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:: 18. Januar 2025

Inhalt

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

Jedes Jahr werden zahlreiche Passagiere mit Schiffen befördert. Dabei kann es vorkommen, dass bösartige Personen ein Teil dieser Passagiere sind. Ein Angriff auf die Steuerung eines Schiffes könnte katastrophale Folgen haben. Dennoch wird die Sicherheit von Schiffen oft vernachlässigt. Wenn die Kommunikation auf einem Can-Bus nicht ausreichend geschützt ist, könnte ein Angreifer durch physischen Zugriff auf das Schiff die Kommunikation manipulieren und so die Steuerung des Schiffes übernehmen. Auf großen Passagierschiffen ist das Risiko besonders hoch, da es schwierig ist, einen solchen Angriff zu bemerken.

1.2 Ziel der Arbeit

1. Dadurch soll aufgezeigt werden, wie wichtig es ist, die Kommunikation auf einem Can-Bus zu schützen und Aufmerksamkeit auf die Sicherheit von Schiffen zu lenken.

Die Arbeit erforscht die Möglichkeit eines solchen Angriffs. Dazu wird ein Rogue Device entwickelt, welches in der Lage ist externe Steuerungsbefehle zu erhalten und mit diesen Kontrollnachrichten auf die Schiffskommunikation zu senden. Dabei wird auf einen Can-Bus zugegriffen, der die Kommunikation zwischen den verschiedenen Steuerungssystemen des

Schiffes ermöglicht. Als Protokolle spielen dabei NMEA 2000, NMEA 0183 und J1939 eine wichtige Rolle. Unter anderem wird auch auf eine serielle Schnittstelle zugegriffen, um mehr Kontrolle über das System zu erlangen.
Hier soll die Machbarkeit eines solchen Angriffs gezeigt werden. Zusätzlich werden die Auswirkungen auf die Steuerung des Schiffes analysiert. Es soll ein Bewusstsein für die Sicherheit von Schiffen geschaffen werden und mögliche Gegenmaßnahmen vorschlagen.

Kapitel 2

Grundlagen

In diesem Kapitel

2.1 CanBus

Bei einem Can-Bus handelt es sich um eine serielle Netzwerktechnologie, welche mehrere Geräte mit einem Draht verbindet. Die Entwicklung des Can-Bus wurde von Bosch im Jahr 1983 begonnen. 1986 wurde der erste Can-Bus Standard veröffentlicht. Die Motivation der Entwicklung war die effiziente Kommunikation zwischen den Steuergeräten in einem Auto. Als günstige Nebenwirkung konnte damit die Kabelmenge reduziert werden, da alle Geräte mit einem Bus verbunden werden können. Durch die höhere Zuverlässigkeit und Funktionalität des Can-Bus, wurde dieser schnell in der Autoindustrie etabliert. Aber auch in anderen Sektoren, wie z.B. der Medizintechnik, der Gebäudeautomation oder der Luftfahrt, spielt der Can-Bus mittlwerweile eine wichtige Rolle. [Vos08, Seiten 2-10]
Man sprich von einem Bussystem, da alle Geräte gleichberechtigt sind. Dass heißt, dass jedes Gerät Nachrichten senden und empfangen kann. 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 nicht bestätigt, da dies zu einer größeren unnötigen Last auf dem Bus führen würde. Bei einer fehlerhaften Nachricht reagieren die Knoten mit einer Fehlermeldung, die wieder der gesamte Bus empfängt. Wenn ein Knoten dauerhaft fehlerhafte Nachrichten sendet, wird dieser vom Bus getrennt. Die auf dem Bus gesendeten Daten werden mit einer Nachrichten-ID versehen, die

die Priorität der Nachricht angibt. Die Nachrichten mit der niedrigsten ID haben die höchste Priorität. Die Maximale Länge einer Nachricht beträgt 8 Byte. Durch die vergleichsweise geringe Länge der Nachrichten, kann eine geringe Latenz erreicht werden. Dabei kann eine höchste Baudrate von 1Mbit/s gesetzt werden. [Vos08, Seiten 13-19]
Alle Knoten in einem Can-Bus sind mit einem Zweiadrigen Kabel verbunden. Diese werden als High(CAN_H) und Low(CAN_L) bezeichnet. Der Bus ist an beiden Enden mit einem Widerstand von 120 Ohm abgeschlossen um Reflexionen zu vermeiden. [Vos08, Seite 132]

