

Claims Investigation Committee Multi-Testing Input 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
 - Project Motivation
 - Our Solution
 - Key Devices Under Test

ZF

ZF

What 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



**Figure 1: Source: google.com
ZF Group Office in Auburn Hills, MI**

Motivation for the Project

Challenges with Current Testing Methods

- Current testing methods are time-consuming and labor-intensive
 - Manual testing is prone to more human error
 - Lack of standardized equipment for multiple devices
 - Difficulty in rapid validation of warranty claims

Need for Improvement

- BST implementation in Daimler's new platform intensifies urgency
 - High production volumes require efficient testing methods
 - Expanding product line increases testing complexity

OUR SOLUTION

Multi-Testing Input Device

Addressed Challenges

- Provides a unified testing platform for multiple devices
 - 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

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
- 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

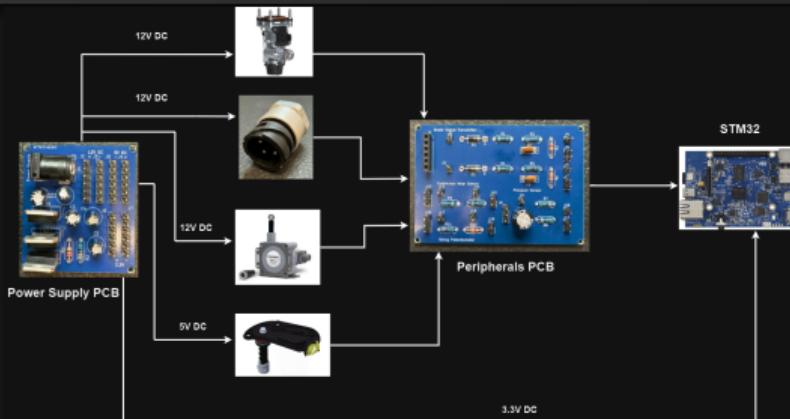
4 Challenges

5 Future Work

6 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 STM32MP157E-DK2 boards

Project Solution and Overview

Project Solution (cont.)

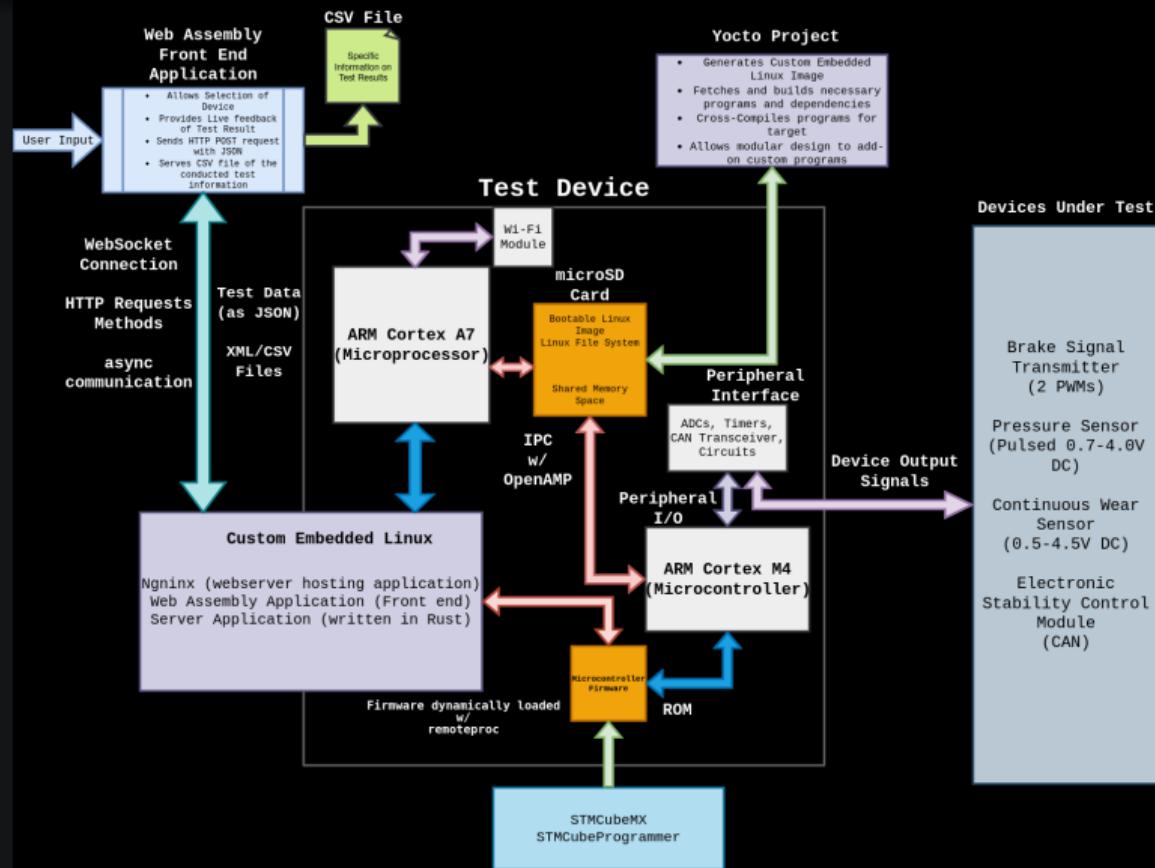
What this project aims to accomplish:

