THE UNIVERSITY OF MELBOURNE

Department of Electrical and Electronic Engineering

ELEN 90081/82 Capstone Project

Final Report

Cyber Secure LEGO System

—— Supervisors ——

Margreta Kuijper

Zhanghan Tang

— Team Members —

Linan Yao, 815387

Qi Chai, 816802

Qingze Yu, 767296

June 3, 2018

Abstract

Nowadays, Cyber-Physical Systems (CPSs) are increasingly penetrating in every facet of life including engineering, manufacturing industry, transportation and so forth, which allows users or companies to efficiently and remotely manage the large-scale facilities in real time. Correspondingly, the security of CPS becomes a concern since the vulnerabilities of a CPS may lead to communication data exposure to the open network environment. Many CPSs work in autonomous manner, which involves sensors to deliver the real-time measurement as the feedback information. Therefore, sensor attacks are always the most common issue of CPSs, where cyber attackers can influence the physical devices by sensor readings.

In this project, we design and assemble a linear CPS paradigm with a Lego Mindstorms robotics kit, which simulates the working mechanism of a driverless vehicle. With this paradigm, we demonstrate the threat of sensor attacks and deliver a security solution including sensor attack detection and false data correction. We propose a detector module based on the theory of Kalman filter to identify the appearance of attack signal and a data correction algorithm to correct the attacked sensor measurement. We then build and test this security solution on the CPS paradigm to verify its validity. In the end, defects and possible improvements of this project are presented.

Keywords: Cyber-Physical System, Sensor Attack, Data Correction, Kalman Filter, Lego Mindstorms NXT.

Acknowledgement

For the completion of this one-year capstone project and associated report, we would like to express our sincere gratitude to our supervisor Prof. Margreta Kuijper for supporting us to pursue this topic, guiding us to complete this project and providing many valuable suggestions.

Additionally, we would like to show our thankfulness to PhD Candidate Mr.Zhanghan Tang for his guidance and encouragement during this whole year.

We also want to give our deepest gratitude to all teachers, colleagues and the others who gave us help.

Symbols and Abbreviations

R to denote real numbers

N to denote natural numbers

ω_{dr}——Angular velocity of right driving wheel

ω_{dl}——Angular velocity of left driving wheel

$\hat{\theta}$——Actual angle of a motor

$\tilde{\theta}$——Redundancy of an attacked sensor

θ_{st}——Angle of steering motor given by sensor

θ_{dr}——Angle of right-driving motor given by sensor

θ_{dl}——Angle of left-driving motor given by sensor

A——State matrix of state-space Representation

B——Input matrix of state-space Representation

C——Output matrix of state-space Representation

CPS —— Cyber-Physical System

DoS —— Denial-of-Service

PID —— Proportional-Integral-Differential

Contents

Abstract i

Acknowledgement ii

Symbols and Abbreviations iii

List of Figures vii

1 Introduction 1

1.1 Background . . . 1

1.1.1 Literature Survey: Application of CPSs . . . 1

1.1.2 Literature Survey: Security problems and solutions in CPS . . . 2

1.1.3 Lego Robotics system . . . 4

1.2 Aim of the Project . . . 5

1.3 Document Structure . . . 5

1.4 Mathematical Preliminaries . . . 6

1.4.1 Kalman Filter in Predictor Form . . . 6

1.4.2 Kalman Filter for Time-varying system . . . 9

2 General CPS and Cyber Attacks 11

2.1 Structure of CPS . . . 11

2.2 Attacks . . . 12

3 The Prototype Driverless Car System 14

3.1 The Introduction of Lego Mindstorms NXT 2.0 . . . 14

3.2 Building the Car System . . . 15

3.2.1 The System Structure . . . 16

3.2.2 The System Parameters . . . 19

3.2.3 Equations of the Moving Car . . . 20

3.3 Modelling the Car Motor . . . 22

3.3.1 Transfer Function . . . 22

3.3.2 Discretization of Transfer Function . . . 23

4 Controlling Method of Prototype Driverless Car 25

4.1 Controlling Method Design . . . 25

4.2 Closed-Loop Control of Steering System . . . 26

4.3 Closed-Loop Control of Driving System . . . 30

5 Sensor Attacks 31

5.1 Denial-of-Service Attack . . . 32

5.1.1 Demonstration of DoS Attack . . . 33
5.2 False Data Injection Attack . . . 34
5.2.1 Demonstration of False Data Injection Attack . . . 35

6 Detection of Sensor Attack 38
6.1 Sensor Attack Detector . . . 38
6.1.1 Detector for General CPS . . . 38
6.1.2 Detector for the prototype driverless car . . . 40
6.2 Detector Design . . . 41
6.2.1 Derivation of State-space Matrices . . . 41
6.2.2 Detector Structure of Prototype Driverless Car . . . 42
6.2.3 Initial Condition of Kalman filter for Detection . . . 43
6.2.4 Detection Threshold . . . 44
6.3 Test of Detector . . . 46
6.4 Optimisation of Detector . . . 48

7 Correction of Sensor Attack 50
7.1 Sensors Redundancy in Prototype Driver-less Car . . . 50
7.2 Correction on Constant Attack Signal . . . 53
7.2.1 Analysis and Design . . . 53
7.2.2 Test of Correction under Constant Attack . . . 56
7.3 Correction on Time-varying Attack . . . 57
7.3.1 Analysis and Design . . . 58
7.3.2 Test of Correction under Time-varying Attack . . . 60

8 Comprehensive Test 64
8.1 Test on Steering System . . . 64
8.2 Test on Driving System . . . 65

9 Discussion and Future Work 66
9.1 Defects . . . 66
9.2 Future Work . . . 67

10 Conclusion 68

References 70

APPENDICES 74
Derivation of Sensor Redundancy . . . 74

List of Figures

1 Lego Mindstorms NXT robot . . . 4
2 Kalman predictor for time-varying system [1] . . . 10
3 General structure of CPS . . . 11
4 Attack scenarios [2] . . . 12
5 General CPS with sensor attack . . . 13
6 Prototype driverless car—Assembled with Lego Mindstorms NXT 2.0 set . . . 14
7 Kit of Lego Mindstorms NXT 2.0 . . . 15
8 Prototype driverless car in the project . . . 16
9 Simple graph of the car demo . . . 16
10 Perspective of motor . . . 17
11 Differential device in a real vehicle . . . 19
12 Reduced graph of the car demo with necessary parameters . . . 20
13 The scenario of the car demo running in a straight line . . . 20
14 The instantaneous of the car demo making a turn . . . 21
15 Simulink model of motor . . . 22
16 Matched line between simulation and actual value . . . 23
17 Distribution model of unfixed sampling time . . . 24
18 Controlling method overview . . . 25
19 Position indication of three motors . . . 26
20 Steering closed-loop control . . . 27
21 Simulation result . . . 27
22 Simulation result with only P control . . . 28
23 Steering closed-loop with PID control . . . 28
24 Simulation result with PID control . . . 29
25 Block diagram of driving systems . . . 30
26 Simulation result of driving system . . . 30
27 Attack model in steering system . . . 31
28 Response of steering system to DoS system . . . 33
29 Response with False Data Injection . . . 35
30 Steering system without attack . . . 35
31 System response when attack appears . . . 36
32 Steady state with attack of 30° . . . 36
33 Attack Simulation in Driving System . . . 37
34 General CPS structure involving sensor attack detector . . . 39
35 Structure of steering system involving sensor attack detector . . . 40
36 Internal structure of detector . . . 42
37 Computing the maximum innovation value for steering system . . . 44

38 Computing the maximum innovation value for driving system 45
39 Steering system test with detector 46
40 Attack and detection test on driving system (right driving motor) 47
41 Decrease the threshold of detector by involving smoothing filter 48
42 Correction process for constant-data sensor attack 55
43 Correction for steering subsystem under constant attack signal 56
44 Correction for driving subsystem under constant attack signal 57
45 Correction process for time-varying sensor attack 59
46 Correction for steering subsystem under time-varying attack signal achieved by **Method 1** 60
47 Correction for driving subsystem under time-varying attack signal achieved by **Method 1** 61
48 Gamepad for control attack signal 62
49 Correction for steering subsystem under time-varying attack signal acheved by **Method 2** 62
50 Correction for driving subsystem under time-varying attack signal achieved by **Method 2** 63
51 General time-varying attack and defence model for steering subsystem 64
52 General time-varying attack and defence model for driving subsystem 65

1 Introduction

1.1 Background

The concept of cyber-physical system (CPS) was first introduced in 2006 and then blossomed rapidly with the development of technology and advanced control systems [3]. CPS integrates computing systems, communication networks and physical environments through 3C (Computation, Communication, and Control) technologies and forms a multidimensional, heterogeneous and complex system that can carry out real-time sensing, dynamic control and information services. The interactions between complex computing integrated systems and the physical world are the most critical parts when researching or designing CPSs. Nowadays, the CPS is more and more advanced and complex. However, the increased usage of CPS has brought more threats which could cause significant consequences. The following sections list several applications and famous security incidents.

1.1.1 Literature Survey: Application of CPSs

CPSs are used in almost everywhere around the world, such as medical systems and devices, traffic control systems, safety systems, advanced automotive systems, infrastructure control, environmental control systems and energy conservation.

A widely-used application of CPS is smart grid. In the past, a traditional electrical grid is achieved by some high-capacity equipment implemented by production as well some medium to low capacity facilities for consumers. The power consumption is increasing. However, most of the facilities are corresponding to inequality energy, which means that the management system of the grid becomes deficient. However, using the new technique of CPS in the grid provides a feasible solution by empowering the decentralised and well-organised scheme. Such solution can offer local grids multiple choices to operate the main grid according to specific situation [4].

Another application example of CPS in manufacturing was researched in 2015 [5]. In manufacturing area, a typical process includes fabrication of each part of a product, product assembly, packaging, transporting, product quality inspection and control. These processes all can be integrated and facilitated by CPSs. The efficiency of the operation is increased while the cost

is decreased. Similarly, CPS shows its advancement in all kinds of areas.

CPSs also appear in healthcare. The CPS applications can not only improve the efficiency of medical surgery but also can accurately collect patients' health conditions to control long-distance treatments. Doctors can monitor devices on patients through the networks. By obtaining and analysing real-time signals of devices, doctors can offer suitable surgeries to patients. Particularly, the CPS applied in the medical equipment can provide people using noninvasive and painless way, which has low cost and high mobility. The new embedded technology of modelling, sensing and controlling connected firmly with physical devices makes the CPS get significant progress of capability and effectiveness [4].

1.1.2 Literature Survey: Security problems and solutions in CPS

Generally, in an integrated CPS, the security problems can be divided into information security and physical system security, and both of them are very vital. The information security is to protect the characteristics of information including confidentiality, availability and integrity [6]. The physical security includes analysing control problems in the network and protecting the system from attacks [7].

Since the environment where the CPS operating is not always under the engineer's control in the real world, there are a large number of dangerous issues to destroy the CPS like cyber-criminals, disgruntled employees, faults of systems, and various attacks. For example, as stated in [8], a cyber-criminal compromised some computers of one water filtering factory in Pennsylvania and used those computers to spreading its software and increasing click-through rates of the webpage in 2006. Another famous 'insider threat' incident is that the Maroochy Shire Council's water control system in Australia was attacked by an ex-employee from a contractor company which installed council's water control system in 2000 [8].

Faults in the CPS will disable the system to operate normally and execute tasks. They can happen in any part of the system and can damage the equipment. In [9], some features of both faults and attacks are stated. Failures of some internal components will lead to unreliable operations of the CPS. However, in attack scenarios, the abnormality of the system occurs due to external injection of the attacker. In 2013, Volkan Gunes, Steffen Peter and Tony Givargis built a SIMULINK framework to simulate faults in the CPS and meanwhile the effects are analysed. A fault-tolerant CPS application is designed by comparing a number of architectures

in which redundancy is added to prevent faults [10]. The detection of faults is various. In [11], it said that the traditional fault diagnosis method cannot detect the fault in complicated nonlinear systems. Therefore, it proposes a new way to detect system faults based on wavelet transform. This approach can use the wavelet transform to detect the short circuit faults in the system. The wavelet transform based on B-spline is used to improve and denoise the fault current signal in [11], and results of this approach show practicability and efficiency in disadvantageous fault initial angles.

Among all threats resulting in unexpected circumstances and system failures, the most common concern is about attacks. Attacks on the CPS can cause severe consequences to the physical system. Similar to faults, attacks can also happen in any part of the system, so it is very difficult to predict and detect. The most common malicious attack usually happening in the CPS is called Denial-of-Service(DoS). An optimal scheduling of DoS attacks in networked control systems was described in the paper [12] in 2016, in which the attack can jam the channel of communication in a certain period, and the system stability under DoS attack was studied as well in the paper. Another type of attacks that also highly occur is False Data Injection, which is about injecting false data into the system. As we discussed before, the appearance of attacks is one of the most urgent security problems in the CPS, so solutions are brought up by a lot of researchers and learners. An excellent example of defending the system from DoS attackers is presented by Ravi Kant Sahu [13] in the vehicular network system. To prevent consequences of attacks, more than one line of defence is built so that the DoS attack is not able to disable the message transmission and retransmission in the entire network. Meanwhile, a decreasing message retransmission rate mechanism is formed along with the various defence lines. In a word, redundancy defence lines are added so that DoS attacks can be handled. Kebina Manandhar researches the way to detect the attacks including FDI in a Smart Grid System, and the Kalman Filter is used as an estimator. Also, people have designed many other types of detectors like the Euclidean detector in [14].

According to the introduction above, the security of CPSs has become a global concern. People has devoted to design and develop robust, efficient and defended CPSs [7].

1.1.3 Lego Robotics system

Figure 1: Lego Mindstorms NXT robot

In 1999, the Lego company started a robotics theme of toys called 'Mindstorms' and has been expanded and updated this branch until now. There are many products in this theme such as Lego Mindstorms 1.0, Lego Mindstorms version NXT and Lego Mindstorms version EV3. The origin of this theme is from a programmable intelligent brick developed at the MIT Lab which is the heart of these robotics sets. The set of one Lego robot contains motors, plastic wheels, many miscellaneous assembled parts and four kinds sensors which can capture action, light, ultrasonic waves and sound. This brick can be programmed by a lot of official software like MATLAB and Python. Also, this brick provides two ways Bluetooth or a USB cable to connect with a host computer.

Since this robotics theme is developed, there are many schools using this set to assemble into various shapes like a robot, bicycle, and tank as one kind of educational aids. For example, in [15], the Robotic Command Explorer which is the first version of the Lego NXT is used for demonstrating embedded systems to freshmen. This Lego Mindstorms NXT is also used for introducing the proportional-integral-differential concept of control courses using C-programming in [16]. Also, this Lego NXT is built into various robots for students to practice concepts of engineering during a lab-based course in Rheinisch-Westfalische Technische Hochschule (RWTH) Aachen University [17]. There are a lot of students and tutors participated in this course which teaches students to use MATLAB in a computer to control the Lego Mindstorms NXT. During

an eight-day period, a convenient and innovated MATLAB toolbox is developed called 'RWTH-Mindstorms NXT Toolbox' which provides a communication approach between MATLAB and the NXT robot without using other software [15]. As stated in [18], one of dramatic advantages of this Lego Mindstorms NXT is that it can be designed into different shapes and then enable people to add control implementation, mathematical concepts and visualisation aids via MATLAB. This toolbox can realise some different functions of the Lego NXT like changing the moving speed and making a turn using the manifold MATLAB to show intelligence of this Mindstorms NXT.

In our project, we use the version of Lego Robotics system called 'Lego Mindstorms NXT 2.0' to assemble a prototype driverless car, for short, we name it the car demo to explore the following content of experiment. We present the detail of the car demo in section 3.2.

1.2 Aim of the Project

This project looks at the scenario of a linear CPS, where the sensor measurement could be tampered by cyber attacks. The project will work on delivering a software security solution to achieve sensor attack detection and data correction. To demonstrate the performance of this software solution, a paradigm of a linear CPS is designed and assembled by a Lego Mindstorms robotics kit. The paradigm simulates the working mechanism and control process of a driverless vehicle. We aim to build and improve the security solution on the paradigm to ensure it can track reference trajectories of speed and angle whenever there is a sensor attack which can change the paradigm movement trajectory.

1.3 Document Structure

There are six main parts in this paper:

Part 1 (section 3 4) In section 3, we present the Lego Mindstorms NXT kit and how we design a car demo by using this kit. In section 4, we explain three main closed-loop systems which are two of driving systems and one steering system.

Part 2 (section 5) The effect of attacks is described in this section. A few most common

types of attackers are introduced as well as their mathematical modelling, regarding which the simulation is implemented in the Lego CPS.

Part 3 (section 6) In this section, we present a structure of sensor attack detector, which utilises the theory of Kalman filter. We then build and test the detector on the Lego car demo. Some improvements are also explained and made afterwards.

Part 4 (section 7) In this section, we propose an algorithm on how to correct the false data caused by sensor attacks. We mainly discuss how to run this correction algorithm under constant and time-varying attack signals. The performance of correction is assessed by testing on the Lego car demo.

Part 5 (section 8) In this section, we test the detection and correction performance under a more realistic situation, where the attack signal can be any form, and the car demo can be driven in dynamic reference speed and orientation.

Part 6 (section 9) We discuss and explain defects existing in detection and correction schemes. Then, we present several feasible improvements to counter defects.

1.4 Mathematical Preliminaries

This project focuses on realising a security solution for the prototype driverless car including sensor attack detection and correction. In Section 6, we present a method to develop a sensor attack detector which applies the theory of Kalman filter. Therefore, in this section, we briefly explain the method of using Kalman filter to make estimation of the system state. After that, we extend the knowledge of Kalman filter model for time-varying systems, which fits the prototype driverless car we assembled.

1.4.1 Kalman Filter in Predictor Form

The Kalman filter is named after the American mathematician Rudolf E. Kalman, and it can solve linear optimal filtering problems recursively in the real time, where the Kalman filter generates estimates of unknown variables, and the filtering output is derived as a weighting

combination between the previous estimate and a new measurement[19]. The predictor form of Kalman filter is analysed in [1], [20], [21], and [22]. It has been shown that the Kalman filter is the optimal minimum mean square model to estimate the state of a system. [23].
Consider a discrete time state-space model of a system below [1]:

$$x(k+1) = Ax(k) + Bu(k) + w(k) \tag{1}$$

$$y(k) = Cx(k) + v(k) \tag{2}$$

In this model, $x \in R^n$ represents the state vector. $u \in R^m$ represents the input vector and in our case u is known, which could be the output of PID controller that is sent to the actuator. $y \in R^p$ is the output vector. $w \in R^n$ is the process noise and $v \in R^n$ is the measurement noise, where both of them are Gaussian white noise with zero mean and we assume the noise w, v and $x(1)$ are uncorrelated. $A \in R^{n\times n}$, $B \in R^{n\times m}$ and $C \in R^{p\times n}$ called system matrices which reflect the system dynamics. The goal of Kalman filter is to generate a linear minimum mean square (lmmse) error estimation for state $x(k+1)$, by using the information up to time at k. The information contain input $u(1), u(2), ..., u(k)$ and sensor observation $y(1), y(2), ..., y(k)$. It is worth mentioning that $y(k)$ and $u(k)$ at each k are stored for solving general linear minimum mean estimation problem, however Kalman filter is a type of sequential linear minimum mean square estimator, so we can save much computation memory by using Kalman filter which make estimates recursively rather than using increasingly large space to store the past data [1]. We define some notations as follows:
1. The state estimate $x(k+1)$ made by given data up to k: $\hat{x}(k+1|k)$;
2. The output estimate $y(k+1)$ made by given data up to k: $\hat{y}(k+1|k)$;
3. The state estimate error and state covariance matrix:

$$\delta(k+1) = x(k+1) - \hat{x}(k+1|k) \tag{3}$$

$$P(k+1) = Cov(\delta(k+1)) \tag{4}$$

we define initial P given by:

$$P(1) = E[x(1)x^T(1)] \tag{5}$$

4. Define the innovation signal as:

$$e(k) = y(k) - \hat{y}(k|k-1) \tag{6}$$

$$e(k) = C\delta(k) + v(k) \tag{7}$$

which reflects the difference between the estimate and the real sensor measurement [21]. The basic idea of Kalman filter for estimating the new state and output is to combine the previous estimation $\hat{x}(k|k-1)$ and the new sensor measurement $y(k)$ with a weighting factor $K(k)$, the

Kalman gain, and the estimate equation is given by:

$$K(k) = AP(k)C^T(CP(k)C^T + V)^{-1} \tag{8}$$

$$\hat{x}(k+1|k) = A\hat{x}(k|k-1) + Bu(k) + K(k)e(k) \tag{9}$$

$$\hat{y}(k|k-1) = C\hat{x}(k|k-1) \tag{10}$$

and equation 9 can be rewritten as:

$$\hat{x}(k+1|k) = A\hat{x}(k|k-1) + Bu(k) + K(k)(y(k) - C\hat{x}(k|k-1)) \tag{11}$$

where V is called measurement noise covariance [1]:

$$V = E[v(k)v(k)^T] \tag{12}$$

and from equation 8, we can see $K(k)$ is determined by $P(k)$, thus $P(k)$ should be updated since δ is also varying and the update equation of $P(k)$ is shown in [1] as:

$$P(k+1) = AP(k)A^T + W - AP(k)C^T(CP(k)C^T + V)^{-1}CP(k)A^T \tag{13}$$

and W is called process noise covariance [1]:

$$W = E[w(k)w(k)^T] \tag{14}$$

Equation 8 and 9 give a reasonable estimation of state $x(k)$. The Kalman gain gives a 'confidence weighting factor' [24]. According to this factor, the filter decides to depend more on whether model $A\hat{x}(k|k-1) + Bu(k)$ or measurement $y(k)$ to get the new estimate $\hat{x}(k+1|k)$. For the car demo, we apply the predictor form of Kalman filter to the steering system and two driving system respectively. And in the closed-loop control manner, we use the computation of Kalman filtering for every sample of each subsystem, and we look at the innovation signal $e(k)$. If $e(k)$ is greater than a threshold value we set, an attack alarm will be triggered. This is how we involve Kalman filtering to detect the sensor attack for both steering and driving motor. Because Kalman filter is shown to be able to make the innovation converge and usually it can converge very fast. Thus a significant $e(k)$ reflects a significant inconsistency between model and measurement [20]. And a false feedback data may result in fatal and irreversible damage in the CPS.

1.4.2 Kalman Filter for Time-varying system

To deploy the Kalman filter for sensor attack detection, the state-space model for the device is needed. In this project, the devices are three servo motors for three subsystems, thus establishing the state-space model, modelling and conducting discretisation are essential stages. It is known that with different sampling period, the model will have different dynamics. Thus to use a time-invariant system model for Kalman filtering, we have to guarantee the sampling interval between each sample from sensor is fixed. In section 3.3.2, we present the problem that there exists an inherent uncertain delay of querying the servomotor's sensor. And we can only use a mean value of sampling interval as a sampling period to discretise the model. Strictly speaking, we cannot directly use the discrete-time model derived in section 3.3.2 to build the Kalman filter, since indeed the sampling interval of the angular sensor of the Lego NXT servomotor is not fixed. This means for every two consecutive samples, the discrete-time models of the servomotor could be different, as well as the state-space matrices. Hence, with an inaccurate model, the innovation signal may not converge so that we cannot get an optimal estimate. Instead, we consider using a dynamic model to fit the model of the servomotors. In this section, we review how to generalise the Kalman filter in prediction form for a time-varying system.

Consider a linear time-varying system:

$$x(k+1) = A(k)x(k) + B(k)u(k) + w(k) \tag{15}$$

$$y(k) = C(k)x(k) + v(k) \tag{16}$$

As we can see, the only difference from the time-invariant system is that the matrices of system model are changed over time. The system model is determined by timing the sampling intervals between consecutive samples. Then the system matrices $A(k)$, $B(k)$, and $C(k)$ are derived by transferring the discrete model to state-space model.

Since the process noise and measurement noise are uncorrelated in the project, thus the covariance matrices are in the same format as the time-invariant case [23].

$$Cov(\begin{bmatrix} v(k) \\ w(k) \end{bmatrix}) = \begin{bmatrix} V(k) & 0 \\ 0 & W(k) \end{bmatrix} \tag{17}$$

The equations for Kalman filter of time-varying model are given by [1]:

1. State estimation and innovation:

$$\hat{x}(k+1|k) = A(k)\hat{x}(k|k-1) + B(k)u(k) + K(k)e(k) \tag{18}$$

$$e(k) = y(k) - C(k)\hat{x}(k|k-1) \tag{19}$$

2. Kalman Gain:

$$K(k) = A(k)P(k)C^T(k)(C(k)P(k)C^T(k) + V(k))^{-1} \tag{20}$$

3. Update of state error covariance:

$$P(k+1) = A(k)P(k)A^T(k) + W(k) - K(k)(C(k)P(k)C^T(k) + V(k))K^T(k) \tag{21}$$

The initialisation scheme is:

$$\hat{x}(1|0) = x_1 \tag{22}$$

$$P(1) = E[x(1)x^T(1)] \tag{23}$$

A block diagram illustrating the procedure of a time-varying Kalman filter is given by figure 2.

