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

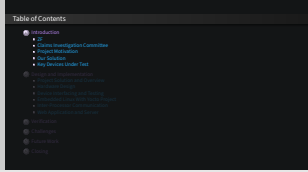
Fall 2024

Faculty Advisor:
Dr. Janos Grantner

Sponsor Manager:
Patrick McNally

1. Hello everyone
2. Today we are excited to present our ZF-sponsored senior design project
3. the Claims Investigation Commitee Multi-Input Testing Device
4. This is brought to you by myself, Daniel Baker
5. Dylan Garza
6. and Rohullah Sah

Table of Contents



ZF

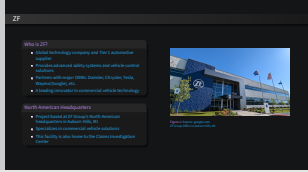
Who is ZF?

- 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
ZF Group Office in Auburn Hills, MI



1. So what is ZF (or zed-eff)
2. ZF is a global leader in automotive technology
3. -specialize in advanced safety systems
4. -vehicle control solutions
5. ZF is a german-based a tier 1 supplier for major automakers like Daimler, Tesla, and Waymo
6. Located in Auburn Hills, ZF North American HQ focuses on commercial vehicle solutions
7. -This is where our project is based
8. -This is also home to the Claims Investigation Center or CIC.

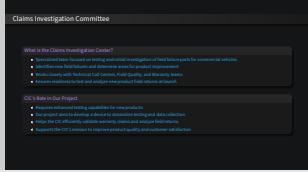
Claims Investigation Committee

What is the Claims Investigation Center?

- Specialized team focused on testing and initial investigation of field failure parts for commercial vehicles
- Identifies new field failures and determine areas for product improvement
- Works closely with Technical Call Centers, Field Quality, and Warranty teams
- Ensures readiness to test and analyze new product field returns at launch

CIC's Role in Our Project

- Requires enhanced testing capabilites for new products
- Our project aims to develop a device to streamline testing and data collection
- Helps the CIC efficiently validate warranty claims and analyze field returns
- Supports the CIC's mission to improve product quality and customer satisfaction



1. The Claims Investigation Center
2. -ZF's specialized team for analyzing field failures
3. Like taking your iphone to apple store for repairs
4. -they ensure components meet reliability standards
5. -help improve testing methods for future products
6. Our project supports their goal to enhance testing efficiency, especially for new product lines like the Brake Signal Transmitter

Motivation for the Project

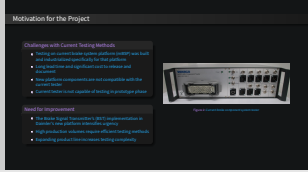
Challenges with Current Testing Methods

- 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*

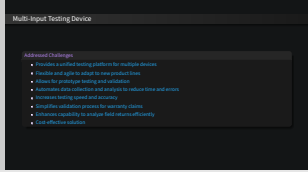


1. The CIC currently relies on the mBSP ABS Component Tester to validate parts like:
2. -Electronic Control Units
3. -Valves
4. -Pressure sensors
5. Tester has key limitations
6. -It's rigid
7. –focus only on predefined components
8. -unsuitable for new or prototype components
9. -It's costly
10. -upgrades require significant effort and manufacturer support
11. -It's slow
12. -manual setup is a step-by-step process
13. -increase testing time and human error
14. With projects such as upcoming release of ZF's Brake Signal Transmitter
15. -Issues become critical
16. BST is launching in 2025
17. -Will be used in Daimler

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

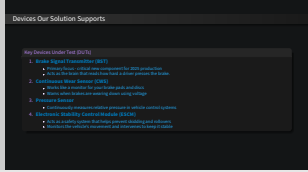


1. Our solution, a multi-input testing device, directly supports CIC
2. it provides a unified testing platform
3. -capable of handling multiple devices across various product lines
4. the device is flexible and adaptable
5. -Designed to accommodate future components
6. -allows us to change acceptance criteria without significant effort
7. it also automates testing.
8. -Reducing manual set up
9. -minimizing human error
10. -improving accuracy

Devices Our Solution Supports

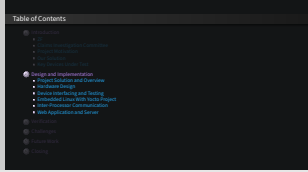
Key Devices Under Test (DUTs)

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

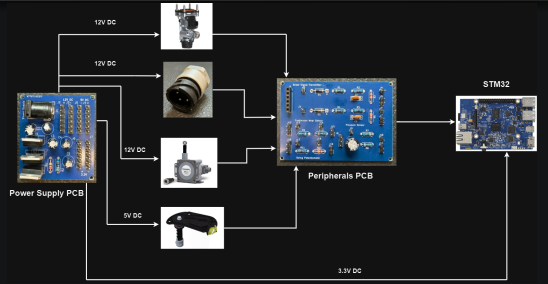


1. Our Multi-Input Testing Device supports several key components
2. -Brake Signal Transmitter or BST
3. -critical new component for 2025 production
4. -interprets the driver brake pedal pressure
5. -Continuous Wear Sensor
6. -Apart of new upcoming brake system platform (mBSP2)
7. -device monitors brake pad and disc wear
8. -provides warnings when brakes need maintenance
9. -Pressure sensor
10. -Apart of previous component tester as well
11. -measures pressure in vehicle control systems
12. -included as part of BST design
13. -Electronic Stability Control Module
14. -Prevents skidding and rollovers
15. -monitors vehicle stability and intervenes if necessary
16. This Test platform enhances CIC's abilities

