

# Claims Investigation Committee Multi-Input Testing Device

ECE-4820: Electrical and Computer Engineering Design II

Dylan-Matthew Garza   Daniel Baker   Rohullah Sah

Department of Electrical and Computer Engineering  
Western Michigan University

ZF Group  
Auburn Hills, MI

Fall 2024



Faculty Advisor:  
Dr. Janos Grantner

Sponsor Manager:  
Patrick McNally

## Table of Contents

- 1 Introduction
    - ZF
    - Claims Investigation Committee
    - Project Motivation
    - Our Solution
    - Key Devices Under Test
  - 2 Design and Implementation
    - Project Solution and Overview
    - Hardware Design
    - Device Interfacing and Testing
    - Embedded Linux With Yocto Project
    - Inter-Processor Communication
    - Web Application and Server
  - 3 Verification
  - 4 Challenges
  - 5 Future Work
  - 6 Future Work
  - 7 Closing

ZF

ZF

## Who is ZF?

- Global technology company and Tier 1 automotive supplier
  - Provides advanced safety systems and vehicle control solutions
  - Partners with major OEMs: Daimler, Chrysler, Tesla, Waymo(Google), etc.
  - A leading innovator in commercial vehicle technology

## North American Headquarters

- Project based at ZF Group's North American headquarters in Auburn Hills, MI
  - Specializes in commercial vehicle solutions
  - This facility is also home to the Claims Investigation Center



**Figure 1: Source: google.com**

## Claims Investigation Committee

## Claims Investigation Committee

## What is the Claims Investigation Center?

## CIC's Role in Our Project

## Project Motivation

## Motivation for the Project

## Challenges with Current Testing Methods

- Testing on current brake system platform (mBSP) was built and industrialized specifically for that platform
  - Long lead time and significant cost to release and document
  - New platform components are not compatible with the current tester
  - Current tester is not capable of testing in prototype phase

## Need for Improvement

- The Brake Signal Transmitter's (BST) implementation in Daimler's new platform intensifies urgency
  - High production volumes require efficient testing methods
  - Expanding product line increases testing complexity



Figure 2: Current brake component system tester

#### Our Solution

## Multi-Input Testing Device

#### **Addressed Challenges**

- Provides a unified testing platform for multiple devices
  - Flexible and agile to adapt to new product lines
  - Allows for prototype testing and validation
  - Automates data collection and analysis to reduce time and errors
  - Increases testing speed and accuracy
  - Simplifies validation process for warranty claims
  - Enhances capability to analyze field returns efficiently
  - Cost-effective solution

## Key Devices Under Test

## Devices Our Solution Supports

## Key Devices Under Test (DUTs)

## 1. Brake Signal Transmitter (BST)

- Primary focus - critical new component for 2025 production
  - Acts as the brain that reads how hard a driver presses the brake

## 2. Continuous Wear Sensor (CWS)

- Works like a monitor for your brake pads and discs
  - Warns when brakes are wearing down using voltage

### 3. Pressure Sensor

- Continuously measures relative pressure in vehicle control systems

#### **4. Electronic Stability Control Module (ESCM)**

- Acts as a safety system that helps prevent skidding and rollovers
  - Monitors the vehicle's movement and intervenes to keep it stable

# Table of Contents

## 1 Introduction

- ZF
- Claims Investigation Committee
- Project Motivation
- Our Solution
- Key Devices Under Test

## 2 Design and Implementation

- Project Solution and Overview
- Hardware Design
- Device Interfacing and Testing
- Embedded Linux With Yocto Project
- Inter-Processor Communication
- Web Application and Server

## 3 Verification

## 4 Challenges

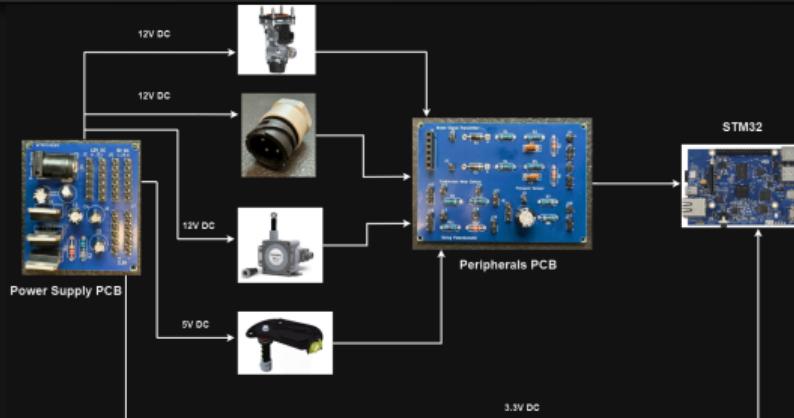
## 5 Future Work

## 6 Future Work

## 7 Closing

Project Solution and Overview

## Project Solution



#### **What this project aims to accomplish:**

## 1. Device Interfacing

## 1.1 Properly read Device Signals using the ARM Cortex-M4 on the onboard microcontroller on the STM32MP157F-DK2:

- PWM duty cycle
  - Frequency
  - Voltages through an analog-to-digital converter (ADC)
  - CAN frames

## 2. Physical Components and Hardware

## 2.1 Printed Circuit Board (PCB) for interfacing with DUT

## 2.2 PCB for scaling and managing power for the DUT and to the microcontroller

### 2.3 Enclosure for PCBs and STM32MP157F-DK2 board

## Project Solution (cont.)

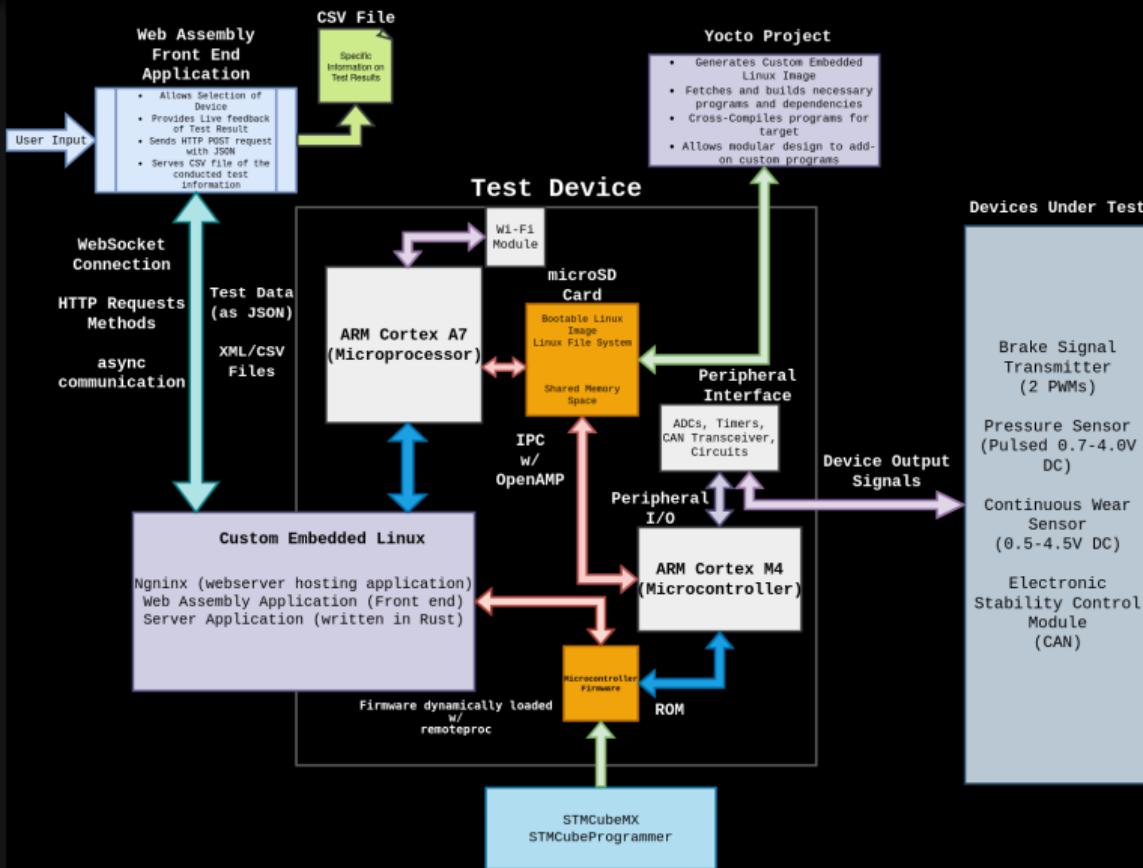
## What this project aims to accomplish:

### 3. Software

- 3.1 Custom embedded **Linux** distribution that will run on the onboard ARM Cortex-A7 microprocessor on the **STM32MP157F-DK2**
  - 3.2 Load firmware to Cortex-M4
  - 3.3 Simple user interface on web-based application
  - 3.4 Custom Webserver to process information from web application to microcontroller
  - 3.5 Communicate collected information from ARM Cortex-M4 to ARM Cortex-A7
  - 3.6 Ability to download measured data, formatted as a CSV, through the web application

## Project Solution and Overview

# Comprehensive System Block Diagram



## Project Solution and Overview

## Gantt Chart



## Project Solution and Overview

## Budget Projection

Project Title		Multi-Signal Automotive Testing Device					
Date		11/22/2024					
Category	Item	Quantity	Estimated Under/Over	Unit Price	Shipping + Tax	Total Costs	Description
HARDWARE	STM32MP157-DK2	4	-	\$109.00	\$0.00	\$436.00	ARM Cortex A7 & ARM Cortex M4
	String Potentiometer	1	-	\$50.00	\$0.00	\$50.00	Detect and measure linear displacement
	Continuous Wear Sensor	2	-	\$335.20	\$0.00	\$670.40	Monitors the wear of brake pads
	Continuous Wear Sensor Harness	2	-	\$0.00	\$0.00	\$0.00	Wear Sensor Connector
	Linear Position Sensor	1	-	\$50.00	\$0.00	\$50.00	Measures the position
	USB to CAN Cable	2	-	\$47.99	\$10.00	\$105.98	Converts USB to CAN
	Pressure Sensor	2	-	\$0.00	\$0.00	\$0.00	Sensor for measuring pressure data
	Electronic Stability Control Module	1	-	\$0.00	\$0.00	\$0.00	Data for vehicle stability
	Brake Signal Transmitter	1	-	\$0.00	\$0.00	\$0.00	Brake Pedal to receive signals
	Anti-static Wrist Band	1	-	\$7.95	\$0.00	\$7.95	Grounding Wristband
	SD Card Reader	1	-	\$20.00	\$0.00	\$20.00	SD Card Reader
	MINI360 Buck Converter	2	-	\$6.99	\$13.66	\$27.64	Voltage Supply Regulator
	LM78xx Buck Converter	1	-	\$13.99	\$0.00	\$13.99	Voltage Supply Regulator
	LM2596 Buck Converter	1	-	\$12.89	\$0.00	\$12.89	Voltage Supply Regulator
	50 Values Resistor Kit	1	-	\$12.99	\$0.00	\$12.99	Signal Conditioning Components
	24 Electrolytic Capacitors	1	-	\$9.99	\$0.00	\$9.99	Signal Conditioning Components
	10 Values Rectifier Diodes	1	-	\$9.99	\$0.00	\$9.99	Signal Conditioning Components
	Screw Terminals	1	-	\$9.99	\$0.00	\$9.99	PCB Mount Terminals
	Male & Female Pin Holders	1	-	\$12.99	\$9.99	\$22.98	Pin Holders on PCB
	Female DC Power Barrel Jacks	1	-	\$5.99	\$0.00	\$5.99	PCB Mount
	PCB Board Kit	1	-	\$13.99	\$0.00	\$13.99	Prototype Kit
	PCB Board Kit Copper	1	-	\$7.99	\$0.00	\$7.99	Prototype Kit
	Custom PCB	10	-	\$1.00	\$19.99	\$29.99	Custom designed circuit Board
Category				Total Costs			
Hardware Costs				\$1,518.75			
Miscellaneous Costs				\$0.00			
Grand Total				\$1,518.75			

## Project Solution and Overview

## Standards

## Used Professional Standards

- IPC-2221A (Generic Standard on PCB Design)
  - ISO 26262 (Automotive Functional Safety)
  - ISO 29119 (Software Testing)
  - IEC 61508 (Functional Safety of Electrical/Electronic Systems)
  - IEEE 802.1X (Port-based network access control)

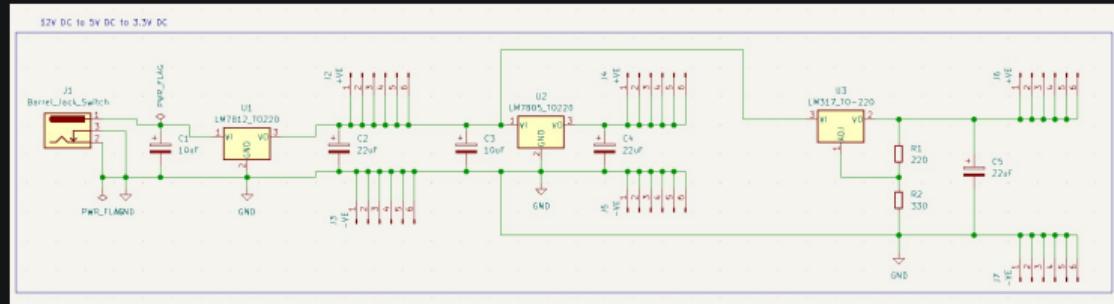
Hardware Design

## Table of Contents

- 1 Introduction
    - ZF
    - Claims Investigation Committee
    - Project Motivation
    - Our Solution
    - Key Devices Under Test
  - 2 Design and Implementation
    - Project Solution and Overview
    - **Hardware Design**
    - Device Interfacing and Testing
    - Embedded Linux With Yocto Project
    - Inter-Processor Communication
    - Web Application and Server
  - 3 Verification
  - 4 Challenges
  - 5 Future Work
  - 6 Future Work
  - 7 Closing

Hardware Design

## Power Supply Schematic Design



## Overview

- 12V DC stable voltage using LM7812 (1A)
  - 12V to 5V DC using LM7805 (1A)
  - 12V to 3.3V using LM317 adjustable regulator

## Key Components

- LM7812, LM7805, LM317 voltage regulators for step-down conversion.
  - Capacitors for noise filtration.
  - Resistors to set voltages for LM317 as  $3.146V$  DC ( $50\mu A$ ).

