

Ultimate Truck Hacking Platform

Unifying Truck Hacking Tools with the Power of Yocto Linux and BeagleBone Hardware

Dr. Jeremy Daily, Associate Professor of Systems Engineering

Rik Chatterjee, Graduate Student in Systems Engineering

Carson Green, Undergraduate in Electrical and Computer Engineering

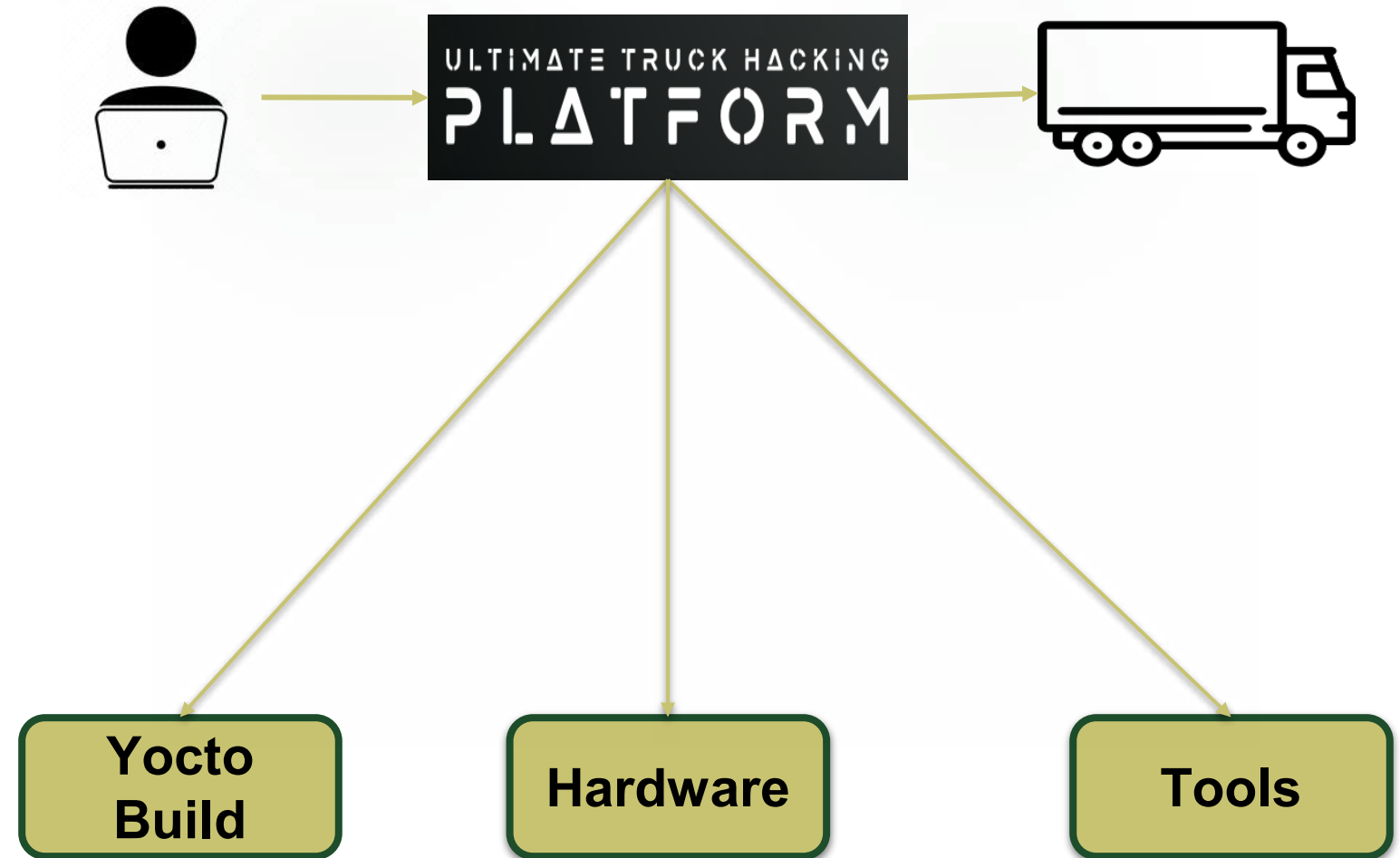


Colorado State University

What is the UTHP ?

A Tool Revolutionizing Vehicle Diagnostics and Security:

- **Unified Solution:** Combining diverse truck hacking tools under one platform for seamless diagnostics and cybersecurity analysis.
- **Yocto-based:** Harnessing the flexibility of Yocto Linux to ensure adaptability and wide-ranging compatibility.
- **Hardware Meets Software:** Introducing a specialized BeagleBone based hardware designed to work hand-in-glove with our software build.
- **Future-Proof:** Regular kernel updates and platform porting to stay ahead in the fast-evolving world of vehicle technology.



Why do we need a new Platform ?

- **Enhanced Vehicle Security:** As trucks become more technologically advanced, the need for comprehensive security diagnostics increases. UTHP offers an all-in-one solution to identify vulnerabilities and potential threats.
- **Unified Toolset:** Currently, professionals and enthusiasts must juggle various tools for different diagnostic tasks. UTHP offers a centralized platform, reducing the learning curve and increasing efficiency.
- **Future of Transportation:** With the rise of autonomous vehicles and smart transportation, it's crucial to stay ahead of potential cyber threats. UTHP provides a foundation for future-proofing.

To prevent something like this due to a cyberattack

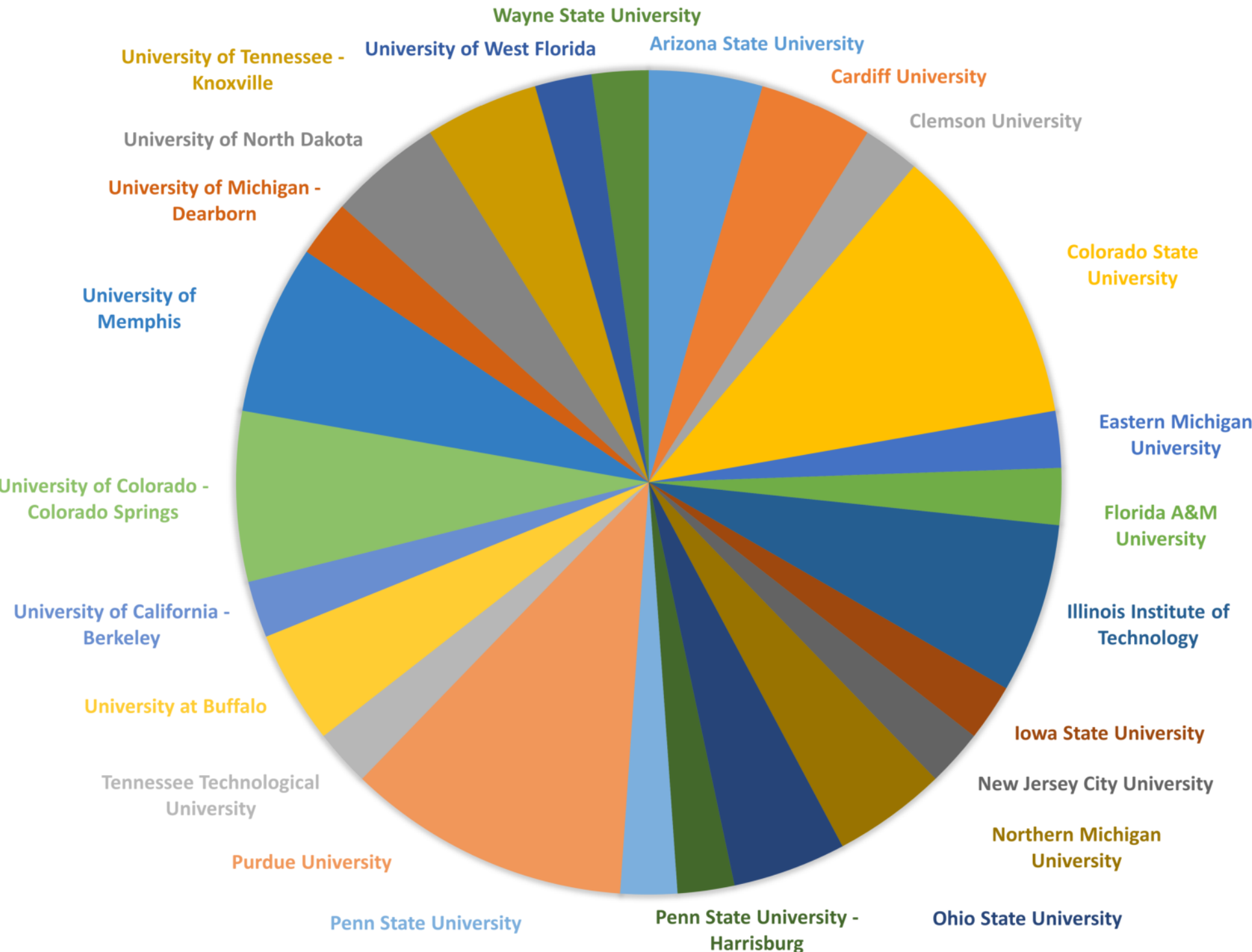


CyberTruck Challenge

Experience to guide in the design decisions and goals.



UNIVERSITIES REPRESENTED AT THE CYBERTRUCK CHALLENGE 2023



Participants

- Students
- Industry
- Instructors
- Mentors

Thank you to the CyberTruck Challenge® sponsors



GEO TAB
management by measurement

PACCAR



BOSCH

NAVISTAR®

Silver
Sponsors



U.S. Department
of Transportation

Federal Motor
Carrier Safety
Administration

DAIMLER

Bronze
Sponsors



SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY

IOActive®



**Macomb
Community College**

Education • Enrichment • Economic Development
Discover. Connect. Advance.™



Contributors

BATTELLE
It can be done

V O L V O



**UNIVERSITY OF
DETROIT
MERCY**
Build A Boundless Future



Sponsors have access to
some outstanding new
talent in cybersecurity.

Warren Michigan, 12-16 June 2023.