Table of Contents

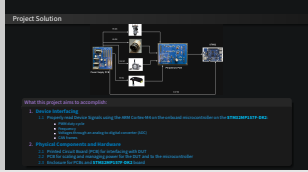


Project Solution



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

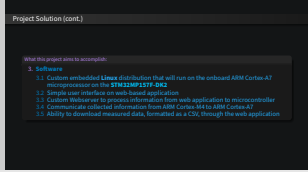


1. using stm32mp157f-dk2, has, Arm Cortex-M4 for external signal handling and Arm Cortex-A7 as an application processor
2. Cortex-M4's primary role is reading signals from the device under test
3. a pwms duty cycle, its frequency, a voltage level through and analog to digital converter and CAN frames
4. processing this information is how the devices will be tested
5. interfacing with the devices under test will also require some hardware for propper signal reading
6. specifically for scaling and managing power and to ensure clean signals with minimal noise

Project Solution (cont.)

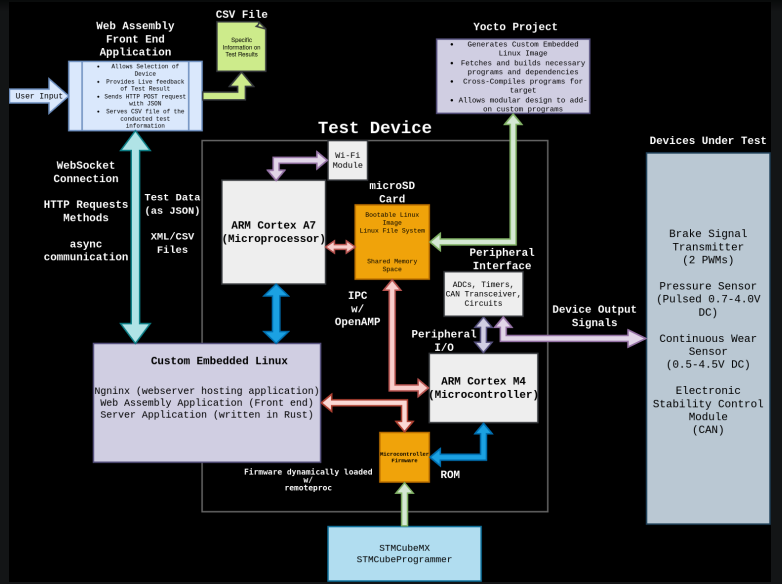
What this project aims to accomplish:

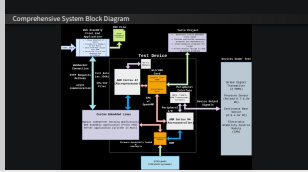
3. **Software**
   - 3.1 Custom embedded **Linux** distribution that will run on the onboard ARM Cortex-A7 microprocessor on the **STM32MP157F-DK2**
   - 3.2 Simple user interface on web-based application
   - 3.3 Custom Webserver to process information from web application to microcontroller
   - 3.4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



1. the software running is the core to the solution, as it aims to simplify the end user, the person running the test on the device
2. The Cortex-A7, the application processor will be running a CUSTOM built linux image
3. it will host a Custom web application in something called web assembly
4. and will interact with a custom webserver that will also handle communication to the Cortex-M4 to gather test data and have it be available for download as a CSV

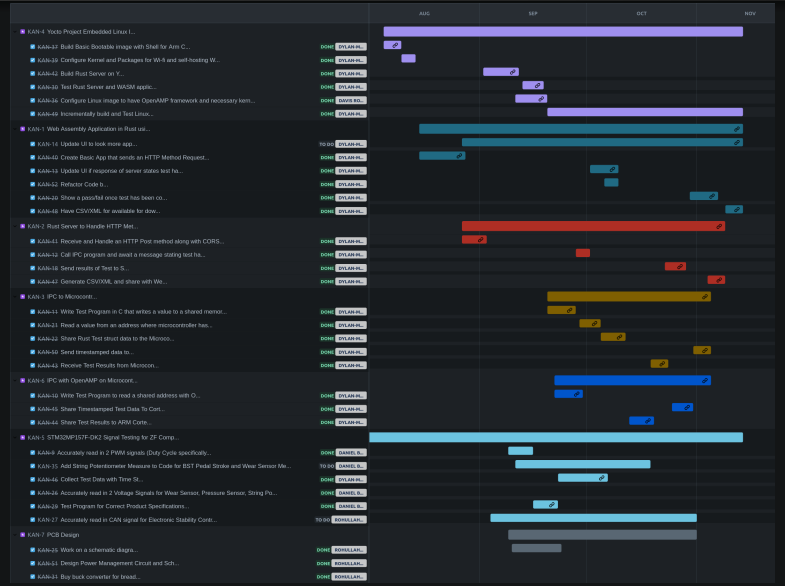
Comprehensive System Block Diagram

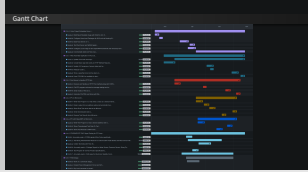




Heres a comprehensive high-level block diagram of everything i just described. Due to time constraints I will have to skip over this but this presentation is available for download if you'd like to look at any section in more detail