3. Software

- 3.1 Custom embedded **Linux** distribution that will run on the onboard ARM Cortex-A7 microprocessor on the **STM32MP157F-DK2**
 - 3.2 Simple user interface web-based application
 - 3.3 Custom Webserver to process information from web application to microcontroller
 - 3.4 Communicate collected information from ARM Cortex-M4 to ARM Cortex-A7
 - 3.5 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 |

Table of Contents

1 Introduction

- ZF
- 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

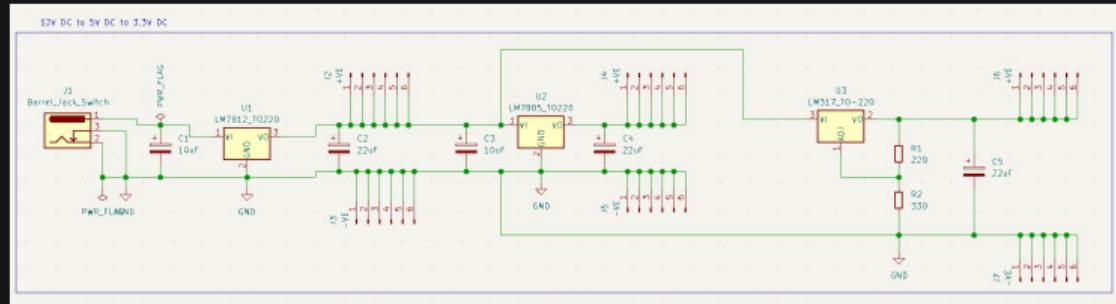
4 Challenges

5 Future Work

6 Closing

Hardware Design

Power Supply Schematic Design



Overview

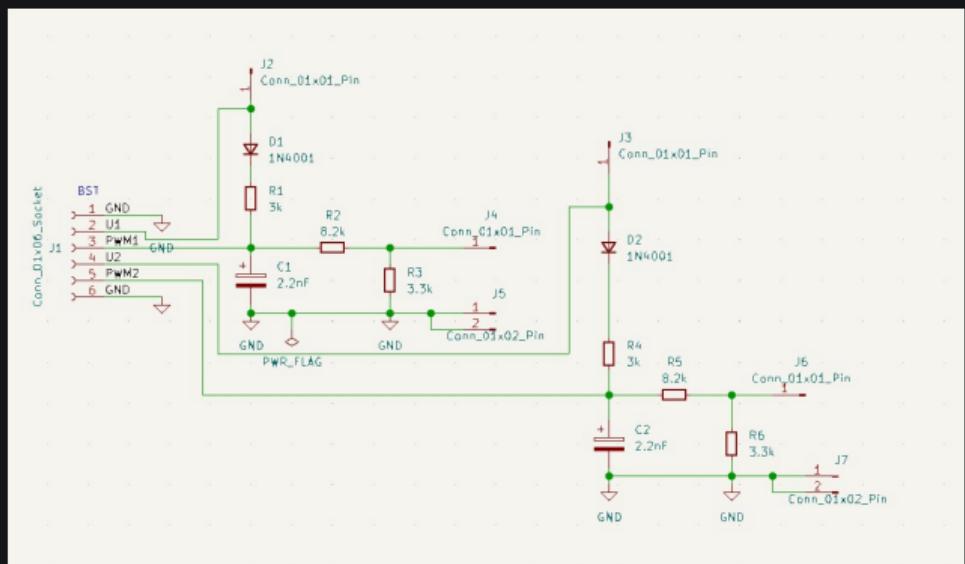
- 12V DC stable voltage using LM7812 (1A)
 - 12V to 5V DC using LM7805 (1A)
 - 12V to 3.3V using LM317 adjustable regulator

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



(12V <50mA)

- Reads PWM signals and wake-up signals
 - Includes resistors and capacitors for signal filtering