CyberTruck Challenge 2023 Schedule

Version:20230516

	Sunday, 11 June	Monday, 12 June		Tuesday, 13 June		Wednesday, 14 June	Thursday, 15 June	Friday, 16 June	Time	
		Group A	Group B	Group A	Group B					
Before 0700	Site Closed	Site Closed							Before 0700	
0700-0730		Breakfast							Breakfast	0700-0730
0730-0800									Student Team Briefs (30 minutes each group)	0730-0800
0800-0830		Welcome // NDA		Wireless Systems	Hardware Reverse Engineering	Safety & Legal Briefing				0800-0830
0830-0900		Vehicle Orientation				Assessment				Assessment
0900-0930				Cryptography	Binary Analysis and Modification					
0930-1000		Software RE	Truck Systems and J1939							
1000-1030										
1030-1100										
1100-1130										
1130-1200										
1200-1230				Lunch						
1230-1300									1230-1300	
1300-1330									1300-1330	
1330-1400		Informal Welcome Reception (offsite)	Binary Decompileation	Diagnostic Systems	Lunch				Site Closed	1330-1400
1400-1430										
1430-1500					Hardware Reverse Engineering	Wireless Systems	Assessment	Assessment		
1500-1530										Truck Systems and J1939
1530-1600					Binary Analysis and Modification	Cryptography				
1600-1630										
1630-1700								1630-1700		
1700-1730										
1730-1800										
1800-1830								Diagnostic Systems		Binary Decompileation
1830-1900	Dinner		Dinner Presentation: Trucking Industry				1830-1900			
1900-1930	Site Closed	Dinner Presentation: Dr. Bratus				Assessment	Free	1900-1930		
1930-2000								1930-2000		
2000-2030								2000-2030		
2030-2100	Site Closed			Assessment Preparation				2030-2100		
2100-2130		Free						2100-2130		
2130-2200								2130-2200		
After 2200		Site Closed							After 2200	

What's it like?



What's it like?



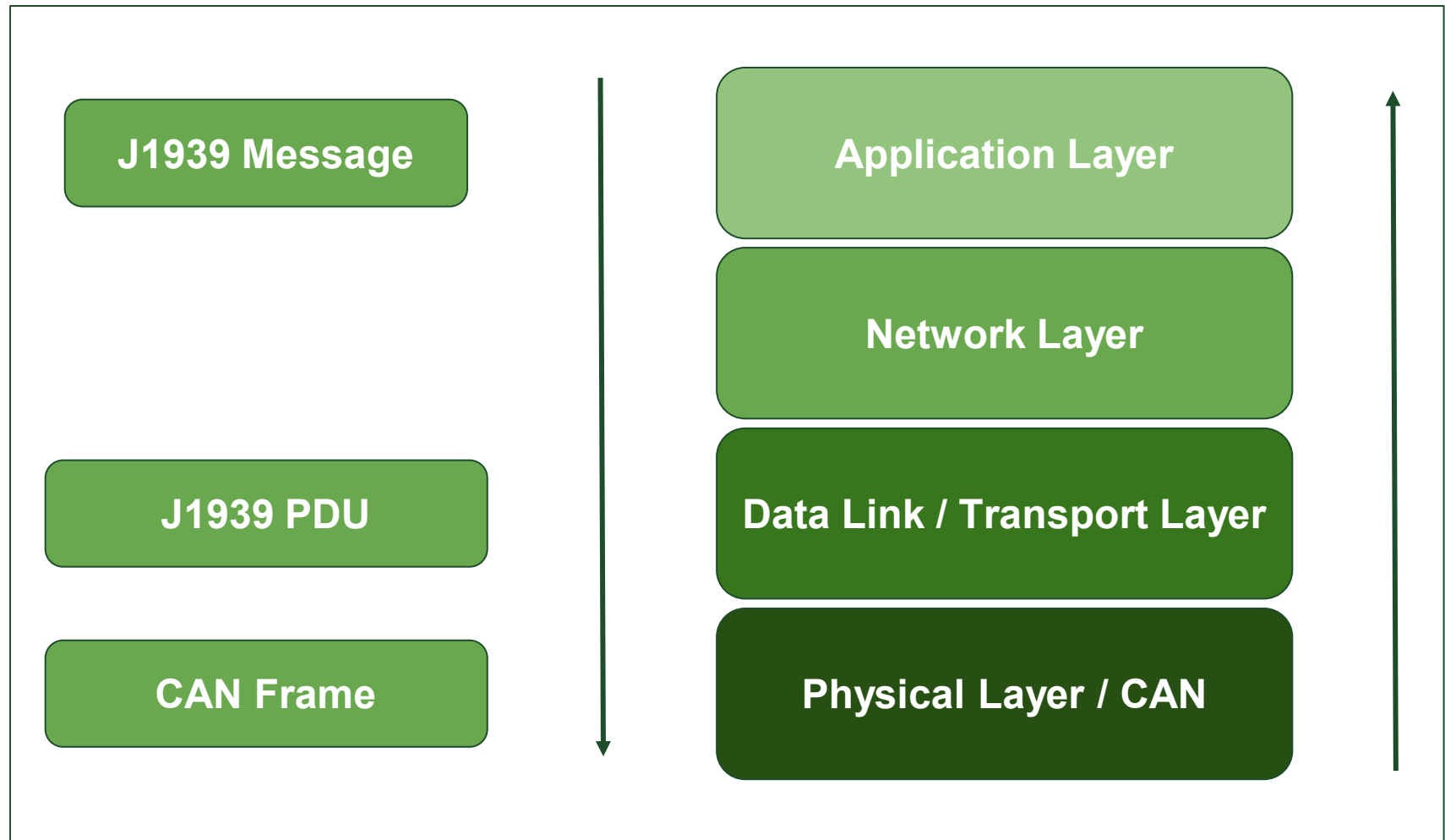
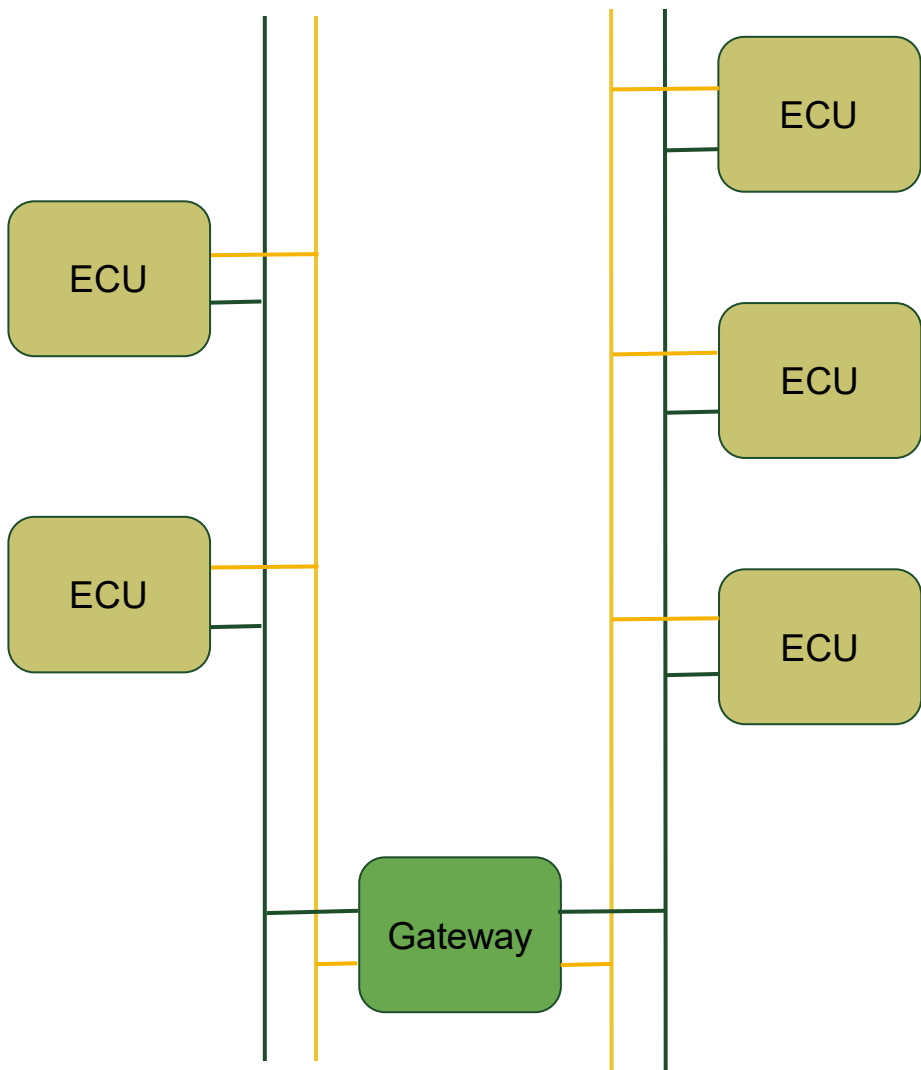
Ultimate Truck Hacking Platform

Brings together useful hardware, tools, and operating systems for a complete, low-cost system for education, training, research and assessments for cybersecurity of heavy vehicles across multiple networking layers in the system.



Colorado State University

Truck Networks



Cybersecurity Threats in Commercial Vehicles

Co-located with the 25th USENIX Security Symposium

WOOT '16 10th USENIX Workshop on Offensive Technologies

AUGUST 8-9, 2016 • AUSTIN, TX

HOME ATTEND PROGRAM SPONSORSHIP PARTICIPATE ABOUT

HELP PROMOTE

WOOT '16
AUGUST 8-9, 2016
AUSTIN, TX
usenix.org/woot16

CONNECT WITH US

Twitter Facebook LinkedIn Google+ YouTube

Truck Hacking: An Experimental Analysis of the SAE J1939 Standard

Authors:
Yelizaveta Burakova, Bill Hass, Leif Millar, and André Weimerskirch, *University of Michigan*

Abstract:
Consumer vehicles have been proven to be insecure; the addition of electronics to monitor and control vehicle functions have added complexity resulting in safety critical vulnerabilities. Heavy commercial vehicles have also begun adding electronic control systems similar to consumer vehicles. We show how the openness of the SAE J1939 standard used across all US heavy vehicle industries gives easy access for safety-critical attacks and that these attacks aren't limited to one specific make, model, or industry.

We test our attacks on a 2006 Class-8 semi tractor and 2001 school bus. With these two vehicles, we demonstrate how simple it is to replicate the kinds of attacks used on consumer vehicles and that it is possible to use the same attack on other vehicles that use the SAE J1939 standard. We show safety critical attacks that include the ability to accelerate a truck in motion, disable the driver's ability to accelerate, and disable the vehicle's engine brake. We conclude with a discussion for possibilities of additional attacks and potential remote attack vectors.

Open Access Media
USENIX is committed to Open Access to the research presented at our events. Papers and proceedings are freely available to everyone once the event begins. Any video, audio, and/or slides that are posted after the event are also free and open to everyone. [Support USENIX](#) and our commitment to Open Access.