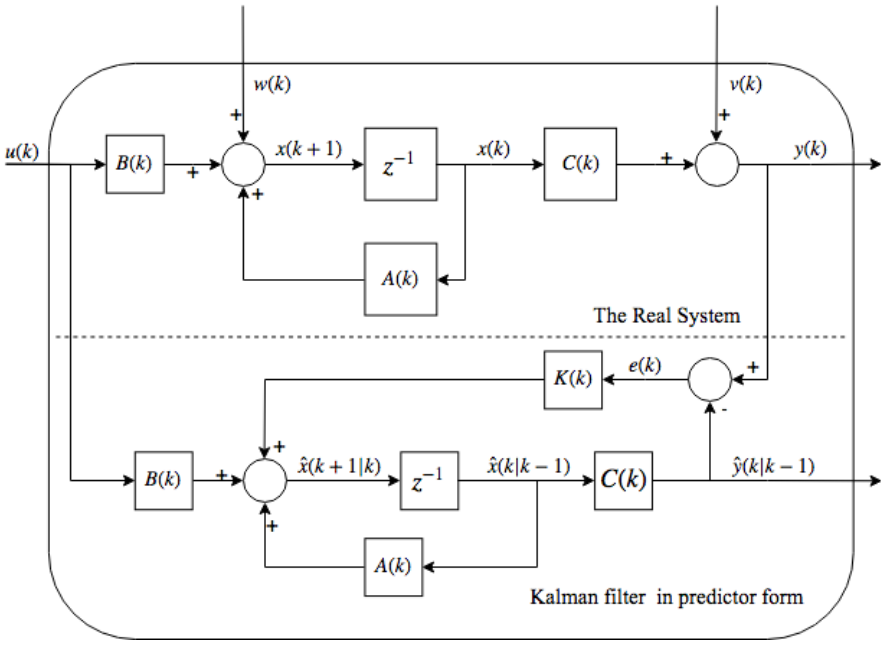

Figure 2: Kalman predictor for time-varying system [1]

The sensor attack detection method is to look at the innovation $e(k)$, which is the difference between the Kalman estimate and the sensor reading at each corresponding time point. Once the innovation is larger than a preset threshold, an attack alarm will be triggered.

2 General CPS and Cyber Attacks

2.1 Structure of CPS

As in figure 3, an integrated CPS includes the cyber control systems, communication networks and physical systems.

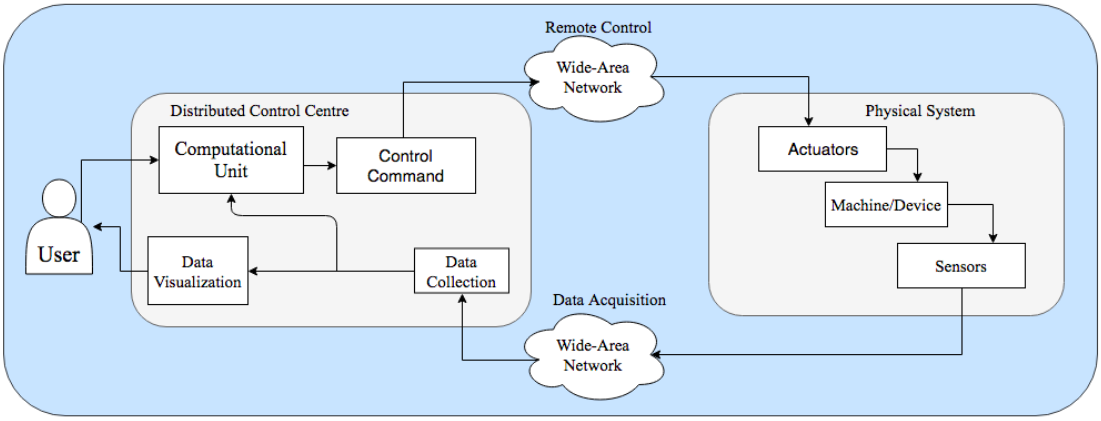

Figure 3: General structure of CPS

The right part of figure 3 is the physical systems, where there are actuators, devices or machines and sensors. The actuator is one of the most important components in a machine. A control signal will be transmitted to the actuator, and the actuator will execute the corresponding task. The control signal can be certain kinds of signals, such as electrical signals, pneumatic pressure and so on. The energy of the signal is converted into mechanical motion by the actuator. The sensor will send the data out to the control centre. The communication system is the data link between the control centre and physical components. The control signal will be sent through the communication channel to enable remote controlling, and the data generated from the sensor will be transmitted and collected by the control centre. The left part is the cyber control terminal. The data transmitted from the sensor through communication link will be collected by the data collection mechanism and stored. Meanwhile, the data will be sent to the computational unit as the feedback signal, which will work on computation and the give the control command based on the computation result. Computation can be conducted automatically or controlled by users. The data can also be visualised and presented to the user.

2.2 Attacks

There are two types of methods of solving security problems in a Cyber-Physical System. One is called information security, which is focusing on the information protection, encryption and data security. The other one is secure control theory. It is more about dealing with the physical dynamics of control systems attacked by cyber-attacks leading to severe consequences, which is also what we are focusing on in this project. In a general integrated control system, information security and secure control theory are both included in the protection system [25]. The secure control theory is focused in this paper, so it is necessary to introduce and explain the sensor attacks that might threat our the car demo system. We explain the features of some sensor attacks in section 5 and demonstrate the corresponding influence on the car demo. There are two main types of attacks existing in cyber-physical systems, which are denial-of-service(DoS) attacks and deception attacks. Attacks can come in the system in any stage of a control system. As described by Saurabh Amin in 2009, the figure 4 can livelily show almost all the possible scenarios in a control system [2].

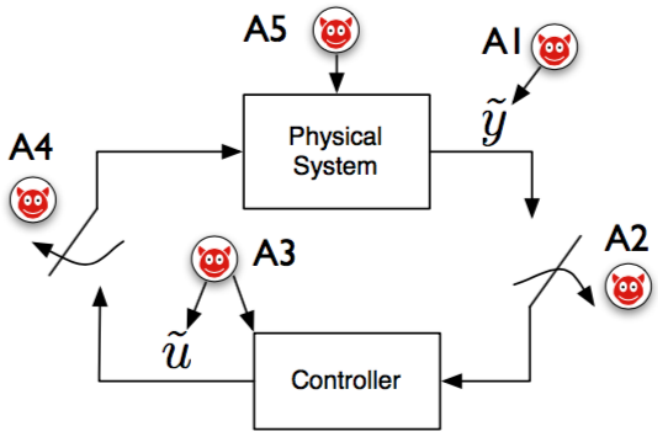

Figure 4: Attack scenarios [2]

In figure 4, $A1$, $A2$, $A3$, $A4$ and $A5$ represent the possible attacks might threat a CPS. y is the real output value from the sensor of the physical system, and u is the real value from the controller, which are not noted in figure 4. $A1$ is a deception attack causing a false data output from the sensor in the system, where the output from sensor $\hat{y}$ is not equal to the real value y. $A3$ has the same effect, and the output from the controller is influenced so that $\hat{u}$ is not the actual value of u. $A2$ and $A4$ here are all DoS attacks (Denial-of-Service). The links between the physical system and the controller are disrupted, which are indicating that entities of communication are prevented by the adversary. DoS attacks can enable the communication channels to lose the ability to send data and affect the system functionality by compromising the

devices in the system. DoS attacks are researched and discussed with a lot of effort nowadays. $A5$ directly attack the actuator in the physical system. The process of actuator operation is impacted by this attack, which leads that the whole system is affected [2].

In this project, we only consider the situation where the output of sensor as feedback is attacked as attack types $A1$ and $A2$ indicated in figure 4. In our CPS, we can model the attack signal as in figure 5. An attack is assumed already successfully hacked into the control systems and can influence the sensor reading of the real physical device, machine or infrastructure. Specifically, the scenario we researched in this project is that the attacker tampers the sensor so that sensor shows us the false output. When the reading results from the sensor are not correct, the user may lose the knowledge of the true behaviour of the car. For sensor security problems, DoS attacks and False Data Injection are explicitly explained in section 5.

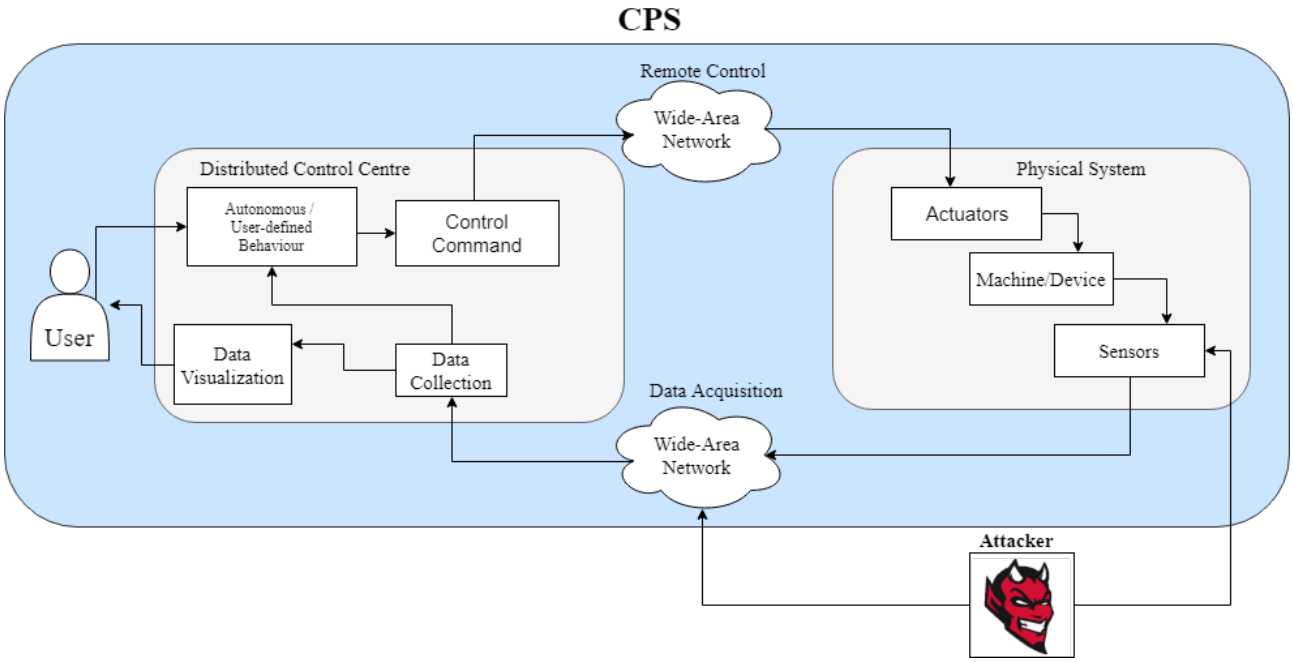

Figure 5: General CPS with sensor attack

3 The Prototype Driverless Car System

Having recognised the importance of the CPS security, we need to choose a proper CPS model to demonstrate the security problem and seek for a solution. According to [26] and requirements of this project, there are four examples of the need: the system should (1) operate in a familiar software environment, e.g., MATLAB or Python; (2) perform a linear (at least) second-order dynamical behaviour; (3) be available for replacement and maintenance; (4) be relatively cheap (in the budget). Considering students are familiar with MATLAB and Simulink, it will be an advantage if the system used in control system experiments interacts with MATLAB and Simulink in real-time [26]. Therefore, the Lego Mindstorms NXT 2.0 gives a valid solution to satisfy all needs as mentioned above. The prototype driverless car is assembled from the Lego Mindstorms NXT 2.0. After designing, programming and controlling of this car demo, it can provide a platform for studying issues of security in the CPS. In this section, we briefly describe The Lego Mindstorms NXT 2.0 at the beginning, then illustrate the car system structure and give the transfer function of the motor.

3.1 The Introduction of Lego Mindstorms NXT 2.0

From the figure 6 and 7, it can be seen components of designing the car demo are all from the Lego Mindstorms NXT 2.0 kit. Therefore, first presenting Lego Mindstorms NXT 2.0 is an acceptable way to introduce this car demo.

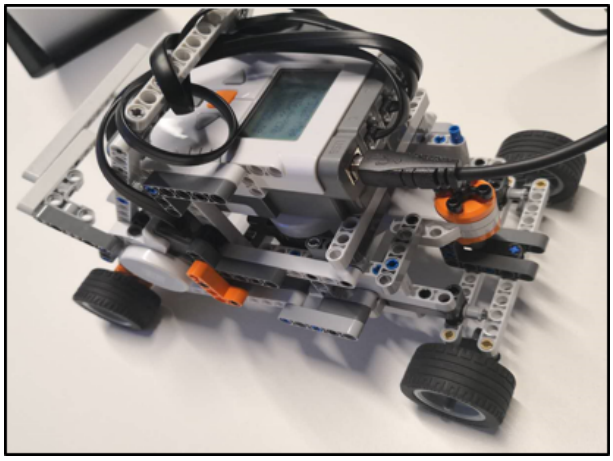

Figure 6: Prototype driverless car—Assembled with Lego Mindstorms NXT 2.0 set

Figure 7: Kit of Lego Mindstorms NXT 2.0

The Lego Mindstorms NXT 2.0 is a programmable robotic kit. This kit has one NXT intelligent brick, four sensors which are touch, light, sound and ultrasonic sensor, three motors and some other parts like plastic gears and wheels. The most main component of this set is the NXT intelligent Brick which has a brick shape and four buttons under the LCD screen. This brick offers an interface between a host computer and various components like motors. Correspondingly, there are two ways the wireless Bluetooth and a USB cable for users to connect this NXT with the host computer.

There are various ways of construction the Lego MINDSTORMS NXT 2.0. In the project, it is built into a prototype driverless car, and this car demo uses the Lego MINDSTORMS NXT toolbox developed by RWTH Aachen University as development code package in MATLAB. With the help of this toolbox, compiled programs of MATLAB can be transferred into the car demo via USB cable, which can directly control this car and retrieve data from sensors to the host computer for analysing the car demo movement.

3.2 Building the Car System

In this part, the report focuses on describing the structure of this prototype car demo which imitates real vehicles in life. Also, the report analyses several significant movement scenarios of this car demo to get relationships and equations existing in a moving car.

3.2.1 The System Structure

The car demo uses the NXT intelligent brick, three motors, a USB cable, and other miscellaneous parts for assembling the car. Two motors are placed symmetrically at the rear part of the NXT brick, and one motor is put in front of the brick like a steering device in a real vehicle. Each motor is connected with one wheel except for the front motor which connects with two wheels so that this structure can ensure the stability of this car demo. The figure 8 shows the car demo used in this project.

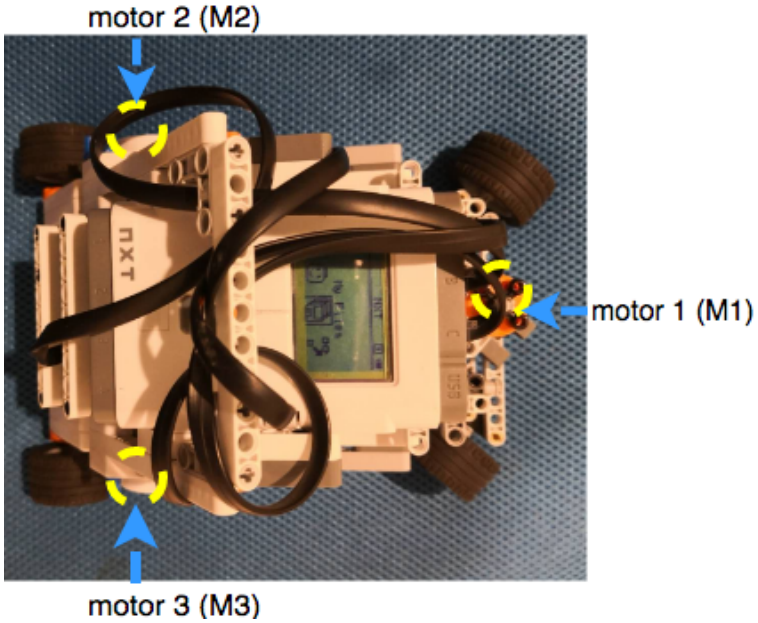

Figure 8: Prototype driverless car in the project

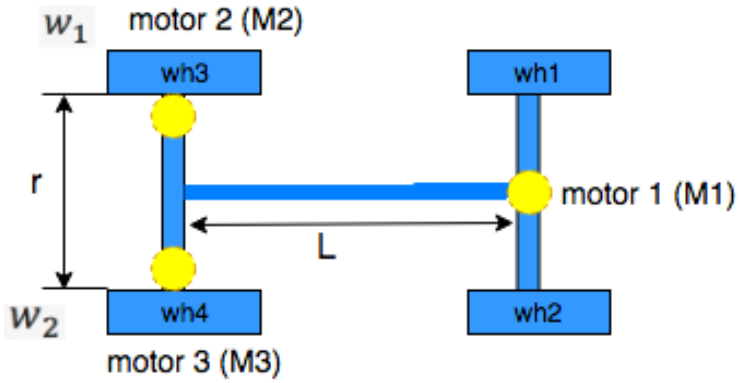

Figure 9: Simple graph of the car demo

(1) In the car demo, the main component NXT intelligent brick takes information from sensors and control motors via MATLAB in the host computer which directly interacts the car

demo. This brick has a high monochrome LCD screen and four buttons which can be used for navigating an interface with hierarchical menus. It has two processors: 32-bit microprocessor programmed by users and 8-bit co-processor used as a servo driver, 64KB RAM, 256KB of flash memory and Bluetooth support. Its power is supplied by two ways which are 4*1.5V batteries and a lithium-ion rechargeable battery [27].

(2) The servo motor in kit contains a gearbox, a built-in encoder and a DC motor. There are three servo motors in the car demo shown in figure 9. Three motors of the car are identical, but they perform diverse functions in accordance with different inputs and references given by users. Therefore, we determine one of these three motors performing as a steering motor M1 and the rest two motors work as driving motors M2, M3. The steering motor is placed in the front of the car and two wheels wh1 and wh2 connected with it control the car make a left or right turn with an angle. Meanwhile, the two rear driving motors are connected with two wheels wh3 and wh4 and control the car move forwards and backwards. The motor of the car is power at 9V, and this voltage value can be changed to control the velocity of the motor. However, one point needs to be concerned is that the speed value is decided by Pulse Width Modulation which is in the range from -100 to +100 [28]. In other words, the voltage value from -9V to 9V given to motor corresponds to number range -100 to +100 in programming command which decides the speed of the motor. The negative sign represents making the motor move backwards.

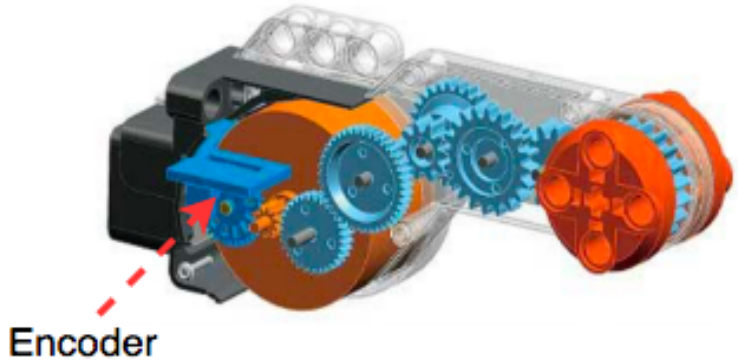

Figure 10: Perspective of motor

(3) The rotary encoders which are built in motors are worked as sensors. There are two main reasons for this design style in this project. The first one is that main content of the project is mainly related to movement and position of the car demo, so the sensor is required to record parameters reflecting the speed and position of the car. Fortunately, the encoder can sense and retrieve its rotation with 1-degree resolution and this information can determine the motor speed and position [29]. The other one is that the encoder is a built-in component of the motor

which is shown in figure 10 [29]. Using the encoder as a sensor can reduce the load of the whole car compared with adding additional sensors, which can decrease the friction between wheels with the ground.

(4) There are two ways to connect the host computer with this car demo which are a USB cable and Bluetooth. As the description in [18], it points that latencies are smaller using USB connections than using Bluetooth approach. Therefore, the USB cable is used instead of the wireless Bluetooth. Additionally, this car demo imitates a real car in life, so it is expected to respond quickly and have less processing time.

The equipment specific to this project include:

(1) hardware: the Lego NXT intelligent brick, three Lego motors and miscellaneous parts in the Lego Mindstorms NXT 2.0

(2) software: MATLAB and RWTH- Mindstorms NXT Toolbox

Although the structure of this car demo imitates a real vehicle in life, it is worth mentioning that there are several differences on the driving system between this car demo and a real vehicle on design structure. Firstly, the driving motor controls two rear wheels together in a real vehicle, while in this project the car demo has two driving motors respectively controlling two rear wheels. Another difference is about the differential. The differential shown in figure 11 is a mechanical device existing in the rear axle of a real vehicle which is a gear train with three shafts. The differential device works when the car is making a turn, and it can determine two different velocities of rear wheels, which ensure the car to steady and safely make a turn. However, this car demo distributes two different angular velocities to two rear wheels for making a turn. Despite these differences mentioned above, when the real vehicle and this car demo are turning, extension cords of steering wheels and rear wheels both intersect to the same point, and there are two concentric circles as shown in figure 14.

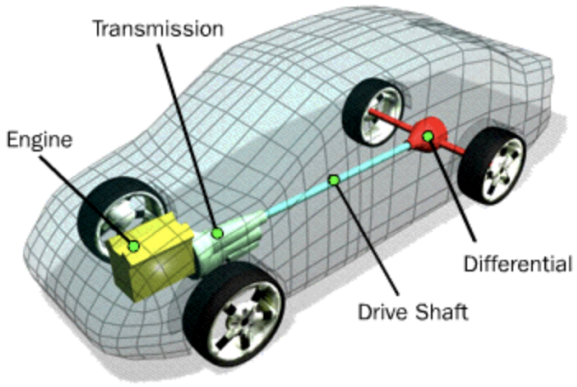

Figure 11: Differential device in a real vehicle

3.2.2 The System Parameters

Through measuring and defining main parameters of the car demo used in this project, the following values and labels are shown in figure 12:

the length of rear axle: r=15.12cm (the same length of the front axle)

the length of transmission shaft: L=19.07cm

the angular velocity of wh3: $\omega 1$ (depends on actual input)

the angular velocity of wh4: $\omega 2$ (depends on actual input)

It needs to be known that the angular velocity of wh3 and wh4 is the angular velocity of driving motors rather than the speed of wheels relative to the ground.

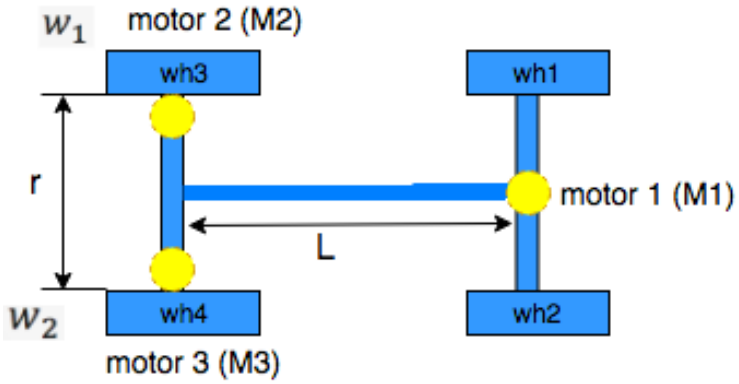

Figure 12: Reduced graph of the car demo with necessary parameters

3.2.3 Equations of the Moving Car

During the movement process of a car, there are several equations among four wheels. When motors are powered, and the brick is compiled by the designed program, the car can move in a specific line. There are two different scenarios of movement:

(1) when the car demo driven in a straight line

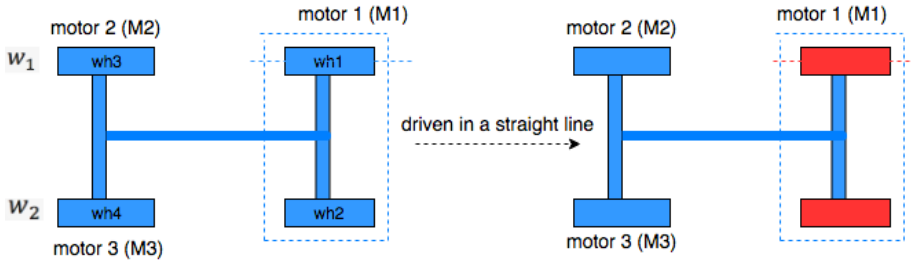

Figure 13: The scenario of the car demo running in a straight line

Since the sensor can send and retrieve angle, the wheels are given the angular velocity [17] for collecting data conveniently in sensors. When the car is driven straight as shown in figure 13, the two driving wheels wh3 and wh4 move together with the steering wheels wh1 and wh2 in the same angular velocity. The steering angle is zero. During this situation:

$$\omega 1 = \omega 2 \tag{24}$$

(2) when the car demo made a specific degree turn

Comparing with driven in a straight-line scenario, the situation of the car demo swerving is more challengeable and complicated. Wheels are placed on each side of the car. Therefore, they have different angular velocities (the angular velocity of the outside wheel is larger than the angular velocity of the inside wheel) when the car performs arc movement rather than going straight, which is modelling the differential device existing in a real vehicle.