Hardware Design

Peripheral Interface Schematic Diagram

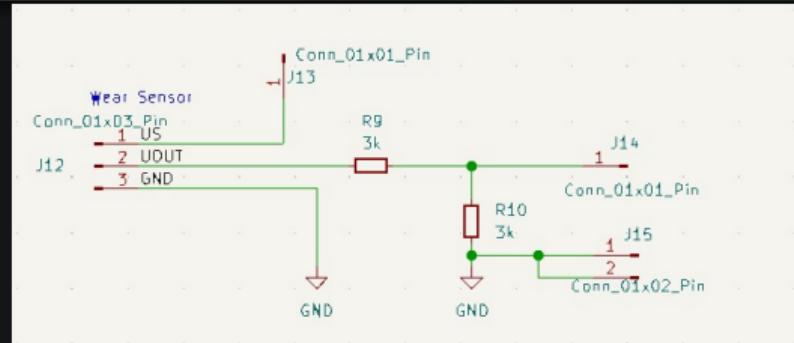


Figure 3: Continuous Wear Sensor Interface Schematic

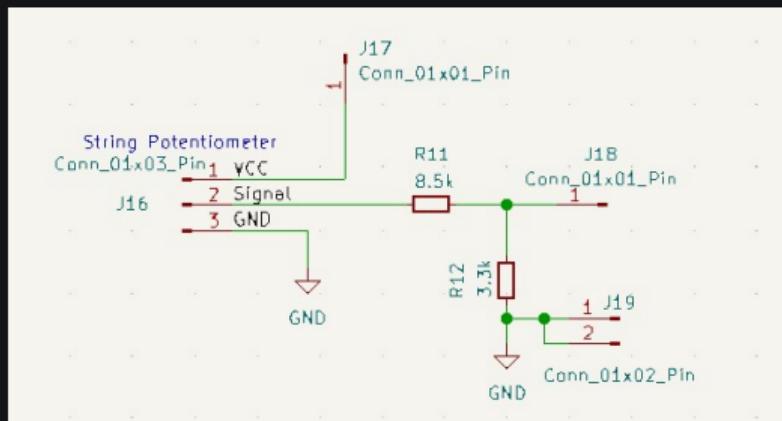


Figure 5: String Potentiometer Interface Schematic

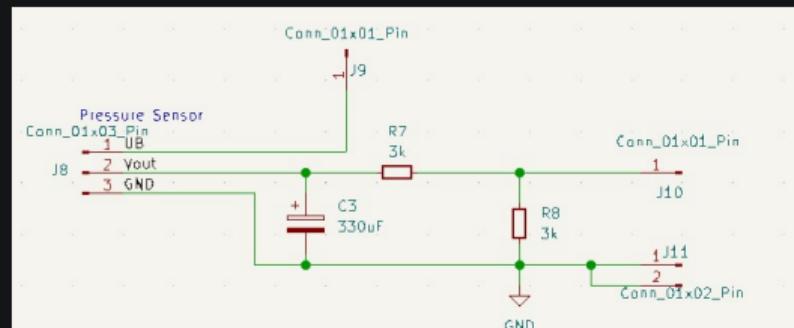


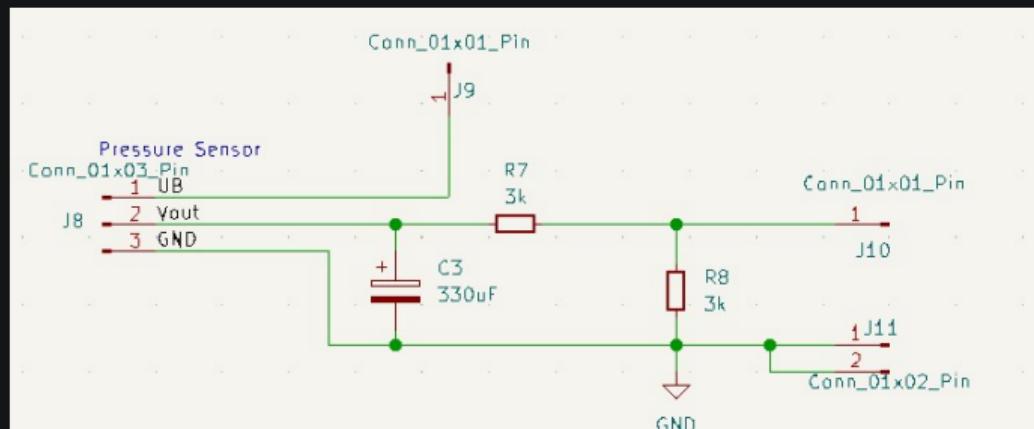
Figure 4: Pressure Sensor Interface Schematic

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

Pressure Sensor (12V <15ma)

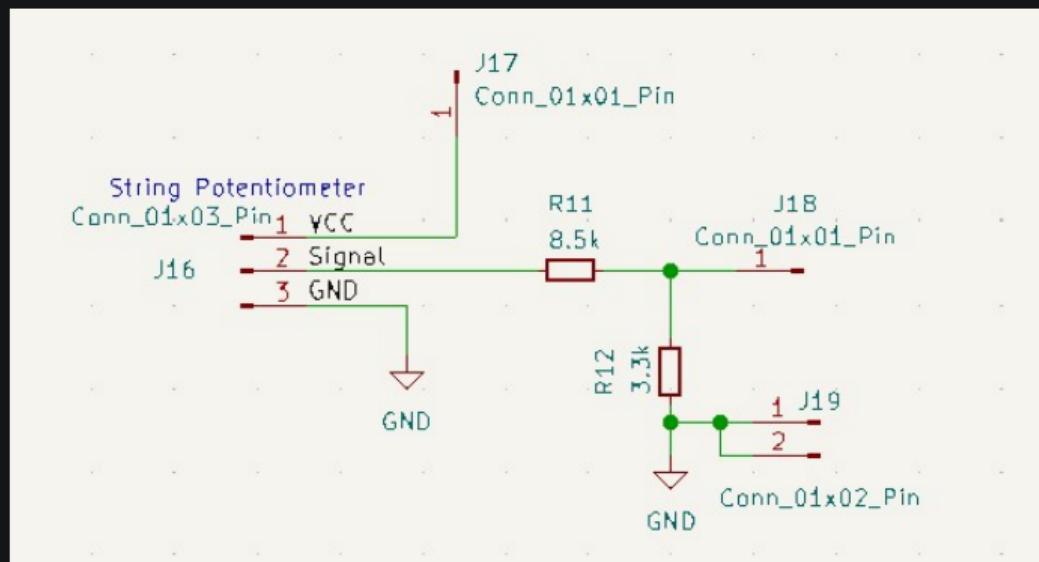


Key Points

- Reads analog signals proportional to pressure.
 - Includes voltage dividers for safe microcontroller input levels.
 - Uses capacitors to stabilize the output.

