600 Jahre Universität Rostock

Traditio et Innovatio

Bachelorarbeit zum Thema

Sicherheitsanalyse durch Entwicklung eines Rogue Device zur Echtzeitmanipulation maritimer Steuerungssysteme

Studiengang:	Informatik
Vorgelegt von:	Jakob Engelbert Tomahogh
Matrikelnummer:	221201101
Bearbeitungszeitraum:	15. November 2024 – 04. April 2025
Erstgutachter:	M. Sc. Marvin Davieds
Zweitgutachter:	Prof. Dr. rer. nat. Clemens H. Cap

Lehrstuhl für Informations- und Kommunikationsdienste
Institut für Informatik
Fakultät für Informatik und Elektrotechnik
Universität Rostock

Gedruckt auf säurefreiem und elementar-chlorfrei gebleichtem Papier.

Stand:: 30. Dezember 2024

Inhalt

1 Einleitung **1**

1.1 Motivation 1

1.2 Ziel der Arbeit 1

2 Grundlagen **2**

2.1 CanBus 2

2.2 Raspberry Pi 3

2.2.1 Raspberry Pi als Rogue Device 3

2.3 State of the Art 3

2.3.1 Anbindung von Raspberry Pi an CanBus 4

2.3.2 Übersetzung von CanBus Nachrichten 4

3 Konzept und Systemdesign **5**

3.1 Aufbau Schiffsysteme 5

3.2 Steuerungslogik des Controllers 5

3.3 Integration des Rogue Device 7

4 Implementierung **8**

4.1 Verbindung Rogue Device - Controller 8

4.2 Übersetzung Signale Controller - Schiff 8

4.3 Eingabe-Interface 9

5 Sicherheit **10**

5.1 Schwachstellen 10

5.2 Schutzmaßnahmen 10

5.3 Relevanz für andere Schiffe 10

6 Abschließende Betrachtung **11**

6.1 Fazit 11

6.2 Ausblick 11

VII Anhang **12**

VII.1 Quellcode 12

VII.2 Schaltpläne 12

Abbildungen **13**

Literatur

Kurzzusammenfassung

Im traditionellen Sinne bezieht sich der Begriff Typografie auf die Gestaltung von Druckwerken mit beweglichen Lettern (Typen). Anfänglich fand dies insbesondere im Bleisatz bzw. dem Satz mit Holzlettern statt.

In der Medientheorie steht Typografie für gedruckte Schrift in Abgrenzung zu Handschrift (Chirografie) und elektronischen sowie nicht literalen Texten.

Heute bezeichnet Typografie meist den medienunabhängigen Gestaltungsprozess, der mittels Schrift, Bildern, Linien, Flächen und Leerräumen alle Arten von Kommunikationsmedien gestaltet. Typografie ist in Abgrenzung zu Kalligrafie, Schreiben oder Schriftentwurf das Gestalten mit vorgefundenem Material.

Abstract

In the traditional sense, the term typography refers to the design of printed works with movable letters (types). Initially, this was done in lead typesetting or wood typesetting.

In media theory, typography stands for printed type in contrast to handwriting (chirography) and electronic as well as non-literal texts.

Today, typography usually refers to the media-independent design process that uses type, images, lines, surfaces and empty spaces to create all kinds of communication media. In contrast to calligraphy, writing or type design, typography is the design with found material.

Kapitel 1

Einleitung

In diesem Kapitel

1.1 Motivation 1
1.2 Ziel der Arbeit 1

1.1 Motivation

Warum ist das Thema wichtig? Warum sollte sich jemand damit beschäftigen? Was ist das Ziel der Arbeit?

1. CanBus ist weit verbreitet in Fahrzeugen, noch nicht ausreichend in Schiffen auf Sicherheit bedacht.
2. In großen Schiffen kann es durchaus vorkommen, dass unbemerkt Personen physischen Zugriff auf Teile des Schiffes haben, die für die Steuerung relevant sind.

1.2 Ziel der Arbeit

1. Entwicklung eines Rogue Device, das in der Lage ist, die Kommunikation auf einem CanBus zu manipulieren.
2. Sicherheitsanalyse der Auswirkungen auf die Steuerung eines Schiffes.
3. Dadurch soll aufgezeigt werden, wie wichtig es ist, die Kommunikation auf einem CanBus zu schützen und Aufmerksamkeit auf die Sicherheit von Schiffen zu lenken.