Hardware Design

## Schematic Design - Brake Signal Transmitter

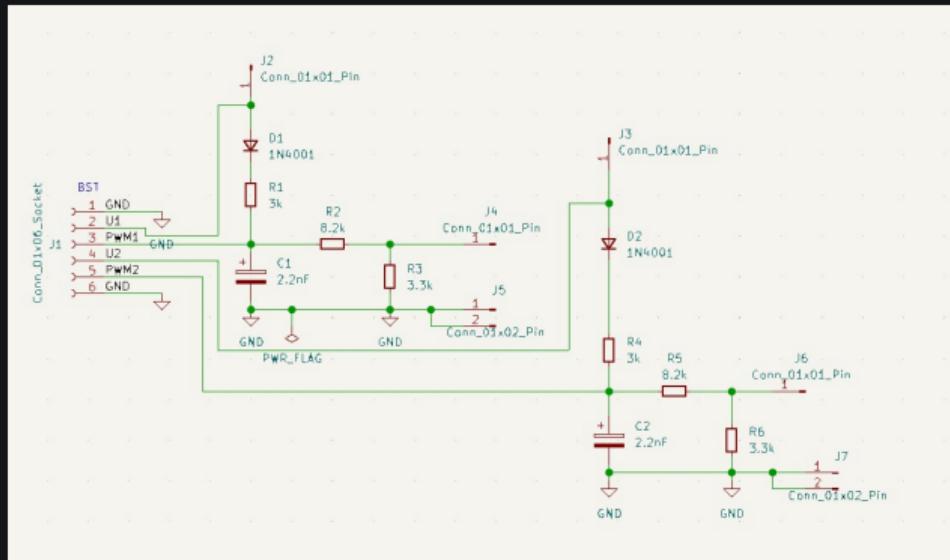


Figure 4: Circuit schematic of Brake Signal Transmitter (BST)

## Key Points

- Captures the output of the Brake Signal Transmitter (BST) in the form of Pulse Width Modulation (PWM) signals.
  - Includes resistors and capacitors for signal filtering.
  - Diodes protect the circuit from voltage surges and reverse polarity.

Hardware Design

# Peripheral Interface Schematic Diagram

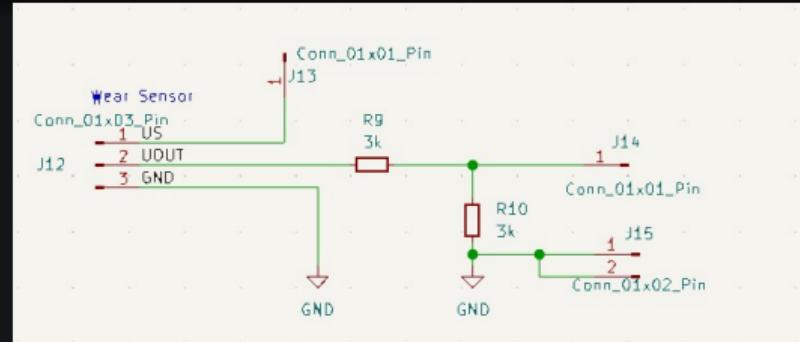


Figure 5: Continuous Wear Sensor Interface Schematic

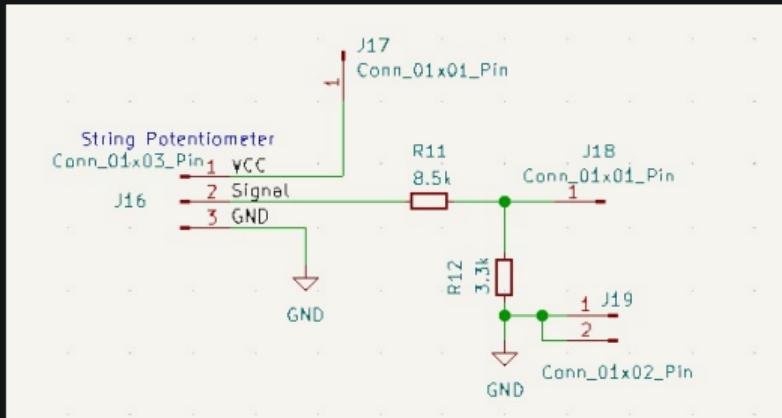
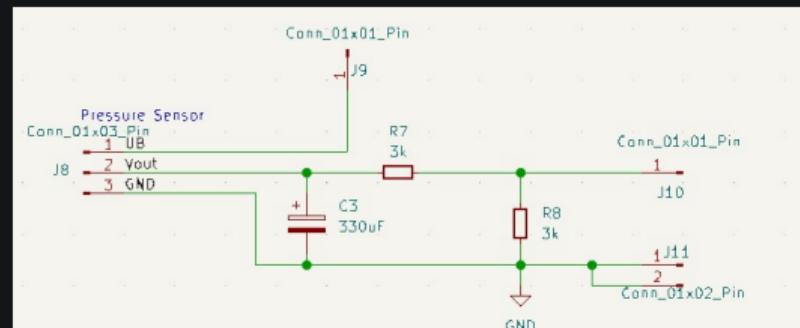


Figure 7: String Potentiometer Interface Schematic



**Figure 6: Pressure Sensor Interface Schematic**

## Key Points

- Captures analog voltage signals to monitor brake wear and pressure sensor and displacement on the string potentiometer.
  - Uses voltage dividers for safe microcontroller input levels.
  - Uses capacitors to stabilize the output.

Hardware Design

# Printed Circuit Board Design

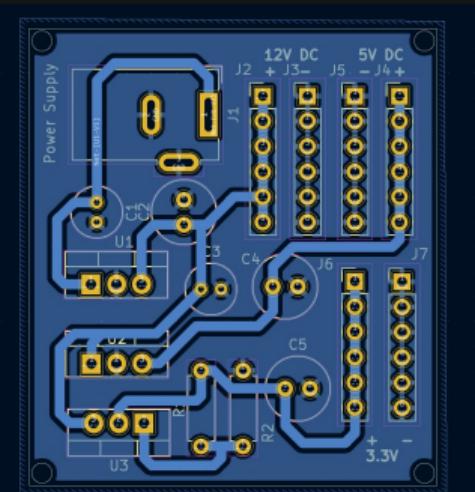


Figure 8: PCB for Power Management

## Overview

- The power supply PCB converts the 12V DC input from the DC jack into regulated output voltages for the system.
    - 12V DC
    - 5V DC
    - 3.3V DC
  - It is designed based on the schematic with components such as voltage regulators (LM7812, LM7805, LM317), capacitors, and resistors.

## Key Components

- DC Jack (J1): Connects the input 12V DC power supply to the board.
  - Output Pins:
    - J2/J3: Provides 12V DC output.
    - J4/J5: Provides 5V DC output.
    - J6/J7: Provides 3.3V DC output.
  - Voltage Regulators
    - Step-down conversion for different voltage levels.
    - Smooth and stable output.
  - Capacitors (C1-C5)
    - Ensure smooth voltage output by filtering noise and ripples.
  - Ground connections: All components are referenced to a common ground for stable operation.