Hardware Design

String Potentiometer (12V/<20mA)

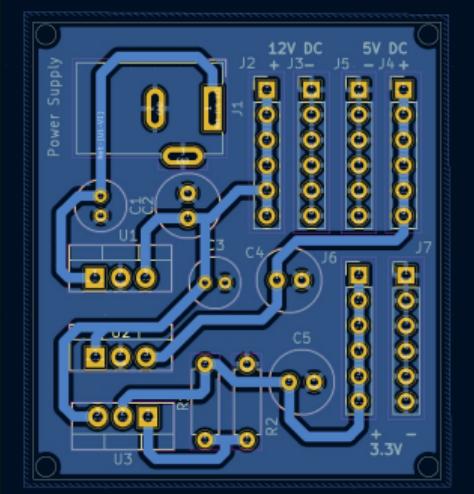


Key Points

- Converts displacement into proportional analog voltage.
 - Includes resistors as voltage dividers for scaling and conditioning.

Hardware Design

Printed Circuit Board Design



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.

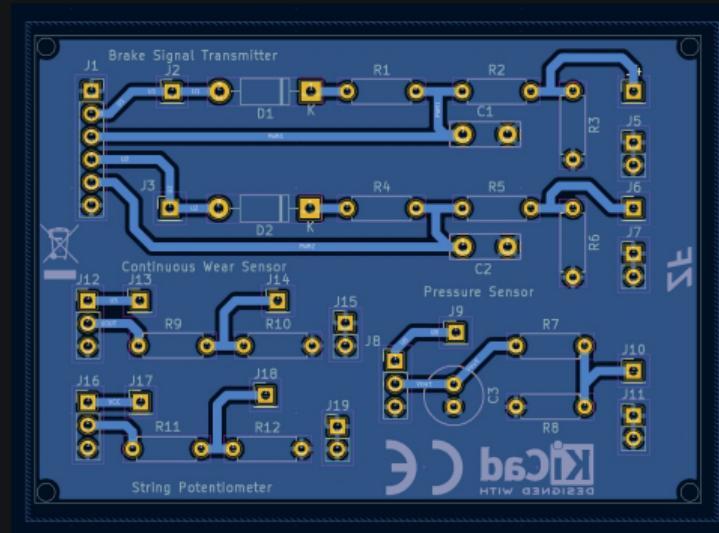
Key Components

Hardware Design

Peripherals Printed Circuit Board

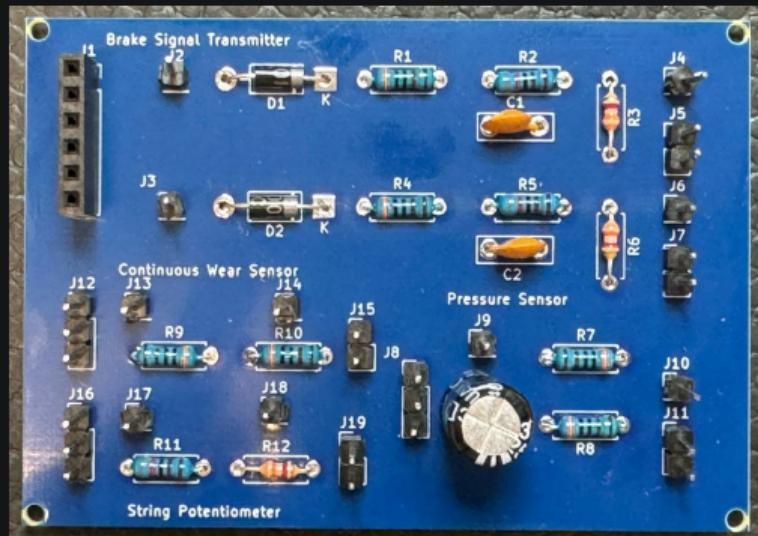
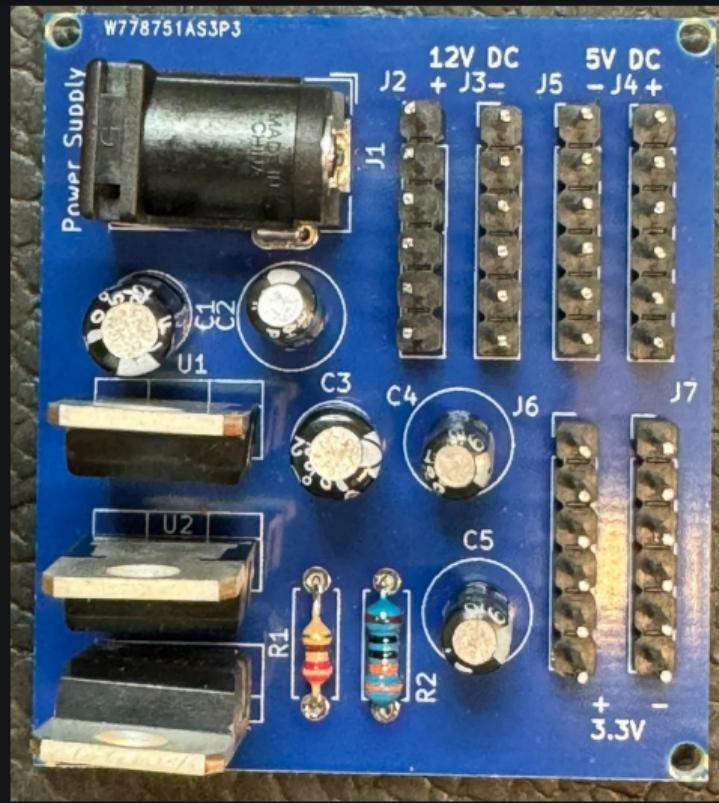
Key Features

- **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.



