: Is it achievable in a complex, interconnected world? *Technovation*, 34(7):382–384, 2014.

[64] Yilin Mo, Tiffany Hyun-Jin Kim, Kenneth Brancik, Dona Dickinson, Heejo Lee, Adrian Perrig, and Bruno Sinopoli. Cyber–physical security of a smart grid infrastructure. *Proceedings of the IEEE*, 100(1):195–209, 2012.

[65] Heather M Roff. The strategic robot problem: Lethal autonomous weapons in war. *Journal of Military Ethics*, 13(3):211–227, 2014.

[66] George Loukas, Diane Gan, and Tuan Vuong. A review of cyber threats and defence approaches in emergency management. *Future Internet*, 5(2):205–236, 2013.

[67] Manuel Flury, Marcin Poturalski, Panos Papadimitratos, Jean-Pierre Hubaux, and Jean-Yves Le Boudec. Effectiveness of distance-decreasing attacks against impulse radio ranging. In *Proceedings of the third ACM conference on Wireless network security*, pages 117–128. ACM, 2010.

[68] Torry Kho, Muataz H Salih, Yan San Woo, Zheng Ng, Janice Jia Min, and Fengjie Yee. Enhance implementation of embedded robot auto-navigation system using fpga for better performance. In *Electronic Design (ICED), 2016 3rd International Conference on*, pages 309–314. IEEE, 2016.

[69] Jonathan Petit, B Stottelaar, M Feiri, and F Kargl. Remote attacks on automated vehicles sensors: Experiments on camera and lidar. *Black Hat Europe*, 11:2015, 2015.

[70] Xu Li, Rongxing Lu, Xiaohui Liang, Xuemin Shen, Jiming Chen, and Xiaodong Lin. Smart community: an internet of things application. *IEEE Communications Magazine*, 49(11), 2011.

[71] Wente Zeng and Mo-Yuen Chow. A reputation-based secure distributed control methodology in d-ncs. *IEEE Transactions on Industrial Electronics*, 61(11):6294–6303, 2014.

[72] Robert Mitchell and Ray Chen. Effect of intrusion detection and response on reliability of cyber physical systems. *IEEE Transactions on Reliability*, 62(1):199–210, 2013.

[73] Po-Yu Chen, Shusen Yang, Julie A McCann, Jie Lin, and Xinyu Yang. Detection of false data injection attacks in smart-grid systems. *IEEE Communications Magazine*, 53(2):206–213, 2015.

[74] Beibei Li, Rongxing Lu, Wei Wang, and Kim-Kwang Raymond Choo. Distributed host-based collaborative detection for false data injection attacks in smart grid cyber-physical system. *Journal of Parallel and Distributed Computing*, 2016.

[75] Fei Miao, Quanyan Zhu, Miroslav Pajic, and George J Pappas. Coding schemes for securing cyber-physical systems against stealthy data injection attacks. *IEEE Transactions on Control of Network Systems*, 2016.

[76] Gulay Oke, George Loukas, and Erol Gelenbe. Detecting denial of service attacks with bayesian classifiers and the random neural network. In *EEE International Fuzzy Systems Conference (FUZZ-IEEE)*. IEEE, 2007.

[77] Chin-Ling Chen and Hsin-Chiao Chen. A rule-based detection mechanism against distributed denial of service attacks. In *the third international conference on digital enterprise and information systems (DEIS2015)*, page 38, 2015.

[78] Muhammad Aamir and Syed Mustafa Ali Zaidi. Denial-of-service in content centric (named data) networking: a tutorial and state-of-the-art survey. *Security and Communication Networks*, 8(11):2037–2059, 2015.

[79] Abdul Quyoom, Raja Ali, Devki Nandan Gouttam, and Harish Sharma. A novel mechanism of detection of denial of service attack (dos) in vanet using malicious and irrelevant packet detection algorithm (mipda). In *Computing, Communication & Automation (ICCCA), 2015 International Conference on*, pages 414–419. IEEE, 2015.

[80] Francesco Palmieri, Sergio Ricciardi, Ugo Fiore, Massimo Ficco, and Aniello Castiglione. Energy-oriented denial of service attacks: an emerging menace for large cloud infrastructures. *The Journal of Supercomputing*, 71(5):1620–1641, 2015.