Kapitel 2

Grundlagen

In diesem Kapitel

2.1 CanBus 2
2.2 Raspberry Pi 3
2.3 State of the Art 3

2.1 CanBus

Wie wird ein CanBus verdrahtet und wie kommunizieren die Geräte? Was ist ein CanBus:

- Technologie für serielle Netzwerke
- 1983 von Bosch für die Autoindustrie entwickelt
- Ist ein zweiadriger Bus, halbduplex
- konventionellen seriellen Technologien überlegen in Funktionalität und Zuverlässigkeit
- Kosteneffiizienter
- Entwickelt für Echtzeitanwendungen mit 1Mbit/s Baudrate
- Verwendung mittlwerweile allen möglichen Fahrzeugen, auch maritimer Bereich und Luftfahrt
- Medizinische Geräte, Industrieanlagen, Gebäudeautomation

[Vos08, Seiten 2-10]
Aufbau: Alle Knoten sind mit zwei Drähten verbunden und sind gleichberechtigt. [Vos08, Seite 132]
Die Nachrichten werden nach Broadcasting-Prinzip übertragen. Jede Nachricht wird von allen Knoten empfangen, aber nur die Knoten, die die Nachricht benötigen, verarbeiten sie. Diese werden aber nicht bestätigt, da dies zu einer größeren Last auf dem Bus führen würde. Bei einer fehlerhaften Nachricht reagieren die Knoten mit einer Fehlermeldung, die wieder der gesamte Bus empfängt. [Vos08, Seite 80]

2.2 Raspberry Pi

- Ein Raspberry Pi ist ein Einplatinencomputer, der von der Raspberry Pi Foundation entwickelt wurde.
- mit diesem kann man viele Dinge machen, wie z.B. programmieren, Musik hören, Videos schauen, im Internet surfen, etc.
- Der Raspberry Pi hat viele Anschlüsse, wie z.B. USB, HDMI, Ethernet, Audio, etc.
- eignet sich gut um einfache Aufgaben zu erledigen, wie z.B. eine Webseite hosten, einen Fileserver betreiben, etc.
-

2.2.1 Raspberry Pi als Rogue Device

Unter einem Rogue Device versteht man ein Gerät, welches sich unautorisiert und unauffällig in ein Netzwerk einbindet. Dies kann ein Raspberry Pi sein, der sich in ein Netzwerk einbindet und Daten abgreift oder manipuliert. Hierüber können sich Angreifer Zugriff auf das Netzwerk verschaffen.

- eigener Code kann auf Rogue Device laufen
- kann von Angreifern von außen gesteuert werden
- kann auch genuzt werden um Informationen über das Netzwerk zu sammeln

2.3 State of the Art

- Was gibt es schon für Lösungen?
- Was ist der aktuelle Stand der Technik?
- Was ist der aktuelle Stand der Forschung?

2.3.1 Anbindung von Raspberry Pi an CanBus

- Raspbian OS hat seit 05.05.2015 eingebundenen Support für den Mikrochip MCP251x

[SKJ16]

- PiCan2 ist ein CanBus Board für den Raspberry Pi
- HAT (Hardware Attached on Top) Standard
- erlaubt dem Raspberry Pi mit dem CanBus zu kommunizieren mit einer Geschwindigkeit von bis zu 1Mbit/s

[PL19]

2.3.2 Übersetzung von CanBus Nachrichten

- J1939 (DBC-Dateien) (https://docs.fileformat.com/de/database/dbc/)
- NMEA 0183
- NMEA 2000 → CanBoat (https://github.com/canboat/canboat)

Kapitel 3

Konzept und Systemdesign

In diesem Kapitel

3.1 Aufbau Schiffsysteme 5
3.2 Steuerungslogik des Controllers 5
3.3 Integration des Rogue Device . 7

3.1 Aufbau Schiffsysteme

Wie werden die Motoren im Schiff angesteuert? Welche Angriffsmöglichkeiten gibt es? Wie wird das Ruder angesteuert? Gibt es noch weitere wichtige Systeme?

3.2 Steuerungslogik des Controllers

Der benutzte Controller ist ein XBox Series X Controller. Dieser wurde gewählt, da er eine gute Haptik hat und viele Tasten besitzt. Zusätzlich ist er kabellos und kann somit frei bewegt werden. Um die Steuerung des Schiffes zu ermöglichen, müssen die Eingabgen des Controllers in Steuerbefehle umgewandelt werden. Dies passiert auf dem Raspberry Pi. Der Controller wird über Bluetooth mit dem Raspberry Pi verbunden. Dort werden die Eingaben des Controllers ausgelesen und in einem Python-Script in Steuerbefehle umgewandelt.
Um eine intuitive Steuerung zu ermöglichen, muss das Konzept des Controllers gut durchdacht sein. Da der XBox Controller viele Tasten besitzt und die Steuerung des Schiffes nicht zu komplex sein soll, werden einige Tasten nicht verwendet.