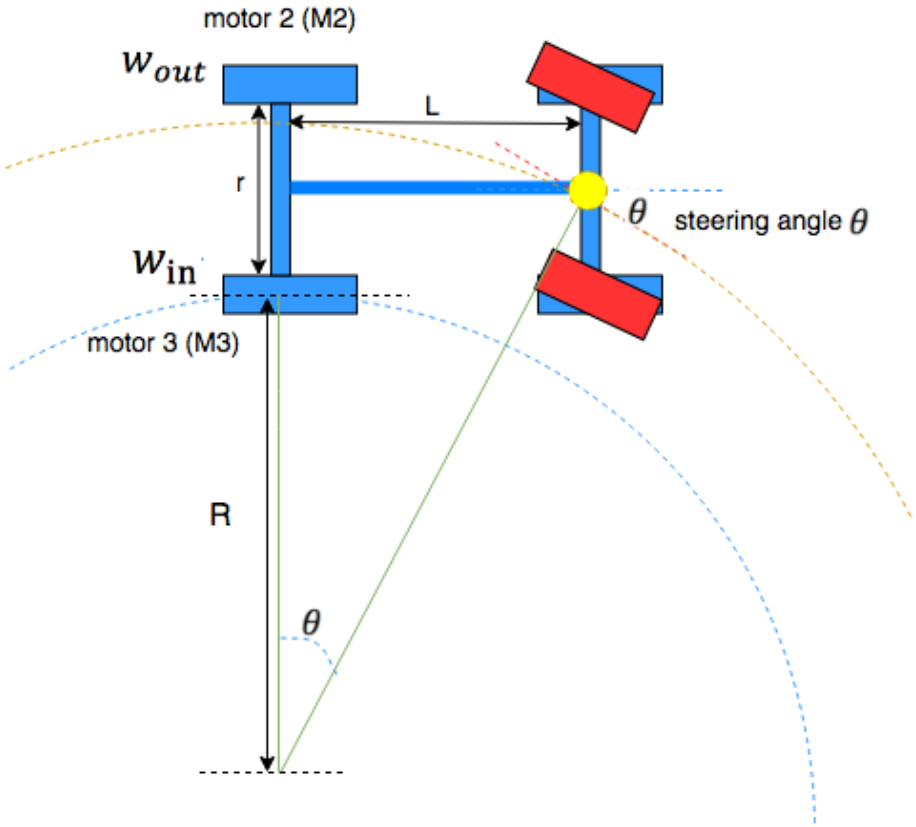

Figure 14: The instantaneous of the car demo making a turn

In steering situation, for avoiding drag and skid, the steering motor and the inner wheel connected with M3 are moving in two concentric circles. In these two concentric circles, extension cords of the rear inner wheel and the steering motor should intersect at the centre of these two concentric circles shown in figure 14 [30]. By analysing two concentric circles, following relationships can be conducted:

$$\frac{\omega_{out} * r_1}{\omega_{in} * r_1} = \frac{\omega_{out}}{\omega_{in}} = \frac{R+r}{R} \tag{25}$$

$$tan\theta = \frac{L}{R + \frac{1}{2}r} \tag{26}$$

$$\frac{\omega_{out}}{\omega_{in}} = \frac{\frac{L}{\tan\theta} + r/2}{\frac{L}{\tan\theta} - r/2} \tag{27}$$

In the equation, ω_{in} is the inner wheel angular velocity, ω_{out} is the outside wheel angular velocity, θ is the steering angle, and r_1 is the radius of a wheel. The car demo has four identical wheels so r_1 can be removed from the equation 25.

3.3 Modelling the Car Motor

3.3.1 Transfer Function

As mentioned before, the Lego Mindstorms NXT 2.0 has been used in many educational experiments for several years. Almost in every experiment, although the NXT is designed into various shapes like a bicycle or a robot, they are all expected to actuate by motors. Therefore, there are many models of motors given by previous users. According to a physical model of the motor in [31] and mathematical analysis of a linear DC motor model in [32], a transfer function of a typical NXT motor is described in the following form:

$$G(s) := \frac{\theta(s)}{V(s)} = \frac{K_m}{s(T_m s + 1)} \tag{28}$$

where $\theta(s)$ represents the turning angle of the motor, $V(s)$ represents the applied voltage, K_m represents the system gain, T_m represents the time-constant parameter and assumed having zero initial condition. Then, the built simulation in figure 16 can be used to find matchable parameters K_m and T_m by comparing the output of the motor with data from the motor encoder. In the following figure 15, the input of the transfer function is given by the product of voltage and square wave generator, which imitates changeable input values of the motor. After several trials, from the figure 16, it can be seen that Simulink Servo Model and data from Motor Encoder are well matched in the Simulink scope, which K_m=8.835 and T_m=0.055.

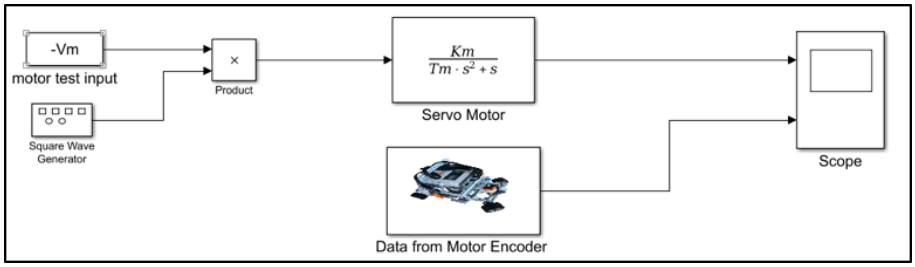

Figure 15: Simulink model of motor

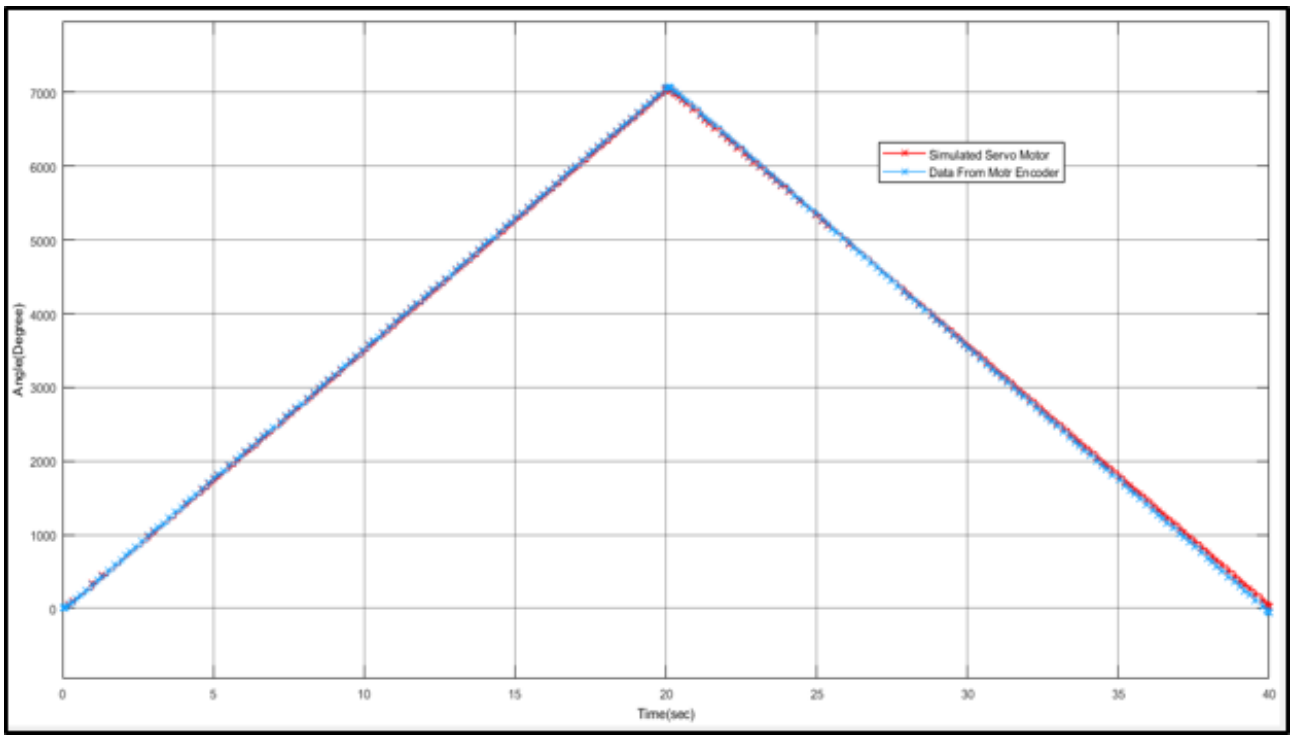

Figure 16: Matched line between simulation and actual value

Therefore, the transfer function is conducted:

$$G(s) := \frac{8.835}{s(0.055s+1)} \tag{29}$$

3.3.2 Discretization of Transfer Function

Since angle values processed by sensor and controller are discrete, so the transfer function needs to be discrete for analysing in the following sections. Before that, the sampling time is a priority. However, there is a problem that the sampling time is not fixed. The reason is that between host computer and NXT communication, the function of sending and receiving data is respective, and there is a 30ms latency during the process of switching transmit mode to receive mode in the Lego NXT intelligent brick, so it can be expected a 60ms latency in a whole process of sensor completing a request [33]. In [33], it also states that one important issue of NXT intelligent brick is that it might lose packets and commands during working because the size of input queue is limited. As users do not know any details about its size, the transmitting and retrieve functions have an option to choose whether they wait between subsequent send commands or not. If functions want to wait, it demands several milliseconds to wait between two successive send operations. Fortunately, the NXT brick is intelligent and it only pauses execution when functions are necessary. Users will not realise the automatic waiting time as it

is not essential if the brick is only required to send a packet in every second. In order to fix this unstable sampling time, we model the sampling time distribution. It can be seen that a standard normal distribution is shown in figure 17 after 10000 times sampling and mean value of the distribution is 0.0189s (18.9ms) which is used as the sample time for discretisation.

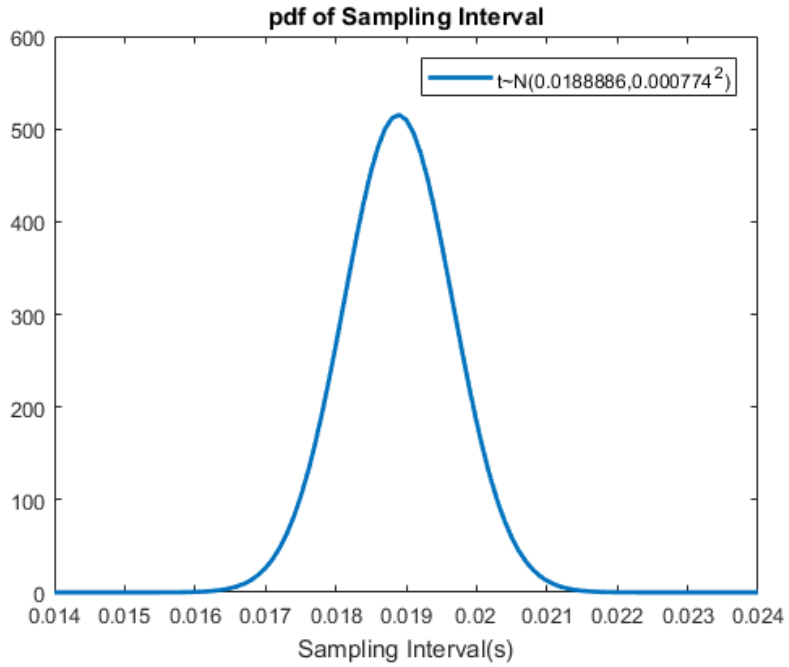

Figure 17: Distribution model of unfixed sampling time

The discrete time transfer function

$$G(z) = \frac{dis_{num}(z)}{dis_{den}(z)}$$

where $dis_{num}(z)$ and $dis_{den}(z)$ can get from a MATLAB command such as c2d

$$G(z) := \frac{0.02593z + 0.02311}{z^2 - 1.708z + 0.7079} \tag{30}$$

4 Controlling Method of Prototype Driverless Car

Over the previous pages, three subsystems are introduced including a steering system and two driving systems. It is essential to choose the method to effectively control systems so that the car can perform well in moving along the desired trajectory. In this section, we will first introduce the overall car demo system controlling method and then explain the specific closed-loop design and controller design for each system.

4.1 Controlling Method Design

Our system has two input reference signals as shown in figure 18. One of them is the reference steering system. It will be directly sent to the steering system, which will decide the direction of the car when moving. When the car is moving in a straight line, the reference steering angle is set to zero. When the car is turning left, we define the steering angle is larger than zero. when the car is turning right, we define the steering angle is smaller than zero. The other input reference signal is defined as the angular velocity of the outside wheel when the car is making a turn. Obviously, when the car is going in a straight line, the input reference input angular velocity will be directly sent to both driving subsystems because the left wheel and the right wheel will have the same angular velocity. We define the driving subsystems as one right driving system and one left driving system and they include their own motor in the corresponding position of the car, which are indicated in figure 19.

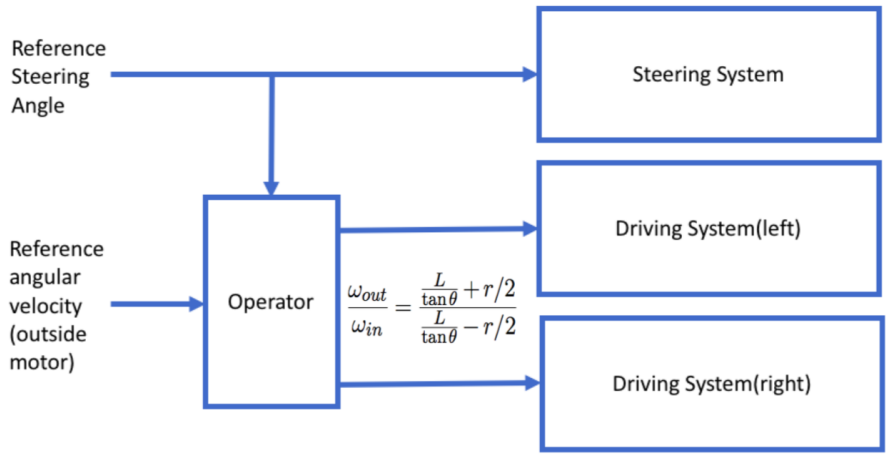

Figure 18: Controlling method overview

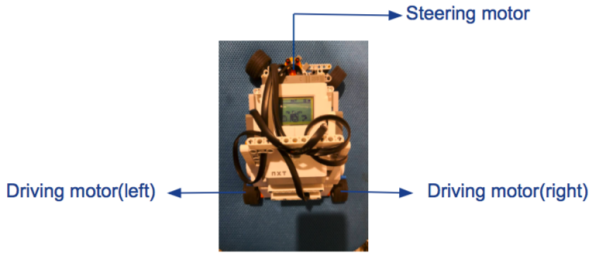

Figure 19: Position indication of three motors

We have the following rules to decide which motor is the outside motor. All these comparison and calculations will be conducted in the operator indicated in figure 18.
1) When the input reference steering angle is zero, we decide the car is moving in a straight line. In this case, the reference angular velocity will be sent to both right motor and left motor.
2) When the input reference steering angle is positive, we decide the car is turning left and the right motor is the outside motor. In this case, the reference angular velocity will be sent to the right motor. And the angular velocity of the left motor will be calculated by equation 31, which we derived before.

$$\frac{\omega_{out}}{\omega_{in}} = \frac{\frac{L}{\tan\theta} + r/2}{\frac{L}{\tan\theta} - r/2} \tag{31}$$

3) When the input reference steering angle is negative, we decide the car is turning right and the left motor is the outside motor. In this case, the reference angular velocity will be given to the left motor. For the right motor, the angular velocity can be calculated by the same equation 31.

After the operator figured out the angular velocities of both right motor and left motor, it will send the two angular velocities to each motor respectively as their reference inputs, which is also indicated in the controlling system structure diagram shown in figure 18.

4.2 Closed-Loop Control of Steering System

In the previous paragraphs, we have described the overall controlling method of our car demo. If we want the car to moving along the desired trajectory with a defined speed, then for each subsystem, we want the output to be almost equal to the corresponding reference signal. For all subsystems, using closed-loop system is necessary. In a closed-loop control manner, the output will be sent to the input and compare with the input signal, and the difference will

be sent to the controller to adjust the system so that the system output can avoid departing from the desired trajectory due to variations of the process dynamics and disturbances. In a word, closed-loop control can strongly stabilise the system by enabling output signal to track a reference signal closely. Furthermore in our case, closed-loop control contributes to the further steps of the project, which are the attacker modelling, detection and reconstruction.

The steering closed-loop system was initially designed as in figure 20, where the data link is simplified as just a line connecting control centre and steering system.

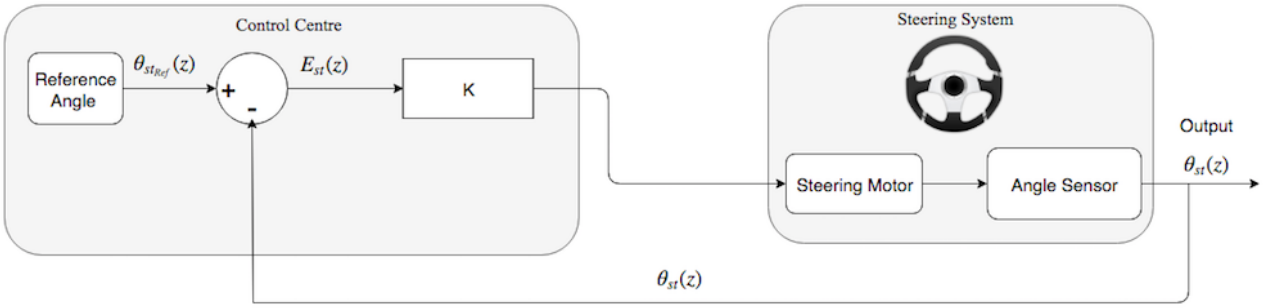

Figure 20: Steering closed-loop control

As explained before, the system is digitalized. The input of the system is the reference steering angle denoted by $\theta_{st_{Ref}}$ and the output is θ_{st} given by sensor. After reading the output value from the sensor, θ_{st} will be sent to input and the difference between the input and the output, denoted as E_{st}, is sent to plant. K is the gain in the closed loop.

When the reference angle $\theta_{st_{Ref}}(z)$ is 40°, we simulated the system and got the step response of the system as in figure 21.

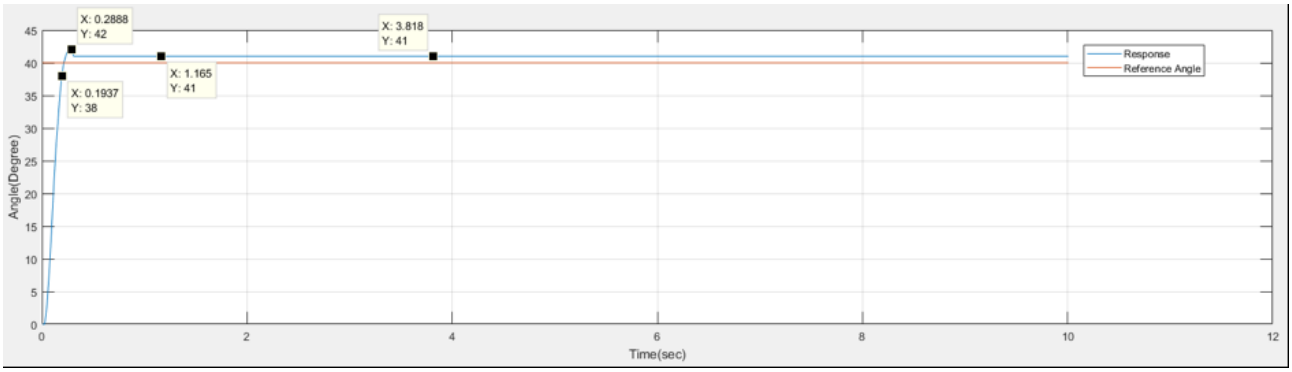

Figure 21: Simulation result

In figure 21, the red line is the reference angle we input to the system, and the blue line is the step response of the closed-loop. It is obvious that a relatively large overshoot appears and the value of the output in steady state is not closely tracking the input. This will lead to some errors that cannot be ignored. Another circumstance is that when we only use proportional control, for some gain with certain values, the response of the systems cannot really reach the input reference signal so that the steady state error is quite nonnegligible as shown in figure 22.

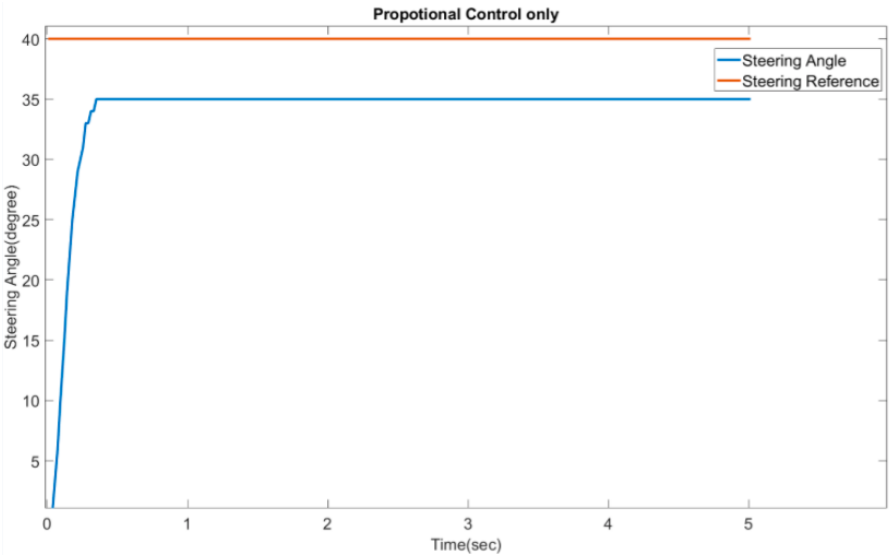

Figure 22: Simulation result with only P control

Fortunately, this problem can be solved by adding PID controller in our system. The theoretical block diagram of steering system with PID controller is shown in figure 23.

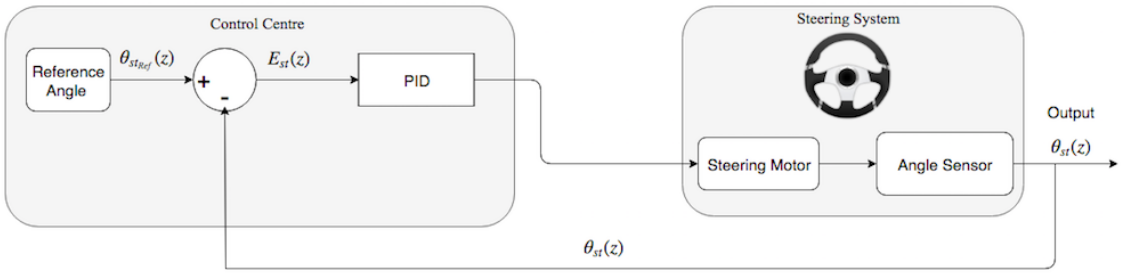

Figure 23: Steering closed-loop with PID control

PID control consists of three different types of control including proportional control, integral control and derivative control. It is known that PID control can help to control the plant even the precise characteristics of the plant are not fully known [34].

The proportional control can help the system to respond quickly with a shorter transient. The derivative control can solve the problem of large overshoot we encountered, and integral control can reduce the steady state error.

In this project, a zero steady-state error is desired. Firstly we implemented PI control. However, after adding derivative control, the system showed less error. Therefore, a PID controller is chosen [34].

For a PID control, the function should be written by equation 32.

$$u(t) = K_p(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}) \tag{32}$$

The discrete PID control can be expressed by equation 33.

$$u(t) = K_p(e(t_k) + \frac{\delta t}{T_i}\sum_{i=1}^{k} e(t_i) + \frac{T_d}{\delta t}(e(t_k) - e(t_{k-1}))) \tag{33}$$

We obtained the values of K_p, T_i and T_d by tuning in the experiments to achieve the optimal performance of closed-loop step response.