Gantt Chart





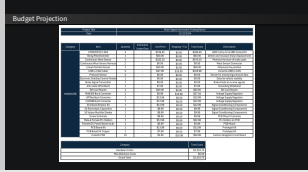
Using Atlassian's Jira, we were able to track the work we needed to do and what tasks to do and what tasks were dependencies on other tasks

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



The following is our budget to show our total costs. The more expensive items like the STM discovery boards and the continuous wear sensor were provided to us by our sponsor, but the rest was provided by us or was a donation

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)

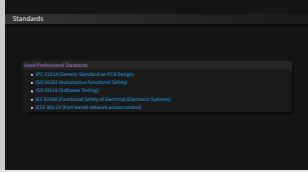
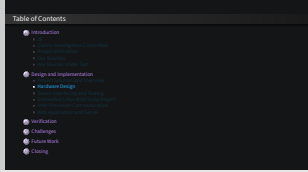


Table of Contents



Power Supply Schematic Design

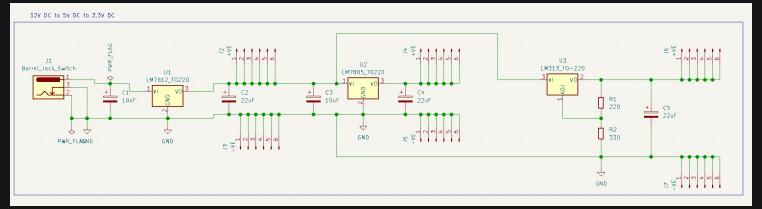


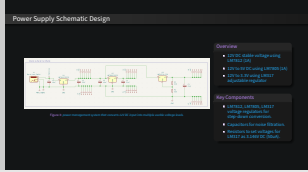
Figure 3: *power management system that converts 12V DC input into multiple usable voltage levels.*

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 (50uA).



1. Let’s start with the power supply design. Here, the input to our system is through a DC Barrel Jack which supplies 12V DC, but the components like sensors and microcontrollers operate at lower voltages. We use three main voltage regulators: this one ensures the input is smooth and regulated to 12V.
2. Over here, it steps down the 12V to 5V.
3. and the last one is adjustable, steps down 12V further to 3.146V using resistors to act as dividers and capacitors to filter out any electrical noise.

Schematic Design - Brake Signal Transmitter

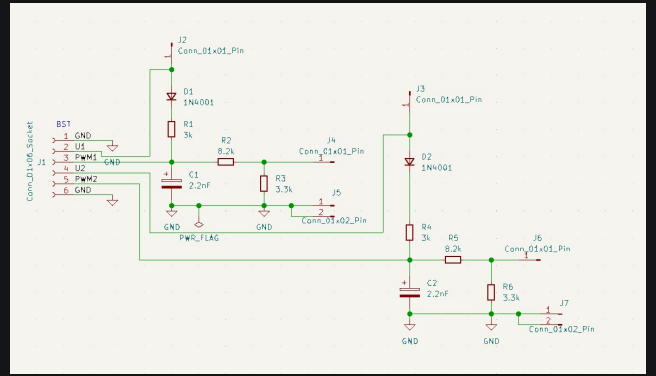
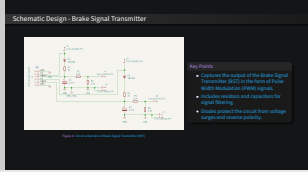


Figure 4: Circuit schematic of Brake Signal Transmitter (BST)

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.



Now, let's move on to the Schematic of the Peripherals. This schematic design shows how the signals of the Brake Signal Transmitter are sent to the microcontroller. The sensor sends PWM signals (.7V – 4.5V) that represent how much force is applied to the brake. Our circuit uses resistors as voltage dividers and capacitors as noise filtering to ensure these signals are stable and within the microcontroller's safe range to read. Diodes act as a safeguard, preventing voltage spikes or reversed connections from damaging the microcontroller.

Peripheral Interface Schematic Diagram

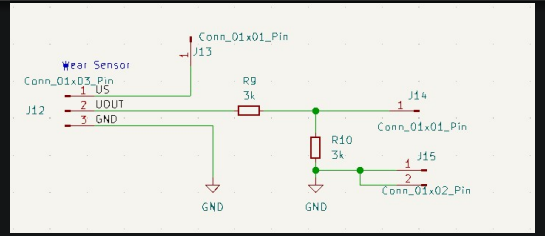


Figure 5: Continuous Wear Sensor Interface Schematic

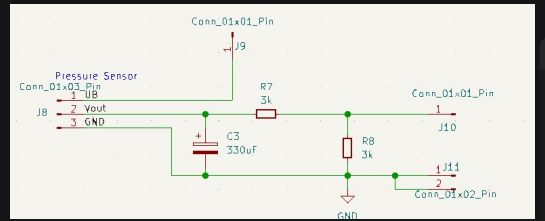


Figure 6: Pressure Sensor Interface Schematic

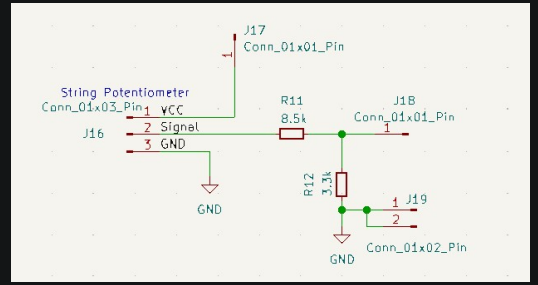
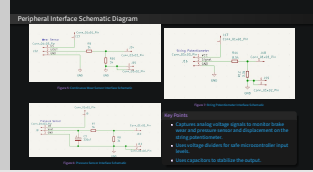


Figure 7: String Potentiometer Interface Schematic

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.