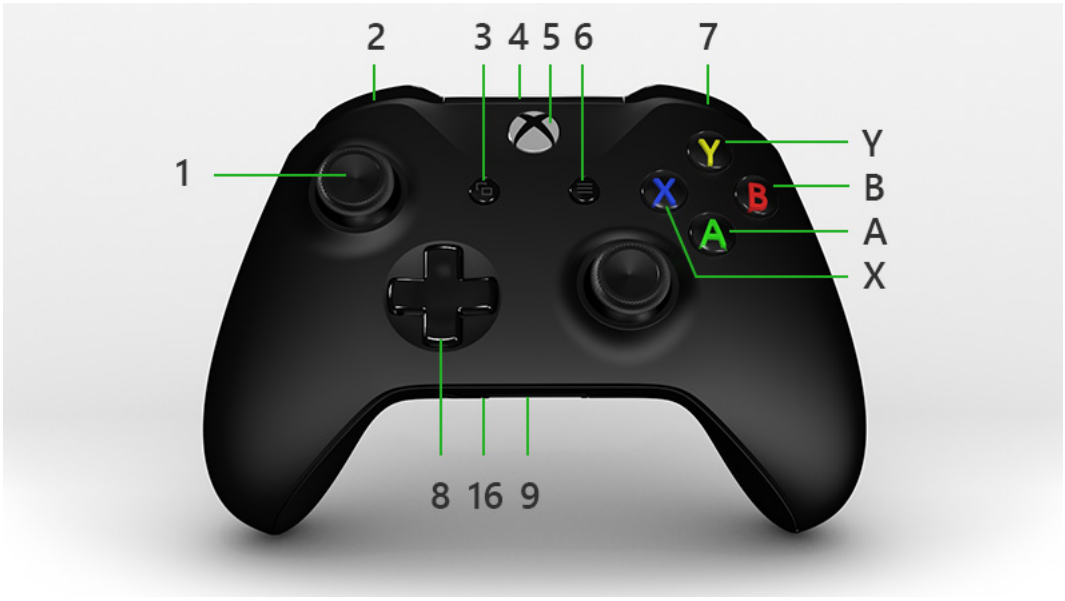

Abbildung 3.1: Vorderseite des Controllers

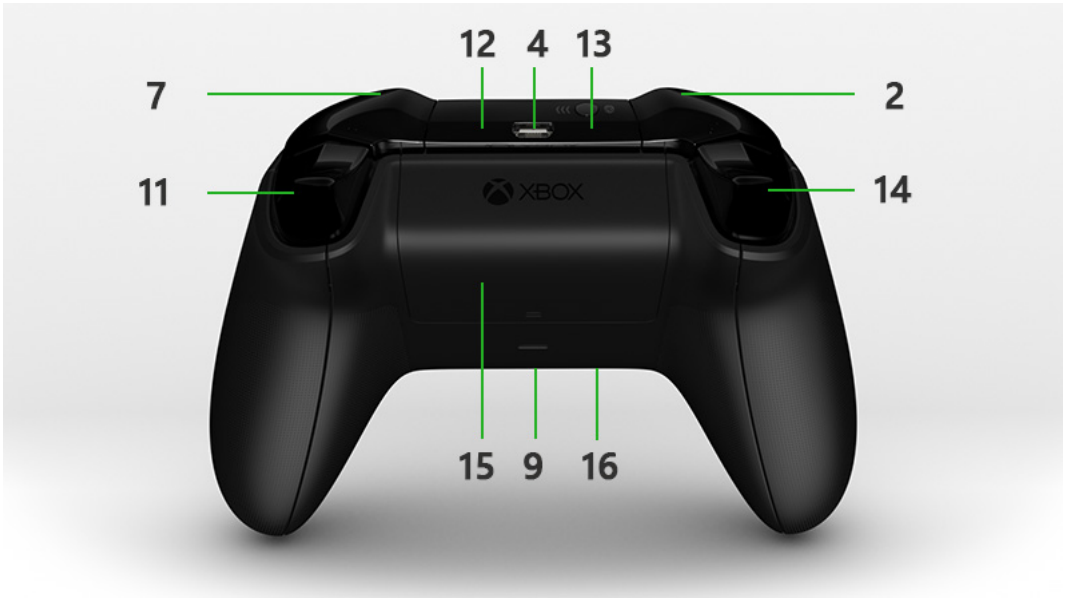

Abbildung 3.2: Rückseite des Controllers

Nummerierung der Taste	Funktion
1	1
1	1

3.3 Integration des Rogue Device

Wie soll es mit dem Controller und dem Schiff verbunden sein?
Was muss ich dabei beachten? Muss eine Rückmeldung für die Eingaben geschehen? Wenn ja, wie? (kleiner OLED-Bildschirm oder App)