[BibTeX](#) [Burakova PDF](#) [View the slides](#)

TWITTER

Application Layer

Network Layer

Data Link / Transport Layer

Physical Layer / CAN

Cybersecurity Threats in Commercial Vehicles

Journals & Magazines > IEEE Transactions on Vehicula... > Volume: 67 Issue: 5

Security Shortcomings and Countermeasures for the SAE J1939 Commercial Vehicle Bus Protocol

Publisher: IEEE [Cite This](#) [PDF](#)

Pal-Stefan Murva ; Bogdan Groza [All Authors](#)

24 Cites in Papers 1443 Full Text Views

Abstract

Document Sections

- I. Introduction
- II. Background
- III. Security Shortcomings of the J1939
- IV. An Authentication Mechanism for J1939

Abstract:

In the recent years, countless security concerns related to automotive systems were revealed either by academic research or real life attacks. While current attention was largely focused on passenger cars, due to their ubiquity, the reported bus-related vulnerabilities are applicable to all industry sectors where the same bus technology is deployed, i.e., the CAN bus. The SAE J1939 specification extends and standardizes the use of CAN to commercial vehicles where security plays an even higher role. In contrast to empirical results that attest such vulnerabilities in commercial vehicles by practical experiments, here, we determine that existing shortcomings in the SAE J1939 specifications open road to several new attacks, e.g., impersonation, denial of service (DoS), distributed DoS, etc. Taking the advantage of an industry-standard CANoe based simulation, we demonstrate attacks with potential safety critical effects that are mounted while still conforming to the SAE J1939 standard specification. We discuss countermeasures and security enhancements by including message authentication mechanisms. Finally, we evaluate and discuss the impact of employing these mechanisms on the overall network communication.

Application Layer

Network Layer

Data Link / Transport Layer

Physical Layer / CAN

Cybersecurity Threats in Commercial Vehicles

NDSS About 2024 Symposium 2023 Symposium Previous Events 2024 Submissions

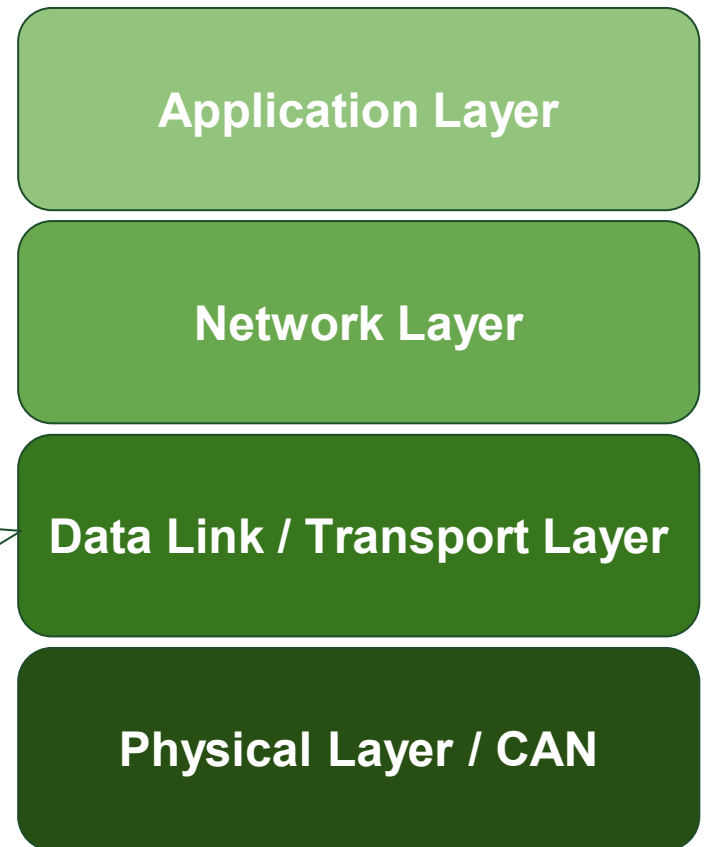
Exploiting Transport Protocol Vulnerabilities in SAE J1939 Networks

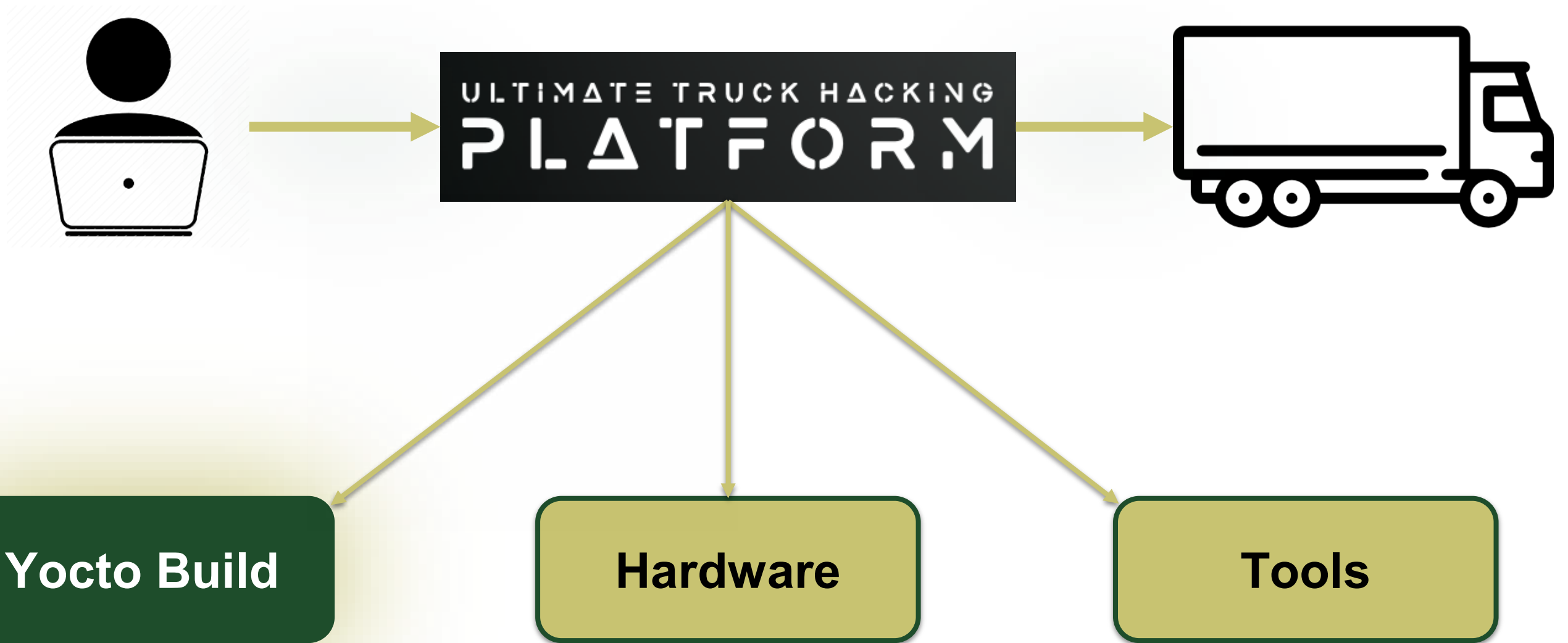
Rik Chatterjee, Subhojeet Mukherjee, Jeremy Daily (Colorado State University)

Modern vehicles are equipped with embedded computers that utilize standard protocols for internal communication. The SAE J1939 protocols running on top of the Controller Area Network (CAN) protocol is the primary choice of internal communication for embedded computers in medium and heavy-duty vehicles. This paper presents five different cases in which potential shortcomings of the SAE J1939 standards are exploited to launch attacks on in-vehicle computers that constitute SAE J1939 networks.

In the first two of these scenarios, we validate the previously proposed attack hypothesis on more comprehensive testing setups. In the later three of these scenarios, we present newer attack vectors that can be executed on bench test setups and deployed SAE J1939 networks. For the purpose of demonstration, we use bench-level test systems with real electronic control units connected to a CAN bus. Additional testing was conducted on a 2014 Kenworth T270 Class 6 truck under both stationary and driving conditions. Test results show how protocol attacks can target specific ECUs. These attacks should be considered by engineers and programmers implementing the J1939 protocol stack in their communications subsystem.

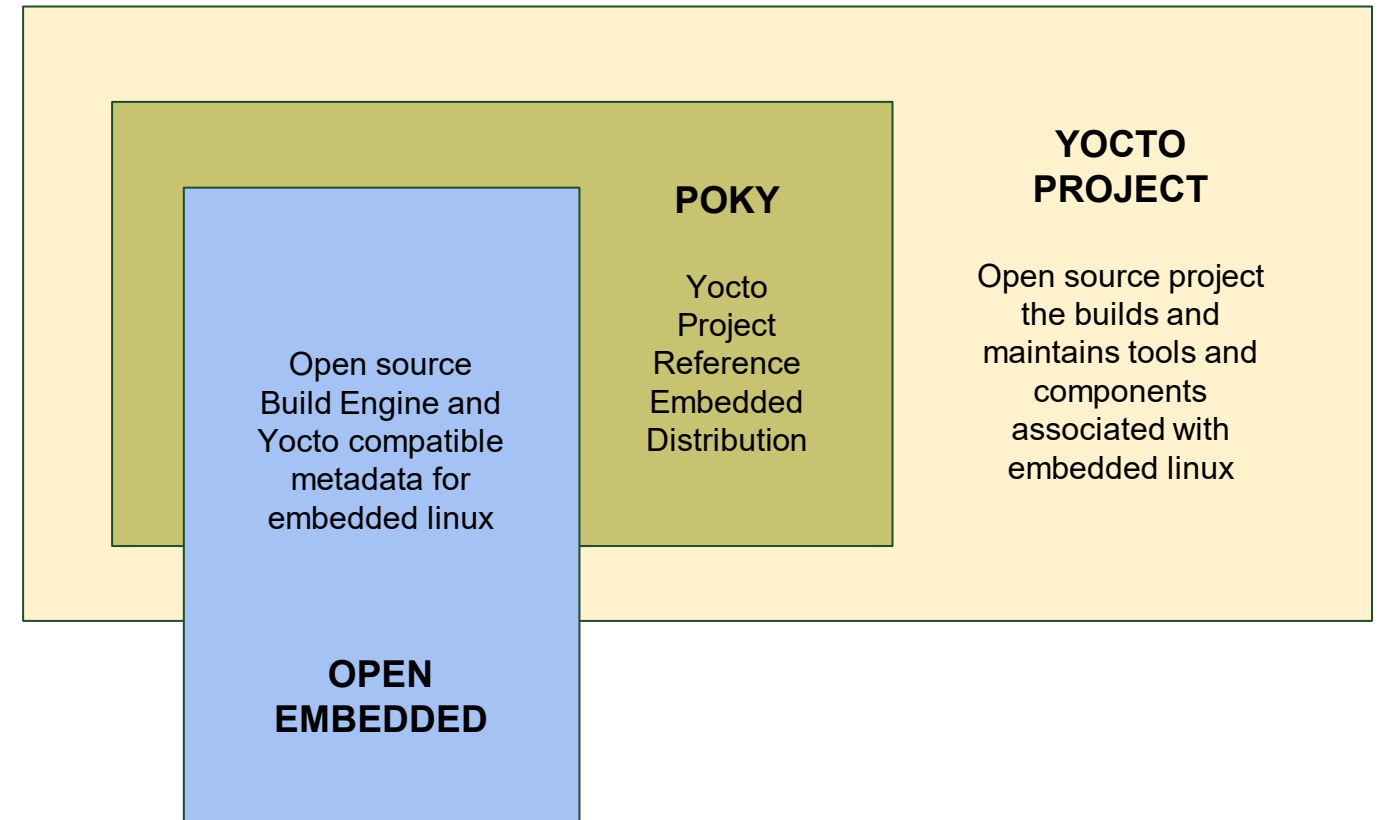
[Paper](#)