Set the reference steering angle to 40°, and we obtained the result as in figure 24.

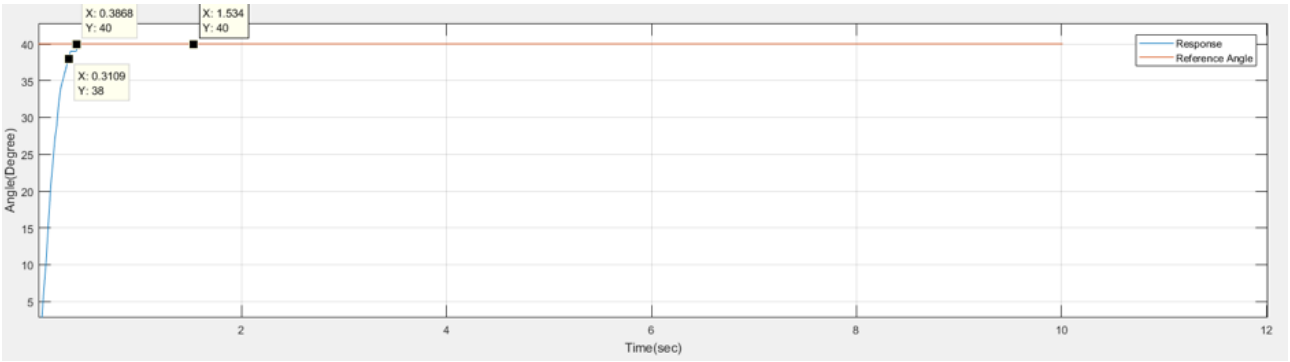

Figure 24: Simulation result with PID control

It can be seen that after adding properly tuned PID controller, the overshoot problem is solved. The response has less steady state error and is closely tracking the reference input value.

4.3 Closed-Loop Control of Driving System

Closed-loop systems are also implemented in two driving subsystems in our project. The two driving systems have the same characteristics and same controller. Ideally, when the car is moving in a straight line, two driving motors will operate in the almost same way. The only difference is that when the car is moving in a circle or changing direction, two wheels will rotating in different angular velocities so that two sensors in these two subsystems will read out different values of angular velocity. In this section, we will take the right motor as an example to explain the driving systems.

The theoretical block diagram of the driving system is in figure 25. Here the input is the angular velocity $\theta_{dr_{Ref}}(z)$. However the sensor can only read the angle, so we need to convert the angular velocity into angle $\theta_{dr_{Ref}}(z)$ as in the first block. Similarly, as discussed in the steering system, we also designed PID controller in driving systems.

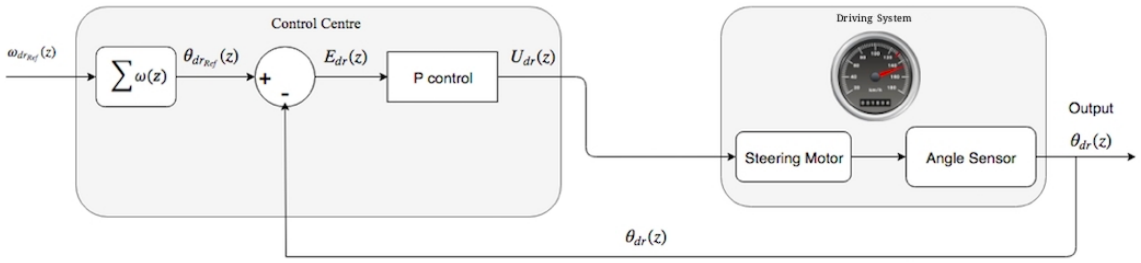

Figure 25: Block diagram of driving systems

After simulation, we can plot the output diagram as in Figure 26. The red line is the reference angle we put in the system after integrator and the blue line is the response from the output.

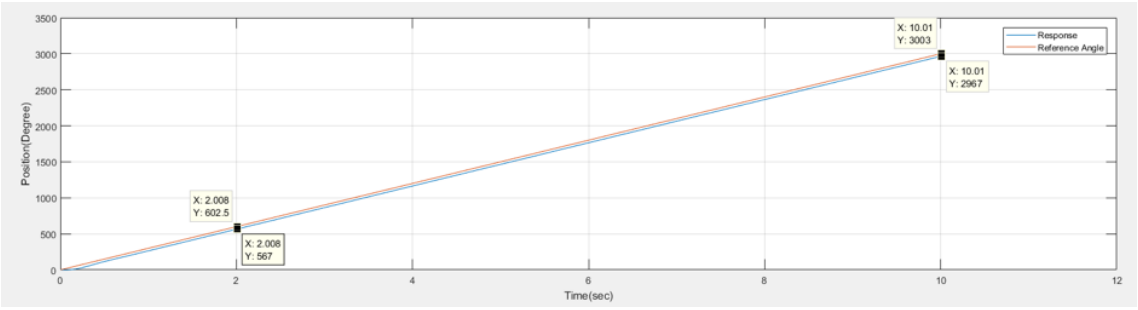

Figure 26: Simulation result of driving system

It can be seen the output is not strictly same as the input signal as the car moving. However, For instance, when the time point is at 2.008, the difference between input and output is

approximately 35.5°. When time reaches around 10, the difference is almost 36°, which means the output is tracking the reference signal with some fixed errors. The response is lagging behind the reference signal, which means it is always lower than the reference signal. This is reasonable, because if in certain time point the output goes beyond the reference signal, the corresponding motor will rotate reversely leading to skidding of the car.

The other driving system has almost the same simulation results because the motors of these two driving systems are the same, so we do not need to repeat here. The relationship of angular velocities of these two wheels was discussed before.

5 Sensor Attacks

As discussed before, in this project, we only focus on the scenarios where the attacks will attack the sensor so that the sensor will give us false reading results. Also, we assume that only one of three sensors will be attacked, this is because if we have more than one sensor attacked, then when we try to reconstruct the attacked system, we cannot work out the real correct output of attacked motor by using redundancy information. Three subsystems have the same sensor attack modelling method. For instance, we can draw the diagram of the steering system as in figure 27 with modelled attack signal which is noted by $a_{st}(z)$. The real output signal is $\hat{\theta}_{st}(z)$, which is then attacked by the attacker, and the feedback sent back as sensor output is denoted by $\theta_{st}(z)$, hence:

$$\theta_{st}(z) = \hat{\theta}_{st}(z) + a_{st}(z) \tag{34}$$

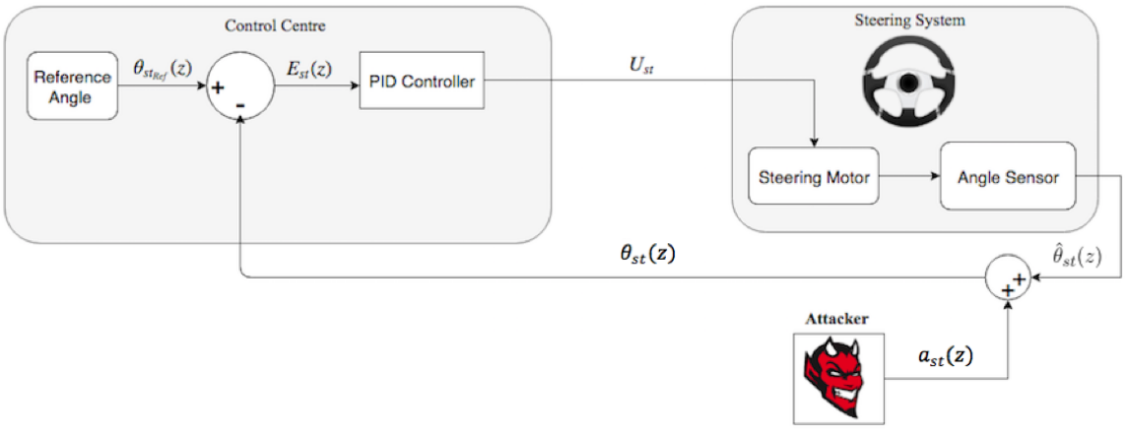

Figure 27: Attack model in steering system

5.1 Denial-of-Service Attack

Denial-of-Service(DoS) is one form of attacks that can disable computers or network systems so that they cannot offer their service. The DoS attackers always have access to the network resources while the services are disrupted. This type of attacks can lead to the lack of resources. For a cyber system, no matter how fast the processing speed is, how huge memory capacity it has or what its network bandwidth is, the consequences of this attack are really difficult to avoid.

For our driverless car model, the DoS attack scenario can be understood as the block brought by the attacker that will make the master computer temporarily or indefinitely lose the knowledge of the measurement of sensors, which is specifically the angle at the real time. In our model, lacking of information from the sensor is easy to detect since the computational unit can instantly realise the difficulties of retrieving the data from sensors. For generality, in this paper, we model the DoS attack as the measurement suddenly drops to zero, which is similar to the experiment done by Manandhar, Cao, Hu and Liu [14].

The attack model of the DoS attack can be described as:

$$y(k) = a(k) * \hat{y}(k) \quad k \in N \tag{35}$$

where $y(k)$ is the data delivered to the server, $\hat{y}(k)$ is the real sensor measurement. Here in this model, the measurement noise in the system is not taken into consideration.
For attack signal, we have its mathematical model like:

$$a(k) = \begin{cases} 0, k \in [k1, k2] \\ 1, otherwise \end{cases} \tag{36}$$

Where $k1$ and $k2$ represent the time points of DoS attacks' appearance and leaving time respectively. It is clear that during the existence of DoS attack, we lose the knowledge of sensor measurement, which means that no matter what the input is, the reading of sensor will be almost zero.

5.1.1 Demonstration of DoS Attack

According to the description of the DoS attack, in our automobile example, to simulate the situation when the DoS attack is launched, we can simply fix the sensor reading to zero for any of the motors. The following plots illustrate the results of DoS attack of steering motor of the car demo.

Set the reference angle to 20°, and then the DoS attack launches after 5 seconds. We know that when no attack appears, the reading of the sensor will track the input signal 20°. When there is an attack, we can plot the response of the system in figure 28, in which the blue line is the sensor measurement. The actual angle is in red, and the input reference angle is in yellow.

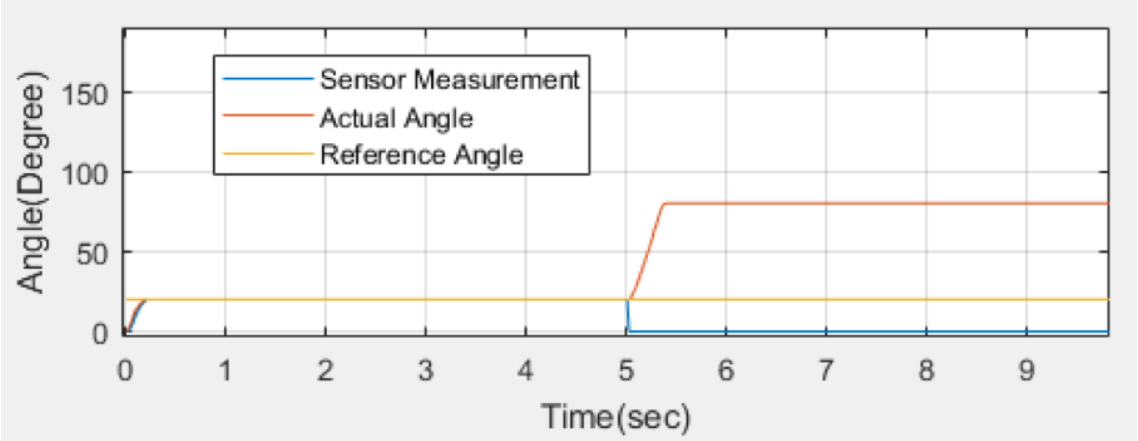

Figure 28: Response of steering system to DoS system

Within the first 5 seconds when the DoS attack is not launched yet, the actual angle and the sensor measurement are strictly tracking the input reference angle. When the attack comes in, the actual angle increases while the sensor measurement value changes to almost zero. The reason why the actual value reaches to almost 80° first and then stay at 80° is because the maximum angle of steering is 80°. Otherwise, the actual value will increase without stop.

We did not implement the DoS attacks in the driving motor here, because the voltage put into the motor will easily go beyond the maximum voltage (9V) at which the motor can operate. The reason is that in the driving system, the input of the controller is the angle of the motor, which is increasing while the car is driven. However, the DoS attacks will make the sensor reading result close to zero, for which the error signal will getting larger and larger because it is the difference between the reference signal and feedback signal (sensor reading). Hence, the input of the motor will be increasing quickly after we launch the attack signal into the system

and finally go beyond the maximum voltage. In this case, the motor will report an error and stop working quickly. In a real vehicle with a very high maximum working voltage, the speed of the car will keep increasing if we launch the DoS attacks into its driving system. In a word, the attacks disable the car to control its speed.

5.2 False Data Injection Attack

False Data Injection attack has been discussed in many papers and mostly been understood as an additive attack that the attack can manipulate the measurement of the sensor with adding a series of value at different time nodes thus alter the data delivered to the server. Furthermore, False Data Injection can be classified into many types by the characteristics of the data injected. It is shown that for different types of False Data Injection attacks, the level of difficulties for detection is different [35].

The attack model of False Data Injection attack can be described as:

$$y(k) = \hat{y}(k) + a(k) \quad k \in N \tag{37}$$

where $a(k)$ represents the attack signal and is a function of time axis k. $y(k)$ is the data delivered to the server and $\hat{y}(k)$ is the real sensor measurement.

The function of attack signal $a(k)$ can be different according to different aims of the attacker. In our case, when we model False Data Injection attacks, the amplitude of $a(k)$ should be chosen to be less than the maximum value of the motor output. For instance, for our steering system, if the amplitude of the false data is very large and the attacked sensor measurement is larger than the maximum steering angle 80°, the system will quickly report an error and stop working. Similarly in driving system, if the amplitude of the attack signal is too large, the voltage input of the driving motor will be larger than the maximum operating voltage of the motor. In this case, the system will report an error. In short, attack signals in our system should be modelled with certain amplitude so that it will not break down the system but impact the behavior of the system.

5.2.1 Demonstration of False Data Injection Attack

To demonstrate the influence of False Data Injection attack, we define the behaviour of an attack of steering sensor as in equation 38.

$$a(k) = \begin{cases} 30°, k \in [5s, 10s] \\ 40°, k \in [15s, 20s] \\ 0, otherwise \end{cases} \tag{38}$$

We also set the reference angle to 20°. The simulation result is plotted as in Figure 29.

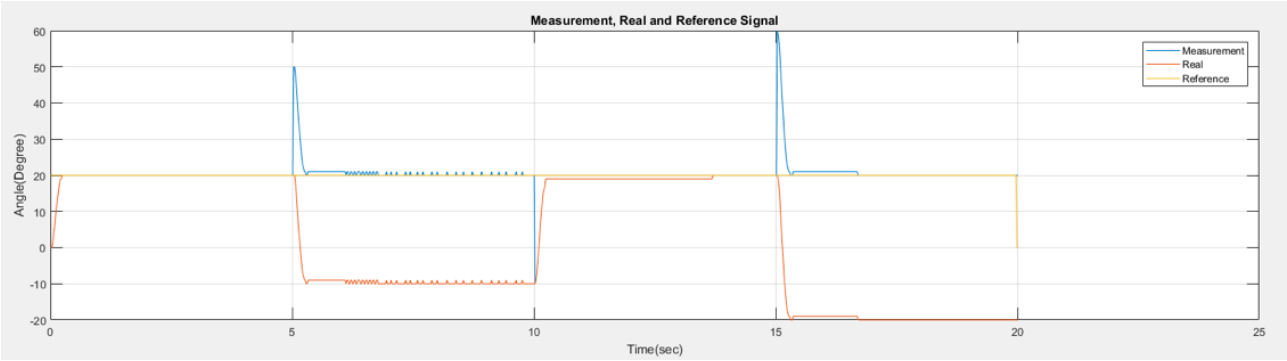

Figure 29: Response with False Data Injection

To explain more clearly how the False Data Injection affects our system, the process of data changes involving attack signal in the steering system is explained with figures from figure 30 to figure 32. The reference value of steering angle is 20°, and a constant attack with value 30° will first come into the system to alter the reading from the sensor. Figure 30 is the block diagram with the steering angle value indicated on every stage when there is no attack.

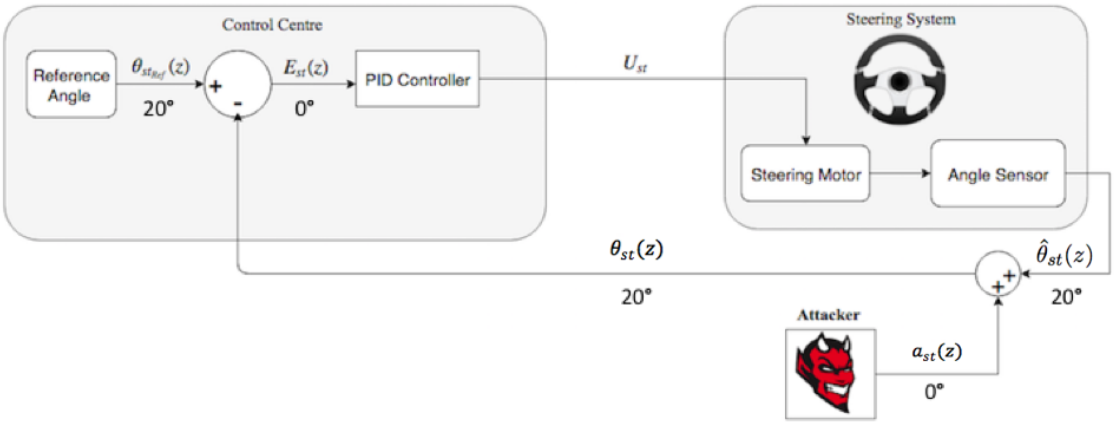

Figure 30: Steering system without attack

Obviously, after the system achieves steady state, the output of the system will track the input of the system, which is expected. Figure 31 shows what will happen when the attack comes in during steady state.

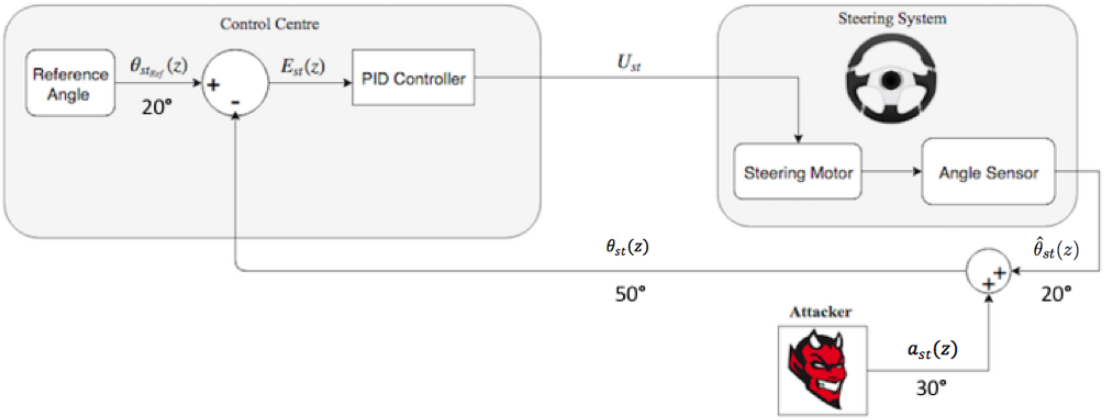

Figure 31: System response when attack appears

Attack signal is denoted by $a_{st}(z)$, $\hat{\theta}_{st}(k)$ is the real value of the output and $\theta_{st}(k)$ is the value read from sensor. When the reference angle is 20°, the output signal will be 20° as well in the steady state. The attack came in and fooled the sensor by adding 30° in output so that the reading of the sensor will be 50°, which will be sent back to the input side. The value of the $E_{st}(k)$ will be 30° after 50° is sent back. The error signal 30° is sent to controller and motor.

Eventually, the output will be forced to turn into −10°. After that, the reading from sensor turns back to 20° as in figure 32. We can also see this in the response curve in figure 29. Till now we did not know anything goes wrong because the sensor is showing us the desired value instead of the real value of the output of the motor after achieving steady state. When the attack value changes to 40°, we can explain it in a similar way.

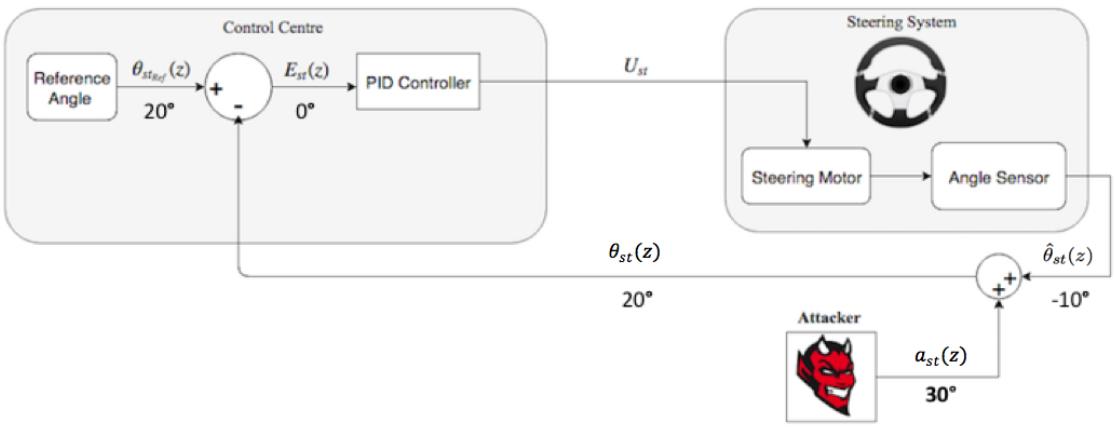

Figure 32: Steady state with attack of 30°

We can also implement a False Data Injection attack with its mathematical model in equation 39 into our driving system, and the simulation result is in figure 33.

$$a(k) = \begin{cases} 0, k \in [0, 1.5s] \\ 50°, otherwise \end{cases} \tag{39}$$

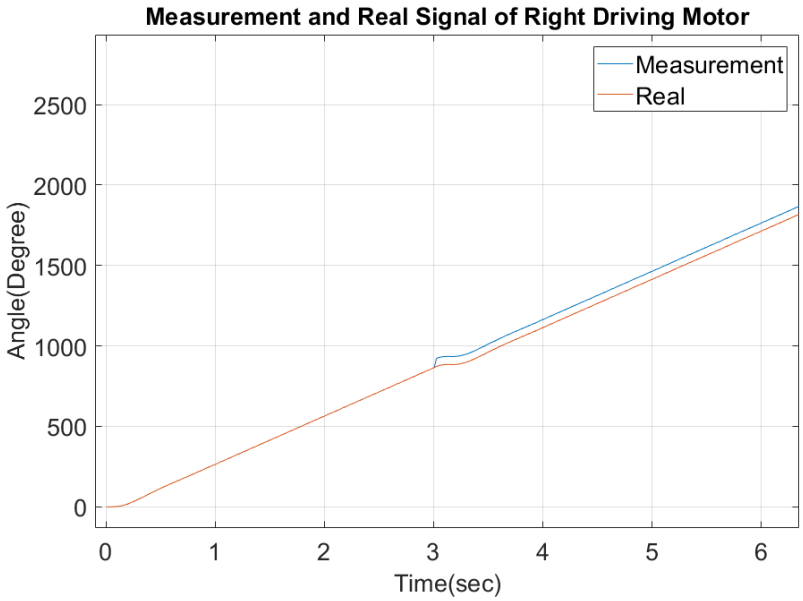

Figure 33: Attack Simulation in Driving System

In figure 33, the blue line is the reading of the sensor, and the red line is the actual value of the output of the driving motor. The motor rotate at a speed of 300° per second, so the angle read by the sensor is increasing by time. Before the attack comes in, these two lines are almost exactly same. At the third second, the sensor is attacked by a False Data Injection attack with amplitude of 50°, which is why the measurement value suddenly increases by 50°. The measurement result, which is also the feedback signal in the driving closed-loop system, suddenly gets larger than the reference signal. In this case, the car stagnates for a very short period, which drags the measurement value back to the reference value. The real angle of the driving motor drops down by 50° due to the short stagnation of the car. Obviously, the False Data Injection attacks can actually affect driving motor leading to skidding or trembling of the car, which is dangerous.