Hardware Design

## Peripherals Printed Circuit Board

## Key Features

- **Input/Power Pins:**
    - Each DUT has a dedicated connector for input signals and a power signal.
    - J1: Inputs for BST (PWM1 and PWM2) | J2/J3: 12V Power Signals for BST (PWM1 and PWM2)
    - J8: Pressure sensor input | J9: 12V DC Power Signal
    - J12: Wear sensor input | J13: 5V DC Power Signal
    - J16: String Potentiometer input | J17: 12V DC Power signal
  - **Output Pins:**
    - Processed signals are sent to the microcontroller through the output pins.
    - J4/J5/J6/J7: BST processed signals
    - J10/J11: Pressure sensor output
    - J14/J15: Wear sensor output
    - J18/J19: String potentiometer output.
  - **Signal Conditioning:**
    - Resistors: Scale signals for safe microcontroller input.
    - Capacitors: Filter noise and stabilize signals.
    - Capacitors: Filter noise and stabilize signals.

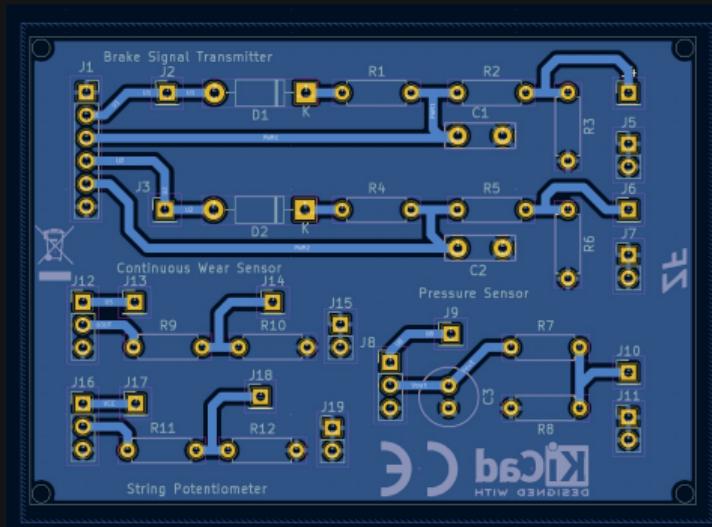
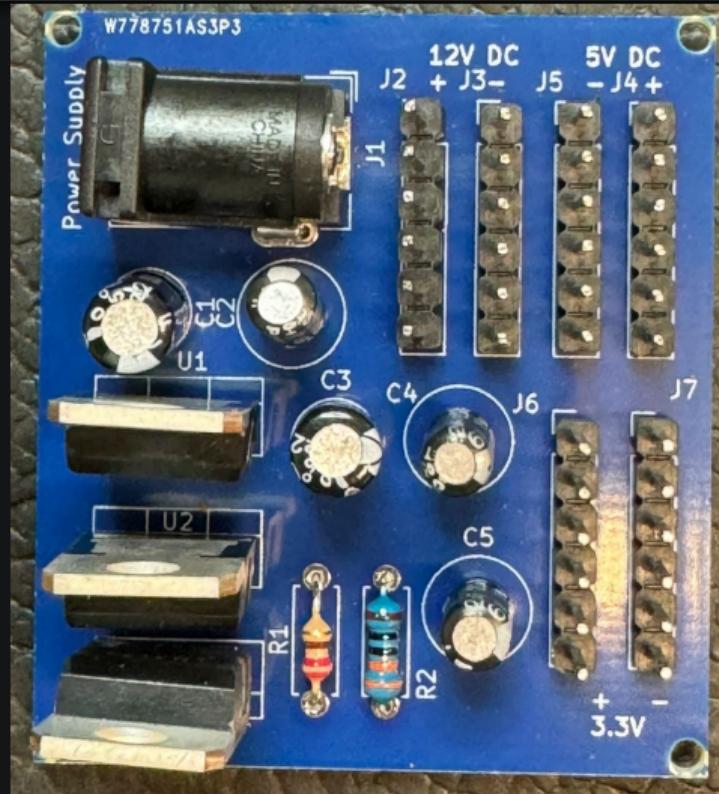


Figure 9: PCB for connecting to peripheral device

Hardware Design

## Fabricated PCB



**Figure 10: Power Management PCB**

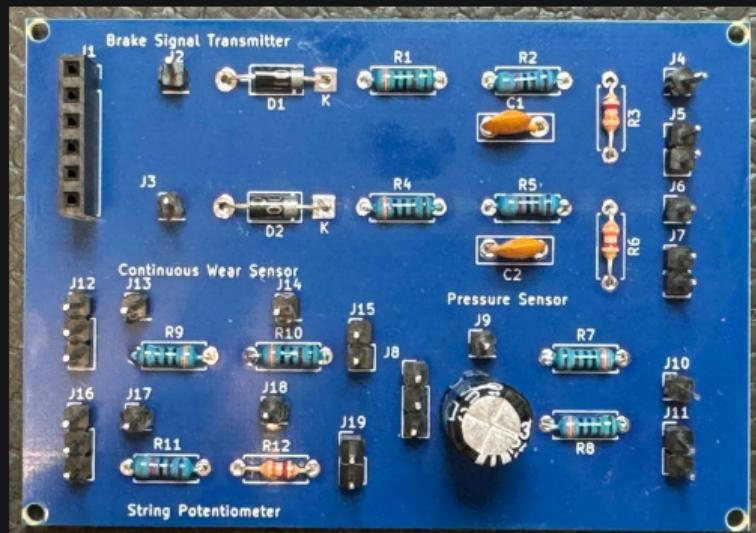


Figure 11: Peripheral Interface PCB

## Hardware Design

## Enclosure Design

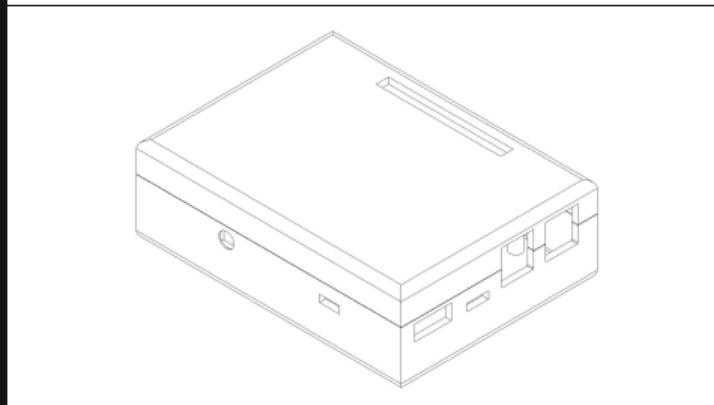
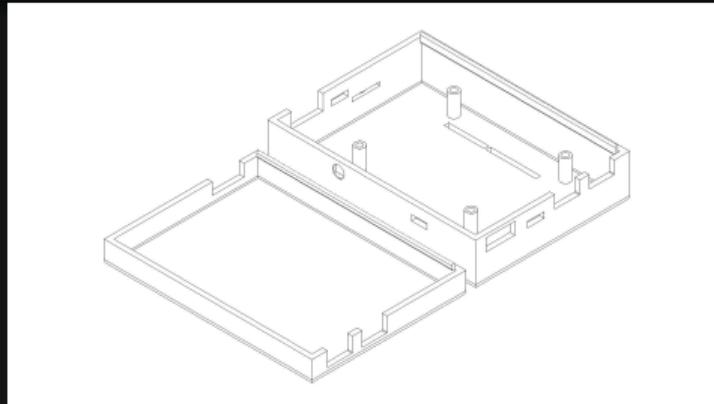