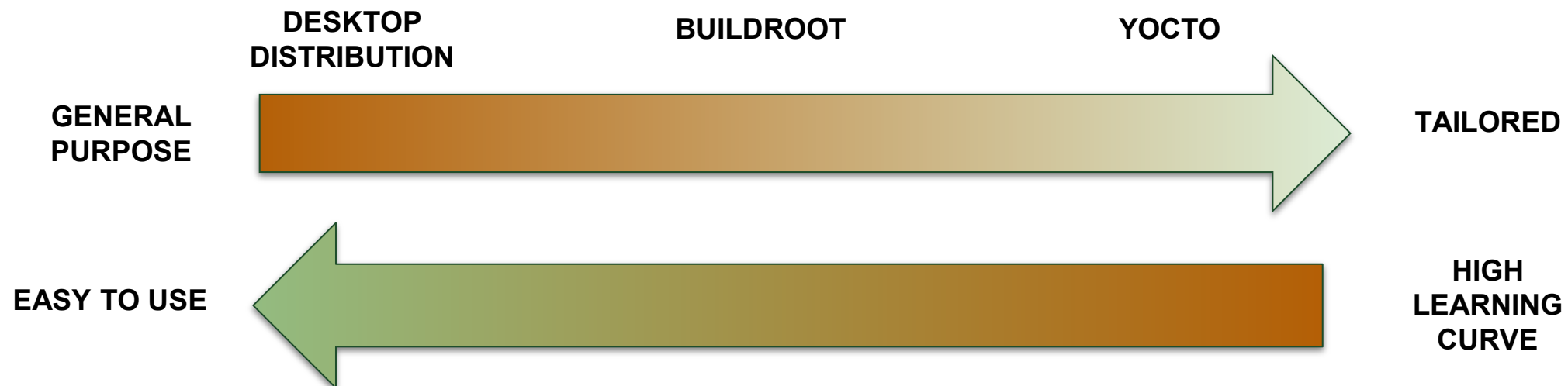
What is Yocto ?

- **Definition:** Yocto is a collaborative project that provides templates, tools, and methods to create custom Linux-based systems for embedded products.
- **OpenEmbedded Build System:** Utilizes the OpenEmbedded build framework, which comprises a collection of "recipes" detailing how software components are compiled and integrated.
- **Layered Approach:** Organizes functionalities in layers, allowing for better modularity and separation of concerns.
- **Not Just Another Distribution:** Yocto isn't a Linux distribution itself but a toolkit to develop distributions tailored for specific needs.
- **Backed by the Linux Foundation:** Ensuring consistent development and maintenance backed by industry professionals.



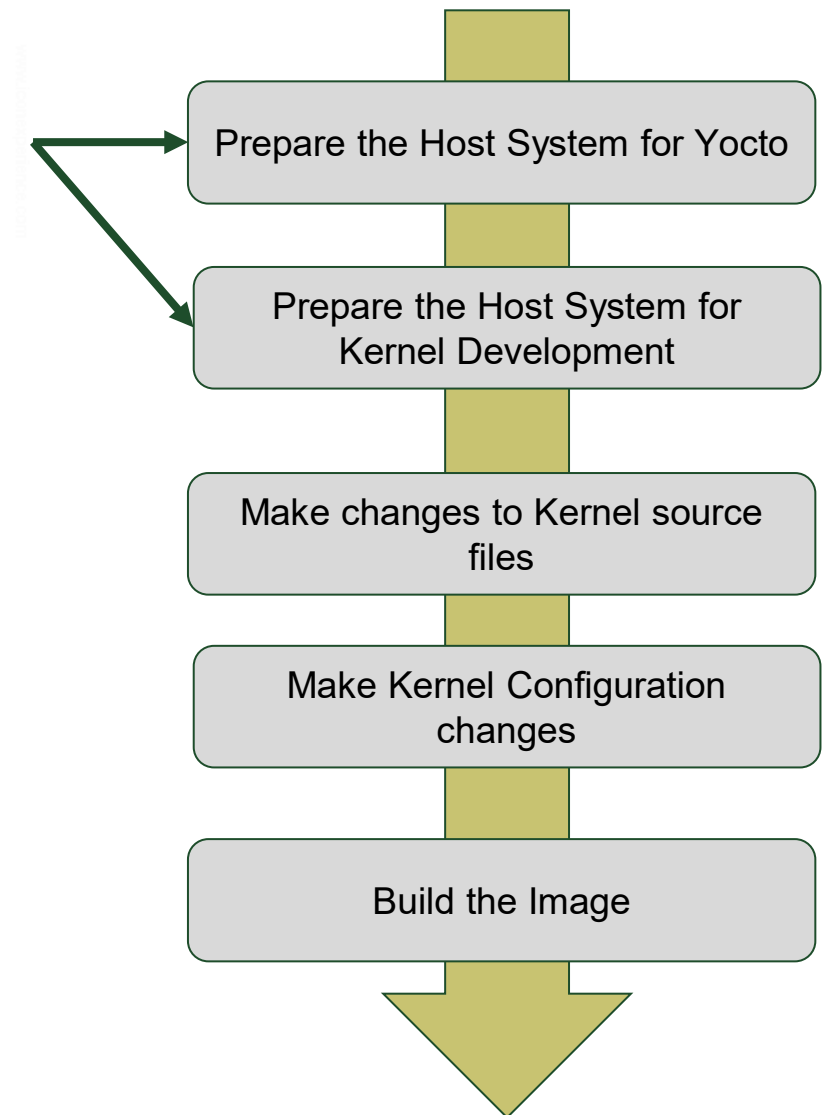
Why Choose Yocto ?

- **Customized Linux Systems:** Yocto crafts a tailor-made OS, optimized for specific needs and hardware architectures.
- **Layer Model:** Yocto's flexible layering system, allows developers to add or remove functionalities as needed.
- **Rich Ecosystem:** A vast network of available layers and configurations on the net accelerates development, especially with layers designed by board founders and manufacturers.
- **Complete Granularity:** Beyond just generating full system images, Yocto offers granularity in generating bootloader, kernel, filesystem, and toolchain, ensuring every component is precisely as intended.
- **Learning Curve vs. Flexibility:** While Yocto has a steeper learning curve, its power and flexibility are unparalleled. When a project requires in-depth customization, particularly for embedded systems aiming for efficiency, Yocto is often the optimal choice.
- **Comparative Advantage:** Compared to desktop distributions, Yocto is more tailored for specific product goals, especially when lightness, speed, and ultra-customization are priorities. While Buildroot offers simplicity, Yocto provides a broader spectrum of packages and capabilities.