1. We have the schematic interface of the other three sensors: The Continuous Wear Sensor, which monitors the brake pad wear by outputting a small voltage signal (.7V-4V) |
2. the Pressure Sensor, which measures brake system pressure, outputting a voltage (.5V-4.5V) proportional to the pressure level. We have a capacitor to smooth out the noise in the signal
3. the String Potentiometer, which measures the displacement of the brake pedal and converts it to a proportional voltage (0V-4.75V). For all three schematic designs, we have resistors, which condition the signal so the microcontroller can process it correctly.

Printed Circuit Board Design

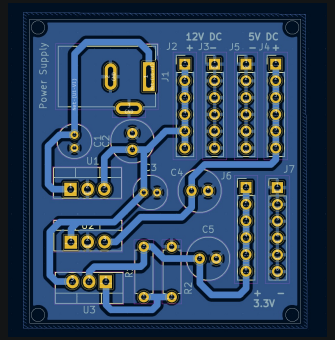


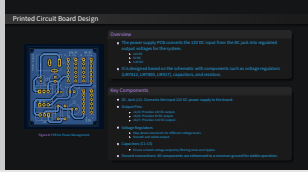
Figure 8: PCB for Power Management

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.



1. Here is the Printed Circuit Board design for our power supply. This board takes 12V DC input from a DC jack and converts it into the usable outputs.
2. This is where we connect the 12V input DC Jack power supply (J1).
3. We have the capacitors over here, to filter out noise and voltage ripples.
4. These are the voltage regulators, to ensure the outputs are stable and consistent even if the input voltage fluctuates.
5. These are the output pins. These ones provide 12V. These provide a regulated 5V and the ones at the bottom provide a 3.146V keeping the resistors as voltage dividers. All components are referenced to a common ground.

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.

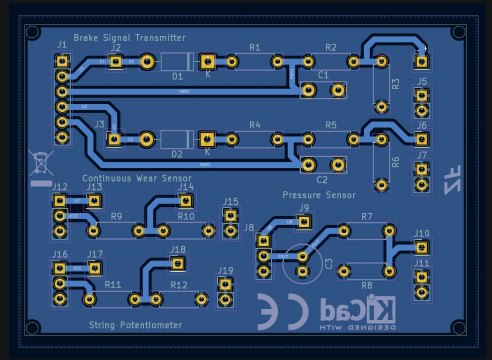
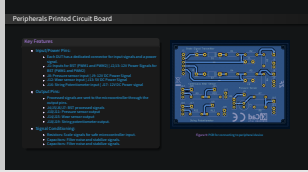


Figure 9: PCB for connecting to peripheral device



1. This is the Peripheral PCB Design. We have kept all the components of each sensor separate. This one over here is for BST, we have the Continuous Wear Sensor over here, the Pressure sensor on the right of the board, and the String Potentiometer at the bottom.
2. Each Devices Under Test has its own dedicated connector for both signal input and power supply pin.
3. These pins (J1) capture the PWM signals from the Brake Signal Transmitter, while these pins (J2 and J3) provide the 12V power needed to operate it.
4. Similarly, J8 reads the Pressure Sensor input, and J9 delivers the 12V power required for its operation.
5. These pins connect the Continuous Wear Sensor input, with J13 supplying it with 5V power.
6. Lastly, the String Potentiometer uses J16 for its signal input and receives its 12V power from J17. We have the corresponding electrical components (resistors, capacitors, diodes).
7. The processed signals are sent through specific output pins, while we have a common ground

Fabricated PCB

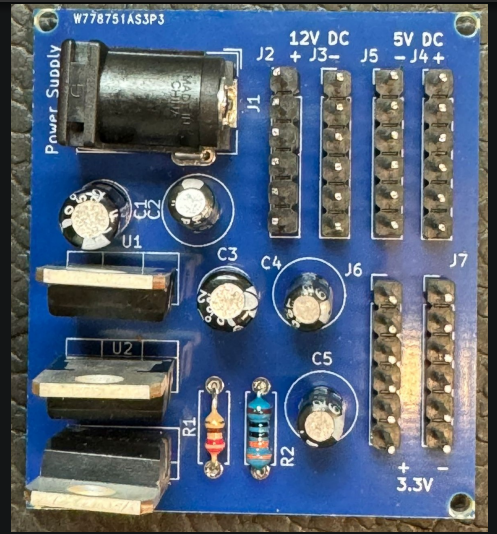


Figure 10: Power Management PCB

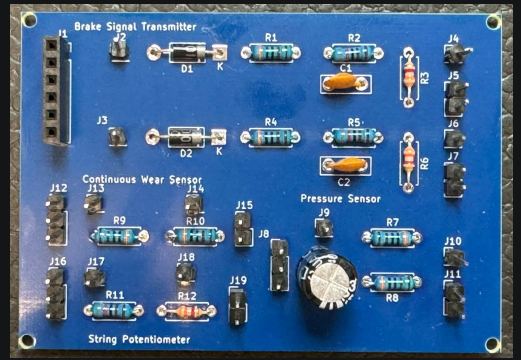
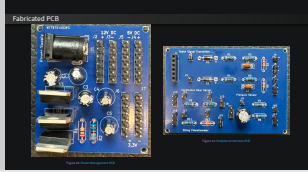


Figure 11: Peripheral Interface PCB



Here is the fabricated and custom assembled PCB's. Power management is on the left, and on the right is the Peripheral PCB.