Figure 12: Enclosure for STM32MP157E-DK2

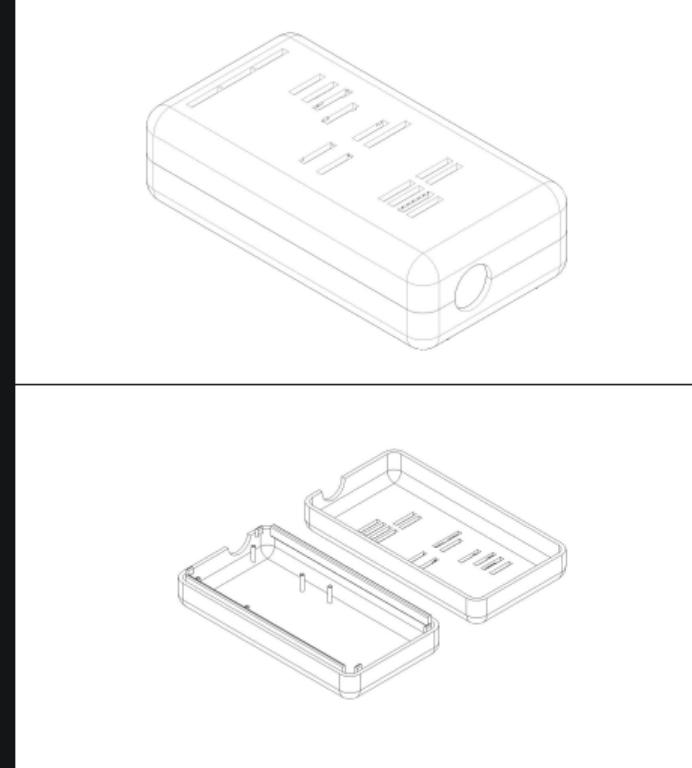


Figure 13: Enclosure for PCBs

# Table of Contents

- 1 Introduction
    - ZF
    - Claims Investigation Committee
    - Project Motivation
    - Our Solution
    - Key Devices Under Test
  - 2 Design and Implementation
    - Project Solution and Overview
    - Hardware Design
    - Device Interfacing and Testing
    - Embedded Linux With Yocto Project
    - Inter-Processor Communication
    - Web Application and Server
  - 3 Verification
  - 4 Challenges
  - 5 Future Work
  - 6 Future Work
  - 7 Closing

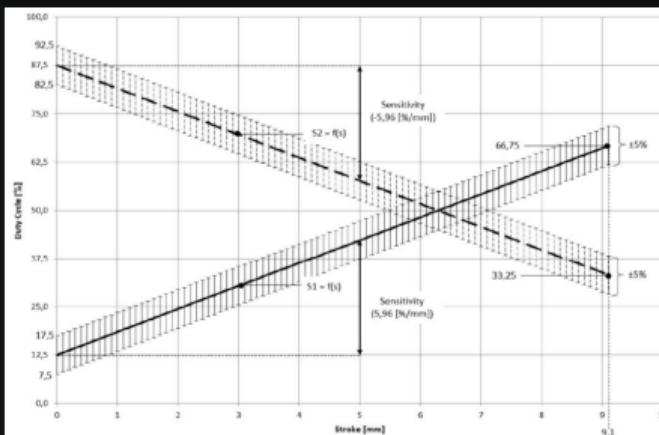
## Device Interfacing and Testing

# Firmware to Test Brake Signal Transmitter (BST)

## Purpose

## Method

- **Input Capture:** Timers capture two PWM signals from the BST
  - **ADC Reading:** Optional string potentiometer for direct analog voltage measurements via ADC
  - **Processing:** Calculates duty cycles, frequencies, and estimated stroke via timer interrupts
  - **Validation:** Compare measurements against expected values according to product specifications to verify BST accuracy
  - **Results:** Sends test results to the main processor for logging and user display



The nominal requirements, are fulfilled:

	S1 (PWM signal 1)	S2 (PWM signal 2)
Offset	12,5 % DC	87,5 % DC
Tolerance	± 5 % DC	
Sensitivity	5,96 % DC/mm	-5,96 % DC/mm
Frequency	200 ± 10 Hz	

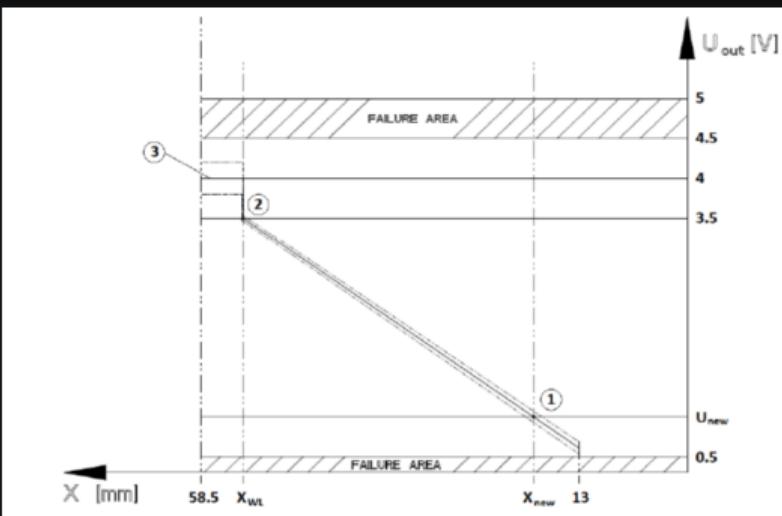
**Figure 14: Product Specifications for BST**

## Firmware to Test Continuous Wear Sensor (CWS)

## Purpose

## Method

1. **ADC Configuration:** Read direct analog voltage via ADC using DMA for efficiency and a timer trigger for consistency
  2. **Wear Calculation:** Mapped the measured voltage to brake pad wear using a linear relationship and handled special conditions (e.g., new pad, worn-out pad) with specific tolerances
  3. **Validation:** Compared wear values against expected values based on product specifications
  4. **Results:** Error thresholds to determine pass/fail and send detailed test outcomes to the main processor for logging and user display