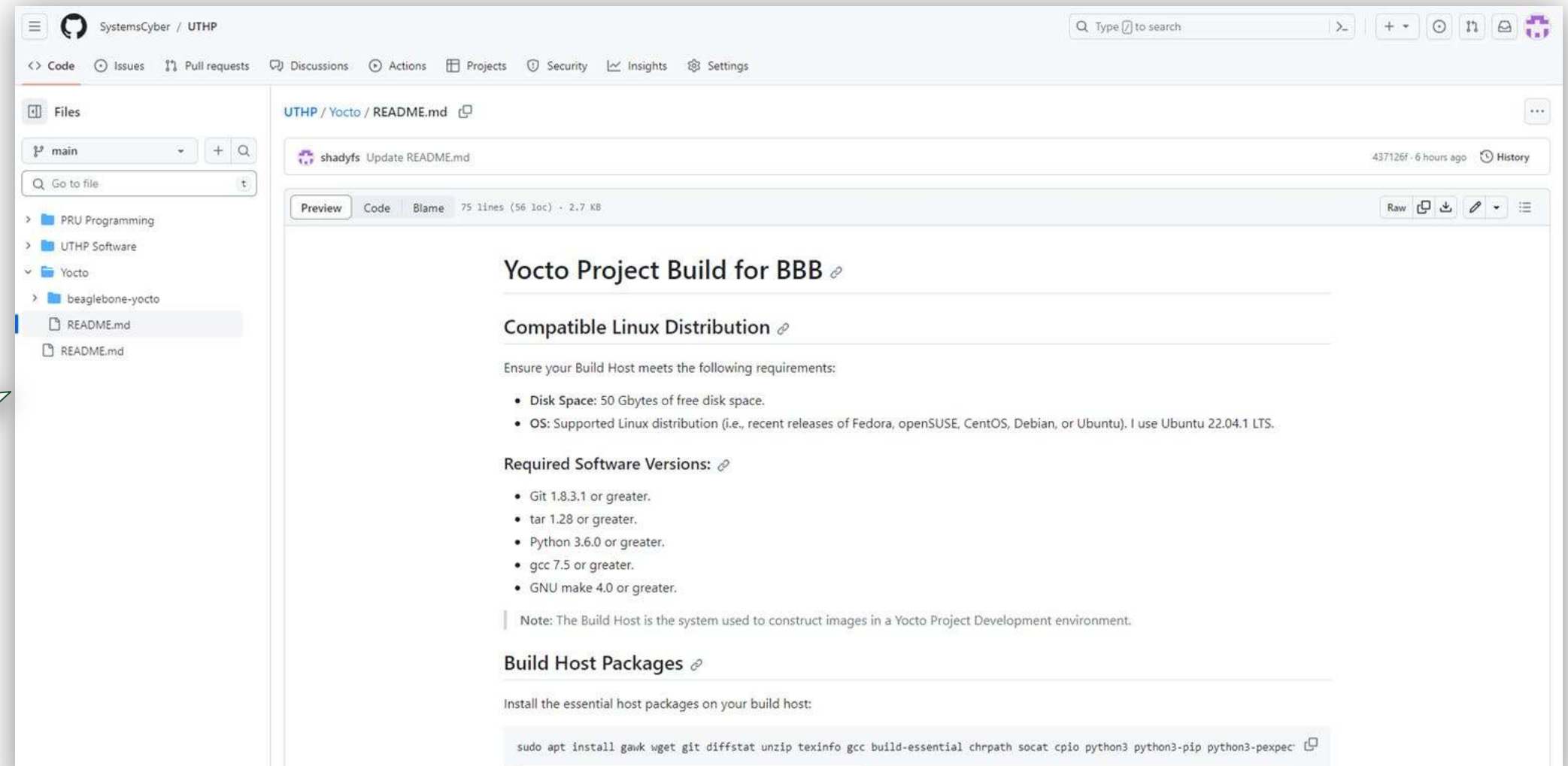
Building with Yocto for UTHP

- **Install Essential Packages:** Tools and dependencies for Yocto.
- **Clone Poky Repository:** The heart of the Yocto Project.
- **Navigate to Poky Directory:** View and choose the relevant branch.
- **Checkout 'kirkstone' Branch:** The designated release for our project.
- **Initialize Build Environment:** Using `oe-init-build-env` within the 'poky' directory.
- **Review & Modify 'local.conf':** Adapt configurations to your project needs.
- **Execute BitBake:** Kickstarts the entire build process.
- **Monitor Build Progress:** Track the construction of your custom Linux distribution.
- **Flash the image to the Beaglebone:** Use 'dd' or and other tool.



Building with Yocto for UTHP

<https://github.com/SystemsCyber/UTHP>



The screenshot shows the GitHub interface for the repository SystemsCyber / UTHP. The left sidebar displays the file structure with folders for PRU Programming, UTHP Software, Yocto, and beaglebone-yocto, each containing a README.md file. The main content area shows the README.md file for the Yocto Project Build for BBB, updated by shadyfs. The README includes sections for Compatible Linux Distribution, Required Software Versions, and Build Host Packages.

Yocto Project Build for BBB

Compatible Linux Distribution

Ensure your Build Host meets the following requirements:

- Disk Space: 50 Gbytes of free disk space.
- OS: Supported Linux distribution (i.e., recent releases of Fedora, openSUSE, CentOS, Debian, or Ubuntu). I use Ubuntu 22.04.1 LTS.

Required Software Versions:

- Git 1.8.3.1 or greater.
- tar 1.28 or greater.
- Python 3.6.0 or greater.
- gcc 7.5 or greater.
- GNU make 4.0 or greater.

Note: The Build Host is the system used to construct images in a Yocto Project Development environment.

Build Host Packages

Install the essential host packages on your build host:

```
sudo apt install gawk wget git diffstat unzip texinfo gcc build-essential chrpath socat cpio python3 python3-pip python3-pexpect
```

The diagram illustrates the architecture of the 'ULTIMATE TRUCK HACKING PLATFORM'. On the left, a user icon (a person at a laptop) is connected by a dotted line to the platform. The platform is represented by a black box with the text 'ULTIMATE TRUCK HACKING PLATFORM' in white. To the right of the platform, there are two vertical lines: a yellow one and a green one. The yellow line connects to a truck icon, and the green line connects to another truck icon. Above each of these connections is a red circle with a white 'X' inside, indicating a restriction or a specific type of connection. The entire diagram is set against a light gray background.

```
ALSA: Restoring mixer settings...
alsa-lib ../../../../alsa-lib-1.2.6.1/src/ucm/main.c:1412:(snd_use_case_mgr_open) error: failed to import hw:0 use case configuration -2
No state is present for card Black
alsa-lib ../../../../alsa-lib-1.2.6.1/src/ucm/main.c:1412:(snd_use_case_mgr_open) error: failed to import hw:0 use case configuration -2
Found hardware: "simple-card" "" "" "" ""
Hardware is initialized using a generic method
No state is present for card Black
Configuring packages on first boot....
(This may take several minutes. Please do not power off the machine.)
Running postinst /etc/rpm-postinsts/100-sysvinit-inittab...
update-rc.d: /etc/init.d/run-postinsts exists during rc.d purge (continuing)
Removing any system startup links for run-postinsts ...
/etc/rcS.d/S99run-postinsts
INIT: Entering runlevel: 5
Configuring network interfaces... cpsw-switch 4a100000.switch: starting ndev. mode: dual_mac
SMSC LAN8710/LAN8720 4a101000.mdio:00: attached PHY driver (miibus:phy_addr=4a101000.mdio:00, irq=PO
LL)
udhcpc: started, v1.35.0
udhcpc: broadcasting discover
udhcpc: broadcasting discover
udhcpc: broadcasting discover
udhcpc: no lease, forking to background
done.
Starting system message bus: dbus.
Starting rpcbind daemon...done.
Starting bluetooth: bluetoothd.
Starting syslogd/klogd: done
* Starting Avahi mDNS/DNS-SD Daemon: avahi-daemon
...done.
Starting Telephony daemon
Starting Linux NFC daemon

Poky (Yocto Project Reference Distro) 4.0.12 beaglebone-yocto /dev/ttyS0

beaglebone-yocto login: root
root@beaglebone-yocto:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 3C:E4:B0:D6:5A:CF
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:11 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1696 (1.6 KiB)  TX bytes:1696 (1.6 KiB)

root@beaglebone-yocto:~#
```

ΛΟΟΓΘΡΕΩΔΥΕΡΟΥΕ-ΛΟΕΡΟ:-#

```

BX 0x00000000:00000000  1X 0x00000000:00000000
C0FFEE0000000000  0000000000000000
1X 0x00000000:00000000  0x00000000:00000000  0x00000000:00000000  0x00000000:00000000
BX 0x00000000:00000000  0x00000000:00000000  0x00000000:00000000  0x00000000:00000000

```


Customizing the build: Recipes & Layers

➤ Recipes:

- Individual instructions for building software components.
- Specify source location, dependencies, and installation procedure.

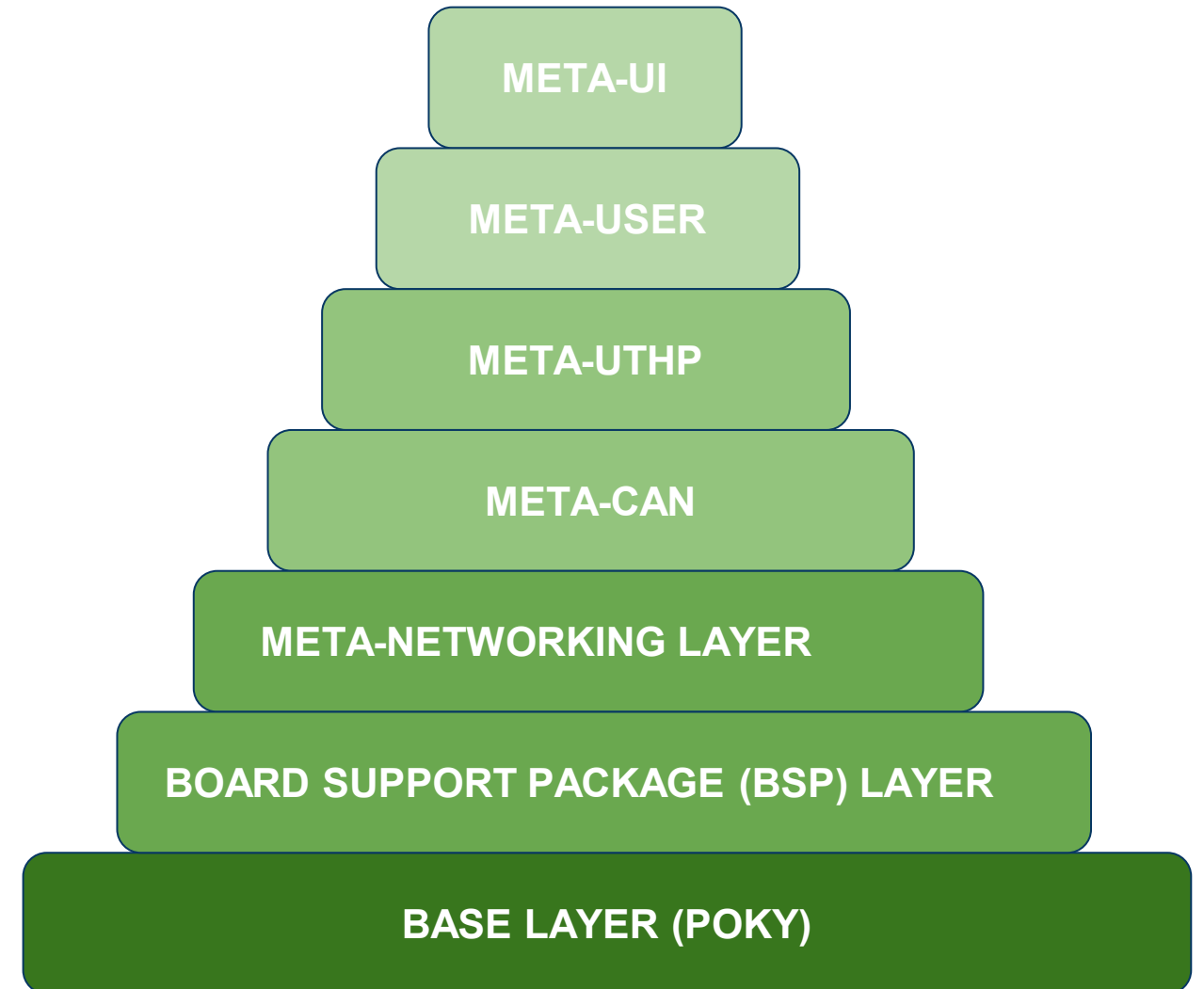
➤ Layers:

- Collections of related recipes and configurations.
- Provide structure and modularity to the build process.