Enclosure Design

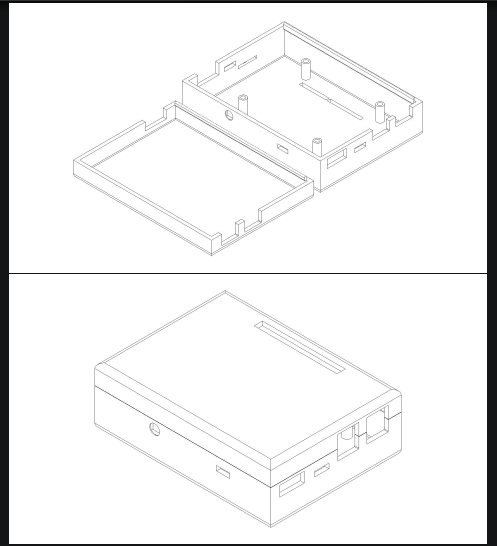


Figure 12: *Enclosure for STM32MP157F-DK2*

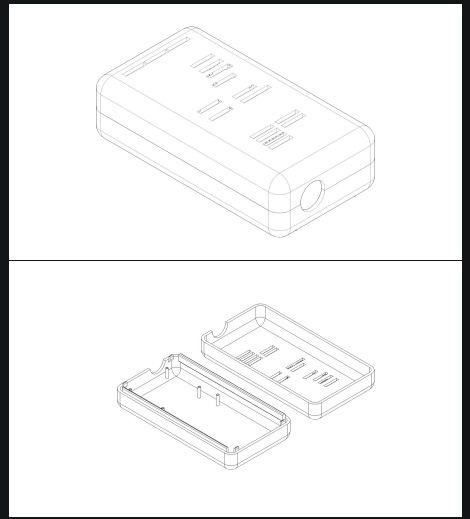
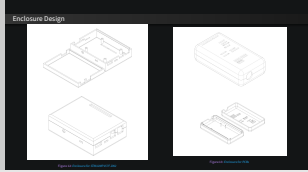
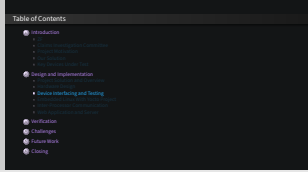


Figure 13: *Enclosure for PCBs*



hese enclosures are custom-designed to securely house the STM board, power supply, and peripheral PCBs. They include slots for the DC jack, ports, and input/output pins, ensuring easy and organized connections. The slots also act as ventilation to keep the temperature stable during testing. Inside, standoffs hold the boards firmly in place, and the two-part assembly allows easy access for repairs, upgrades, or debugging.

Table of Contents



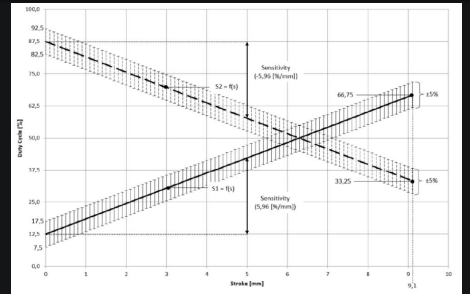
Firmware to Test Brake Signal Transmitter (BST)

Purpose

- Developed firmware on the onboard Cortex-M4 microcontroller to validate BST
- Ensures brake actuation is accurate to distance moved by brake pedal
- **Key Specifications:** Output range 1 mm to 9 mm, Sensitivity 5.96% DC/mm, Output signals PWM1 and PWM2 (S1 and S2)

Method

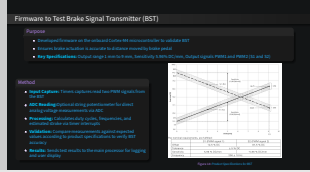
- **Input Capture:** Timers captures read two PWM signals from the BST
- **ADC Reading:**Optional string potentiometer for direct analog voltage measurements via ADC
- **Processing:** Calculates duty cycles, frequencies, and estimated stroke via timer interrupts
- **Validation:** Compare measurements 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*



1. For the BST, the Cortex-M4 receives signals in the form of pulse-width modulation, or PWM
2. This is a method where signals are turned on and off rapidly
3. 2 key parameters define PWM
4. the frequency is how fast these signals are read
5. for BST specifically reads at 200 Hz
6. the pulse is repeated 200 cycles per second
7. The duty cycle of the pulse is what percentage the signal is "on"
8. so a duty cycle of 50
9. The code measures these signals to calculate the brake stroke
10. ensures it matches the sensitivity of 5.96
11. The test also offers option to use a string potentiometer
12. this provides an analog voltage proportional to the brake stroke
13. read by ADC on the cortex-M4

Firmware to Test Continuous Wear Sensor (CWS)

Purpose

- Developed firmware on the onboard Cortex-M4 microcontroller to validate the Continuous Wear Sensor (CWS)
- Ensures accurate measurement of brake pad wear levels to enhance vehicle safety
- **Key Specifications:** Output range 0.7V (18 mm or new pad) to 4.0 V (53 mm or worn pad), Sensitivity 0.08 V/mm, Voltage divider ratio 2:1