Hardware Design

Fabricated PCB



Hardware Design

Enclosure Design

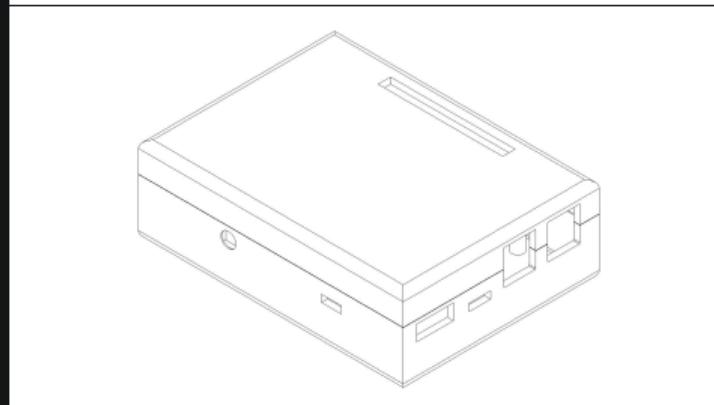
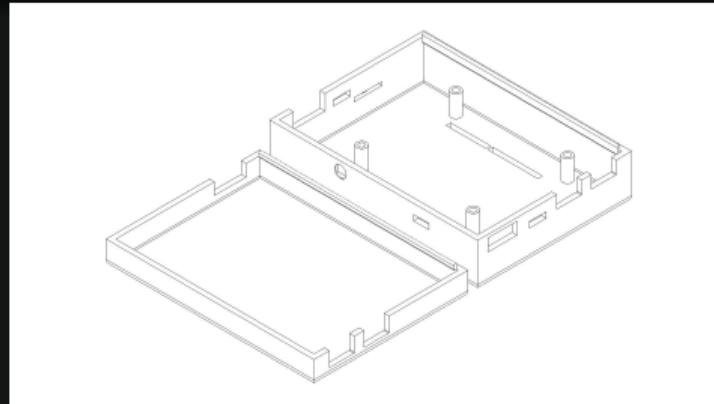


Figure 6: Enclosure for STM32MP157F-DK2

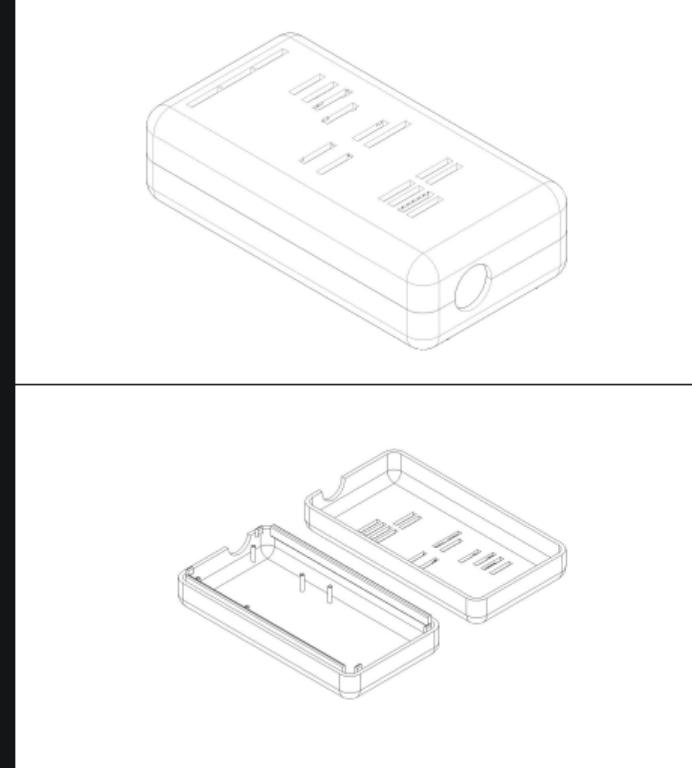


Figure 7: Enclosure for PCBs

Table of Contents

- 1 Introduction
 - ZF
 - 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 Closing

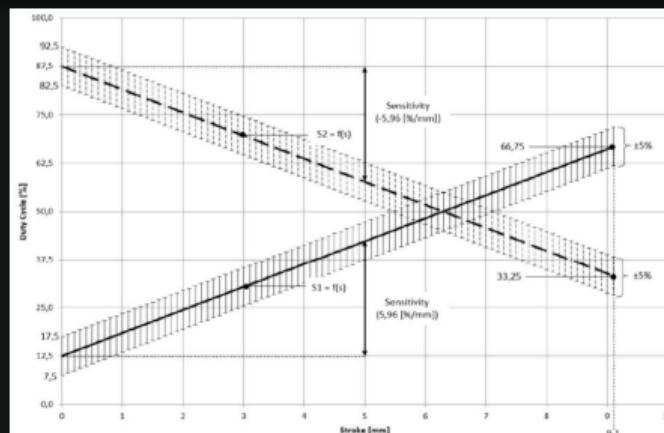
Device Interfacing and Testing

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 8: Product Specifications for BST