**Figure 15: Product Specifications for CWS**

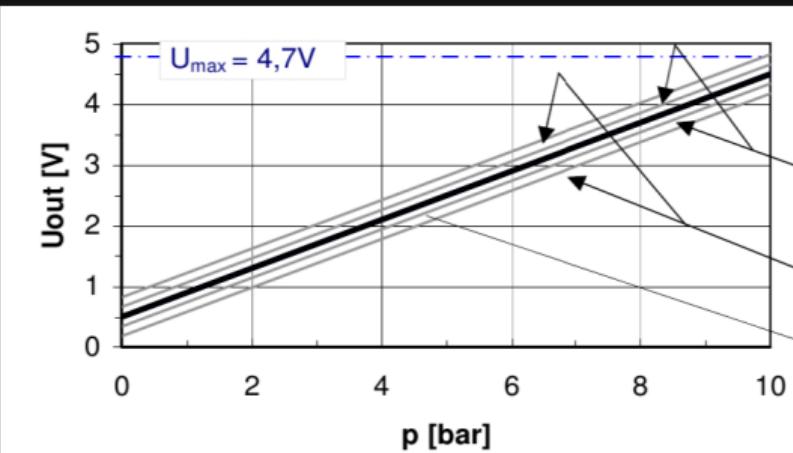
Device Interfacing and Testing

## Firmware to Test Pressure Sensor

## Purpose

## Method

1. **ADC Configuration:** Configured the ADC to read analog voltage from the Pressure Sensor using DMA for efficient data transfer and utilized a timer to trigger ADC conversions periodically
  2. **Pressure Calculation:** Mapped the measured voltage to pressure using a linear relationship given in the product specifications with the addition of converted pressure from bar to psi
  3. **Validation:** Compared calculated pressure against expected values based on product specifications with voltage tolerances to determine pass/fail status
  4. **Results:** Sent detailed test outcomes to the main processor for logging and user display



**Figure 16: Product Specifications for CWS**

# Table of Contents

## 1 Introduction

- ZF
- Claims Investigation Committee
- Project Motivation
- Our Solution
- Key Devices Under Test

## 2 Design and Implementation

- Project Solution and Overview
- Hardware Design
- Device Interfacing and Testing
- **Embedded Linux With Yocto Project**
- Inter-Processor Communication
- Web Application and Server

## 3 Verification

## 4 Challenges

## 5 Future Work

## 6 Future Work

## 7 Closing

Embedded Linux With Yocto Project

# Embedded Linux

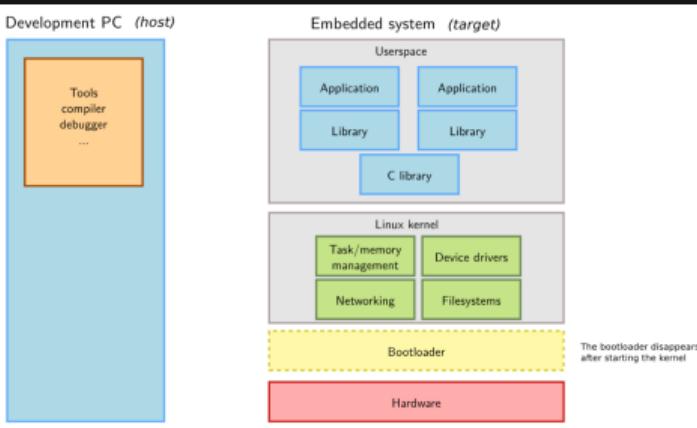


Figure 17: Source:<https://bootlin.com/>  
*Embedded Linux system architecture*

## Why use embedded Linux?

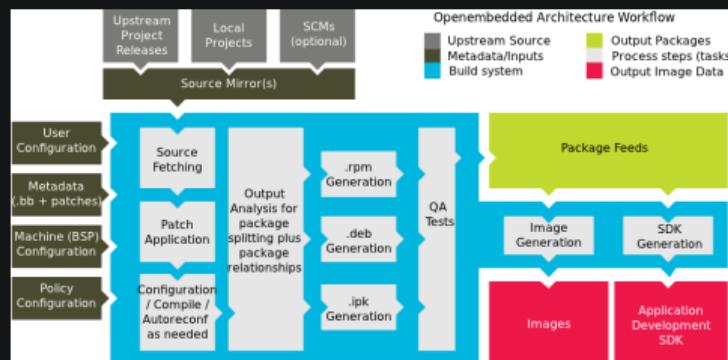
- Industry standard for any embedded operating system
  - Access to open-source software (OSS) and tools
  - Networking and connectivity made easy
  - Easily save/access data with filesystem

Embedded Linux With Yocto Project

# Using The Yocto Project to Build a Custom Distribution

## What is the Yocto Project and why?

- Most popular set of tools for embedded Linux Development
  - Collection of OSS tools to make a custom Linux distribution
  - Independent of target architecture
  - **bitbake** build tool handles **metadata**
  - **MetaData** can be in the form of
    - software build/patch instructions
    - configuration files for software
  - **MetaData** organized in its **Layer Model**



**Figure 18:** Source: <https://docs.yoctoproject.org>  
*High-level diagram representing how builds work using The Yocto Project*

# Custom Linux Image for the STM32MP157F-DK2

## What is used in the deployed image?

- ST's BSP (board support package) layer provides metadata
  - Hardware drivers
  - Kernel Configurations
  - Devicetree
- Custom layer **meta-zf-project**
  - **nginx** (webserver), **wpa\_supplicant** (Wi-Fi access client/ IEEE 802.1X supplicant)
  - recipes for custom applications (Web application, Server, Cortex-M4 Firmware)
  - Kernel configurations and custom Devicetree

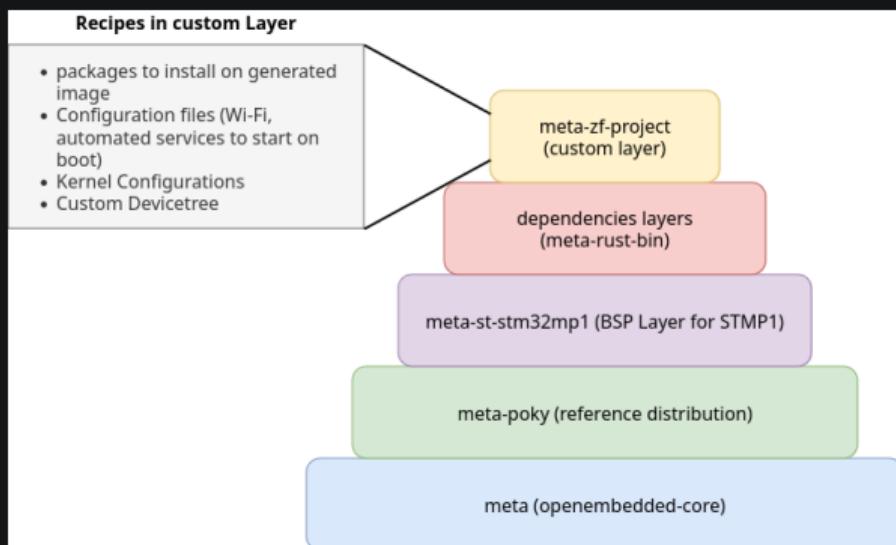


Figure 19: Layer Model representation of this project for deploying onto a STM32MP1-DK2

## Inter-Processor Communication

# Table of Contents

## 1 Introduction

- ZF
- Claims Investigation Committee
- Project Motivation
- Our Solution
- Key Devices Under Test

## 2 Design and Implementation

- Project Solution and Overview
- Hardware Design
- Device Interfacing and Testing
- Embedded Linux With Yocto Project
- **Inter-Processor Communication**
- Web Application and Server