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

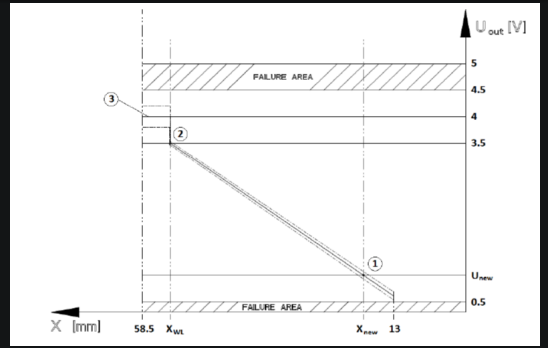
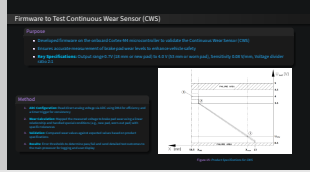


Figure 15: *Product Specifications for CWS*



1. The Continuous Wear Seansor outputs analog voltages to indicate brake pad wear
2. ranges from 0.7V for new pads to 4.0V for worn pads.
3. The Cortex-M4 reads these voltages using Analog-to-Digital Converter or ADC
4. the firmware code maps the readings to pad thickness using linear calibration shown in graph

Firmware to Test Pressure Sensor

Purpose

- Developed firmware to validate Pressure Sensor readings on the Cortex-M4 microcontrooler
- 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

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

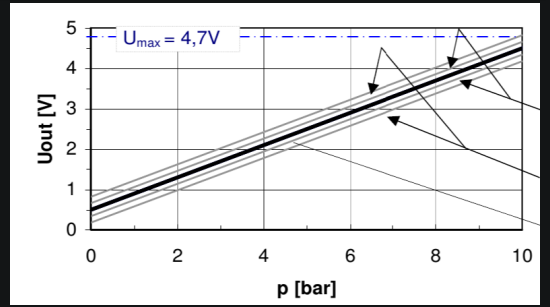
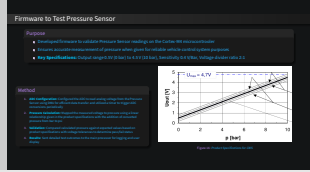
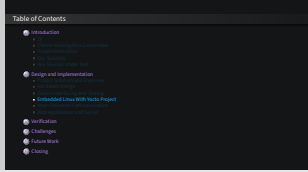


Figure 16: *Product Specifications for CWS*



1. Similarly, the Pressure Sensor outputs voltages proportional to pressure
2. 0.5V for 0 bar to 4.5V for 10 bar
3. The ADC reads these signals
4. our firmware converts them to meaningful pressure values
5. Both these tests incorporate thresholds to determine pass or fail outcomes
6. Validation in all test occurs on the Cortex-M4
7. comparing calculated values with expected specifications.
8. Results are then sent to the Cortex-A7
9. logged and displayed for user

Table of Contents



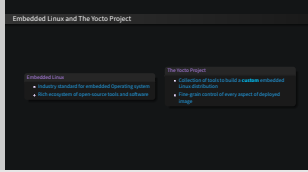
Embedded Linux and The Yocto Project

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



1. due to time constrains i'll go over the contents of the following to slides
2. As i mentioned earlier, the application processor is runnning embedded linux
3. It gives a lot of benefits, namely its virtually the only option for any serious embedded operating system
4. as it gives a rich ecosystem of open-source tools and software
5. The Yocto Project was used to build a custom linux distribution
6. Since it provides fine grain control of the resulting image that will be used on the device

Embedded Linux

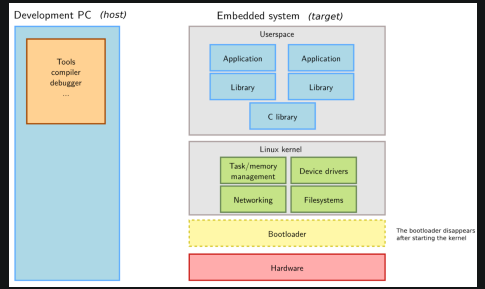
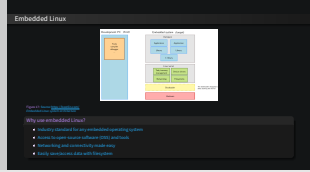


Figure 17: Source:https://bootlin.com/
Embedded Linux system architecture

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



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**

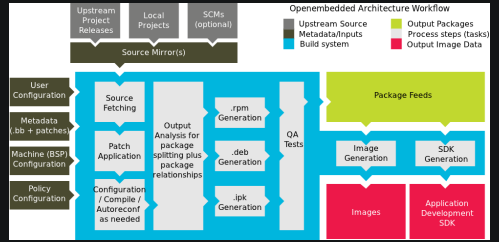
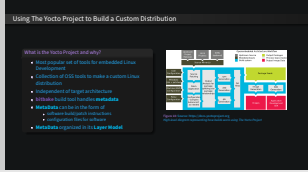


Figure 18: Source: https://docs.yoctoproject.org
High-level diagram representing how builds work using The Yocto Project