Kapitel 4

Implementierung

In diesem Kapitel

4.1 Verbindung Rogue Device - Controller 8
4.2 Übersetzung Signale Controller - Schiff 8
4.3 Eingabe-Interface 9

4.1 Verbindung Rogue Device - Controller

- benutzt wird Python 3.12.8, da einfache Syntax und gute Bibliotheken
- Bibliothek: Pygame für die Controllereingabe (https://www.pygame.org/docs/(abgerufen am 08.12.2024))
- Pygame braucht ein GUI, andersfalls wird es Fehler ausgeben
- Beispielscode für Pygame: github (abgerufen am 08.12.2024)

Benutzte Hardware, Protokolle, Libraries

4.2 Übersetzung Signale Controller - Schiff

Welches Dateiformat wird für Controllersignale benutzt? Wie werden diese effizient genug in Motorsignale übersetzt? Kann ich einfach originale Steuerungssignale unterdrücken?

Es gibt zuerst ein Programm, welches die Signale des Controllers erhält und in passende Variablen in Python interpretiert. Diese Variablen müssen dann in Nachrichten für den Can-Bus umgewandelt werden. Hierfür gibt es ein weiteres Programm. Damit die beiden Programme miteinander kommunizieren können, wird Inter-Process-Communication (IPC) benutzt. Als Methode werden hierbei Pipes benutzt. Diese sind einfach zu implementieren und haben eine automatische Synchronisierung zwischen den Prozessen. Das bedeutet, dass die Prozesse nicht aufeinander warten müssen, sondern einfach weiterarbeiten können. Es wird durch den Puffer der Pipe sichergestellt. Wenn dieser voll ist, wird der schreibende Prozess angehalten, bis der lesende Prozess den Puffer geleert hat. Dies ist ein einfaches und effizientes Verfahren, um die beiden Prozesse zu synchronisieren [VJ15].

- Anfangsbeispiel: geeksforgeeks (abgerufen am 18.12.2024)
- Python: thelinuxcode (abgerufen am 19.12.2024)

4.3 Eingabe-Interface

Wie wird die Rückmeldung tatsächlich aussehen?

Kapitel 5

Sicherheit

In diesem Kapitel

5.1 Schwachstellen 10
5.2 Schutzmaßnahmen 10
5.3 Relevanz für andere Schiffe . . 10

5.1 Schwachstellen

Welche habe ich benutzt und welche weiteren möglichen Schwachstellen habe ich gefunden?

5.2 Schutzmaßnahmen

Welche gibt es bereits? Was sind weitere Möglichkeiten?

5.3 Relevanz für andere Schiffe

Gibt es solche Angriffsmöglichkeiten auch auf anderen Schiffen?

Kapitel 6

Abschließende Betrachtung

In diesem Kapitel

6.1 Fazit 11
6.2 Ausblick 11

6.1 Fazit

Was wurde geschafft? Was kann damit ausgesagt werden?

6.2 Ausblick

Wo kann noch weiter geforscht werden? Was wurde nicht geschafft?

Kapitel VII

Anhang

In diesem Kapitel

VII.1 Quellcode 12
VII.2 Schaltpläne 12

VII.1 Quellcode

VII.2 Schaltpläne

Abbildungen

3.1 Vorderseite des Controllers 6
3.2 Rückseite des Controllers 6

Literatur

[PL19] Sudarshan Pant und Sangdon Lee. „Design and Implementation of a CAN Data Analysis Test Bench based on Raspberry Pi“. In: *Journal of Multimedia Information System* 6.4 (Dez. 2019), S. 239–244. ISSN: 2383-7632. DOI: 10.33851/jmis.2019.6.4.239 (siehe S. 4).

[SKJ16] A. A. Salunkhe, Pravin P Kamble und Rohit Jadhav. „Design and implementation of CAN bus protocol for monitoring vehicle parameters“. In: *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. IEEE, Mai 2016, S. 301–304. DOI: 10.1109/rteict.2016.7807831 (siehe S. 4).

[VJ15] Aditya Venkataraman und Kishore Kumar Jagadeesha. „Evaluation of inter-process communication mechanisms“. In: *Architecture* 86.64 (2015) (siehe S. 9).

[Vos08] Wilfried Voss. *A comprehensible guide to controller area network*. Copperhill Media, 2008 (siehe S. 3).

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, sind als solche kenntlich gemacht.

Die Arbeit ist noch nicht veröffentlicht und ist in ähnlicher oder gleicher Weise noch nicht als Prüfungsleistung zur Anerkennung oder Bewertung vorgelegt worden.

Rostock, 30. Dezember 2024