Device Interfacing and Testing

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

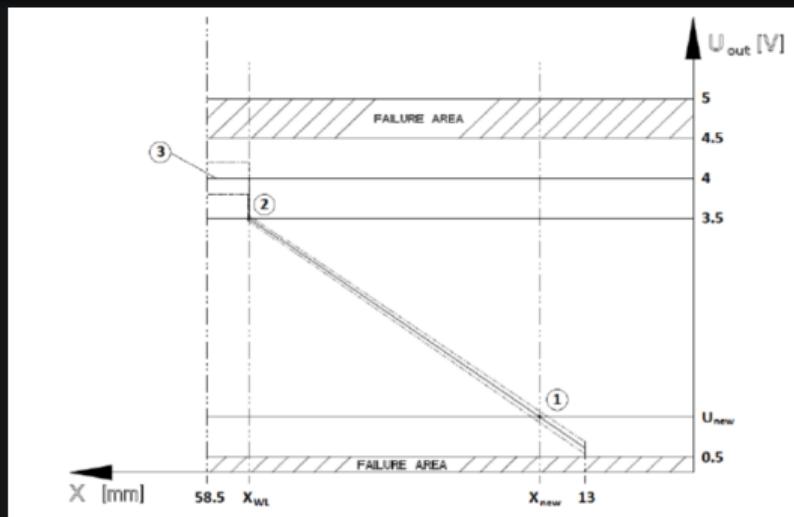


Figure 9: Product Specifications for CWS

Firmware to Test Pressure Sensor

Purpose

- Developed firmware to validate Pressure Sensor readings on the Cortex-M4 microcontroller
- Ensures accurate measurement of pressure when given for reliable vehicle control system purposes
- Key Specifications:** Output range 0.5V (0 bar) to 4.5 V (10 bar), Sensitivity 0.4 V/Bar, Voltage divider ratio 2:1

Method

- 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
- 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
- Validation:** Compared calculated pressure against expected values based on product specifications with voltage tolerances to determine pass/fail status
- Results:** Sent detailed test outcomes to the main processor for logging and user display

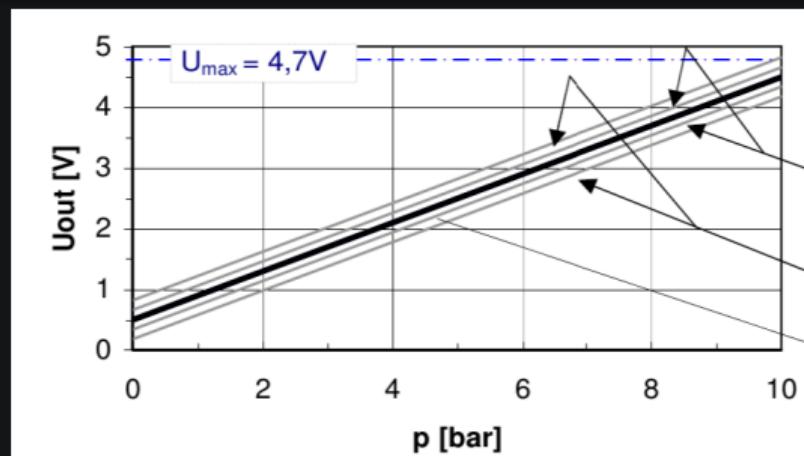


Figure 10: Product Specifications for CWS

Table of Contents

- 1 Introduction
 - ZF
 - 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 Closing

Embedded Linux and The Yocto Project

Embedded Linux

- Industry standard for embedded Operating system
- Rich ecosystem of open-source tools and software

The Yocto Project

- Collection of tools to build a custom embedded Linux distribution
- Fine-grain control of every aspect of deployed image