➤ Importance:

- Easily add or remove software components.
- Separate customizations or configurations.
- Maintain third-party software, hardware adaptations, or proprietary software.

```
meta-<layer-name>
|---- recipes-<category>
|---- <recipe-name>
|---- <recipe-name>.bb
```



Customizing the build: Adding CAN

➤ Kernel Recipe Append Creation:

- Create an append recipe specific for CAN: `linux-yocto_5.15-can.bbappend`.

➤ Specify Custom Configuration:

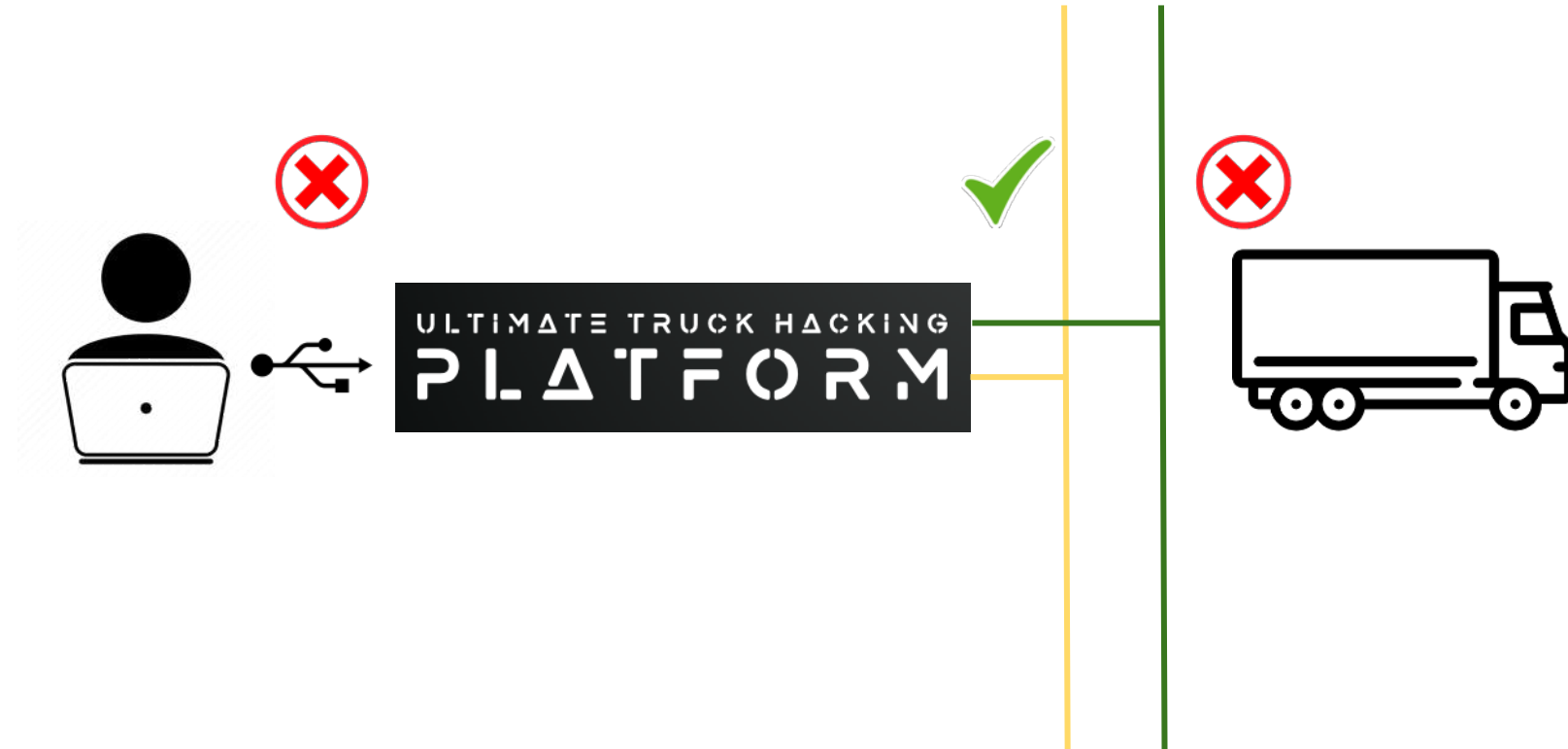
- Add configurations in a separate `can-support.cfg`.
 - `CONFIG_CAN=y`
 - `CONFIG_CAN_RAW=y`
 - `CONFIG_CAN_BCM=y`

➤ Update Kernel Append Recipe for CAN:

- Extend `SRC_URI` to include `can-support.cfg`.

➤ Integrate CAN-Utills:

- Update image recipe: to include `"can-utils"`



Customizing the build: USB ETHERNET

➤ Kernel Recipe Append Creation:

- Name: `linux-yocto_5.15-usb.bbappend`.
- Purpose: Enable USB gadget features.

➤ Specify Custom Configuration:

- Configuration Details:
 - `CONFIG_USB_ETH=y`
 - `CONFIG_USB_G_NCM=m`
 - `CONFIG_USB_MASS_STORAGE=y`

➤ Update Kernel Append Recipe:

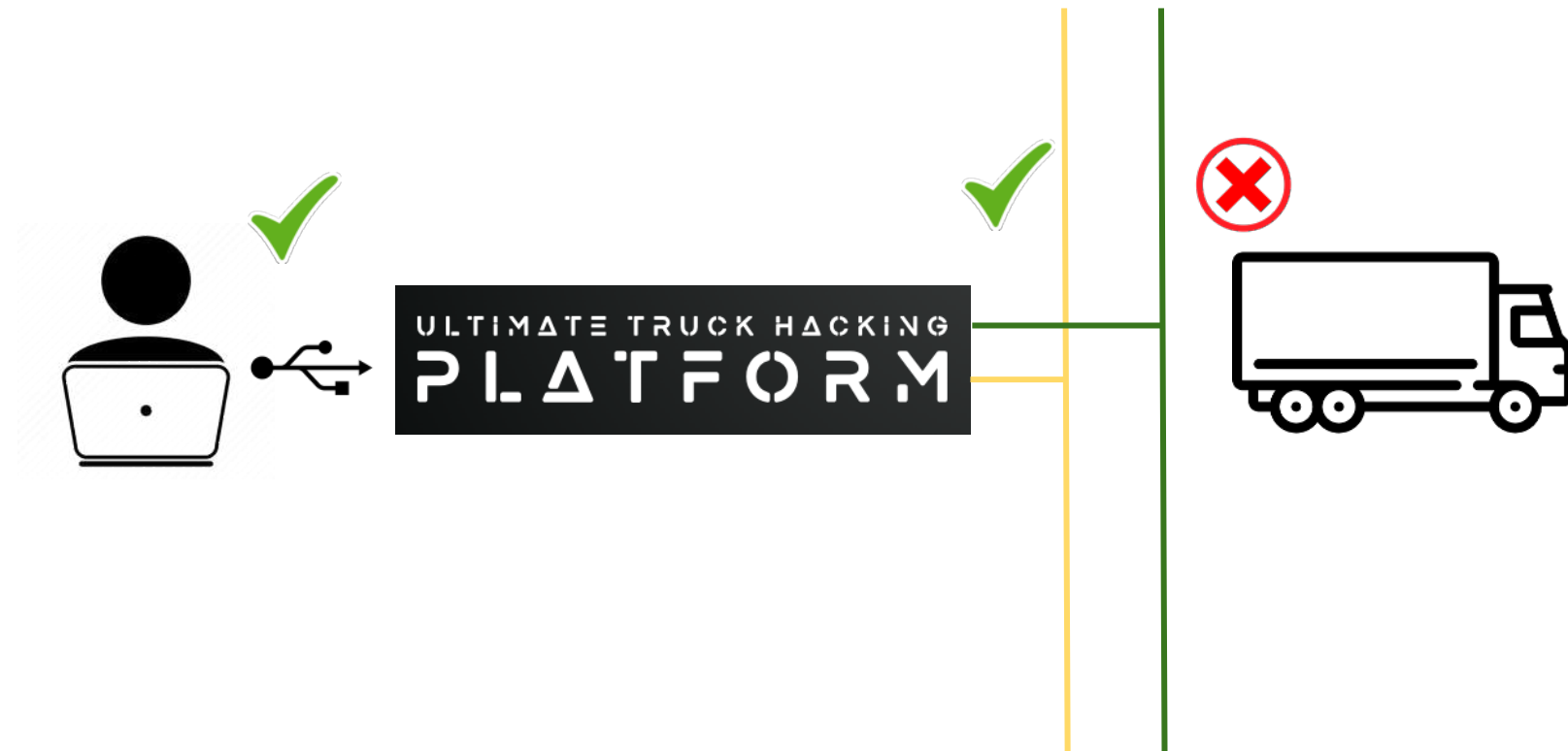
- Extend `SRC_URI` to include `usb.cfg`.

➤ Network Configuration:

- Define static IP for USB Ethernet.
- Update `/etc/systemd/network/usb0.network` on BeagleBone's root filesystem.

➤ Host Configuration:

- Manually set IP for the new network connection.