[81] Ahmad Y Javaid, Weiqing Sun, Vijay K Devabhaktuni, and Mansoor Alam. Cyber security threat analysis and modeling of an unmanned aerial vehicle system. In *Homeland Security (HST), 2012 IEEE Conference on Technologies for*, pages 585–590. IEEE, 2012.

[82] Ahmad Y Javaid, Weiqing Sun, and Mansoor Alam. Uavsim: A simulation testbed for unmanned aerial vehicle network cyber security analysis. In *Globecom Workshops (GC Wkshps), 2013 IEEE*, pages 1432–1436. IEEE, 2013.

[83] Katrina Mansfield, Timothy Eveleigh, Thomas H Holzer, and Shahryar Sarkani. Unmanned aerial vehicle smart device ground control station cyber security threat model. In *Technologies for Homeland Security (HST), 2013 IEEE International Conference on*, pages 722–728. IEEE, 2013.

[84] Eray Yağdereli, Cemal Gemci, and A Ziya Aktaş. A study on cyber-security of autonomous and unmanned vehicles. *The Journal of Defense Modeling and Simulation*, 12(4):369–381, 2015.

[85] Jeff Coffed. The threat of gps jamming: The risk to an information utility. *Report of EXELIS, Jan*, 2014.

[86] Alan Grant, Paul Williams, Nick Ward, and Sally Basker. Gps jamming and the impact on maritime navigation. *Journal of Navigation*, 62(2):173187, 2009.

[87] Hui Hu and Na Wei. A study of gps jamming and anti-jamming. In *Power Electronics and Intelligent Transportation System (PEITS), 2009 2nd International Conference on*, volume 1, pages 388–391. IEEE, 2009.

[88] Daniel Chou, Liang Heng, and GX Gao. Robust gps-based timing for phasor measurement units: A position-information-aided approach. In *Proceedings of the ION GNSS+ Meeting*, 2014.

[89] Kyle Wesson, Mark Rothlisberger, and Todd Humphreys. Practical cryptographic civil gps signal authentication. *Navigation*, 59(3):177–193, 2012.

[90] Daniel P Shepard, Jahshan A Bhatti, Todd E Humphreys, and Aaron A Fansler. Evaluation of smart grid and civilian uav vulnerability to gps spoofing attacks. In *Proceedings of the ION GNSS Meeting*, volume 3, 2012.

[91] UT Austin Researchers Successfully Spoof an $80 million Yacht at Sea, Jul 2013.

[92] Bharat B Madan, Manoj Banik, and Doina Bein. Securing unmanned autonomous systems from cyber threats. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, page 1548512916628335, 2016.

[93] Ralph Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, 9(3):49–51, 2011.

[94] TH Witte and AM Wilson. Accuracy of non-differential gps for the determination of speed over ground. *Journal of biomechanics*, 37(12):1891–1898, 2004.

[95] Michael R Moore, Robert A Bridges, Frank L Combs, Michael S Starr, and Stacy J Prowell. Modeling inter-signal arrival times for accurate detection of can bus signal injection attacks: a data-driven approach to in-vehicle intrusion detection. In *Proceedings of the 12th Annual Conference on Cyber and Information Security Research*, page 11. ACM, 2017.

[96] Stefano Nolfi and Dario Floreano. Synthesis of autonomous robots through evolution. *Trends in cognitive sciences*, 6(1):31–37, 2002.

[97] Pedro Garcia-Teodoro, J Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1):18–28, 2009.

[98] Song Han, Miao Xie, Hsiao-Hwa Chen, and Yun Ling. Intrusion detection in cyber-physical systems: Techniques and challenges. *IEEE Systems Journal*, 8(4):1052–1062, 2014.

[99] Tamara Bonaci, Jeffrey Herron, Tariq Yusuf, Junjie Yan, Tadayoshi Kohno, and Howard Jay Chizeck. To make a robot secure: an experimental analysis of cyber security threats against teleoperated surgical robots. *arXiv preprint arXiv:1504.04339*, 2015.

[100] Hamed Orojloo and Mohammad Abdollahi Azgomi. A method for modeling and evaluation of the security of cyber-physical systems. In *Information Security and Cryptology (ISCISC), 2014 11th International ISC Conference on*, pages 131–136. IEEE, 2014.

[101] Shahir Majed, Suhaimi Ibrahim, and Mohamed Shaaban. Energy smart grid cyber-threat exposure analysis and evaluation framework. In *Proceedings of the 16th*

International Conference on Information Integration and Web-based Applications & Services, pages 163–169. ACM, 2014.