6 Detection of Sensor Attack

In previous sections, relevant security problem of the CPS and the classification of sensor attacks of the CPS have been stated, and a prototype driverless car demo has been developed as an example to illustrate the principle of the CPS and the security problem caused by sensor attacks. It is obvious that without any defence mechanism, the attack signal can influence physical devices and cyber hackers can intercept the communication data or gain the authority to change the measurements of sensors. It is worth mentioning that a DoS attack in this report can be simulated as a False Data Injection attack presented in section 5.1.1 by treating the value of attack signal equal to the opposite number of the sensor measurement, thus in this project we focus on building a security solution for the CPS against False Data Injection attacks.

In this project, a detection and a correction algorithm are proposed, which can indicate the appearance of attack signal by analysing the receiving data from sensors and protects the physical plant from being messed by cyber attacks. This algorithm deploys a Kalman-filter-based sensor attack detector to identify the attack signal, where similar methods were proposed in [35] and [14]. However, we focus more on linear time-varying systems which are relevant to the Lego car demo presented in this project as an example of CPS. In other parts of this section the design, testing result and improvement on detection performance are presented. The correction algorithm is presented in section 7.

6.1 Sensor Attack Detector

6.1.1 Detector for General CPS

A general structure of cyber-physical system involving sensor attacks is presented as figure 5, where an attacker can tamper the sensor measurements by injecting false data or making the measured data unavailable to control centre. We proposed a new structure for CPS by introducing a Kalman-filter based 'Sensor Attack Detector' which can perceive the existence of attack signal, given as figure 34[14].

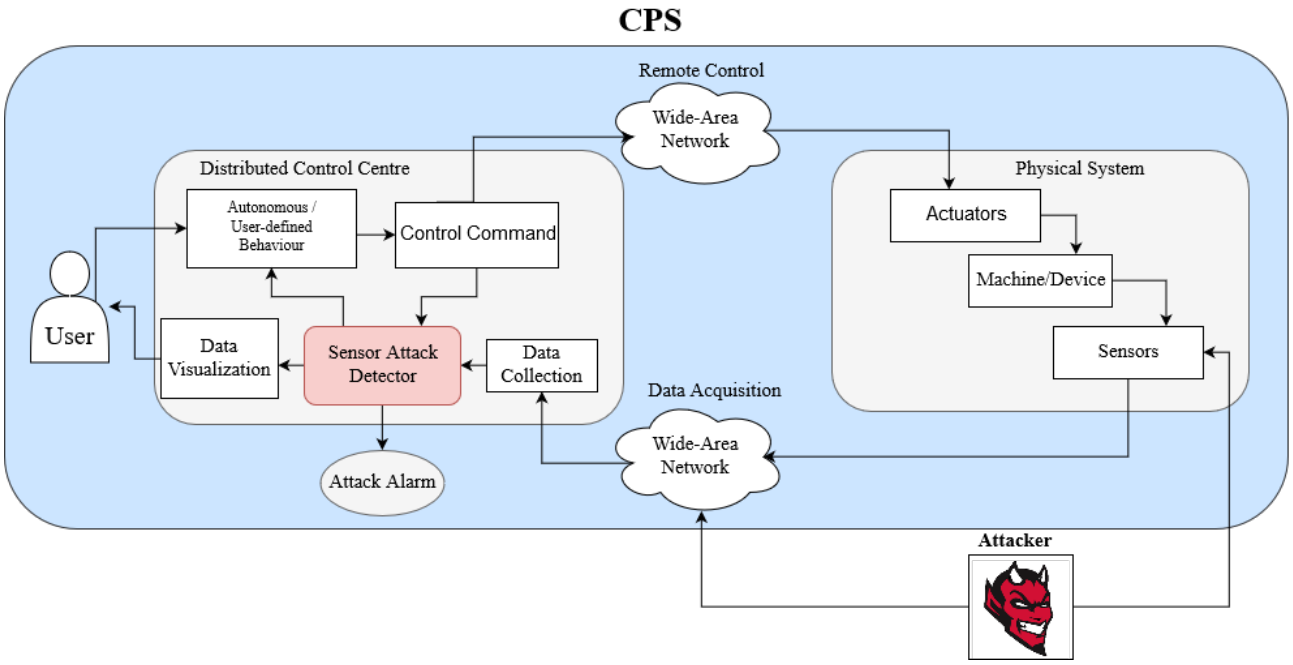

Figure 34: General CPS structure involving sensor attack detector

In this figure, we use the module 'Attack Alarm' as an alert that will be triggered to raise attention when an attack signal is detected by the detector. In figure 34, the other outward-arrow paths of the detector indicate two subsequent strategies when the presence of a attack signal is perceived: to visualise the attack signal to users as and inform the control centre to generate an control command that will stop the remote machines. If no attack alarm is triggered, these two outward paths will just pass sensor measurement data to the following modules.

In our project, we develop a detection algorithm with references to this car structure as the detector, and the algorithm of sensor attack correction is proposed in section 7, with which the detector can be shown to have the ability of identifying the attack signal in quantity. In addition, in figure 34, to achieve the attack detection,the detector needs to access to every incoming data from sensors transmitted by the communication network and the detector also can access to the control command data, which is going to be sent to the physical system. The figure 34 illustrates the structure of our proposal regarding sensor attack detection for general cyber-physical systems, and for the Lego car demo in this project, more detail about the detector structure are presented in the following sections.

6.1.2 Detector for the prototype driverless car

In this section, the internal structure and working process of the detector for the Lego car demo are presented. In section 4, the control method of the car demo is given in detail, where there are three servomotors being controlled separately within three subsystems (one steering subsystem and two driving subsystems). These subsystems associate with each other to drive the car with given reference of the steering angle and speed. Therefore, three servomotors are physical plants in these three subsystems. Each subsystem can be considered as a complete closed-loop system. For simplicity in this section, we use the steering subsystem as an example to demonstrate the mechanism of sensor attack detector, the structure for steering subsystem is given by figure 35.

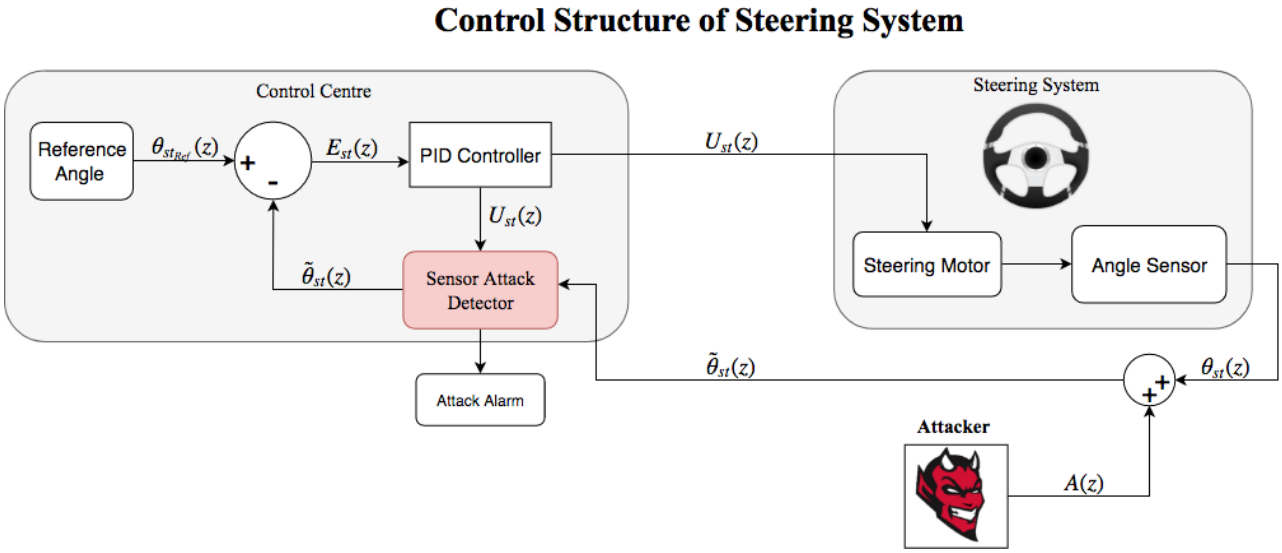

Figure 35: Structure of steering system involving sensor attack detector

In figure 35, it can be seen that there is no other differences between it and the figure 27 apart from the detection related modules. The controlling method has been detailed in section 4, and the influence of attack signal is presented in section 5. The detector is responsible for examining the existence of attack signal by analysing the incoming data from the angle sensor in steering subsystem, and in this project the physical data transmission network has been omitted and the attack signal is simulated as an additive signal at the output of the angle sensor. The detector consists of the algorithm of Kalman filter so that it is able to detect the attack signal. The idea of involving Kalman filter is widely used in many designs regarding sensor faults or the sensor attack detection such as [36], [37], [38], and [39]. The basic task of a Kalman filter for a system is to estimate the state of the system in real time, which some linear algebra calculations to amend a noisy sensor measurement [35], [14], [38]. In this project, each subsystem has an exclusive Kalman filter, and for each sensor measurement sample, the detection algorithm will

compare the estimation of Kalman filter with its sensor reading. With the result of comparison, the sensor attack can be identified. To estimate the system state and output, the Kalman filter needs to know the information of the system parameters, which are represented by state-space matrices. And the Kalman filter is shown can estimate the system state and sensor measurement at time k with information before k [1], therefore, it is possible to use the estimation of the Kalman filter to check the existence of attack signal $a(k)$ at time k by comparing the estimation with the sensor reading at k, and we examine the deviation between them which is named by innovation [1]. We compare the innovation $e_{st}(k)$ with a pre-computed threshold value $thre_{st}$, once innovation $e_{st}(k)$ is greater than $thre_{st}(k)$, we say an attack is detected at time k [35], [14]. This is because the estimation of Kalman filter can be shown converge around the real output of the physical plant, and if the sensor reading is tampered by the attack signal, then the larger inconsistency will be found which indicate the abnormality of sensor. In section 1.4, we gradually explain the procedure of applying the Kalman filter and in the following section, we present how to build the detection algorithm for the Lego car demo and improve its performance on detection.

6.2 Detector Design

6.2.1 Derivation of State-space Matrices

In the previous section, we addressed the delay problem of the sampling rate of the servomotor by fitting its model to a time-varying system. However, the algorithm complexity will be dramatically increased in building the time-varying Kalman filter on the car demo, because we have to run discretisation and derive the state-space model between every two consecutive samples. And we find that if we include these series operation in each iteration, it will increase the time consumption by ten times of the original sampling period.

In programming, we fix the aforementioned problem by designing a dictionary object in MATLAB, whose key is including every possible sampling interval and values are the corresponding system matrices. It means we can do the discretisation and create state-space model for every possible sampling period beforehand. When the Kalman filter starts to work, between each two samples, the dynamic model can be derived by timing the sampling interval and refer to the dictionary. With this configuration, we can shrink time complexity 10 times than without using the dictionary method. This is the improvement we made in this project which significantly

upgrades performance of detection because higher time complexity for controlling the car demo will result in bed performance of closed-loop control as well as the sensitivity of detection. In the steps of discretisation and deriving state-space matrices, we can further reduce the algorithm complexity by deriving the state-space model in controllable canonical form, which gives a fixed input matrix B. Therefore, the dictionary we make can only contain the information of state matrices A and output matrices C thus the memory for storing the dictionary could be reduced.

6.2.2 Detector Structure of Prototype Driverless Car

Because we utilise the RWTH - Mindstorms NXT Toolbox for MATLAB and it does not have a graphic interface, we have to develop every component of each subsystem by coding. For visualisation, We present the internal structure of the detector as a block diagram in figure 36.

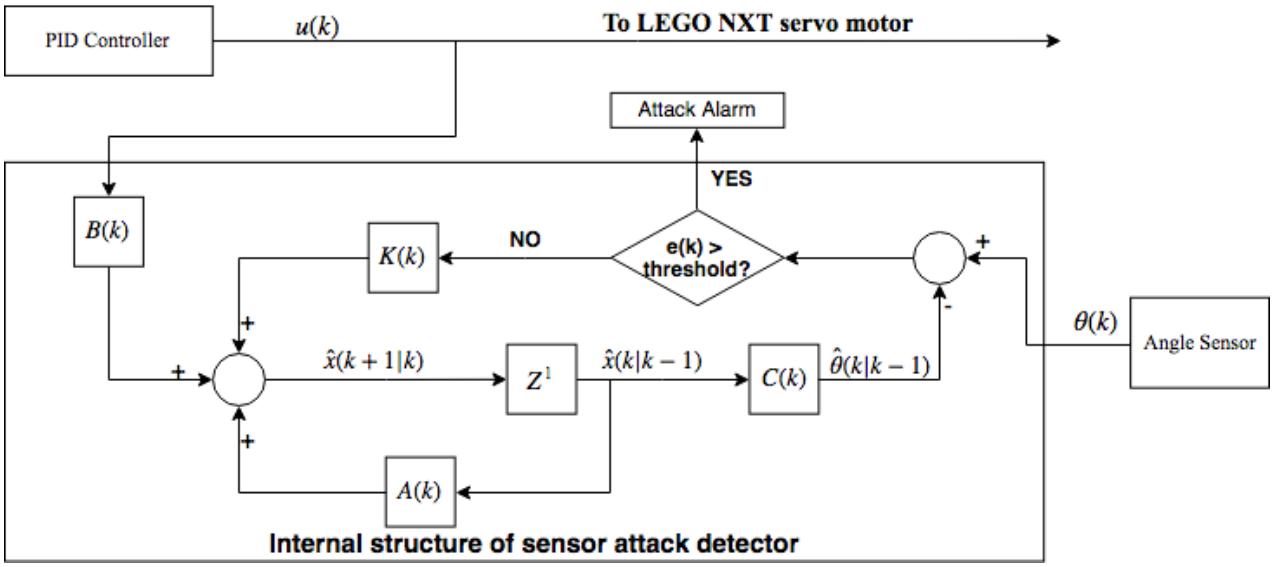

Figure 36: Internal structure of detector

The detector shown in figure 36 can be used for both steering and driving subsystem of the car demo since they both use the same servomotors. The value of detection threshold in some literature is set according to $x^2 - detector$ or *Euclidean detector* presented in [14] which involves the long-term detector. However, in our project, we simplify the method to derive the detection threshold. The method to find a proper threshold value is to build and repeatedly test the Kalman filter to seek for the maximal innovation for each subsystem on the car demo, which we need to make the car demo navigate under different environment and speed for many times. Thus we choose to build the detector with a relatively large threshold on the steering

and driving subsystem at first, then we introduce some techniques to minimise the threshold so that the smaller attack signal can also be detected.

6.2.3 Initial Condition of Kalman filter for Detection

The statistic characteristics of input noise and measurement noise of Lego NXT servomotors are given by [40]:

$$v(k) \sim N(0, 0.5^2) \tag{40}$$

$$e_i \sim N(0, 5^2) \tag{41}$$

where e_i represents the error of normalised input voltage, since the input error will propagate into system state, thus we can transfer which to state process noise. The model of the servomotor is known as a second order system, and B matrix is known fixed $B(k) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, thus the process noise covariance matrix W is given by [41]

$$W(k) = \sigma_{ei}^2 * E[B(k)B(k)'] \tag{42}$$

and from equation 40, we have the measurement noise covariance matrix given by:

$$V(k) = E[v(k)v(k)'] = \sigma_v^2 \tag{43}$$

Assume sensor measurement $y(0)$ and initial state $x(0)$ are zero. To confirm the initial estimation and covariance matrix P_1, by referring to equation 11, we can see $x(1|0) = x1$ is related to the distribution characteristic of input signal $u(0)$ [1]. Thus we have;

$$x_1 = \begin{bmatrix} u(0) \\ 0 \end{bmatrix} \tag{44}$$

$$P(1) = E[x(1)x(1)^T] = \begin{bmatrix} \sigma_{ei}^2 & 0 \\ 0 & 0 \end{bmatrix} \tag{45}$$

Then, we have enough condition to build a Kalman filter in the detector, the process of using Kalman filter in prediction form is given in section 1.4.

6.2.4 Detection Threshold

In the last section, we present the method to find a proper detection threshold of innovation signal of Kalman filter in differentiating attack signal and generally acceptable error in estimation. In this section, we build a preliminary detector whose threshold is not given, and attack signal is not launched. The aim of this section is to find the maximum estimation error that the Kalman filter may have, which will then be treated as the detector threshold. Thus a fault alarm will be triggered. However, we also need to admit the truth that a large threshold cannot perceive a relevant smaller attack signal. Thus an optimised method or improvement is needed to decrease the threshold.

In the car demo, we have two different subsystems: the steering subsystem and the driving subsystem. To find the maximum estimation error (innovation signal) for the steering system, we set the reference angle to be the maximum steering angle (60°), and we activate the preliminary detector to estimate the system output in real-time. We repeatedly make a ten-second close-loop control and the maximum innovation value appears in figure 37.

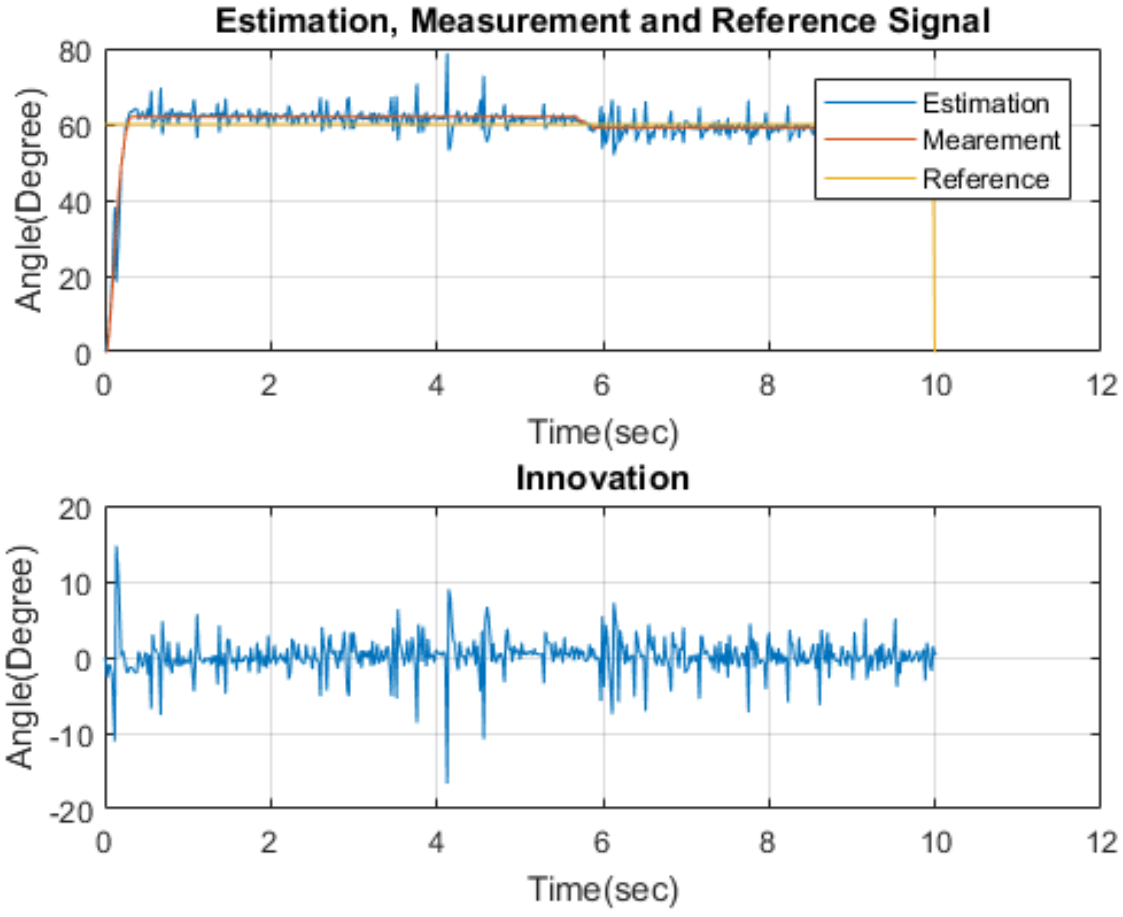

Figure 37: Computing the maximum innovation value for steering system

In figure 37, the first figure gives the information of estimation output $\hat{\theta}_{st}(k|k-1)$, sensor measurement $\theta_{st}(k)$ and reference steering angle $\theta_{st_{Ref}}(k)$. As we can see $\hat{\theta}_{st}(k|k-1)$ appears with significant oscillation, and in the second figure the innovation signal is given. The largest amplitude is shown as 18° which is fairly large. An optimisation method is presented in section 6.4. For now, we temporally set the threshold of steering system as ±20°. For the driving system, to find the largest innovation value, we control the driving motor with reference angular velocity to 700° per second, since the sensor measures angle rather than angular velocity. Thus we still use the Kalman filter to estimate the angle of driving motors. The maximum innovation value signal of driving motors appears in figure 37 (We just present the result of the right driving motor).

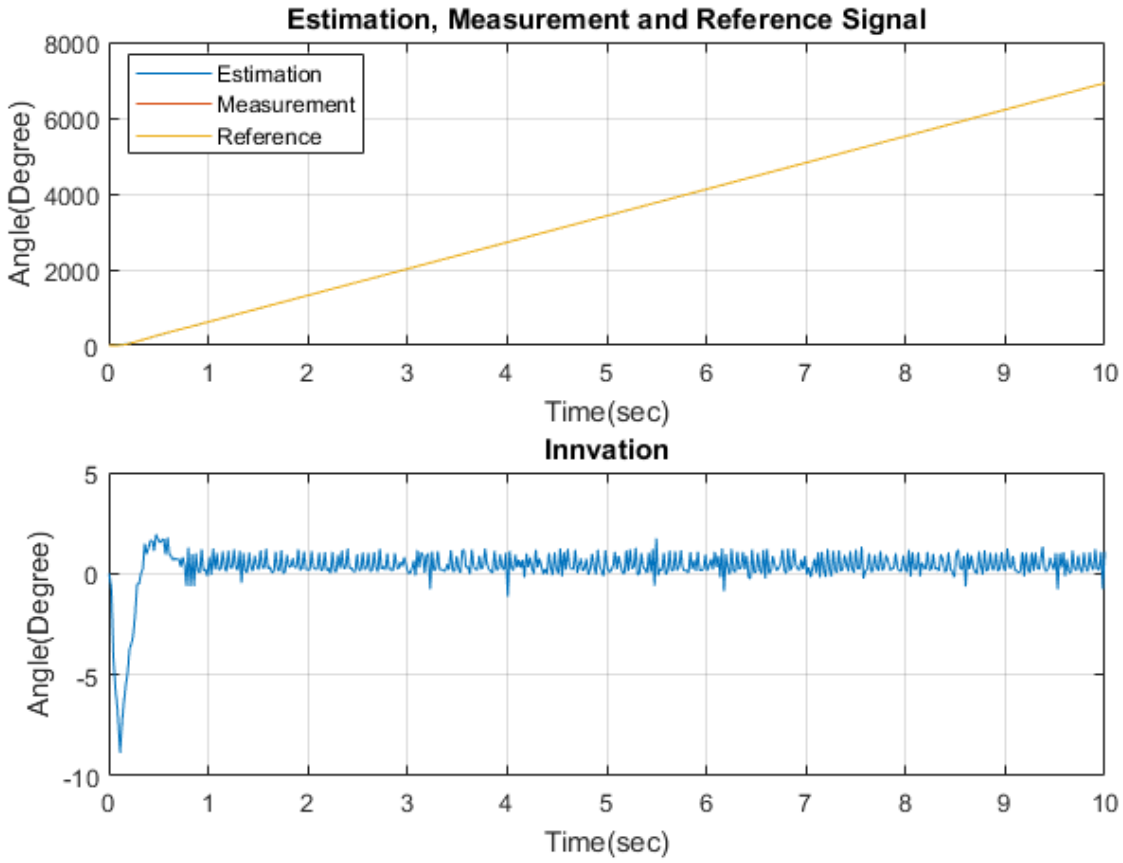

Figure 38: Computing the maximum innovation value for driving system

In figure 38, we compare the real-time value between estimation $\hat{\theta}_{dr}(k|k-1)$, the sensor measurement and reference angle of the driving motor (integral of reference angular velocity by time) $\theta_{dr_{Ref}}$. As we can see, the largest innovation value is found over 5° , thus we can set the threshold for detection as ±10°.

6.3 Test of Detector

In this section, we test the performance of the preliminary detectors of steering and driving systems. We use the same configuration presented in 5.2, then differences between the scenario with and without detector are given. We define the attack alarm as Boolean variable as follows:

$$alarm(k) = \begin{cases} 1 & \text{for } k \in \tau \\ 0 & \text{otherwise} \end{cases}$$

where τ indicates the time when the attack signal exists. To test the performance of detector for the steering system, we still set the reference steering angle to 20°, the steering system start to be controlled at 0s, and we add an attack signal of 30° from 5s to 10s and the other attack signal of 40°launched between 15s to 20s. We active the detector at all times. The experiment result is given in figure 39.