Customizing the build: Adding J1939

➤ Kernel Recipe Append Creation:

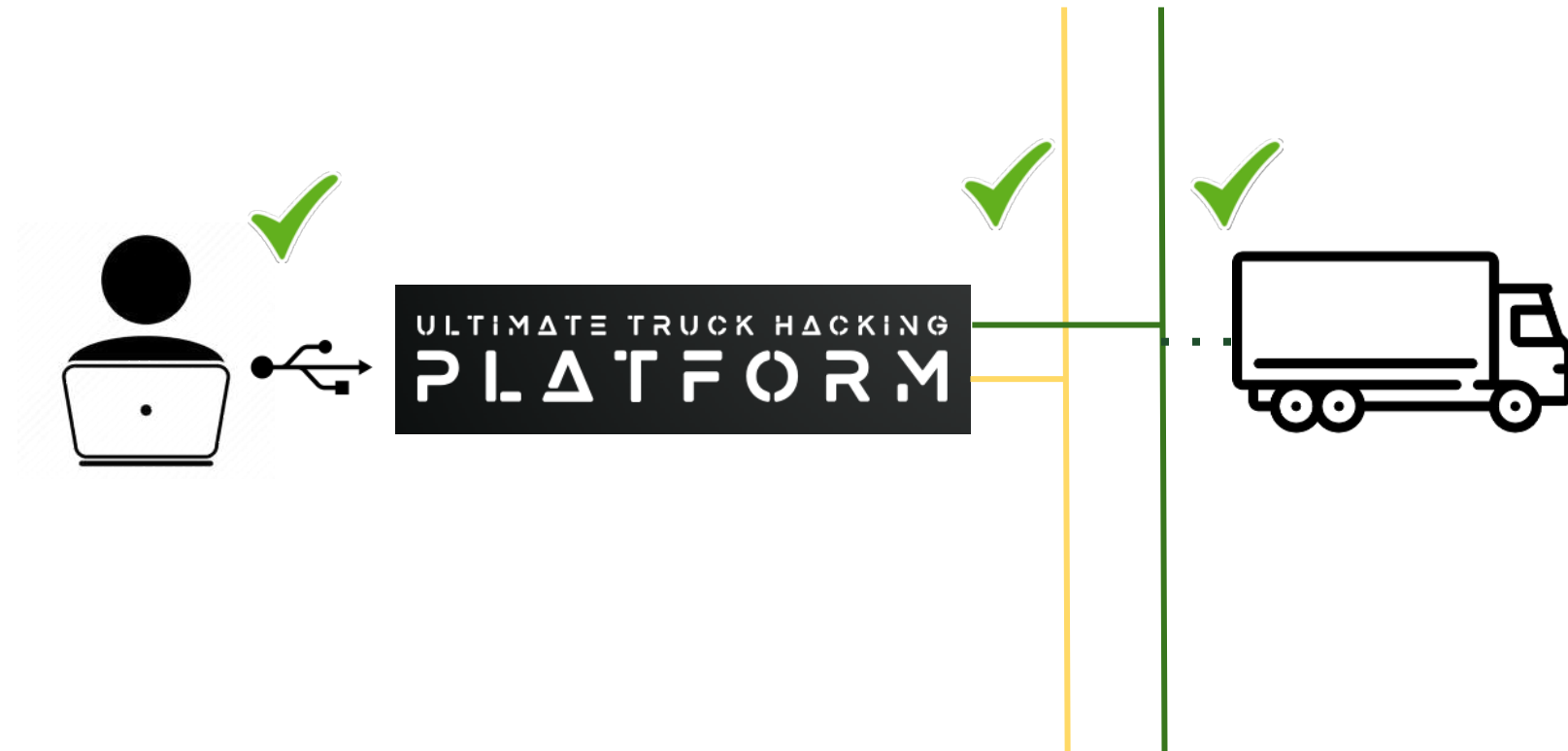
- Create a J1939 specific append:
 - `linux-yocto_5.15-j1939.bbappend.`

➤ Specify Custom Configuration:

- Define in `j1939-support.cfg.`
 - `CONFIG_CAN_J1939=y`

➤ Update Kernel Append Recipe for J1939:

- Extend `SRC_URI` to include `j1939-support.cfg.`



Current Status

➤ USB Ethernet Integration:

- Kernel Configuration: Successfully added and enabled.
- Image Recipe: `usbinit` integrated.
- Connectivity: Confirmed stable USB Ethernet connection via SSH.

➤ CAN Protocol Integration:

- Kernel Configuration: CAN support enabled.
- Image Recipe: `can-utils` integrated and functioning.
- Communication: Successful test transmission and reception on CAN bus.

➤ J1939 Protocol Integration:

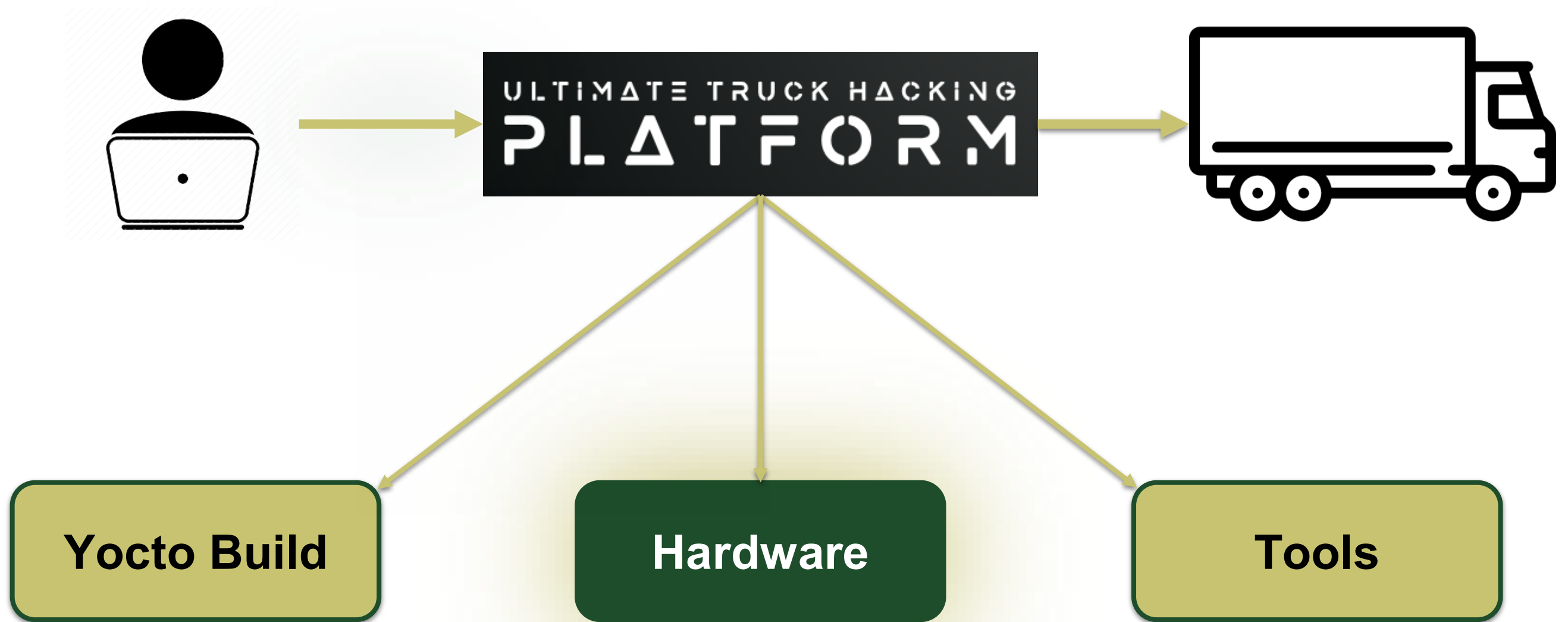
- Kernel Configuration: J1939 support enabled.
- Image Recipe: J1939 utilities and tools integrated.
- Protocol Testing: Successful J1939 communication verified.

```
rik@rik-pc:~/Documents/poky/build$ ping 192.168.7.2
PING 192.168.7.2 (192.168.7.2) 56(84) bytes of data.
64 bytes from 192.168.7.2: icmp_seq=1 ttl=64 time=0.761 ms
64 bytes from 192.168.7.2: icmp_seq=2 ttl=64 time=0.354 ms
64 bytes from 192.168.7.2: icmp_seq=3 ttl=64 time=0.328 ms
64 bytes from 192.168.7.2: icmp_seq=4 ttl=64 time=0.313 ms
64 bytes from 192.168.7.2: icmp_seq=5 ttl=64 time=0.404 ms
64 bytes from 192.168.7.2: icmp_seq=6 ttl=64 time=0.301 ms
64 bytes from 192.168.7.2: icmp_seq=7 ttl=64 time=0.283 ms
64 bytes from 192.168.7.2: icmp_seq=8 ttl=64 time=0.366 ms
64 bytes from 192.168.7.2: icmp_seq=9 ttl=64 time=0.224 ms
^C
--- 192.168.7.2 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8182ms
rtt min/avg/max/mdev = 0.224/0.370/0.761/0.146 ms
rik@rik-pc:~/Documents/poky/build$ ssh root@192.168.7.2
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!    @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ED25519 key sent by the remote host is
SHA256:dHLs+mFnjyeLNQn6EX0HT7c+ovp1+00xJNsCJbbe4zE.
Please contact your system administrator.
Add correct host key in /home/rik/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /home/rik/.ssh/known_hosts:2
  remove with:
    ssh-keygen -f "/home/rik/.ssh/known_hosts" -R "192.168.7.2"
Host key for 192.168.7.2 has changed and you have requested strict checking.
Host key verification failed.
rik@rik-pc:~/Documents/poky/build$ sudo rm -rf /home/rik/.ssh/known_hosts
rik@rik-pc:~/Documents/poky/build$ ssh root@192.168.7.2
The authenticity of host '192.168.7.2 (192.168.7.2)' can't be established.
ED25519 key fingerprint is SHA256:dHLs+mFnjyeLNQn6EX0HT7c+ovp1+00xJNsCJbbe4zE.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.7.2' (ED25519) to the list of known hosts.
Last login: Fri Mar  9 12:35:39 2018
root@beaglebone-yocto:~#
```