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

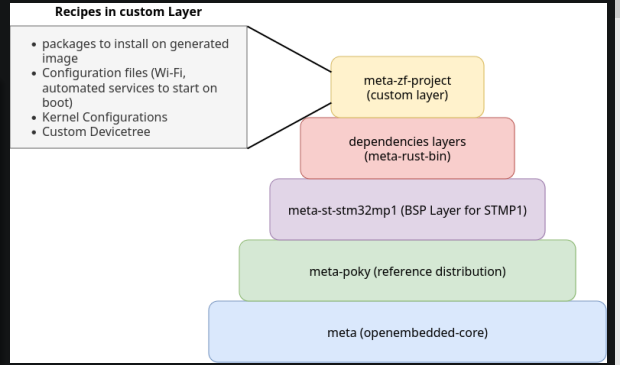
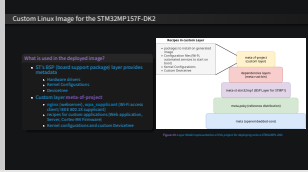


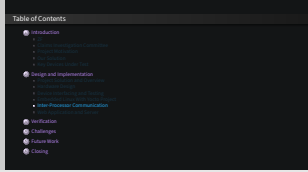
Figure 19: *Layer Model representation of this project for deploying onto a STM32MP1-DK2*



The yocto project was essential for building and configuring

1. hardware drivers
2. Linux Kernel configurations
3. Linux kernel device tree
4. and using the layer structure created by the yocto project a custom layer of meta-data was made to build and configure
5. nginx, a webserver to host the web application
6. wpa_supplicant, a wi-fi access client
7. among the meta-data there were build recipes for
8. written programs like the web-application, web server, and the Cortex-M4 firmware
9. a custom kernel and and a custom devicetree

Table of Contents



Inter-Process Communication on a Heterogenous Architecture

With a heterogenous architecture (ARM Cortex-A7 and ARM Cortex-M4) how can information be shared?

Hetergenous 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 recieving/transmitting messages over **RPMsg protocol**
- STMP1 layer automatically enables the **RPMSG 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

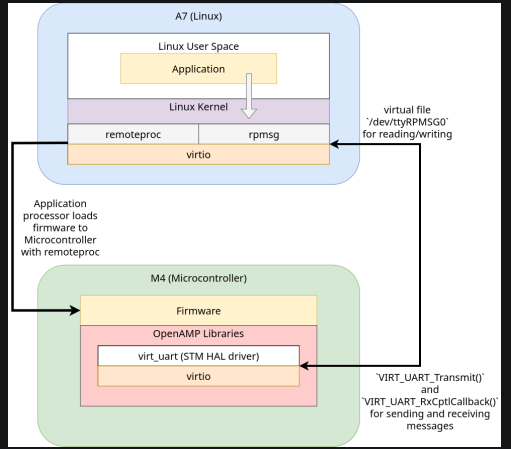
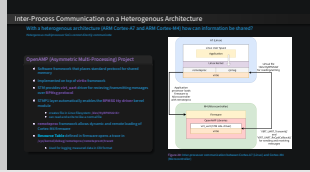
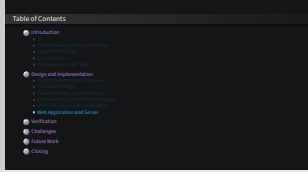


Figure 20: Inter-processor communication between Cortex-A7 (Linux) and Cortex-M4 (Microcontroller)



1. a major problem to solve in this project was having the two processors communicate to eachother
2. the cortex-m4 and cortex-A7 are on the same system on chip so having them communicate is not for free
3. OpenAMP is a project that aims to solve this project by standardizing how heterogenous architectures communicate between cores
4. to briefly summarize how its used in this project:
5. it allows linux to load the cortex-M4 firmware after a device under test is determined
6. communication between the two processors is possible, with the cortex-m4 utilizing some abstraction called virtuart and linux having a virtual file to read and write to
7. and logging from the cortex-m4 is done through writing to a linux kernel trace file, which will be used for logging the CSV data

Table of Contents



Rust

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

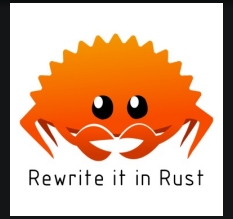
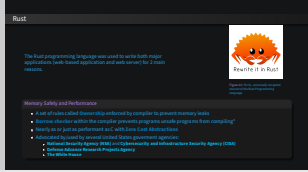


Figure 21: *Ferris, universally accepted mascot of the Rust Programming language*

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



1. The Rust Programming language was used to write the following 2 pieces of software
2. as it is a modern and safe language that is being looked into as an alternative to the C programming language

Web-Application for User Interface

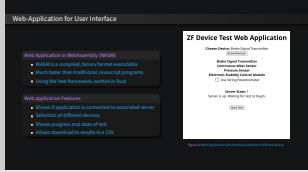
Web Application in WebAssembly (WASM)

- 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*



1. The way the tester interacts with our solution is through the web application as a web assembly application ,which is a modern solution to slow web programs which are usually written in javascript.
2. Web assembly is much faster than javascript as it is ran in web browsers as a compiled binary program
3. the web application allows a selection of the device they wish to test, and the starting of the test, which sends serialized JSON data to the web server which we will talk about next