Data-Frames eines Can-Bus:

- Start of Frame (SOF): Startbit
- Arbitration Field: Identifier
- Control Field: wird zur Datengröße und Nachrichtslänge verwendet
- Data Field: eigentliche Nutzdaten
- CRC Field: Prüfsumme
- End of Frame (EOF): Stopbit

[Vos08, Seite 36]

Erweitertes Can-Protokoll: Standard Can-Protokoll hat 11 Bit Identifier, erweitertes Can-Protokoll hat 29 Bit Identifier. Higher-Layer-Protokolle SAE J1939 wurde für die Kommunikation in Nutzfahrzeugen entwickelt. SAE = Society of Automotive Engineers
Identifier wurde mit J1939 auf 29 Bit erweitert um mehr verschiedene Nachrichten zu ermöglichen.
Auf einem Can-Bus können der Standard 11 Bit Identifier und der erweiterte 29 Bit Identifier gleichzeitig verwendet werden. Wenn zwei Nachrichten den gleichen 11 Bit Identifier haben, wird die Nachricht mit dem 11 Bit Identifier bevorzugt immer die höhere Priorität haben.

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
- UCAN ist ein USB-CAN Adapter, der es ermöglicht ein beliebiges USB-Gerät mit dem CanBus zu verbinden

[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)
- cantools Python Library

Kapitel 3

Konzept und Systemdesign

In diesem Kapitel

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?

- einzelnen Can-Bus für jeden Gashebel
- Serielle Kommunikation bei Autopilot
- Autopilot kann nur das Ruder steuern

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.