## 3 Verification

## 4 Challenges

## 5 Future Work

## 6 Future Work

## 7 Closing

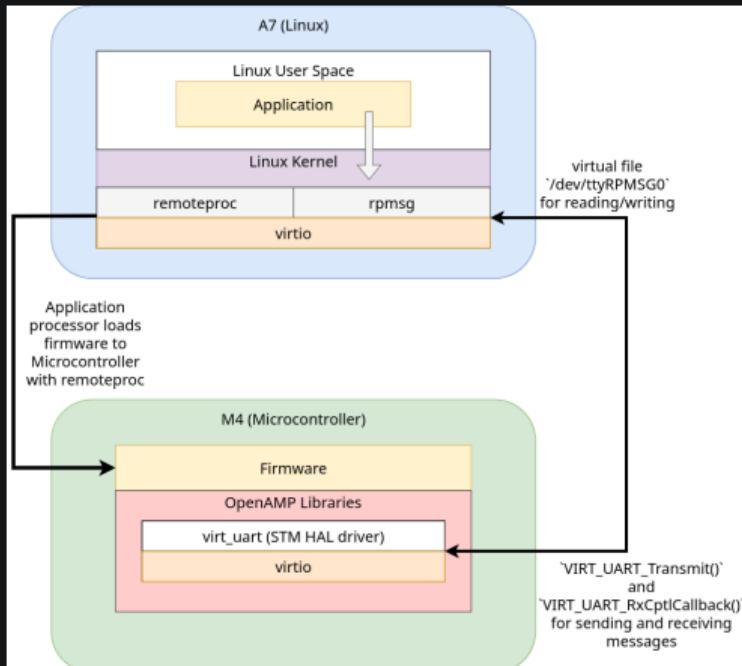
## Inter-Processor Communication

# Inter-Process Communication on a Heterogenous Architecture

## *Heterogenous multiprocessor SoCs cannot directly communicate*

OpenAMP (Asymmetric Multi-Processing) Project

- Software framework that places standard protocol for shared memory
  - Implemented on top of **virtio** framework
  - STM provides **virt\_uart** driver for receiving/transmitting messages over **RPMMsg protocol**
  - STMP1 layer automatically enables the **RPMMSG tty driver** kernel module
    - creates file in Linux filesystem: `/dev/ttyRPMSG<X>`
    - can read and write to like a normal file
  - **remoteproc** framework allows dynamic and remote loading of Cortex-M4 firmware
  - **Resource Table** defined in firmware opens a trace in `/sys/kernel/debug/remoteproc/remoteproc0/trace0`
    - Used for logging measured data in CSV format



# Table of Contents

## 1 Introduction

- ZF
- Claims Investigation Committee
- Project Motivation
- Our Solution
- Key Devices Under Test

## 2 Design and Implementation

- Project Solution and Overview
- Hardware Design
- Device Interfacing and Testing
- Embedded Linux With Yocto Project
- Inter-Processor Communication
- **Web Application and Server**

## 3 Verification

## 4 Challenges

## 5 Future Work

## 6 Future Work

## 7 Closing

Rust

The Rust programming language was used to write both major applications (web-based application and web server) for 2 main reasons.



**Figure 21: Ferris, universally accepted mascot of the Rust Programming language**

Memory Safety and Performance

- A set of rules called **Ownership** enforced by compiler to prevent memory leaks
  - **Borrow checker** within the compiler prevents programs unsafe programs from compiling\*
  - Nearly as or just as performant as C with **Zero Cost Abstractions**
  - Advocated by/used by several United States government agencies:
    - **National Security Agency (NSA)** and **Cybersecurity and Infrastructure Security Agency (CISA)**
    - **Defense Advanced Research Projects Agency**
    - **The White House**

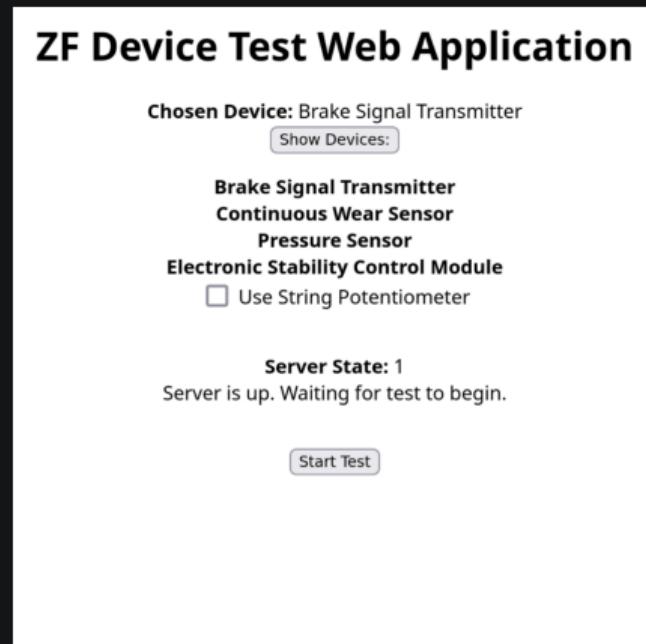
## Web-Application for User Interface

## Web Application in WebAssembly (WASM)

- WASM is a compiled, binary format executable
  - Much faster than traditional Javascript programs
  - Using the Yew framework, written in Rust

## Web application Features

- Shows if application is connected to associated server
  - Selection of different devices
  - Shows progress and state of test
  - Allows download to results in a CSV



**Figure 22:** Web application with dropdown selection of different devices

Web Application and Server

## Custom API Web Server

## Web Server features

- Handles **HTTP requests** from web application
  - Dynamically loads M4 Firmware for selected device with **remoteproc**
  - Polls for results by reading and writing to **/dev/ttyRPMSG0**
  - Saves information from **/sys/kernel/debug/remoteproc/remoteproc0/trace0** as CSV for download