[102] Adam Hahn and Manimaran Govindarasu. Cyber attack exposure evaluation framework for the smart grid. *Smart Grid, IEEE Transactions on*, 2(4):835–843, 2011.

[103] Robert Mitchell and Ing-Ray Chen. On survivability of mobile cyber physical systems with intrusion detection. *Wireless Personal Communications*, 68:1377–1391, 2013.

[104] Mark Yampolskiy, Péter Horváth, Xenofon D Koutsoukos, Yuan Xue, and Janos Sztipanovits. A language for describing attacks on cyber-physical systems. *International Journal of Critical Infrastructure Protection*, 8:40–52, 2015.

[105] Mark Mateski et al. *Cyber threat metrics*. Sandia National Laboratories, 2012.

[106] Chirag Modi, Dhiren Patel, Bhavesh Borisaniya, Hiren Patel, Avi Patel, and Muttukrishnan Rajarajan. A survey of intrusion detection techniques in cloud. *Journal of Network and Computer Applications*, 36(1):42–57, 2013.

[107] M. Strohmeier, V. Lenders, and I. Martinovic. Intrusion detection for airborne communication using phy-layer information. In *12th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*. Springer, 2015.

[108] J.S. Warner and R.G. Johnston. GPS spoofing countermeasures. *Homeland Security Journal*, 25(2):19–27, 2003.

[109] Paul Brutch and Calvin Ko. Challenges in intrusion detection for wireless ad-hoc networks. In *Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on*, pages 368–373. IEEE, 2003.

[110] Robert Mitchell and Ing-Ray Chen. Specification based intrusion detection for unmanned aircraft systems. In *Proceedings of the first ACM MobiHoc workshop on Airborne Networks and Communications*, pages 31–36. ACM, 2012.

[111] Masaaki Sato, Hirofumi Yamaki, and Hiroki Takakura. Unknown attacks detection using feature extraction from anomaly-based ids alerts. In *Applications and the Internet (SAINT), 2012 IEEE/IPSJ 12th International Symposium on*, pages 273–277. IEEE, 2012.

[112] Khattab M Ali Alheeti and Klaus McDonald-Maier. Hybrid intrusion detection in connected self-driving vehicles. In *Automation and Computing (ICAC), 2016 22nd International Conference on*, pages 456–461. IEEE, 2016.

[113] Khattab M Ali Alheeti, Anna Gruebler, and Klaus McDonald-Maier. Intelligent intrusion detection of grey hole and rushing attacks in self-driving vehicular networks. *Computers*, 5(3):16, 2016.

[114] Peter Waszecki, Philipp Mundhenk, Sebastian Steinhorst, Martin Lukasiewycz, Ramesh Karri, and Samarjit Chakraborty. Automotive electrical/electronic architecture security via distributed in-vehicle traffic monitoring. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2017.

[115] Min-Joo Kang and Je-Won Kang. Intrusion detection system using deep neural network for in-vehicle network security. *PloS one*, 11(6):e0155781, 2016.

[116] George Loukas, Yongpil Yoon, Georgia Sakellari, Tuan Vuong, and Ryan Heartfield. Computation offloading of a vehicles continuous intrusion detection workload for energy efficiency and performance. *Simulation Modelling Practice and Theory*, 2016.

[117] Andreas Theissler. Detecting known and unknown faults in automotive systems using ensemble-based anomaly detection. *Knowledge-Based Systems*, 2017.

[118] Silvia Gil Casals, Philippe Owezarski, and Gilles Descargues. Generic and autonomous system for airborne networks cyber-threat detection. In *Digital Avionics Systems Conference (DASC), 2013 IEEE/AIAA 32nd*, pages 4A4–1. IEEE, 2013.

[119] M Silva, F Pereira, F Soares, CP Leão, J Machado, and V Carvalho. An overview of industrial communication networks. In *New Trends in Mechanism and Machine Science*, pages 933–940. Springer, 2015.

[120] Carlos Santos, Manuel Mazo, Enrique Santiso, Felipe Espinosa, and Miguel Martínez. Adaptive self-triggered control for remote operation of wifi linked robots. In *ROBOT2013: First Iberian Robotics Conference*, pages 541–554. Springer, 2014.

[121] Kuo-Huang Lin, Hsin-Sheng Lee, and Wei-Ting Chen. Implementation of obstacle avoidance and zigbee control functions for omni directional mobile robot. In *Advanced robotics and Its Social Impacts, 2008. ARSO 2008. IEEE Workshop on*, pages 1–5. IEEE, 2008.