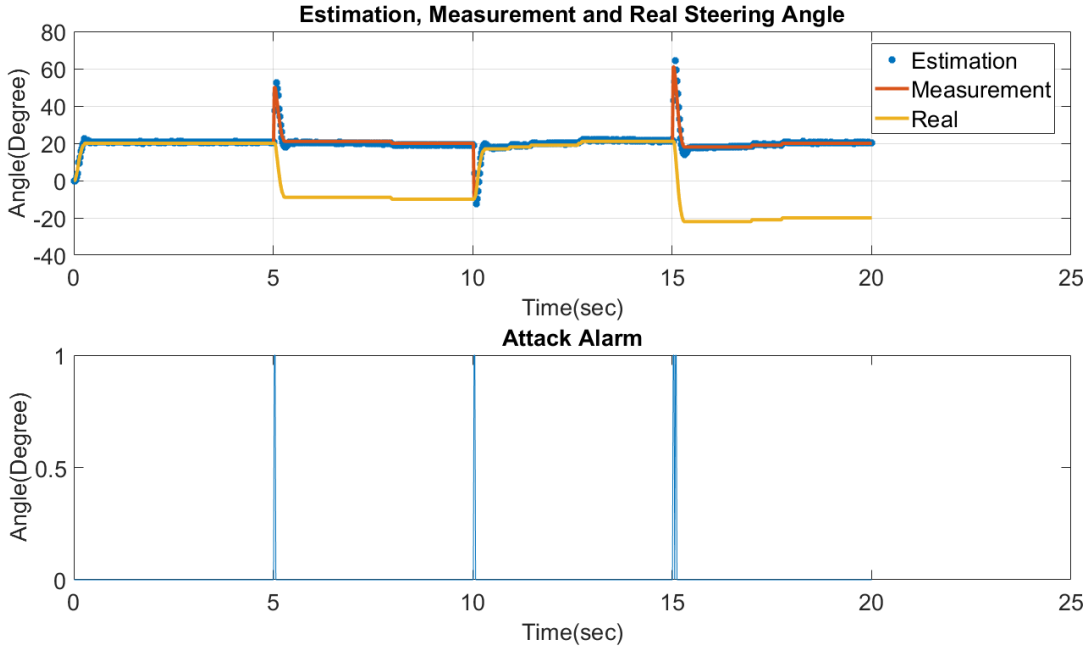

Figure 39: Steering system test with detector

In the first plotting, we can observe estimation signal is with some oscillation, which is why we need to set a threshold to avoid false attack alarm. The second plotting shows the value of attack alarm, where three spikes can be found and the first spike corresponds to the time when the first attack signal was launched (at 5s). The second spike indicates when the first attack left (at 10s), and the third spike indicates when the second attack signal was launched (15s). The second attack signal persisted until we stop the controlling of the steering system. Therefore, there is no spike indicating the departure of the second attack signal. It is clear that

in this figure, the attack alarm signal is not what we expect since we define the attack alarm should be kept alarming until attack signal has left. The reason is we still send the false data to the controller rather than terminating the system or using data reconstruction scheme. The detector can detect the appearance and departure of the attack signal, proved by these spikes. However, if we still enable this closed-loop manner, the Kalman filter would use the false data to estimate the output of the next stage. That is why before the departure of the attack signal, the value of attack alarm signal jumped back to zero. We optimised the visualisation of attack alarm signals in section 7 by introducing the technique of reconstructing the correct sensor measurement from the false data mixed with attack signals.

To test the performance of detector on the driving system for ten seconds, we set the reference angular velocity to 300°per second, the driving system starts to be controlled at 0s, and we add an attack signal of 50° from 5s to 10s. We active the detector at all times. The experiment result is given in figure 40.

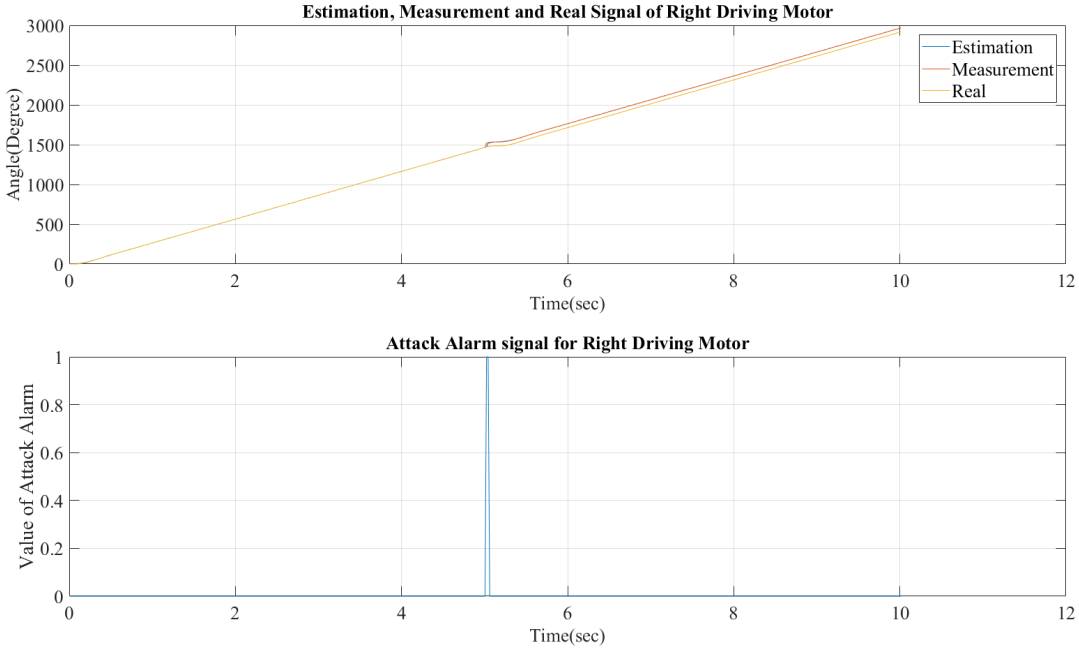

Figure 40: Attack and detection test on driving system (right driving motor)

where it has the same problem of the attack alarm signal with the testing result of the steering system, but as expected, the detector can figure out the appearance of attack signals.

6.4 Optimisation of Detector

A common defect of the sensor attack detector for steering and driving systems in the car demo is presented and shown in figure 37 and figure 38, where the theoretical threshold value is too large because the large maximal innovation. In this detection algorithm, the detector identifies the existence of attack signal by checking whether the innovation signal is over the threshold or not, so any value of innovations below the threshold will not be considered compromised by the attacker. Thus, a very large threshold such as $\pm 20°$ is not allowed since any attack signal with amplitude lower than the large threshold can easily bypass the detection. To optimise the detector and make it more resistive to attacks, we deploy a moving average filter after the Kalman filter, which smooths the estimation $\hat{\theta}$ of the Kalman filter to decrease the maximum value of the innovation signal so that the detection threshold can be set smaller. The testing of the detector for the steering system associated with a smoothing filter is given in figure 41.

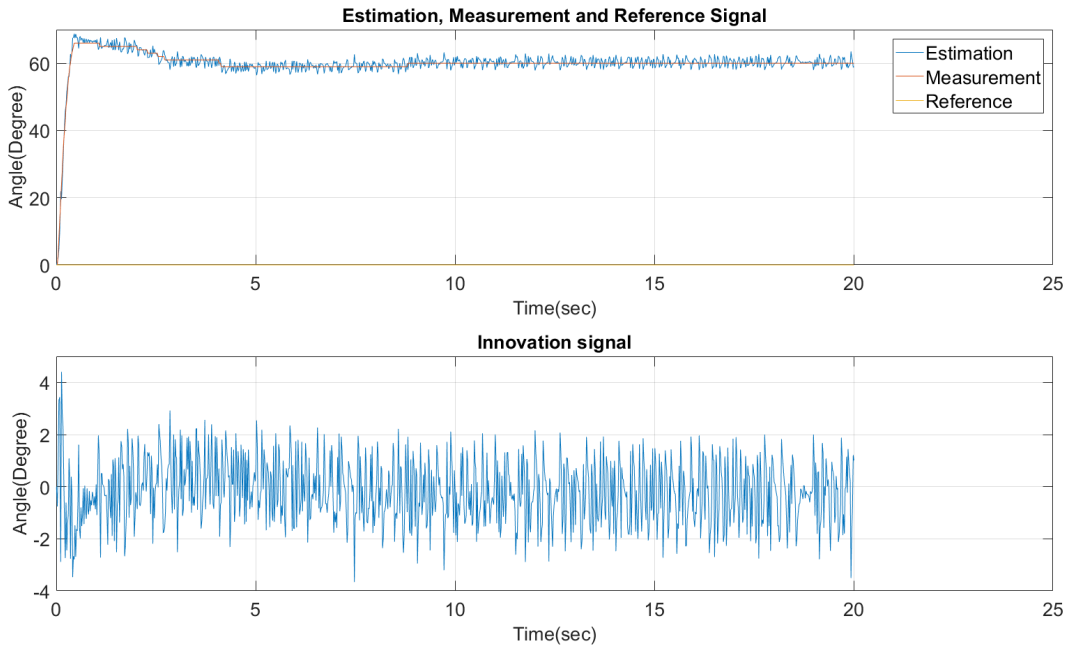

Figure 41: Decrease the threshold of detector by involving smoothing filter

Compared with figure 37, the smoothing filter shrinks the maximal possible innovation value within $\pm 5°$. Thus we can change the detection threshold to a smaller value, and the value of any signal which is higher than the threshold value will identify as the attack signal.

However, there are still some other defects existing in the detector designed for the car demo. For some clever attackers who know the characteristics of the detector [14], they can bypass the detection by continuously injecting small false data whose value is below the threshold. Thus,

the long-term detection method are involved to counter this problem rather than checking the existence of the attack signal by a sensor measurement at an instant moment, and relevant techniques can be found in [14] and [35].

7 Correction of Sensor Attack

In section 6, a feasible approach for detecting sensor attack is presented for CPS and tested on the prototype car. In addition to the technique of detection, the ability of data reconstruction from sensor attack in CPS is also needed in order to guarantee the high robustness and reliability of the system and high resistance against attack signal [42]. Most CPSs are working autonomously in real time, and sometimes resilience and sustainability are also needed rather than terminate the physical ends once an attack is identified. For instance, the GPS measurement could be distorted, but it is impossible to force a craft to turn off engine and land immediately since the craft could be in thousands of inches high and the position information is still needed by the control tower to arrange emergency landing. A recovery mechanism should be included in the discussion of the security of CPS. In this section, we present a data correction algorithm in countering the threat sensor attack for CPS which involves the thinking of Redundancy. We also build and test the algorithm on this car demo, after which the testing result and defects are given.

7.1 Sensors Redundancy in Prototype Driver-less Car

In many CPS applications and investigation, the redundancy mechanism is mostly used and proposed when facing sensor attack issues such as [39], [25], [43], [42] and [8]. In this project, we also use the redundancy information from healthy sensors to achieve sensor attack correction. Essentially, the car demo system is a combination of three individual subsystems: one steering subsystem and two driving subsystems, and they work on correlative reference signals during navigation, which is presented in section 3.2. Given any two reference signals at a specific moment, we can derive the third reference signal based on the geometry structure presented in section 3.2. Since the three subsystems are all controlled by closed-loop manner, when these motors are in steady state, the output of each subsystem is equal to the given reference signal. Therefore, at steady state, we can estimate the output of one subsystem by knowing the sensor measurements of the other two subsystems. This is how we achieve the sensor attack correction. In other words, any two subsystems can provide redundancy for the other subsystem. Because with the detection algorithm presented before, each subsystem has the ability to identify the abnormality of the sensor measurement when attack signal is launched. The false data cannot be considered as the real feedback signal for the closed-loop system so it will be discarded. Thus, suppose there is one subsystem whose sensor measurement is detected under attack, then the correction algorithm can formulate a correct feedback signal for it

according to the redundancy. Specifically, the sensor attack correction algorithm in this project will estimate a correct feedback signal for the attacked subsystem with the data from other two healthy subsystems. Nevertheless, it is obvious that if there are two or more subsystems being attacked, the correction algorithm will fail, which could be considered as a constraint existing in this solution.

In some other applications like [44], [42] and [45], the redundancy amongst sensors are also used to detect sensor attack. However, the reason we did not employ this method to develop detection algorithm is because there is a constraint that the number of attacked sensor cannot be over the limit determined by the characteristic of the system. Otherwise, the system could fail on identifying the existence of attack signal. Instead, the Kalman filter detector allows a subsystem individually detect the sensor attack. Thus, we intend to add detectors for each subsystem so that there would be no constraint on the number of attacked sensors, which means even if all sensors are under attack, each subsystem can still identify the attack signal.

In this project, the detection algorithm can indicate instantly when attack signal appear, and the correction algorithm can start to estimate correct sensor measurement for that instant. The correction algorithm is explained as follows.

Recall the geometry structure which analysed with figure 14, when all the three subsystems reach into steady state, the outputs of all three sensors of three subsystems are almost equal to their given reference signals because of the closed-loop control manner. The three reference signals are referring to the desired trajectory set by user. Thus if we lost the sensor measurement of one subsystem, by referring to the other two subsystems' measurements, we could derive the lost measurement data. Denote the measured angle of steering by θ_{st}, the angular velocity of the right driving motor by ω_{dr} and angle of the left driving motor by ω_{dl}. Suppose the car demo is commanded to navigate, when the three subsystems reach steady state. We have:

$$\theta_{st} = \theta_{st_{Ref}} \tag{46}$$

$$\omega_{dr} = \omega_{dr_{Ref}} \tag{47}$$

$$\omega_{dl} = \omega_{dl_{Ref}} \tag{48}$$

This means the sensor measurement of each subsystem is equal to their given reference. $\theta_{st_{Ref}}$ is the reference steering angle; $\omega_{dr_{Ref}}$ is the reference angular velocity of the right driving motor; $\omega_{dl_{Ref}}$ is the reference angular velocity of the left driving motor. Then, if only steering

subsystem starts to be attacked at time τ, and its attack detector works well.

$$\theta_{st}(\tau) = \hat{\theta}_{st}(\tau) + a(\tau) \tag{49}$$

This means the sensor reading of steering subsystem θ_{st} is no longer equal to its reference signal but the sum of the real steering angle at τ and the attack signal. Here $\hat{\theta}_{st}(\tau)$ represents the real steering angle of steering motor. In this case, the sensor attack detector will detect the appearance of attack signal $a(\tau)$ and avoid using $\theta_{st}(\tau)$ as the feedback signal for controlling. Then, the correction algorithm will refer to the sensor readings of two other subsystems ω_{dr} and ω_{dl}:

$$\tilde{\theta}_{st}(\tau) = f(\omega_{dr}(\tau), \omega_{dl}(\tau)) \tag{50}$$

$$\tilde{\theta}_{st}(\tau) = \hat{\theta}_{st}(\tau) \tag{51}$$

We denote $\tilde{\theta}_{st}(k)$ as the redundancy of $\hat{\theta}_{st}$ at time τ and equation 50 and 51 illustrates that actual angle of steering motor can be determined by a function of the angular velocity of two other driving motors. The function f can be deduced by transforming equation 25 and 26. And in the other hand, the information of the driving motors are given as angle rather than angular velocity. In order to get the angular velocity, we can calculate the derivative of angle as:

$$\omega(k) = \frac{\theta(k) - \theta(k-1)}{\Delta t} \tag{52}$$

where $\Delta t(k)$ is the interval between the angle sample at time k and the sample at time $k-1$. This is how correction algorithm is used to estimate the correct sensor reading for steering subsystem. If attacks happen on one of the driving subsystem, we present different functions to extract the corresponding redundancy, which are given in Appendix from equation 62 to equation 78.

However, it is possible for an attack signal to be launched at the time when the car demo has not reached in steady state, we then re-formulate the correction algorithm as follows (for steering subsystem):

$$\tilde{\theta}_{st}(\tau) = f(\omega_{dr}(\tau), \omega_{dl}(\tau)) \tag{53}$$

$$\tilde{\theta}_{st}(\tau) = \hat{\theta}_{st}(\tau) - error_{est}(\tau) \tag{54}$$

where $error_{est}(k)$ represents the error in using redundancy to estimate the sensor reading for steering angle, which could be a random process. The reason of the existing error is because during transient process, the real angle of the steering motor is changing fast comparing with that in the steady state, and the redundancy provided by the other two healthy subsystems can

only reflect a trend of the steering angle with a roughly estimated angle value. Hence, there must be existing an estimation error that we default and consider as a zero. In this case, using redundancy to reflect the real sensor measurement will introduce error and a large error will cause failure of correction and the car demo will go out of control. In our correction algorithm, once an estimation has been made, we still use the detection to check the correctness of it and if the estimation of the Kalman filter is close to the estimation made by sensor redundancy, the system will confirm the correctness of the redundancy and use the estimation as feedback signal for time τ. Otherwise, if the estimated sensor measurement shows large inconsistency with the estimation made by the Kalman filter, we then conclude the failure of correction.

We then give the precondition of realising correction of sensor attack for the car demo:
1. The imposed value of attack signal on sensors should be over the detection threshold (sufficient and necessary condition for an attack to be detected).
2. In one specific moment, for the car demo, there could be at most one sensor being attacked so that correction can be done. Otherwise, it is impossible to realise correction since redundancy for one sensor requires the other two sensors are not under attack.

Theses assumptions are made only for the car demo, for different CPS with a different model, the tolerant number of sensors under attack could be different. A terminology called Security Index is presented in [46], which presents the relevant investigation, definition and method to derive the tolerant number.

7.2 Correction on Constant Attack Signal

7.2.1 Analysis and Design

It is shown that under some conditions, once the attack signal is launching to the sensor of one subsystem, the correction algorithm can recover the correct sensor reading instantly. However, getting the redundancy at a single moment is not sufficient to realise correction. The car demo is driving autonomously in real time and the correct sensor measurement of each subsystem at every time point is needed to realise robust navigation. Thus we then encounter the problem if the attack signal is not only launched at a single time point τ but also the following consecutive samples. In other words, the correction algorithm should be able to keep providing correct sensor measurement for the attacked subsystem even if the attack signal is

launched for a period of time.

In brief, the problem comes to estimate the correct sensor measurement at time $\tau+1$, if the sensor measurement has been tampered by attack signal at time τ. On the other hand, the attack signal could be either a time-varying signal or a constant signal. The correction algorithm has different scheme on account of the characteristics of attack signal. In this section, the correction scheme under constant attack signal is presented. The scheme for correcting under time-varying attack signal is given in next section.

We assume steering subsystem is still attacked by attack signal after time τ. The attack signal is constant over time, then the sensor measurement is given by:

$$\theta_{st}(k) = \hat{\theta}_{st} + a \qquad k > \tau \tag{55}$$

where we can see the sensor reading keeps being influenced by attack signal.
Recall at time τ, we can estimate the sensor reading with the redundancy amongst sensors:

$$\tilde{\theta}_{st}(\tau) = f(\omega_{dr}(k\tau), \omega_{dl}(k\tau)) \tag{56}$$

$$\tilde{\theta}_{st}(\tau) = \hat{\theta}_{st}(\tau) \tag{57}$$

Assume the detector for the steering system works well and the attack signal a is larger than the detection threshold. An attack alarm will start to be triggered at τ and if the difference between $\tilde{\theta}_{st}(\tau)$ and the estimation sensor measurement made by the Kalman filter is smaller than the threshold, the algorithm will then record $\theta_{st}(\tau)$ and $\tilde{\theta}_{st}(\tau)$ into memory. Meanwhile, the detector will treat $\tilde{\theta}_{st}(\tau)$ as the real sensor measurement of steering angle rather than $\theta_{st}(\tau)$ to continuously estimate the sensor reading for $\tau+1$. Then, the following step is to get the knowledge of the angle of steering motor at $\tau+1$ so that the steering motor can still be controlled in closed-loop manner. The algorithm will set $\tilde{\theta}_{st}(\tau)$ as a foundation denoted by F. Since the attack signal is constant, thus at time $\tau+1$, we can extract the changes of steering motor from the attacked sensor measurement by:

$$D(\tau+1) = \theta_{st}(\tau+1) - \theta_{st}(\tau) \tag{58}$$

$$\tilde{\theta}_{st}(\tau+1) = F + D(\tau+1) \tag{59}$$

where $D(k)$ represents the dynamic change of the angle of steering motor from time $k-1$ to k, and we can see $D(k)$ does not contain the attack signal a which is cancelled by subtraction of equation 58. Therefore, we still can recover the real measurement of the sensor which is $\tilde{\theta}_{st}(k)$. In addition, to correct, the detector should keep detection between estimation of the

Kalman filter and sensor measurement continuous and select the $\tilde{\theta}_{st}(k)$ from equation 59 as the redundancy of the steering system then treat it as the feedback signal to control the steering motor. The algorithm will keep working until the attack signal does not exist any more. When the innovation signal is lower than threshold again, we can reuse the sensor measurement as the feedback signal.

The correction algorithm can resolve the influence of DoS attack or the False Data Injection attack with constant amplitude. The flowchart of correction algorithm is given in figure 42.

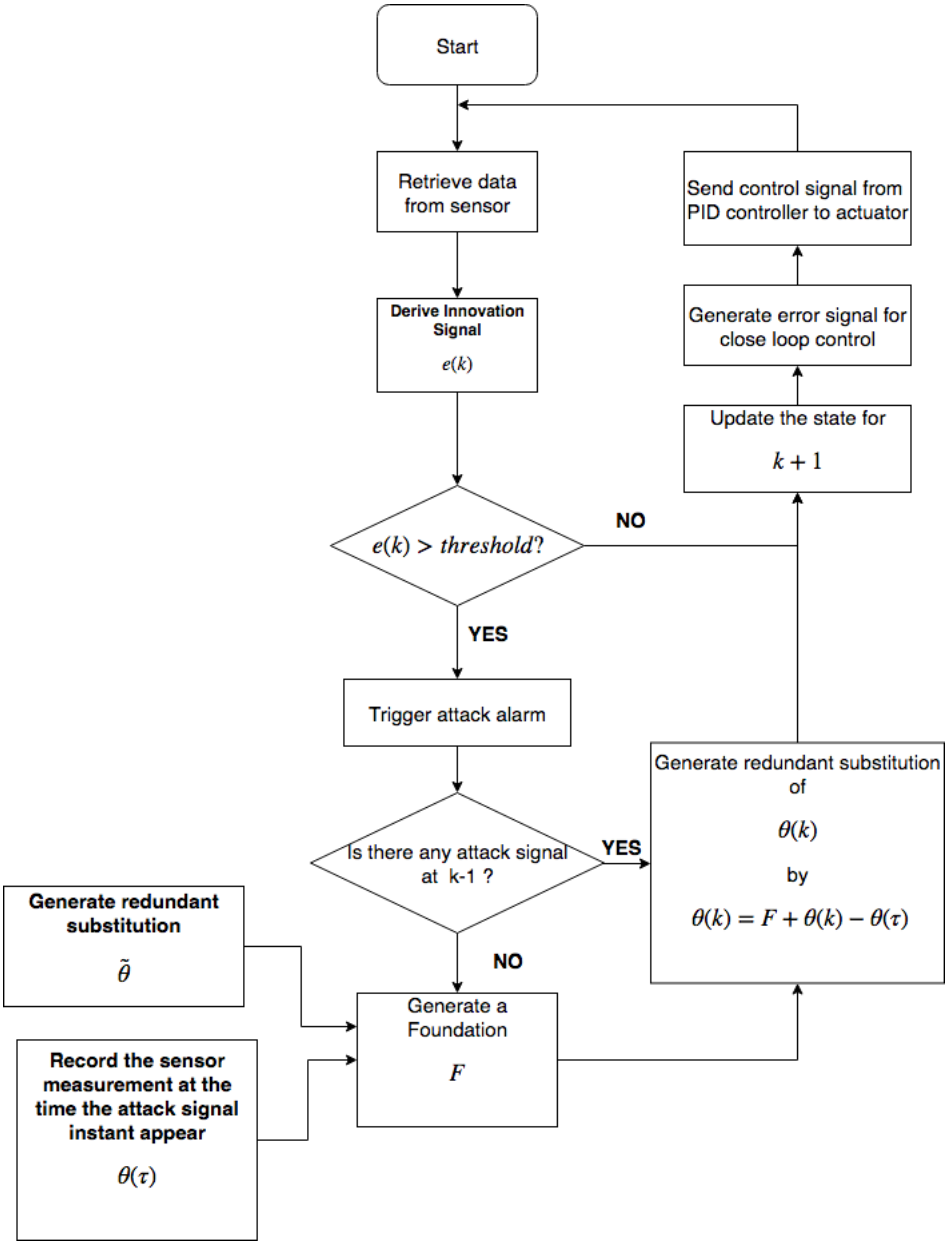

Figure 42: Correction process for constant-data sensor attack

7.2.2 Test of Correction under Constant Attack

In this section, we build the correction algorithm in the car demo associated with the detector. We test the performance of the algorithm towards to the constant-amplitude attack on both steering system and driving system. We use the same settings of attack signal presented in Section 6.3, and we activate the sensor attack detection algorithm and involve the correction algorithm for constant attack signal.