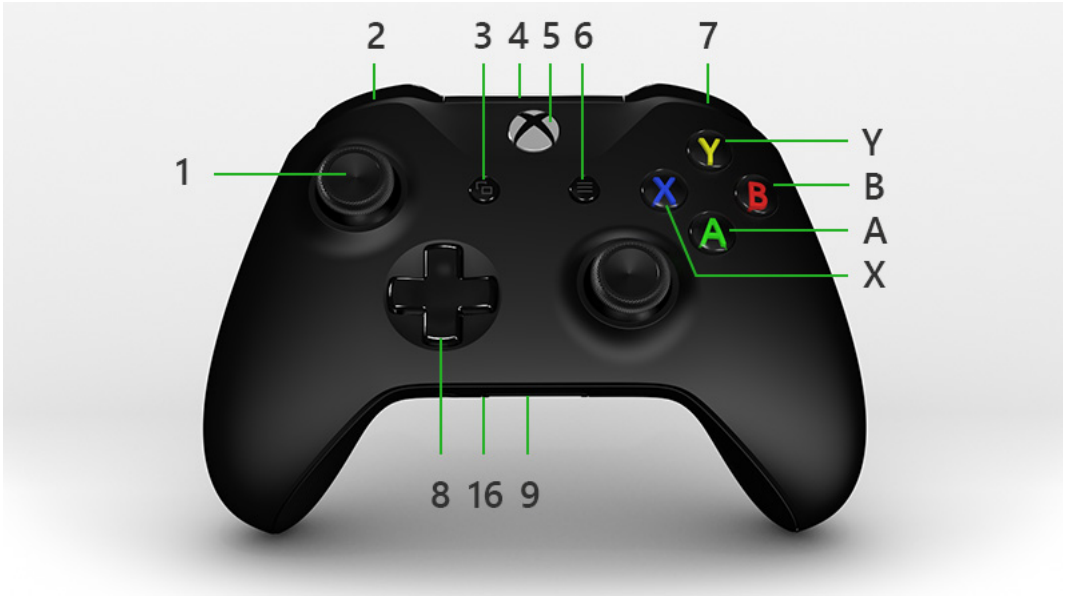

Abbildung 3.1: Vorderseite des Controllers

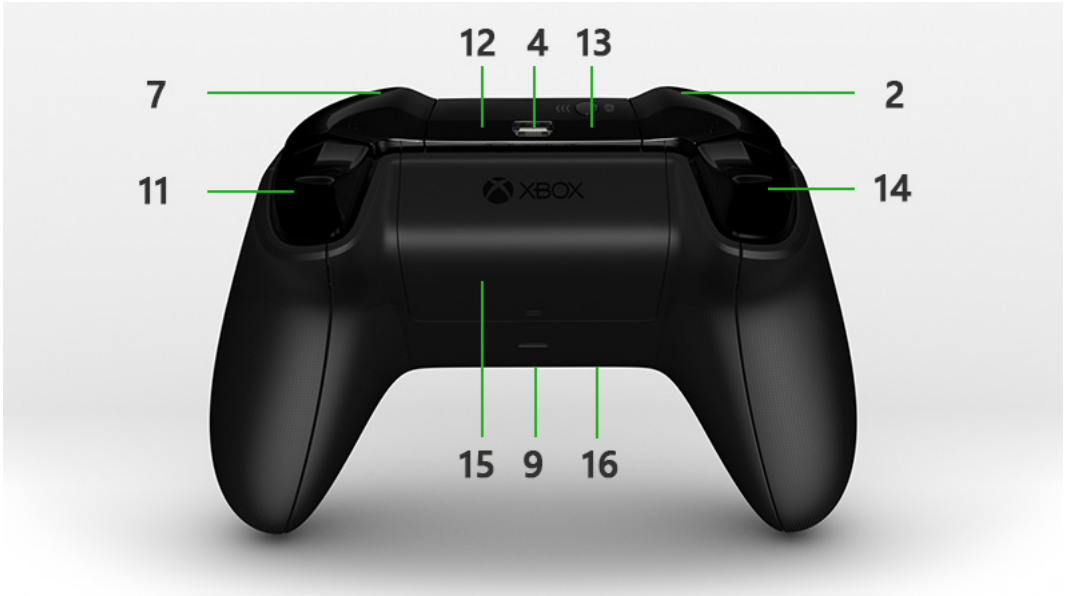

Abbildung 3.2: Rückseite des Controllers

Nummerierung der Taste	Funktion
1	Bewegung des Ruders
2	Reduzierung der linken Gashebelposition
7	Reduzierung der rechten Gashebelposition
11	Erhöhung der rechten Gashebelposition
14	Erhöhung der linken Gashebelposition
B + 2 + 7	Einlegen des Rückwärtsgangs

3.3 Integration des Rogue Device

Damit der Controller die Steuerbefehle an das Schiff senden kann, muss das Rogue Device in das System integriert werden. In diesem Fall ist das Rogue Device der Raspberry Pi. Damit dieser möglichst unbemerkt in das System integriert werden könnte, muss der Controller drahtlos verbunden werden. Um die Kommunikation von dem Rogue Device zu dem Schiff zu ermöglichen, müssen die einzelnen Systeme angesteuert werden. Um die Gashebelposition zu verändern, wird der Raspberry Pi mit dem Can-Bus des Schiffes verbunden.
Sollte der Gashebel in der normalen Benutzung vom Schiffsführer benutzt werden, wird ein Signal an den Can-Bus gesendet. Dieses Signal wird dann an die Motoren weitergeleitet. Um diese Eingabe zu verhindern, muss auf diese Nachricht geachtet und reagiert werden. Mit dem Lesen des Can-Bus kann die entsprechende Nachricht entdeckt werden. Dann kann eine Nachricht von dem Rogue Device gesendet werden, um die Gashebelposition zu überschreiben.
Um eine Rückmeldung zu erhalten, wie die gewünschte Gashebel- und Ruderposition ist, sollte mit einer einfachen App gearbeitet werden. Diese App kann dann die gewünschten Positionen anzeigen. Ein kleiner OLED-Bildschirm könnte auch benutzt werden. Allerdings hat dieser den Nachteil, dass er physisch an den Raspberry Pi angeschlossen werden muss. Damit ist keine Rückmeldung möglich, wenn dieser als Rogue Device in dem System versteckt angeschlossen ist. Um die Kommunikation zwischen dem Raspberry Pi und dem Handy mit der App zu ermöglichen, wird Bluetooth benutzt.