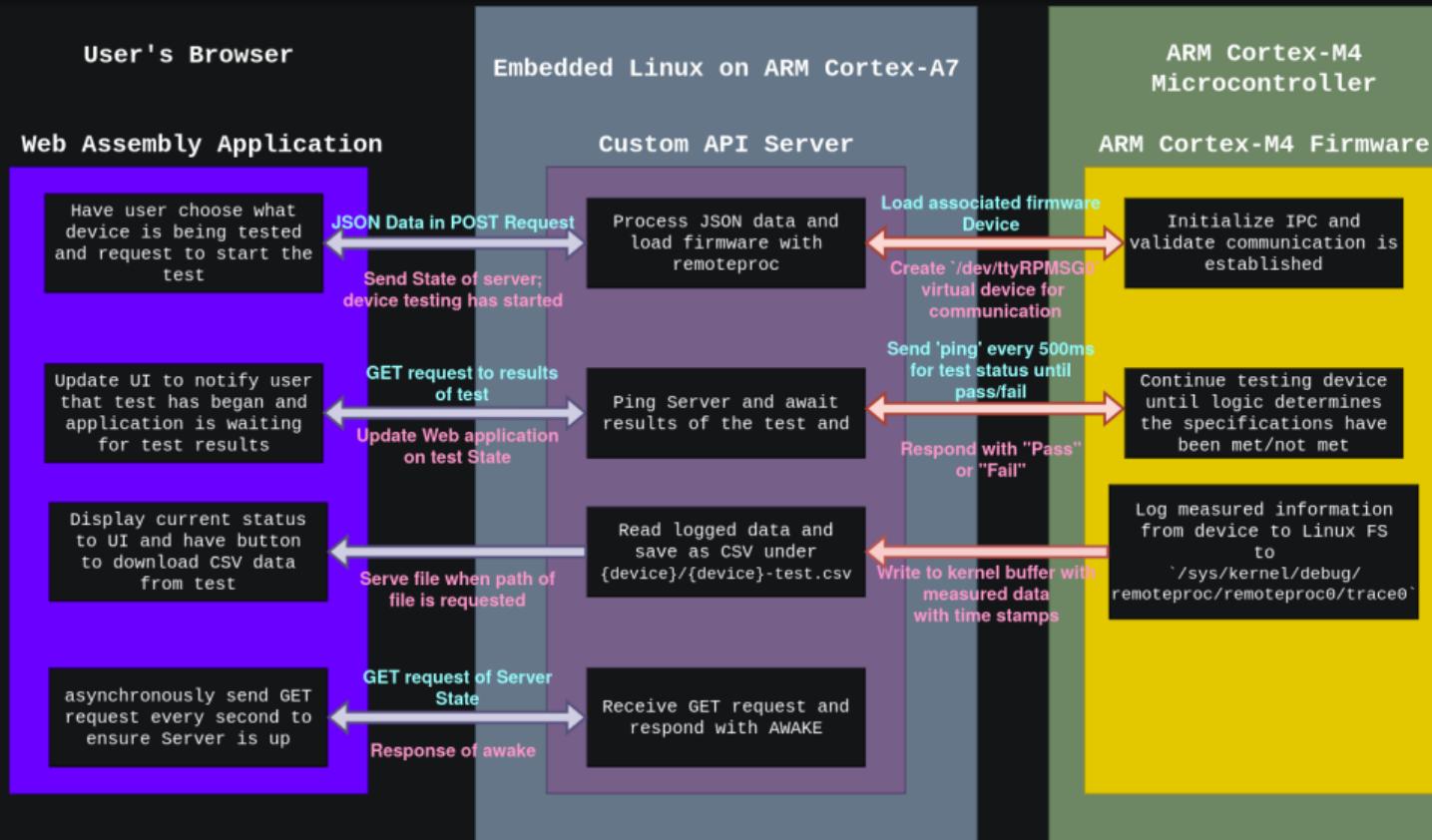
```
____Rust Server for Web Assembly Application_____
Server Listening http://172.20.10.7:8080...
[REDACTED]

Attempting to read from device...
Message
    ping
written successfully!
Response was:
    Pass

Successfully created data/BST-test.csv of test
Firmware for BST has been deloaded: fw_cortex_m4.sh: fw_name=BST-Firmware.elf
```

**Figure 23:** Console logging of server application

# Software Architecture



# Table of Contents

## 1 Introduction

- ZF
- Claims Investigation Committee
- Project Motivation
- Our Solution
- Key Devices Under Test

## 2 Design and Implementation

- Project Solution and Overview
- Hardware Design
- Device Interfacing and Testing
- Embedded Linux With Yocto Project
- Inter-Processor Communication
- Web Application and Server

## 3 Verification

## 4 Challenges

## 5 Future Work

## 6 Future Work

## 7 Closing

Introduction  
oooooo

Design and Implementation  
oooooooooooooooooooooooooooo

Verification  
○●

Challenges  
○○

Future Work  
○○

Future Work  
○○

Closing  
○○○

Link to video demonstration

<https://dylxndy.xyz/senior-design-presentation/verification>

# Table of Contents

## 1 Introduction

- ZF
- Claims Investigation Committee
- Project Motivation
- Our Solution
- Key Devices Under Test

## 2 Design and Implementation

- Project Solution and Overview
- Hardware Design
- Device Interfacing and Testing
- Embedded Linux With Yocto Project
- Inter-Processor Communication
- Web Application and Server

## 3 Verification

## 4 Challenges

## 5 Future Work

## 6 Future Work

## 7 Closing

# Challenges

- System Clock configuration with Devicetree
- Timer configuration for PWM signals
- Mini-360 Buck Converter
- PCB Creation

# Table of Contents

## 1 Introduction

- ZF
- Claims Investigation Committee
- Project Motivation
- Our Solution
- Key Devices Under Test

## 2 Design and Implementation

- Project Solution and Overview
- Hardware Design
- Device Interfacing and Testing
- Embedded Linux With Yocto Project
- Inter-Processor Communication
- Web Application and Server

## 3 Verification

## 4 Challenges

## 5 Future Work

## 6 Future Work

## 7 Closing

# Future Work

- Finish CAN implementation for ESCM
  - USB to CAN used currently
  - enabled **CAN\_GS\_USB** module in Linux Kernel
- Implement CAN through the STM32MP1's CAN interface
  - Requires pin multiplexing and device tree configuration
- Improve Web application appearance
- Ease change of acceptance criteria
- Professional-Grade Enclosure and Connectors

# Table of Contents

## 1 Introduction

- ZF
- Claims Investigation Committee
- Project Motivation
- Our Solution
- Key Devices Under Test

## 2 Design and Implementation

- Project Solution and Overview
- Hardware Design
- Device Interfacing and Testing
- Embedded Linux With Yocto Project
- Inter-Processor Communication
- Web Application and Server

## 3 Verification

## 4 Challenges

## 5 Future Work

## 6 Future Work

## 7 Closing

# Future Work

- Finish CAN implementation for ESCM
  - USB to CAN used currently
  - enabled **CAN\_GS\_USB** module in Linux Kernel
- Improve Web application appearance
- Have test acceptance criteria be provided by user rather than hard-coded
- Professional-Grade Enclosure and Connectors
  - Current design uses 3D-printed enclosure and wires connected directly to the poles
  - upgrading to more professional standard similar to the mBSP tester would greatly enhance usability and durability even further

# Table of Contents

## 1 Introduction

- ZF
- Claims Investigation Committee
- Project Motivation
- Our Solution
- Key Devices Under Test

## 2 Design and Implementation

- Project Solution and Overview
- Hardware Design
- Device Interfacing and Testing
- Embedded Linux With Yocto Project
- Inter-Processor Communication
- Web Application and Server

## 3 Verification

## 4 Challenges

## 5 Future Work

## 6 Future Work

## 7 Closing

# Special Thanks

- Dr. Grantner (faculty advisor)
- David Florida (lab technician)
- Patrick McNally (Head of Engineering at ZF Group - Auburn Hills, MI)
- Davis Roman (Senior Staff Software Engineer at Rivian - Palo Alto, CA)

# Thank you

Any Questions?

## Project Sources

- **Custom Yocto Project Layer:**
  - <https://github.com/DMGDy/meta-zf-project>
- **Custom Web Server in Rust**
  - <https://github.com/DMGDy/zf-webserver-app>
- **Web Application in WASM**
  - <https://github.com/DMGDy/zf-yew-app>
- **Microcontroller Firmware**
  - <https://github.com/danb127/Brake-System-Tester>
- **This Presentation**
  - <https://github.com/DMGDy/ECE4820-Presentation>