The test result of steering subsystem is given by figure 43. In the first figure, with the reference angle of 20°, the real steering angle (shown by the yellow line) did not change although there were two constant attack signals launching from 5s to 10s (with 30°) and from 15s to 20s (with 40°). The sensor measurements are shown tampered during the period of attack, but the estimation of Kalman filter was not influenced and overlapped over the real trajectory since the correction algorithm replace the false measurement data with the estimation based on the redundancy for steering subsystem. In the second figure, the amplitude of attack signal was identified in real time, which is equal to the innovation signal (difference between the sensor reading and estimation of the Kalman filter). The experimental result shows the detection and correction algorithm worked well.

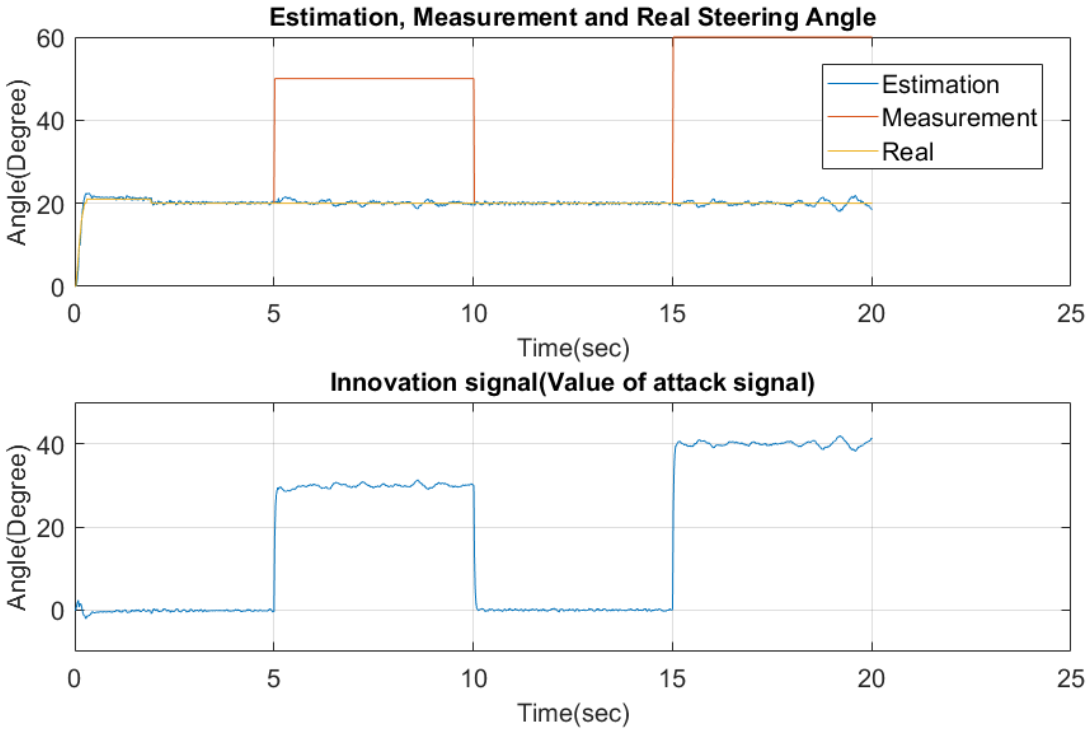

Figure 43: Correction for steering subsystem under constant attack signal

The test for driving subsystem under constant attack signal is shown in figure 44.

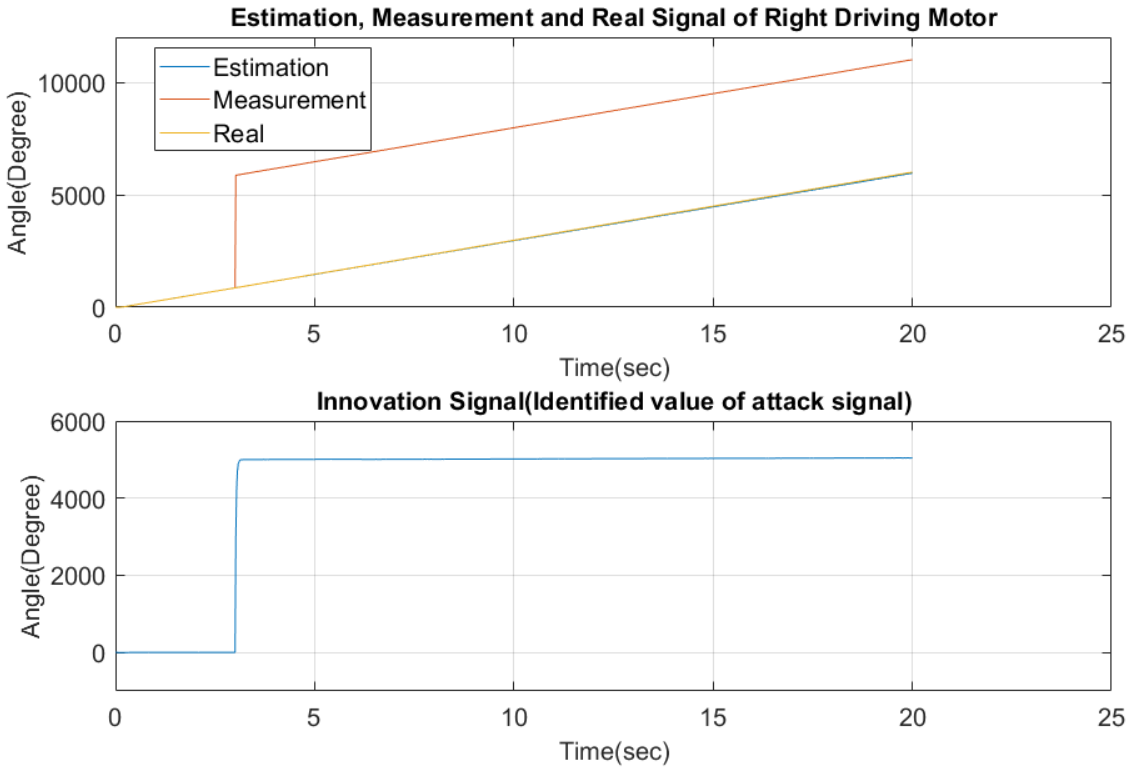

Figure 44: Correction for driving subsystem under constant attack signal

We use the right driving system to demonstrate the correction test. We set the reference angular velocity to 300° per second and launched a constant attack signal. We launched a constant attack signal with attack value equal to 5000° after 3s and the total time of navigation is also 20s. The result illustrates a good performance of the detection and correction algorithm for the driving subsystem under the constant sensor attack signal. In the first figure, the sensor measurement shows a large jump at time $k = 3s$, which means it is tampered by the attack signal. However, the real value angle of driving motor and the estimation by the Kalman filter are protected by the detection and correction algorithm. The angular velocity is still 300° per second, which is still equal to the reference signal.

7.3 Correction on Time-varying Attack

In the previous section, we present the method on how to correct a sensor measurement under constant attack. However, it will lose the generality if a security solution of a CPS can only resist the constant attack. In real CPS security problem, many False Data Injection attacks are

modelled as a random signal. This means the attackers could inject a random value of data to mess the measurement and cause the physical machines to be out of control. And some other kinds of attackers can also control the physical machines by injecting particular time-varying data series [25]. Therefore, in many cases, the attack signal cannot be simply described by constant except for DoS attack (the sensor reading is always equal to zero). In this section, we present the correction scheme for time-varying attack signal.

7.3.1 Analysis and Design

In Section 7.2, we explained the algorithm which set the redundancy of the attacked sensor as a foundation F at the time when the attack signal is launched. And we combine F and the dynamic changes $D(k)$ of sensor measurement to formulate a redundancy of sensor measurement. To counter time-varying attack signals, we firstly need to ensure the attack signal has changed in amplitude. Suppose we still consider the attack signal does not change at time $\tau+1$, and the scheme is to estimate the sensor measurement by equation 58 and 59. However, in real case, equation 58 cannot cancel the attack signal since its amplitude is no longer a constant. Therefore, equation 59 will result in false estimation of $\tilde{\theta}_{st}(\tau+1)$. To indicate the change of attack signal, the algorithm will firstly feed $\tilde{\theta}_{st}(\tau+1)$ to the Kalman filter again and check the difference innovation and detection threshold. The motivation of this step is to check the correctness of the estimation made by redundancy amongst sensors. Thus if the estimation made by equation 58 and 59 cannot pass the checking, we tend to believe the attack signal has changed.

Once the attack signal is detected changed, the correction algorithm will then re-estimate the sensor reading with the redundancy amongst sensors, which means if $a(\tau) \neq a(\tau+1)$, the following equations will be re-calculated for correction.

$$\tilde{\theta}_{st}(\tau+1) = f(\omega_{dr}(\tau+1), \omega_{dl}(\tau+1)) \tag{60}$$

$$F = \hat{\theta}_{st}(\tau+1) \tag{61}$$

F will be updated once attack signal is found changed, and we update the new false sensor reading $\theta_{st}(\tau+1)$ in the memory. Therefore, during the existence of the attack signal, the detector can figure out the appearance of attack signal and correction algorithm will use the redundancy as the real sensor measurement. Meanwhile, the detector will also test the consistency between estimation made by the Kalman filter and the estimation given by the redundancy amongst sensors. The consistency is verified by the pre-estimated detection threshold as presented in

previous section. And when the difference between them is larger than the threshold, the detector will confirm that the attack signal has changed, and the algorithm will start to re-calculate a new foundation of the attacked sensor and record the new measurement of the false sensor measurement θ_{st} in order to keep getting the knowledge of the dynamic change of steering motor. The flowchart of correction on time-varying attack signal is given by figure 45.

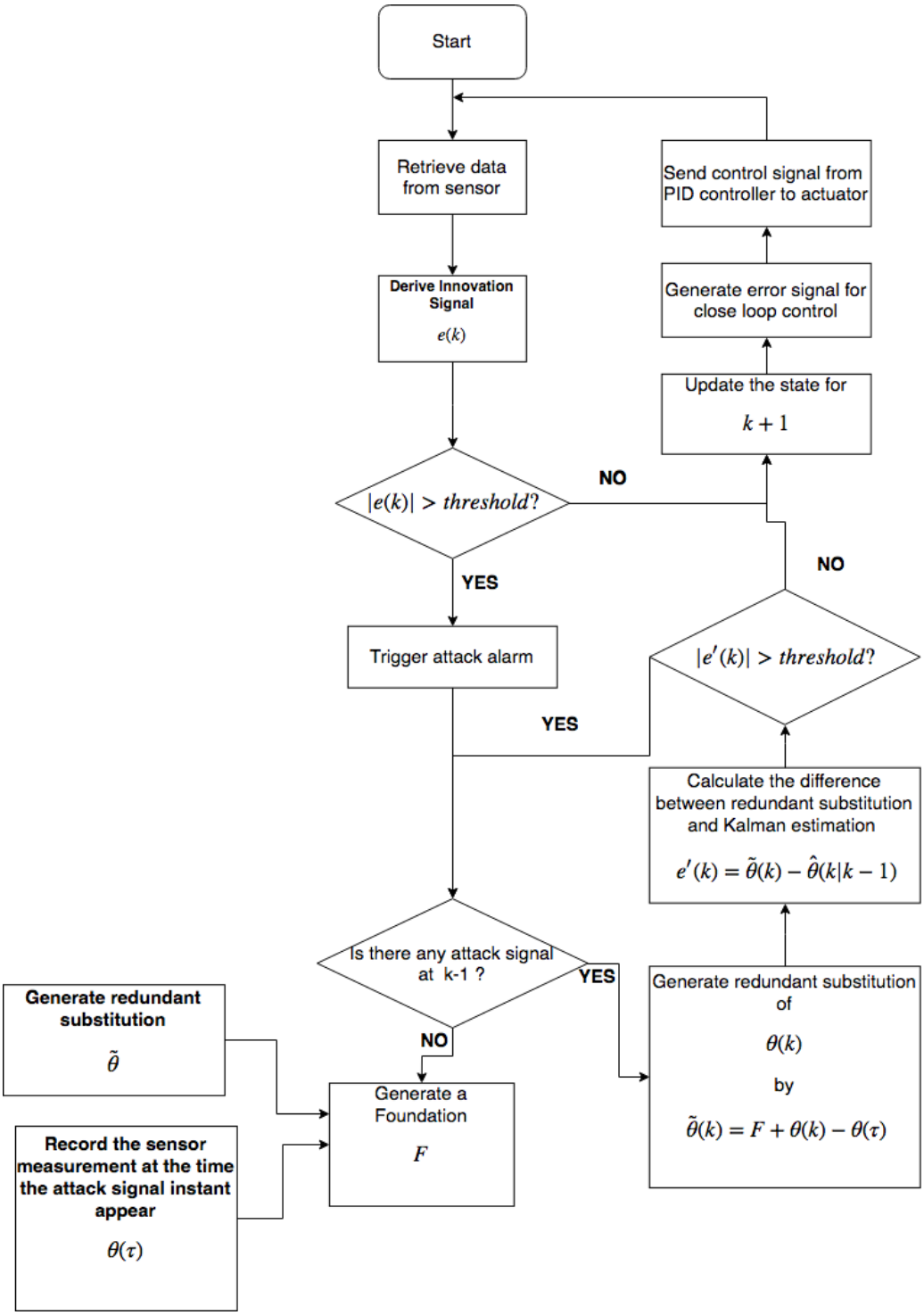

Figure 45: Correction process for time-varying sensor attack

7.3.2 Test of Correction under Time-varying Attack

In this section, we create two methods of generating time-varying attack signals. The first method is to pre-define an attack signal denoted by **Method 1**, which is finished by programming thus we can have the knowledge of attack signal in advance. The second method is to control the value and appearance of the attack signal by a gamepad denoted by **Method 2**, which simulates the scenario that the attack signal can be launched without any discipline. In this section, we present the result of correction for steering and driving systems under the time-varying attack signal finished by a different method.

1. Correction of the steering subsystem under the time-varying attack signal (**Method 1**)

We create an aperiodic square wave attack signal. The experiment result is given as figure 46.

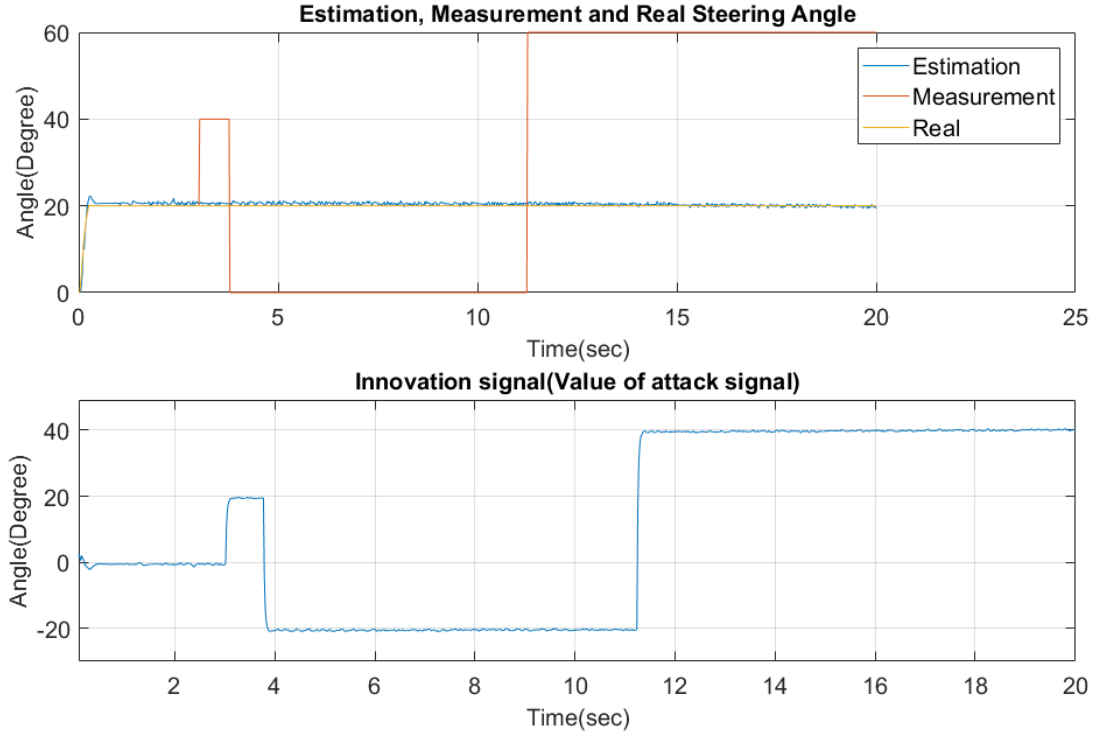

Figure 46: Correction for steering subsystem under time-varying attack signal achieved by **Method 1**

And the result shows the correction algorithm can identify and correct the attack signal very well where the estimation and real steering angle are unchanged, and the innovation signal outlines the change of the attack signal.

2. Correction of driving subsystem under time-varying attack signal (**Method 1**)

For driving subsystem, we create a time-varying attack signal as a square wave with larger amplitude. And the testing result is given in figure 47.

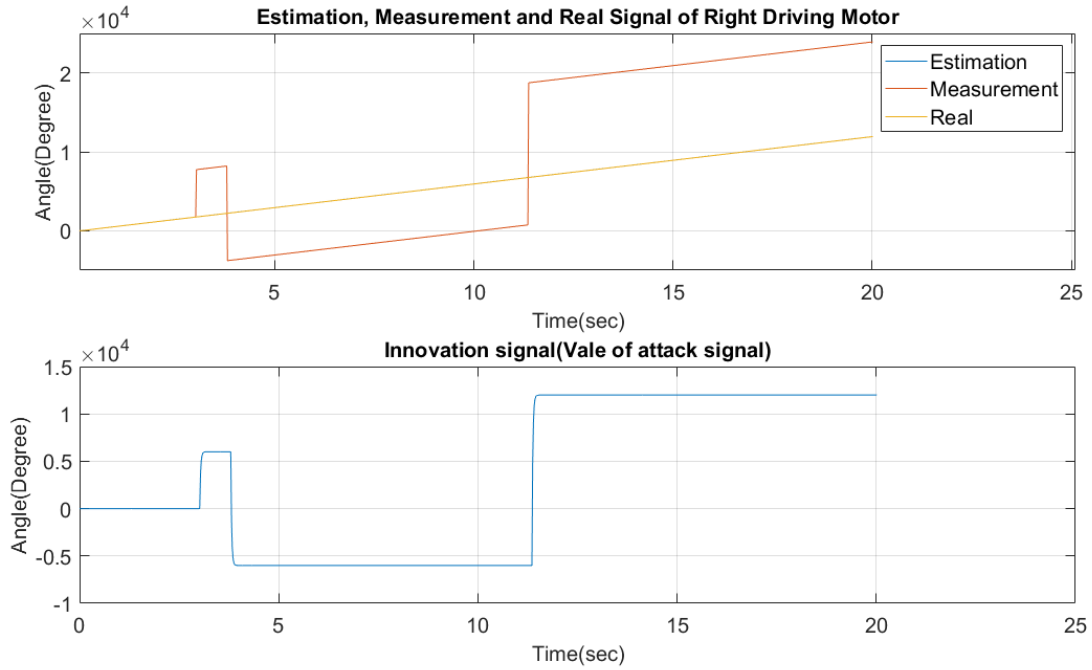

Figure 47: Correction for driving subsystem under time-varying attack signal achieved by **Method 1**

where the attack signal is in the same form with figure 46 multiplied by 300 times. And the measurement is given obvious false and isolated by the correction algorithm. The Kalman filter can still make the correct estimation, and the real angle of the driving motor keeps following the reference angle. In this test, we give the driving subsystem $600°/s$ reference angular velocity, and both the estimation and the real angle of driving motor show the final value is 12000° after 20 seconds' controlling, which meets the expectation that the real value of the sensor measurement can be recovered.

3. Correction of steering subsystem under time-varying attack signal (**Method 2**)

In **Method 2**, the reference signal is fixed and we use an Xbox gamepad to vary the attack signal in real time, where the time-varying attack signal is generated manually. The configuration of the gamepad shown in figure 48.

Figure 48: Gamepad for control attack signal

where we apply different attack value regarding the steering subsystem to buttons: 'A', 'B', 'X', and 'Y': Let 'A' be attack of 20°; let 'B' be attack of −40°; let 'X' be attack of 60°; let 'Y' be attack of −80°. For the driving system, Let 'A' be attack of 200°; let 'B' be attack of −400°; let 'X' be attack of 6000°; let 'Y' be attack of −8000°. We simulate the behaviour of the attacker to launch the attack signal manually and casually, and the correction test for the steering subsystem is given in figure 49.

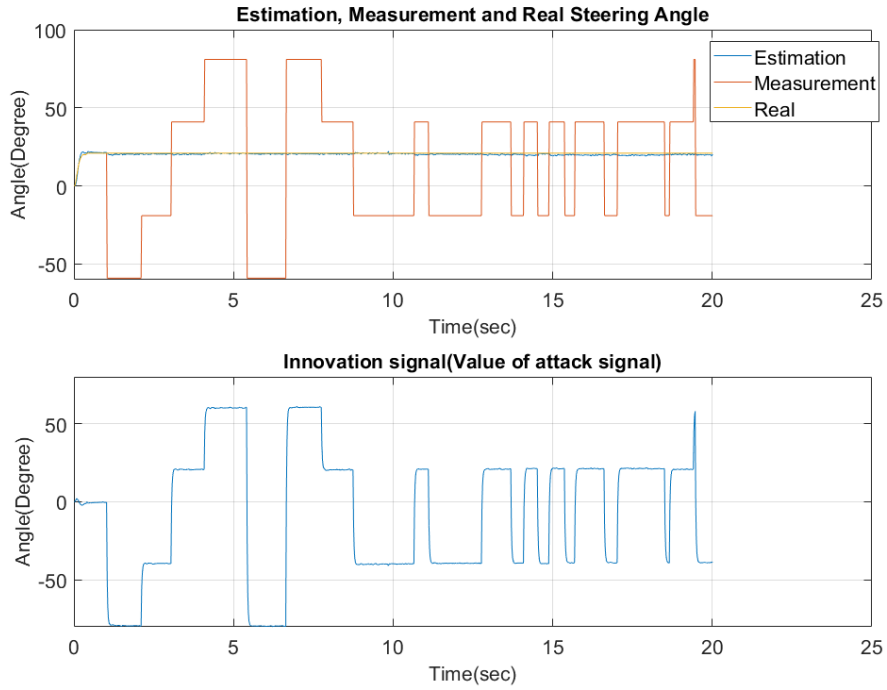

Figure 49: Correction for steering subsystem under time-varying attack signal acheved by **Method 2**

4. Correction of the driving subsystem under the time-varying attack signal (**Method 2**)

The reference angular velocity is set to $600°/s$, and the correction test is shown in figure 50.

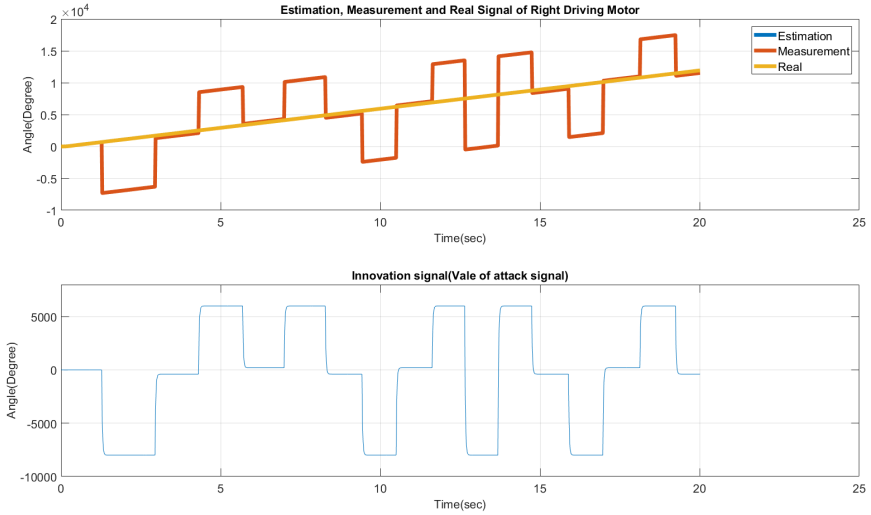

Figure 50: Correction for driving subsystem under time-varying attack signal achieved by **Method 2**

8 Comprehensive Test

In the previous section, we design and test the detection and correction solution for the Lego car demo under a simple case, where both the reference signal of steering and driving subsystems are constant (The reference steering angle and reference angular velocity of driving motors did not change over time). To prove the generality and reliability of the security solution on account of normal driving of a driverless vehicle, we increased the complexity of the test, where we use joysticks of a gamepad to control the reference angle of one steering subsystem and the reference angular velocity of two driving subsystems. This acts as remote operation of the control centre of General CPS and the remote driver can manually orient the car demo and regulate its speed. Meanwhile, we use the buttons of the gamepad to enable the attack signals with the same multiple electable values set presented by **Method 2**. The test results are given for steering and driving subsystem respectively.