- Was passiert bei Veränderung der Gashebelposition?
- Wie passiert die Rückmeldung

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

- 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))
- Beispielscode für Pygame: github (abgerufen am 08.12.2024)

Im ersten Schritt wird der Controller mit dem Raspberry Pi verbunden. Der Raspberry Pi wird mit Raspberry Pi OS betrieben. Dieses Betriebssystem ist eine auf Debian basierende Distribution, die speziell für den Raspberry Pi entwickelt wurde. Als Standard-Bluetooth-Treiber wird BlueZ verwendet. Dieser ist bereits vorinstalliert und muss nicht extra installiert werden. Da es sich in diesem Fall um einen XBox Controllers handelt, müssen die richtigen Treiber (xboxdrv) installiert werden. Diese können im apt-Repository gefunden werden. Zusätzlich muss der Enhanced Re-Transmission Mode (ERTM) deaktiviert werden. Dieser Modus ist standardmäßig aktiviert und verhindert die korrekte Verbindung des Controllers. Zum Schluss soll der Controller auf die aktuelle Firmware geupdatet werden. Dies kann

über die Xbox Accessories App gemacht werden, allerdings ist dies nur auf Windows möglich. Nun kann der Controller mit dem Raspberry Pi verbunden werden. Dies geschieht über die Bluetooth-Einstellungen des Raspberry Pi. Um die Verbindung zu testen, kann eine Webapplikation benutzt werden. Diese zeigt die Eingaben des Controllers an. Zum Beispiel: hardwaretester (abgerufen am 02.01.2025)
Als Programmiersprache wird Python 3 benutzt. Dies ist eine einfache Sprache, die viele Bibliotheken hat. Für die Controller-Eingaben wird die Bibliothek Pygame benutzt. Diese Bibliothek ist einfach zu benutzen und hat viele Funktionen. Sobald der Controller mit dem Raspberry Pi verbunden ist, wird dieser von Pygame erkannt. Die Eingaben des Controllers können dann in Variablen gespeichert werden. Dabei gilt es zu beachten, dass der Gashebel oder die Ruderposition nicht zu schnell verändert werden. Dies könnte zu unerwünschten Ergebnissen führen. Daher werden diese Werte mit Tasten des Controllers eingegeben, welche nicht nur eine binäre Eingabe haben. Diese werden als Achsen bezeichnet. Diese ermöglichen eine stufenlose Eingabe. Bei einer vollständigen Eingabe soll die Gashebelposition nach 20 Sekunden 100% erreichen. So ist sichergestellt, dass die Gashebelposition nicht zu schnell verändert wird. Die Ruderposition soll nach 2 Sekunden in jede Richtung jeweils 100% erreichen. Dies ist ein Kompromiss zwischen Geschwindigkeit und Genauigkeit.

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)
- encoding von canbus nachrichten in Python: cantools

4.3 Eingabe-Interface

Wie wird die Rückmeldung tatsächlich aussehen?

Kapitel 5

Sicherheit

In diesem Kapitel

5.1 Schwachstellen

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

- Kommunikation auf dem Can-Bus ist nicht verschlüsselt
- Keine Authentifizierung der Nachrichten
- Einfach, weiteres Gerät in das System zu integrieren
- Serielle Schnittstelle am Autopiloten unverschlüsselt
- lediglich physischer Zugriff notwendig
- Daten in bestimmten Format, allerdings mit recht wenig Aufwand zu verstehen

5.2 Schutzmaßnahmen

Welche gibt es bereits?

- richtige Baudrate muss eingestellt sein
- keine dedizierten Schutzmaßnahmen auf dem Forschungsschiff

Was sind weitere Möglichkeiten?

- Verschlüsselung der Kommunikation
- Authentifizierung der Nachrichten
- Überwachung der Kommunikation
- regelmäßige Überprüfung der Kommunikation
- Auf größeren Schiffen: Trennung der Passagiernetzwerke von den Steuerungssystemen

5.3 Relevanz für andere Schiffe

Gibt es solche Angriffsmöglichkeiten auch auf anderen Schiffen?

- Can-Bus häufig genutzt, häufig unverschlüsselt
- besonders auf größeren Schiffen ist das Ruder nicht mehr mechanisch, sondern elektronisch
- Angriff auf das Steuerungssystem könnte katastrophale Folgen haben

Kapitel 6

Abschließende Betrachtung

In diesem Kapitel

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

VII.2 Schaltpläne

Abbildungen

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. 5).

[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. 5).

[VJ15] Aditya Venkataraman und Kishore Kumar Jagadeesha. „Evaluation of inter-process communication mechanisms“. In: *Architecture* 86.64 (2015) (siehe S. 11).

[Vos08] Wilfried Voss. *A comprehensible guide to controller area network*. Copperhill Media, 2008 (siehe S. 3, 4).

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, 18. Januar 2025