[122] V Ramya, B Palaniappan, and T Akilan. Embedded system for robotic arm movement control using web server and zigbee communication. In *International Conference on Research Trends in Computer Technologies, Proceedings published in International Journal of Computer Applications®(IJCA)(0975–8887), pg*, pages 30–34, 2013.

[123] Thoraya Obaid, Haleemah Rashed, Ali Abou-Elnour, Muhammad Rehan, Mussab Muhammad Saleh, and Mohammed Tarique. Zigbee technology and its application in wireless home automation systems: a survey. *International Journal of Computer Networks & Communications*, 6(4):115, 2014.

[124] Kechar Bouabdellah, Houache Noureddine, and Sekhri Larbi. Using wireless sensor networks for reliable forest fires detection. *Procedia Computer Science*, 19:794–801, 2013.

[125] Olaf Diegel, Aparna Badve, Glen Bright, Johan Potgieter, and Sylvester Tlale. Improved mecanum wheel design for omni-directional robots. In *Proc. 2002 Australasian Conference on Robotics and Automation, Auckland*, pages 117–121, 2002.

[126] Ministry of Defence. Defence standard 23-09: Generic vehicle architecture (gva). Technical report, UK Ministry of Defence, August 2010.

[127] Flavio Bergamaschi, Dave Conway-Jones, and Nicholas Peach. Generic vehicle architecture for the integration and sharing of in-vehicle and extra-vehicle sensors. volume 7694, pages 76940B–76940B–8, 2010.

[128] Joshua Wright. Killerbee: practical zigbee exploitation framework. In *11th ToorCon conference, San Diego*, 2009.

[129] G. Loukas and G. Oke. Likelihood ratios and recurrent random neural networks in detection of denial of service attacks. In *Proceedings of International Symposium of Computer and Telecommunication Systems (SPECTS)*. SCS, 2007.

[130] Richard John Anthony. *Load Sharing in Loosely-Coupled Distributed Systems: A rich-information approach*. PhD thesis, University of York, 2000.

[131] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *Usenix Security Symposium*, 2011.

[132] Eric Goodman, Joe Ingram, Shawn Martin, and Dirk Grunwald. Using bipartite anomaly features for cyber security applications. In *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*, pages 301–306. IEEE, 2015.

[133] Yang Liu, Armin Sarabi, Jing Zhang, Parinaz Naghizadeh, Manish Karir, Michael Bailey, and Mingyan Liu. Cloudy with a chance of breach: Forecasting cyber security incidents. In *USENIX Security*, pages 1009–1024, 2015.

[134] Anna L Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2):1153–1176, 2016.

[135] James E. Smith. Characterizing computer performance with a single number. *Communications of the ACM*, 31(10):1202–1206, 1988.

[136] Jack J Dongarra, Piotr Luszczek, and Antoine Petitet. The linpack benchmark: past, present and future. *Concurrency and Computation: practice and experience*, 15(9):803–820, 2003.

[137] Anthony Costa Constantinou, Norman Fenton, William Marsh, and Lukasz Radlinski. From complex questionnaire and interviewing data to intelligent bayesian network models for medical decision support. *Artificial intelligence in medicine*, 67:75–93, 2016.

[138] Chee Kian Leong. Credit risk scoring with bayesian network models. *Computational Economics*, 47(3):423–446, 2016.

[139] Anhtuan Le, Yue Chen, Kok Keong Chai, Alexandr Vasenev, and Lorena Montoya. Assessing loss event frequencies of smart grid cyber threats: Encoding flexibility into fair using bayesian network approach. In *Smart Grid Inspired Future Technologies: First International Conference, SmartGIFT 2016, Liverpool, UK, May 19-20, 2016, Revised Selected Papers*, pages 43–51. Springer, 2017.

[140] Jinsoo Shin, Hanseong Son, and Gyunyoung Heo. Cyber security risk evaluation of a nuclear i&c using bn and et. *Nuclear Engineering and Technology*, 2016.

[141] Marco Scutari. Learning bayesian networks with the bnlearn r package. *arXiv preprint arXiv:0908.3817*, 2009.

[142] Cristoph Norbert Bergmeir and José Manuel Benítez Sánchez. Neural networks in r using the stuttgart neural network simulator: Rsnns. *Journal of Statistical Software*, 46:1–26, 2012.