8.1 Test on Steering System

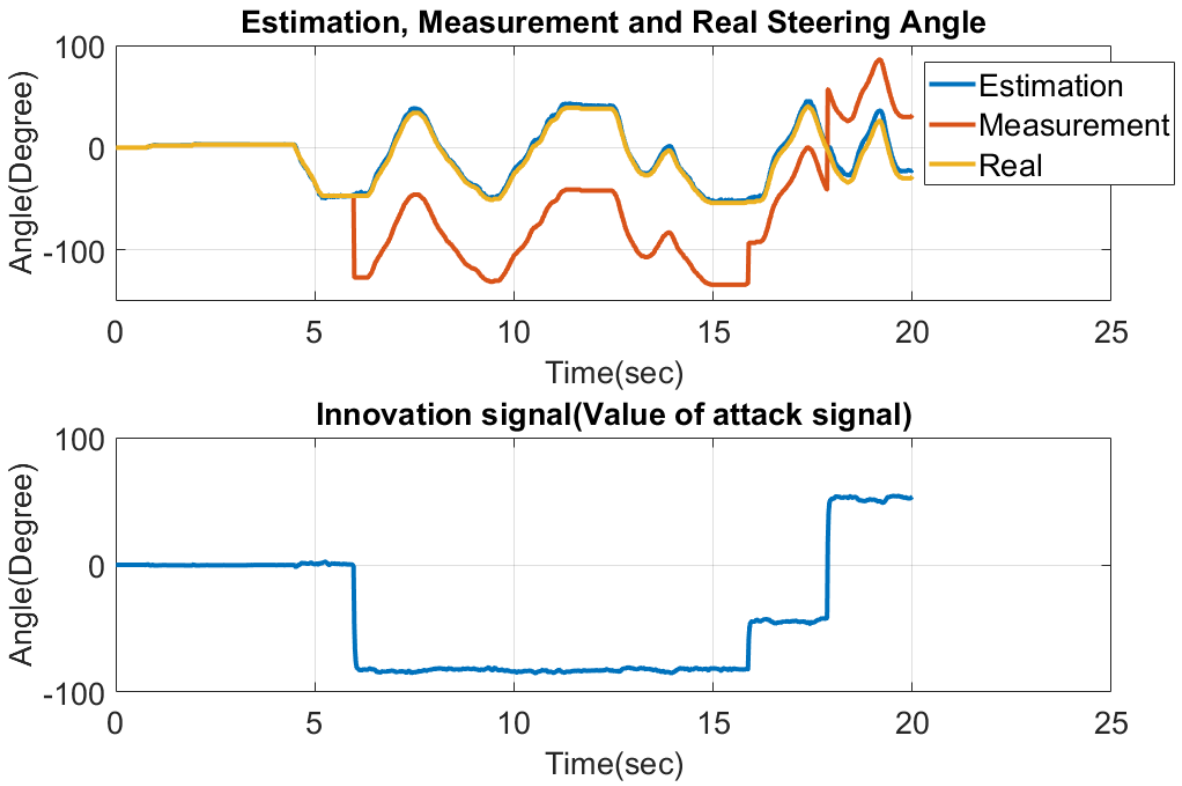

Figure 51: General time-varying attack and defence model for steering subsystem

From figure 51, we can see that the steering angle can be changed according to control centre at any time, and the time-varying sensor attack appears at around 6s and keeps existing. However, the detector can identify the attack signal with small uncertainty (given in the second figure), and the real steering angle does not suffer from the sensor attack. The steering angle estimation is close to the real steering angle (given in the first figure). Visually, with the help of the detection and correction solution, the steering motor of the car demo will not be tampered by sensor attack.

8.2 Test on Driving System

This report uses only the right driving motor for demonstration, but the detection and correction schemes for the left driving subsystem is identical with the right driving subsystem. The test result is shown in figure 52.

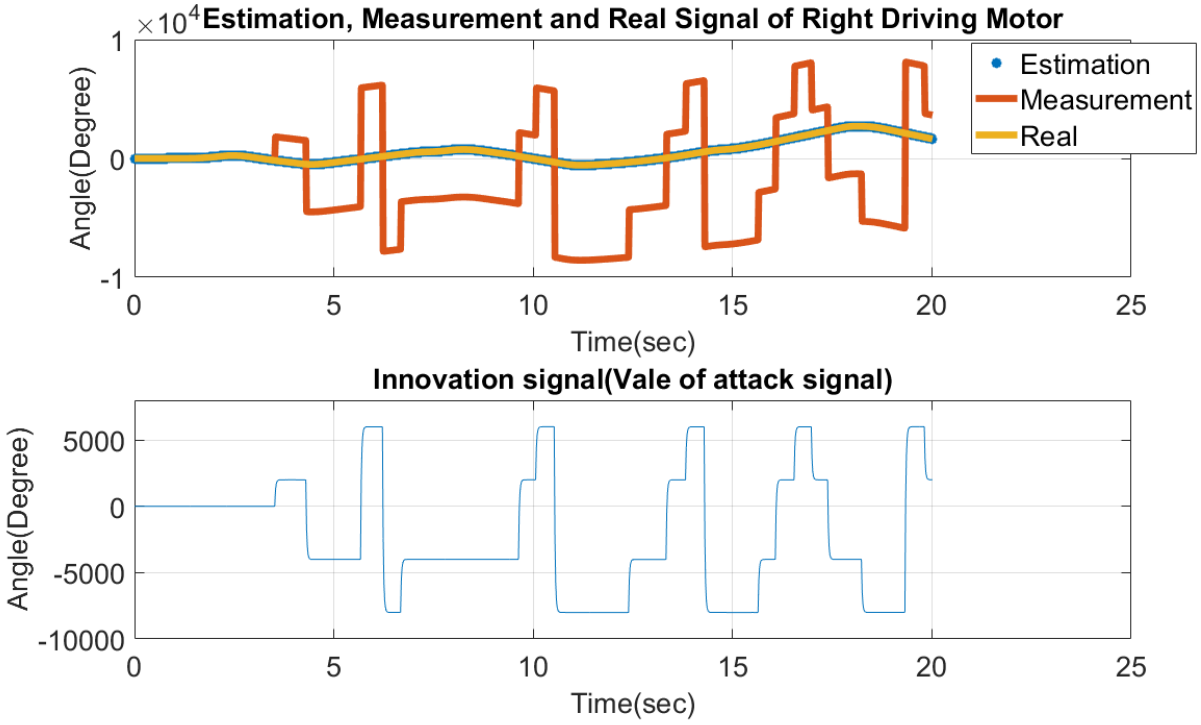

Figure 52: General time-varying attack and defence model for driving subsystem

We can see the driving motor can drive the car with different velocities (shown in the first figure) without being influenced by the attack signal (shown in the second figure).

9 Discussion and Future Work

9.1 Defects

In this project, the security solution including detection and correction is shown effectively against DoS and False Data Injection attacks. However, there are two main limitations can be found existing in this security solution which may cause failure of the detection and correction.

Defect 1: The detection scheme depends highly on the detection threshold. A small detection threshold guarantees the success and accuracy of detection. To make the threshold as small as possible, an extremely precise system model of the device (servomotor) is needed. However, because of the uncertainty of sampling time interval and the error involved during the step of system parameters identification, the system parameters cannot be measured very accurately and the innovation signal (distance between output estimation and sensor measurement) is hard to be minimised in this case. On the other hand, if the threshold for a detector is too small, it is possible for occurrence of false alarm, which could happen during the period that the Kalman gain is not completely converged, which is why we can find in many test results across this report that the maximal innovation value is mostly found at the start of detector. It is found that for a large detection threshold, some carefully designed attack signals can easily bypass the detection if the amplitude of attack signals is always smaller than the value of threshold [14]. Therefore, in our project, the main challenge is to more accurately derive system parameters, so that the estimate of Kalman filter will give a smaller innovation signal, then we can further decrease the detection threshold.

Defect 2: At the stage of sensor attack correction, we introduce the redundancy of an attacked sensor. The redundancy is defined as the reliable feedback signal for the attacked system derived from the measurements of the other sensors. However, in strict sense, the redundancy is sometimes unreliable considering the process of the transient in a closed-loop system. For example, since we have three individual closed-loop systems, they will respond to their given reference signals, and the transient is the period before they reach the state state. During the transient state, if one of sensors is attacked, the derived redundancy will contain much randomness. Therefore, in designing the correction algorithm and testing, we have to assume the attack signal can only access to a sensor when the corresponding servomotor reaches in steady state, which decreases the generality.

9.2 Future Work

1. Future work for defect 1

To overcome the limitations found in this projects, in the real application, it is recommended to setup the security solution on high-quality platform with accurate and a fixed sampling rate. It is also recommended to use the industrial-grade sensing device which can give an instant and precise measurement. In terms of the Kalman filter, it is important that a precise state-space model can be obtained as well as the characteristics of measurement and the process noise. A higher order smoothing filter can be designed to upgrade the performance of smoothing the Kalman estimate so that the threshold can be made smaller. In some other investigations such as [39], [25], [43] and [42], more complicated detection mechanisms are presented which involve long-term detection strategies, and are shown more resilient in terms of small-scale False Data Injection attacks.

2. Future work for defect 2

From [45],[42] and [44], we are inspired that to increase the successful rate of sensor attack correction and isolate the attack signal, it is recommended to develop the integrated level of the physical system. For example, the reason why the correction algorithm in this project is not reliable in some cases is that the steering and driving subsystems are lack of interaction, in other words, there is no factor involving the state of two driving motors to generate the angle of the steering motor and vice-versa. These three subsystems are highly uncorrelated except for in the case of steady state. A better configuration for a driverless vehicle is to increase the integrated level across each actuator so that the system can be shown has higher Security Index and restorability [46]. On the other hand, an alternative and straightforward improvement for the correction algorithm in this project could be designing desired controllers for three subsystems so that the transient period of their closed-loop control can be compressed extremely short and it would be a tiny possibility for arrivals of the attack signal happening during the transient. Therefore, less errors would be introduced by the correction algorithm.

10 Conclusion

The CPS is widely-used which improves efficiency and management of industry, and it is also involved in almost all aspects of life. However, incidents occurred in recent years are drawing more and more attention on security issues of CPS. The CPS attack is usually released via the Internet network. In recent years, some technologies already exist in preventing attacks from hacking in the system. However, it is also critical to develop some methods to prevent the attack from affecting actual physical systems when attacks are already successfully penetrated through networks.

This project uses the Lego Mindstorms NXT 2.0 to design a prototype driverless car which is connected with the host computer. Based on the designed system, we simulated the car demo and got results when attacks imposed on sensors and tampered sensor measurements. In this project, we also designed a security solution to detect and correct this car demo under attacks, which provides a design reference for the security issue of the CPS

The main achievements of this project are as follows:
(1) Review and explain the working mechanism of the CPS and understand its potential safety hazards;

(2) Use the Lego Mindstorm NXT 2.0 robotics product to assemble a prototype car demo which imitates a real vehicle in life containing three sets of motors and angular sensors;

(3) Based on the CPS framework, achieve three closed-loop control of three motors through MATLAB toolbox developed by (RWTH) Aachen University;

(4) Choose the DoS attack and False Data Injection attack patterns and demonstrate their consequences on the car demo;

(5) A sensor attack detector model is designed based on the theory of Kalman filter, and its performance is optimised on increasing the detection accuracy by applying a smoothing filter;

(6) Through the correlation of sensors in the CPS, the data correction algorithm is designed by redundancy information under attack of the sensor;

(7) When successfully compiling the detection and correction algorithm into the car, we analyse the performance and defects of this security solution through the car demo's behaviour and also prospect this design.

From testing results, it can be seen that the detection and reconstruction algorithm can be conducted successfully for attacks which are larger than a certain value. When an attack is detected, the attack signal will be isolated so that the false data will not be sent back as a feedback signal, which will be replaced by the calculated signal from redundancy information. Therefore, the car demo can still go in the correct trajectory. The limitation of the designed algorithm is that for relatively small attack signals, the algorithm is not able to detect and reconstruct. Furthermore, the system is not highly integrated so that the capability of reconstruction for this system is limited. Increasing the degree of integration of the system and improving the accuracy of the system are essential when better performance in the attack detection and reconstruction is expected. This design is an attempt at solving the CPS safety problems in the auto-driving car project. Although we can find the CPS applied in different areas, all CPSs have the similar structure. The achievement of this project can be further extended to other CPSs.

References

[1] S. M. Kay, "Fundamentals of statistical signal processing, volume i: Estimation theory (v. 1)," *PTR Prentice-Hall, Englewood Cliffs*, 1993.

[2] S. Amin, A. A. Cárdenas, and S. S. Sastry, "Safe and secure networked control systems under denial-of-service attacks," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2009, pp. 31–45.

[3] H. Gill, "From vision to reality: cyber-physical systems," in *HCSS National Workshop on New Research Directions for High Confidence Transportation CPS: Automotive, Aviation, and Rail*, 2008.

[4] R. D. B. J. S. B. C. Song, Houbing, *Cyber-Physical Systems*. London: Academic Press, 2017.

[5] L. Wang, M. Törngren, and M. Onori, "Current status and advancement of cyber-physical systems in manufacturing," *Journal of Manufacturing Systems*, vol. 37, pp. 517–527, 2015.

[6] R. Von Solms and J. Van Niekerk, "From information security to cyber security," *computers & security*, vol. 38, pp. 97–102, 2013.

[7] Y. Ashibani and Q. H. Mahmoud, "Cyber physical systems security: Analysis, challenges and solutions," *Computers & Security*, vol. 68, pp. 81–97, 2017.

[8] A. Cardenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig, S. Sastry *et al.*, "Challenges for securing cyber physical systems," in *Workshop on future directions in cyber-physical systems security*, vol. 5, 2009.

[9] A. W. Werth, "Towards distinguishing between cyber-attacks and faults in cyber-physical systems," Ph.D. dissertation, Vanderbilt University, 2014.

[10] V. Gunes, S. Peter, and T. Givargis, "Modeling and mitigation of faults in cyber-physical systems with binary sensors," in *Computational Science and Engineering (CSE), 2013 IEEE 16th International Conference on*. IEEE, 2013, pp. 515–522.

[11] L. Chen and P. Zhang, "Fault detection based on wavelet transform and its application in cps," in *Control and Automation, 2002. ICCA. Final Program and Book of Abstracts. The 2002 International Conference on*. IEEE, 2002, pp. 218–218.

[12] H. Zhang, P. Cheng, L. Shi, and J. Chen, "Optimal dos attack scheduling in wireless networked control system," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 3, pp. 843–852, 2016.

[13] A. M. Malla and R. K. Sahu, "Security attacks with an effective solution for dos attacks in vanet," *International Journal of Computer Applications*, vol. 66, no. 22, 2013.

[14] K. Manandhar, X. Cao, F. Hu, and Y. Liu, "Detection of faults and attacks including false data injection attack in smart grid using kalman filter," *IEEE transactions on control of network systems*, vol. 1, no. 4, pp. 370–379, 2014.

[15] S. H. Kim and J. W. Jeon, "Introduction for freshmen to embedded systems using lego mindstorms," *IEEE Transactions on Education*, vol. 52, no. 1, pp. 99–108, 2009.

[16] B. S. Heck, N. S. Clements, and A. A. Ferri, "A lego experiment for embedded control system design," *IEEE control systems*, vol. 24, no. 5, pp. 61–64, 2004.

[17] A. Behrens, L. Atorf, R. Schwann, B. Neumann, R. Schnitzler, J. Balle, T. Herold, A. Telle, T. G. Noll, K. Hameyer *et al.*, "Matlab meets lego mindstormsâĂŤa freshman introduction course into practical engineering," *IEEE Transactions on Education*, vol. 53, no. 2, pp. 306–317, 2010.

[18] R. A. University. (2012) Rwth aachen mindstorms nxt toolbox for matlab. [Online]. Available: http://www.mindstorms.rwth-aachen.de/tra

[19] S. S. Haykin *et al.*, *Kalman filtering and neural networks*. Wiley Online Library, 2001.

[20] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear estimation*. Prentice Hall, 2000, no. EPFL-BOOK-233814.

[21] B. D. Anderson and J. B. Moore, "Optimal filtering," *Englewood Cliffs*, vol. 21, pp. 22–95, 1979.

[22] S. S. Haykin, *Adaptive filter theory*. Pearson Education India, 2008.

[23] M. S. Grewal, "Kalman filtering," in *International Encyclopedia of Statistical Science*. Springer, 2011, pp. 705–708.

[24] P. J. Escamilla-Ambrosio and N. Mort, "Multi-sensor data fusion architecture based on adaptive kalman filters and fuzzy logic performance assessment," in *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, vol. 2. IEEE, 2002, pp. 1542–1549.

[25] C. Kwon, W. Liu, and I. Hwang, "Security analysis for cyber-physical systems against stealthy deception attacks," in *American Control Conference (ACC), 2013*. IEEE, 2013, pp. 3344–3349.

[26] Y. Kim, "Control systems lab using a lego mindstorms nxt motor system," *IEEE Transactions on Education*, vol. 54, no. 3, pp. 452–461, 2011.

[27] C. Budaciu and L.-D. Apostol, "Dynamic analysis and control of lego mindstorms nxt bicycle," in *System Theory, Control and Computing (ICSTCC), 2016 20th International Conference on*. IEEE, 2016, pp. 145–149.

[28] B. H. Ferri, S. Ahmed, J. E. Michaels, E. Dean, C. Garyet, and S. Shearman, "Signal processing experiments with the lego mindstorms nxt kit for use in signals and systems courses," in *American Control Conference, 2009. ACC'09.* IEEE, 2009, pp. 3787–3792.

[29] H.-S. Chuang, Y.-C. Chuang, and C.-H. Yang, "Development of a low-cost platform for control engineering education," in *Machine Learning and Cybernetics (ICMLC), 2014 International Conference on*, vol. 2. IEEE, 2014, pp. 444–448.

[30] L. H. Prayudhi, A. Widyotriatmo, and K.-S. Hong, "Wall following control algorithm for a car-like wheeled-mobile robot with differential-wheels drive," in *Control, Automation and Systems (ICCAS), 2015 15th International Conference on*. IEEE, 2015, pp. 779–783.

[31] J. U. Liceaga-Castro, I. I. Siller-Alcalá, J. Jaimes-Ponce, and R. Alcántara-Ramírez, "Series dc motor modeling and identification," in *2017 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*. IEEE, 2017, pp. 248–253.

[32] M. Canale and S. C. Brunet, "A lego mindstorms nxt experiment for model predictive control education," in *Control Conference (ECC), 2013 European*. IEEE, 2013, pp. 2549–2554.

[33] R. A. University. (2012) Pc-nxt communication. [Online]. Available: http://www.mindstorms.rwth-aachen.de/documents/downloads/doc/version-4.07/pc_nxt_communication.html

[34] A. Visioli, *Practical PID control*. Springer Science & Business Media, 2006.

[35] Y. Mo, E. Garone, A. Casavola, and B. Sinopoli, "False data injection attacks against state estimation in wireless sensor networks," in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 5967–5972.

[36] R. K. Mehra and J. Peschon, "An innovations approach to fault detection and diagnosis in dynamic systems," *Automatica*, vol. 7, no. 5, pp. 637–640, 1971.

[37] R. Isermann, "Process fault detection based on modeling and estimation methodsâĂŤa survey," *automatica*, vol. 20, no. 4, pp. 387–404, 1984.

[38] G. H. B. Foo, X. Zhang, and D. M. Vilathgamuwa, "A sensor fault detection and isolation method in interior permanent-magnet synchronous motor drives based on an extended kalman filter," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 8, pp. 3485–3495, 2013.

[39] X. Wei, M. Verhaegen, and T. van Engelen, "Sensor fault detection and isolation for wind turbines based on subspace identification and kalman filter techniques," *International Journal of Adaptive Control and Signal Processing*, vol. 24, no. 8, pp. 687–707, 2010.

[40] A. Behrens, L. Atorf, and T. Aach, "Teaching practical engineering for freshman students using the rwth-mindstorms nxt toolbox for matlab," *Matlab-Modelling, Programming and Simulations, SCIYO*, pp. 41–65, 2010.

[41] D. Simon, "Kalman filtering," *Embedded systems programming*, vol. 14, no. 6, pp. 72–79, 2001.

[42] R. H. Chen and J. L. Speyer, "Sensor and actuator fault reconstruction," *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 2, pp. 186–196, 2004.

[43] M. Zhu and S. Martínez, "On the performance analysis of resilient networked control systems under replay attacks," *IEEE Transactions on Automatic Control*, vol. 59, no. 3, pp. 804–808, 2014.

[44] M. Roshanzadeh and S. Saqaeeyan, "Error detection & correction in wireless sensor networks by using residue number systems," *International Journal of Computer Network and Information Security*, vol. 4, no. 2, p. 29, 2012.

[45] Z. Tang, M. Kuijper, M. Chong, I. Mareels, and C. Leckie, "Linear system security–detection and correction of adversarial attacks in the noise-free case," *arXiv preprint arXiv:1711.05400*, 2017.

[46] M. S. Chong and M. Kuijper, "Vulnerability of linear systems against sensor attacks–a system's security index," *arXiv preprint arXiv:1602.06594*, 2016.

APPENDICES

All the MATLAB scripts including the LEGO NXT toolbox, the car demo driving script and the detection-and-correction program are copied in the flash drive.

Derivation of Sensor Redundancy

The sensor attack correction method is presented in section 7, where we present the concept of replace the false sensor data with redundancy. The functions of extracting the redundancy for servomotor in each subsystem are given by:

1. Assume the sensor of steering system is under attack:

$$\theta_{st}(k) = \hat{\theta}_{st}(k) + a(k) \tag{62}$$
$$\omega_{dr}(k) = \hat{\omega}_{dr}(k) \tag{63}$$
$$\omega_{dl}(k) = \hat{\omega}_{dl}(k) \tag{64}$$
$$\omega_{in}(k) = max(\omega_{dr}(k), \omega_{dl}(k)) \tag{65}$$
$$\omega_{in}(k) = min(\omega_{dr}(k), \omega_{dl}(k)) \tag{66}$$
$$\tilde{\theta}_s t(k) = \tan^{-1}\left(\frac{L/r}{(\omega_{in}(k) + \omega_{out}(k))/(2(\omega_{out}(k) - \omega_{in}(k)))}\right) \tag{67}$$

2. Assume the sensor of right driving system is under attack

$$\theta_{dr}(k) = \hat{\theta}_{dr}(k) + a(k) \tag{68}$$
$$\omega_{st}(k) = \hat{\omega}_{st}(k) \tag{69}$$
$$\omega_{dl}(k) = \hat{\omega}_{dl}(k) \tag{70}$$
$$\tilde{\omega}_{dr}(k) = \omega_{dl}(k)\frac{L/\tan|\theta_{st}(k)| + r/2}{L/\tan|\theta_{st}(k)| - r/2} \quad \text{for } \theta_{st}(k) > 0 \tag{71}$$
$$\tilde{\omega}_{dr}(k) = \omega_{dl}(k)/\frac{L/\tan|\theta_{st}(k)| + r/2}{L/\tan|\theta_{st}(k)| - r/2} \quad \text{for } \theta_{st}(k) < 0 \tag{72}$$
$$\tilde{\theta}_{dr}(k) = \hat{\theta}_{dr}(k-1) + \tilde{\omega}_{dr}(k)\Delta t \tag{73}$$

3. Assume the sensor of left driving system is under attack

$$\theta_{dl}(k) = \hat{\theta}_{dl}(k) + a(k) \tag{74}$$

$$\omega_{st}(k) = \hat{\omega}_{st}(k) \tag{75}$$

$$\omega_{dr}(k) = \hat{\omega}_{dr}(k) \tag{76}$$

$$\tilde{\omega}_{dl}(k) = \omega_{dr}(k)\frac{L/\tan|\theta_{st}(k)| + r/2}{L/\tan|\theta_{st}(k)| - r/2} \quad \text{for } \theta_{st}(k) < 0 \tag{77}$$

$$\tilde{\omega}_{dl}(k) = \omega_{dr}(k)/\frac{L/\tan|\theta_{st}(k)| + r/2}{L/\tan|\theta_{st}(k)| - r/2} \quad \text{for } \theta_{st}(k) > 0 \tag{78}$$

$$\tilde{\theta}_{dl}(k) = \hat{\theta}_{dl}(k-1) + \tilde{\omega}_{dl}(k)\Delta t \tag{79}$$