Embedded Linux With Yocto Project

Embedded Linux

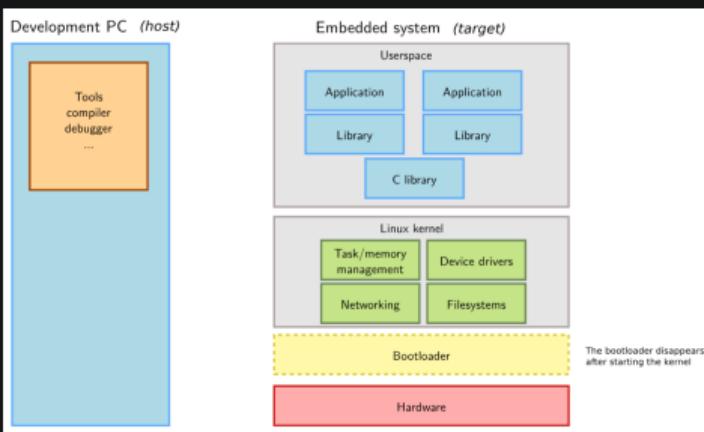


Figure 11: Source:<https://bootlin.com/>
Embedded Linux system architecture

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**

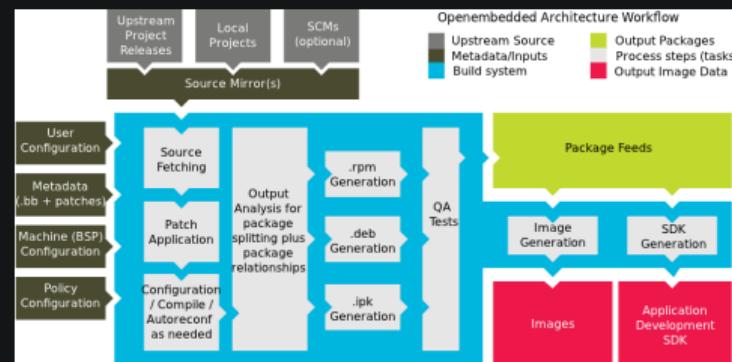


Figure 12: Source: <https://docs.yoctoproject.org>
High-level diagram representing how builds work using The Yocto Project

Custom Linux Image for the STM32MP1-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

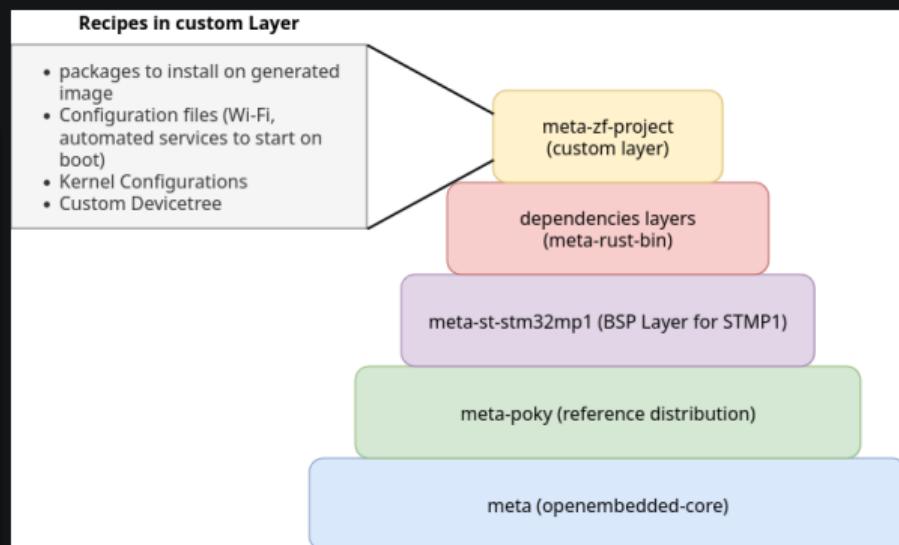


Figure 13: Layer Model representation of this project for deploying onto a STM32MP1-DK2

Inter-Processor Communication

Table of Contents

1 Introduction

- ZF
- 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 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**
 - STM1 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

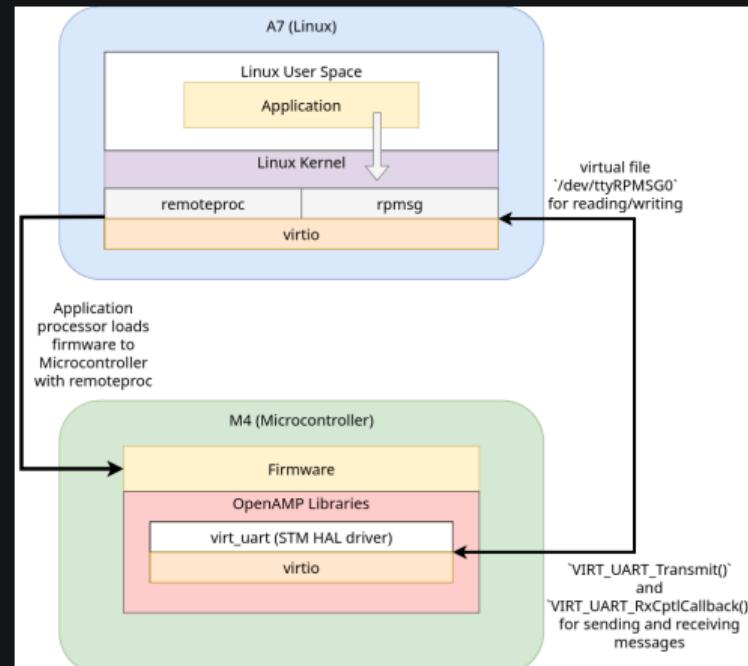


Figure 14: Inter-processor communication between Cortex-A7 (Linux) and Cortex-M4 (Microcontroller)

Table of Contents

1 Introduction

- ZF
- 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 Closing

Rust

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



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 goverment agencies:
 - **National Security Agency (NSA)** and **Cybersecurity and Infrastructure Security Agency (CISA)**
 - **Defense Advance Research Projects Agency**
 - **The White House**