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

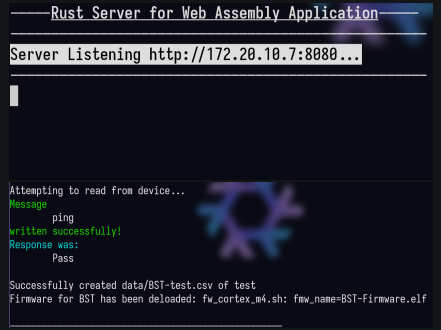
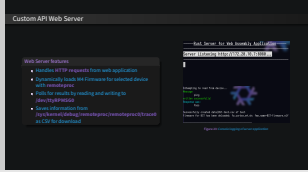


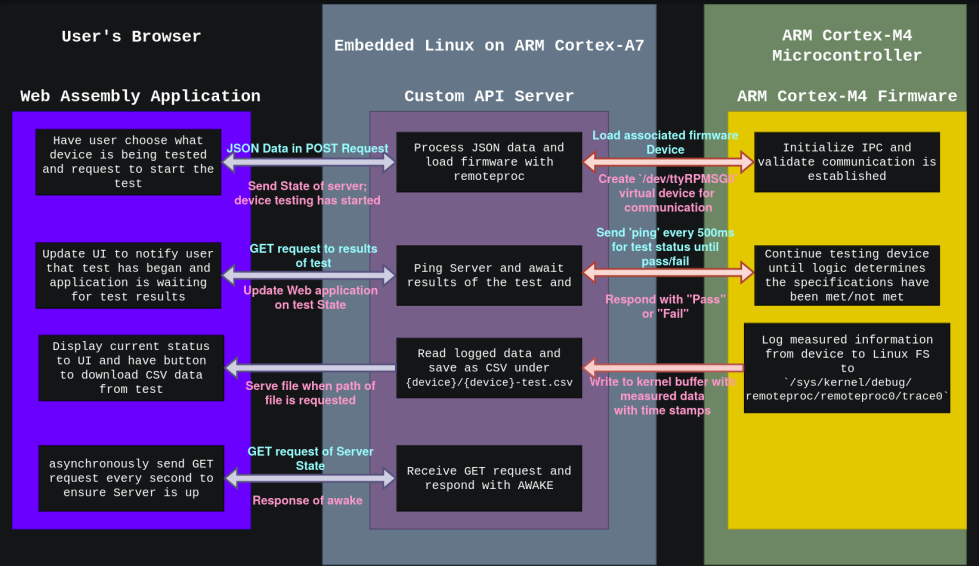
Figure 23: *Console logging of server application*

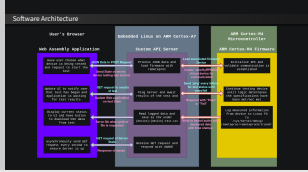


the web server, also developed in rust, has multiple responsabilities

1. it handles http requests from the web application
2. responsible for the interprocess communication between the two processors
3. it loads the firmware to the cortex-m4 from the information recieved by the web-application
4. It polls the Cortex-M4 for test completion and once it is, it saves the written data in the kernel trace and saves it for download
5. **image shows the debug logging of the server application

Software Architecture

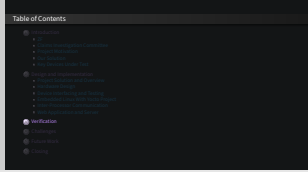




this diagram shows a comprehensive interactions of all 3 moving parts of the project

1. it begins with the user choosing what device to test
2. it sends a HTTP request to the server and then loads the necessary firmware
3. once its loaded, a message is sent back to the server and the server updates the state of the web application user interface
4. the server polls the test for completion and once it is, it notifies the server
5. the server again updates the user interface on the web application and also shows a button to download the CSV and to restart another test

Table of Contents



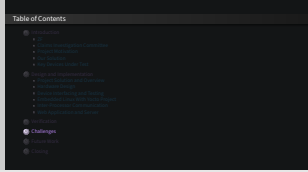
Link to video demonstration

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

Link to video demonstration
https://dylxndy.xyz/senior-design-presentation/verification

1. top center: BST physical setup with PCB and STM board
2. bottom center: Osciliscope visualization of PWM
3. top right: web application user interface
4. bottom right: live readings of PWMs with duty cycle and frequency
5. Right side: will be excel file

Table of Contents



Challenges

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

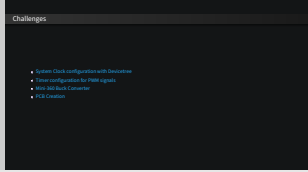
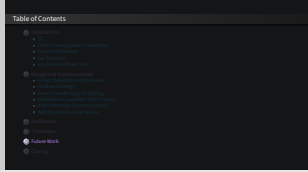


Table of Contents



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 acception criteria by 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

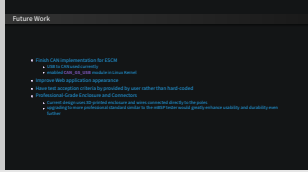
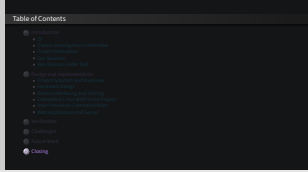
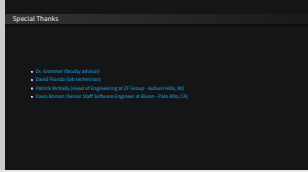


Table of Contents



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)

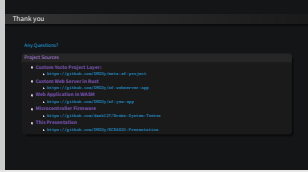


Thank you

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



1. some note here