Web Application and Server

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

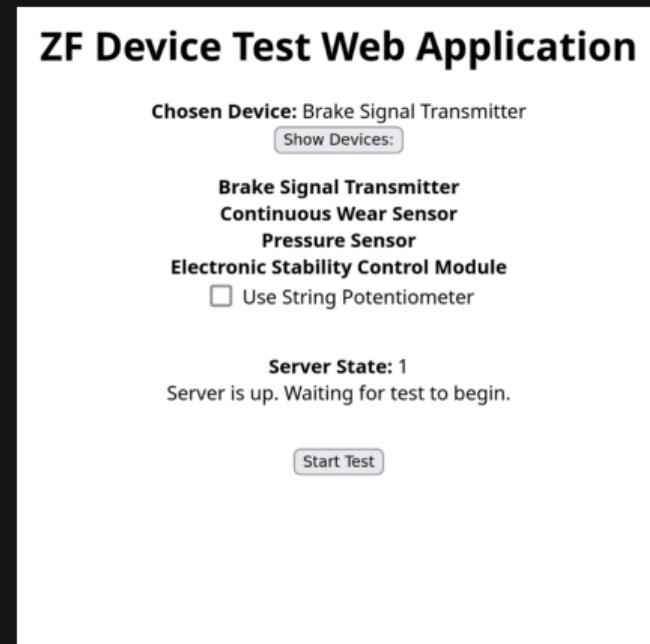


Figure 16: Web application with dropdown selection of different devices

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 17: Console logging of server application

Web Application and Server

Software Architecture

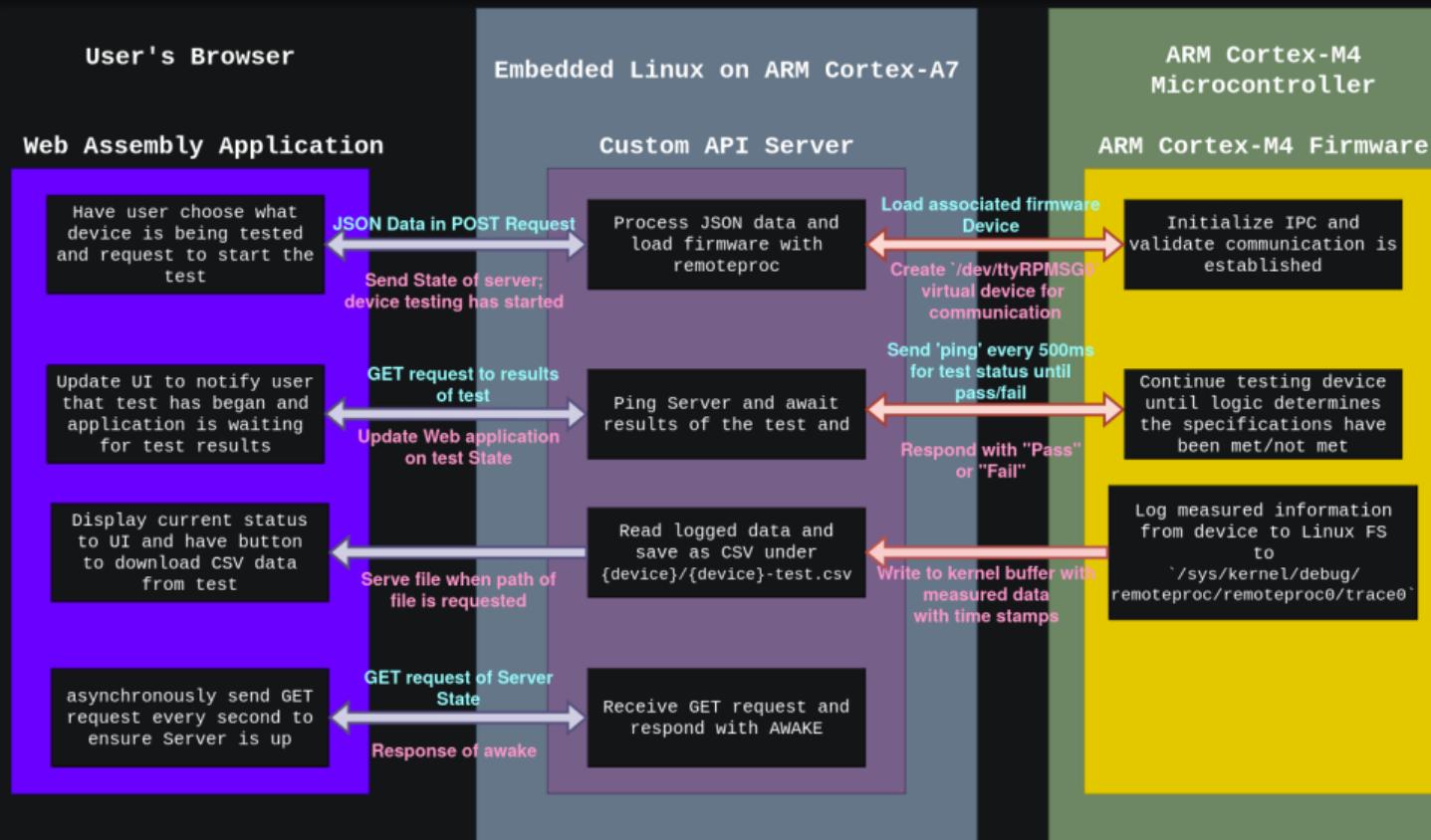


Table of Contents

1 Introduction

- ZF
- 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 Closing

Link to video demonstration

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

Table of Contents

1 Introduction

- ZF
- 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 Closing

Challenges

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

Table of Contents

1 Introduction

- ZF
- 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 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

Table of Contents

1 Introduction

- ZF
- 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 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>