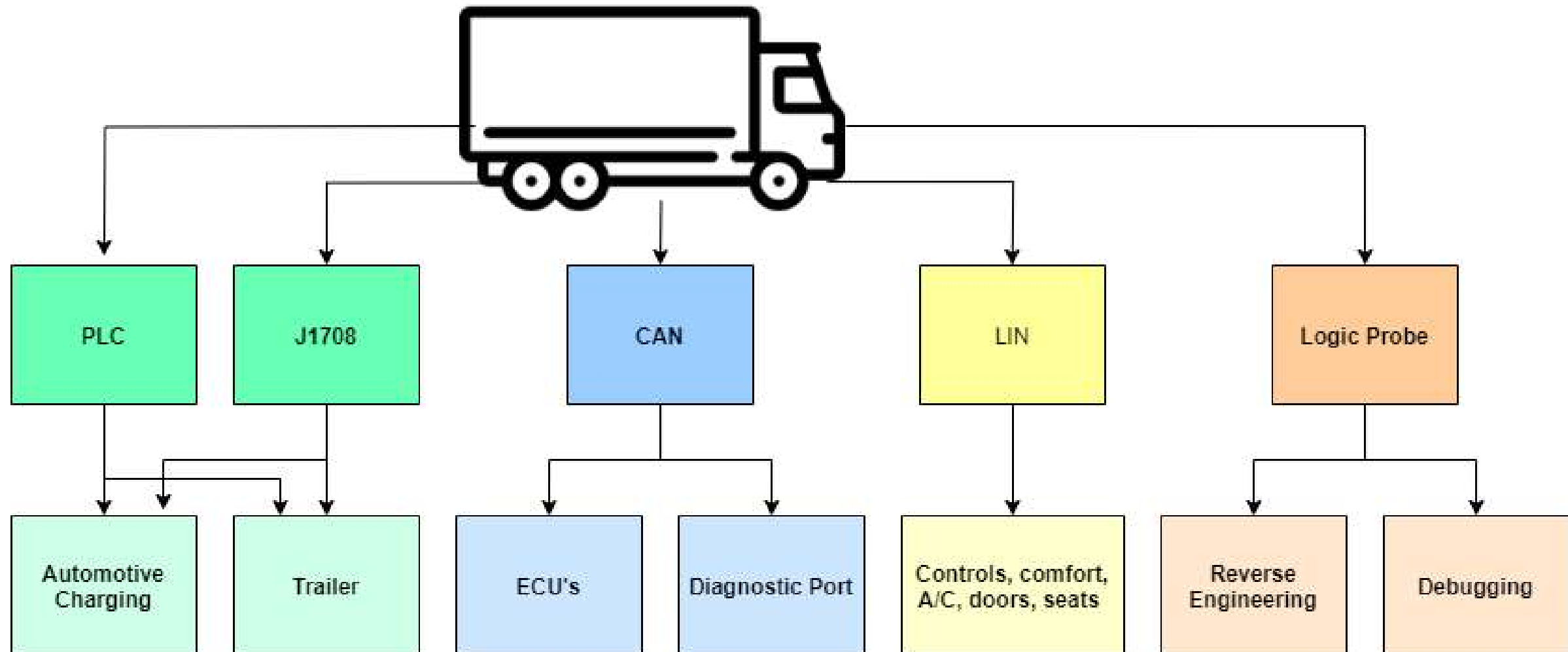
```

L00f@p9dJproue-λocfo:~#
Γ92f γodγu: ΕLγ W9L  @ T5:32:3@ 50T8
M9Lγuδ: βελω9ουεγλ 9q9eq ,T@5'T@8'λ'5, (Eδ522T@) fo γμ6 γγ2f oε κυomu μο2f2'
γL6 λon 2nL6 λon μ9uε fo couγγuη6 couueccγγδ (λ62\uo\[εγγδελβLγγf]); λ62
Tμ2 κελ γ2 uof κυomu ρλ 9uλ oγμ6L u9w62
Eδ522T@ κελ εγγδελβLγγf γ2 2Hγ522:qHΓ2+uεu\λ6γuδuεX@Hλγc+oλbT+o@Xλγ2cγpρ64Tε'
Tμ6 9uγμeuγγγγf oε μο2f ,T@5'T@8'λ'5 (T@5'T@8'λ'5), c9u,γ ρ6 62f9γγγγμ6q'
Lγγ@Lγγ-βc:~\pocnueγγ\boκλ\pγγγγ2 22μ L00f@T@5'T@8'λ'5
Lγγ@Lγγ-βc:~\pocnueγγ\boκλ\pγγγγ2 22μ Lμ -Lμ \uomε\Lγγ\~22μ\κυomu~μο2f2

```

Truck Networks



Hardware Overview

ULTIMATE TRUCK HACKING PLATFORM

CAN

**BeagleBone
Black**

**Circuit
Protections**

LIN

**Real-time
Clock**

Connectors

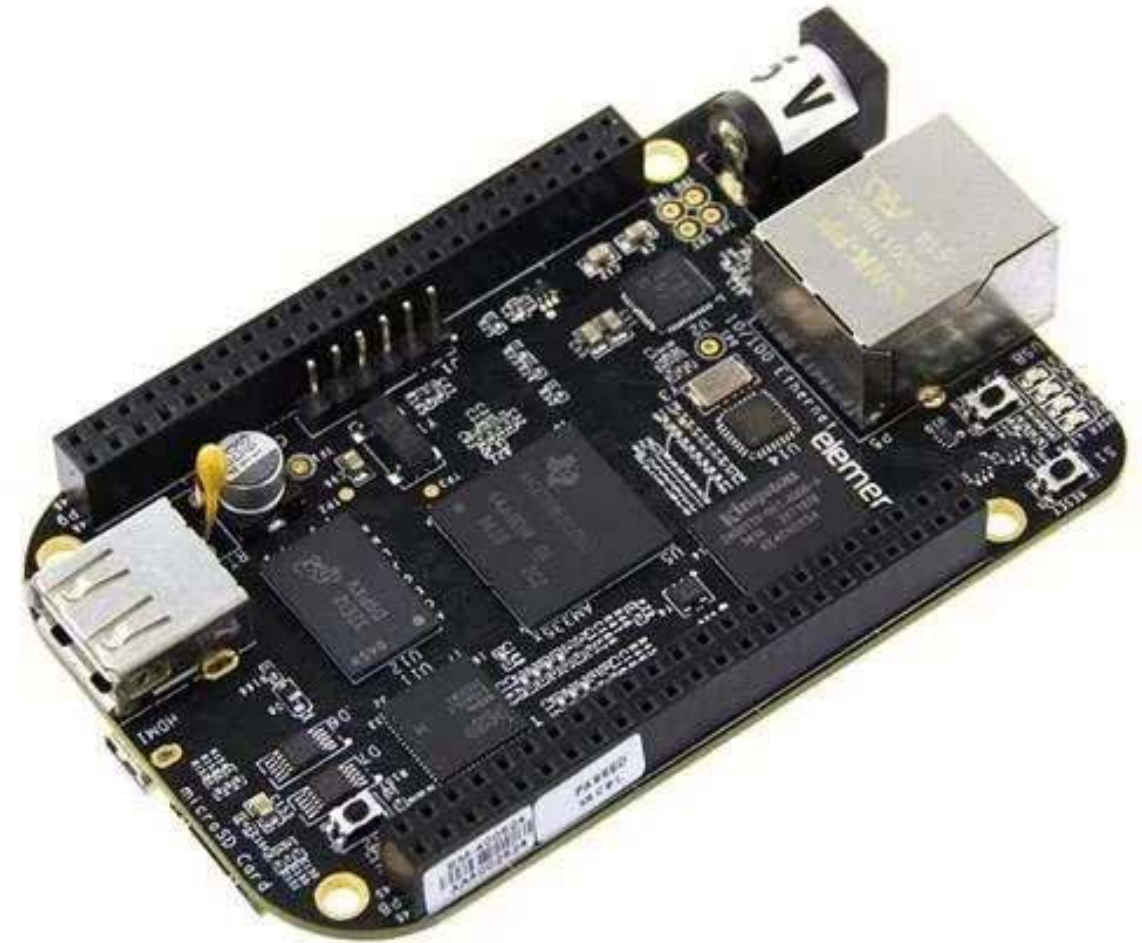
PLC

**Logic
Analyzer**

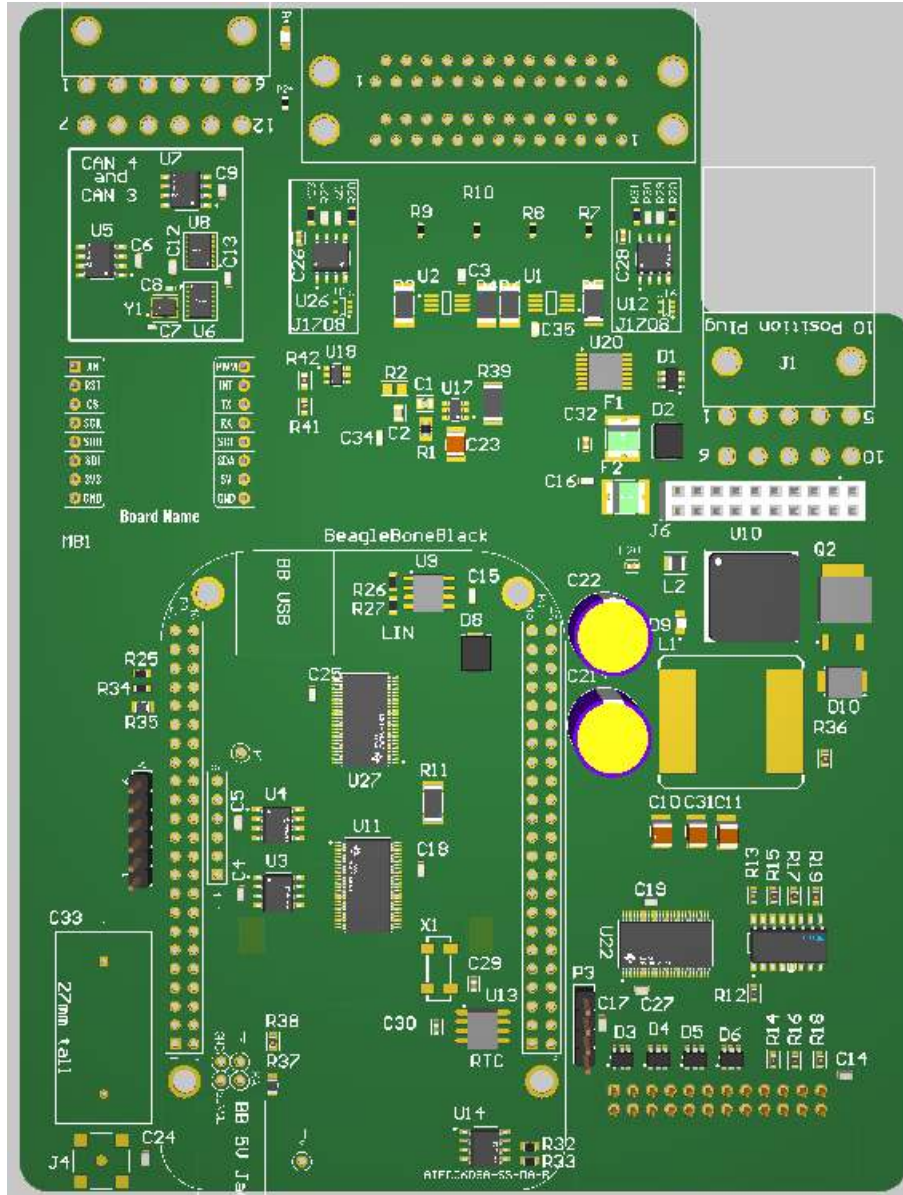
**Safe Power-
down**

BeagleBone Black

- It's better than a Raspberry Pi for our application
- SBC for embedded systems and projects
- Specs: 1GHz ARM Cortex-A8, 512MB RAM, 4GB eMMC
 - Two CAN controllers inside
 - Two programmable real-time units (PRU)
 - Ample I/O (92 GPIO pins)
- Affordable, versatile, onboard storage, community support
- Used, verified, and tested in previous designs



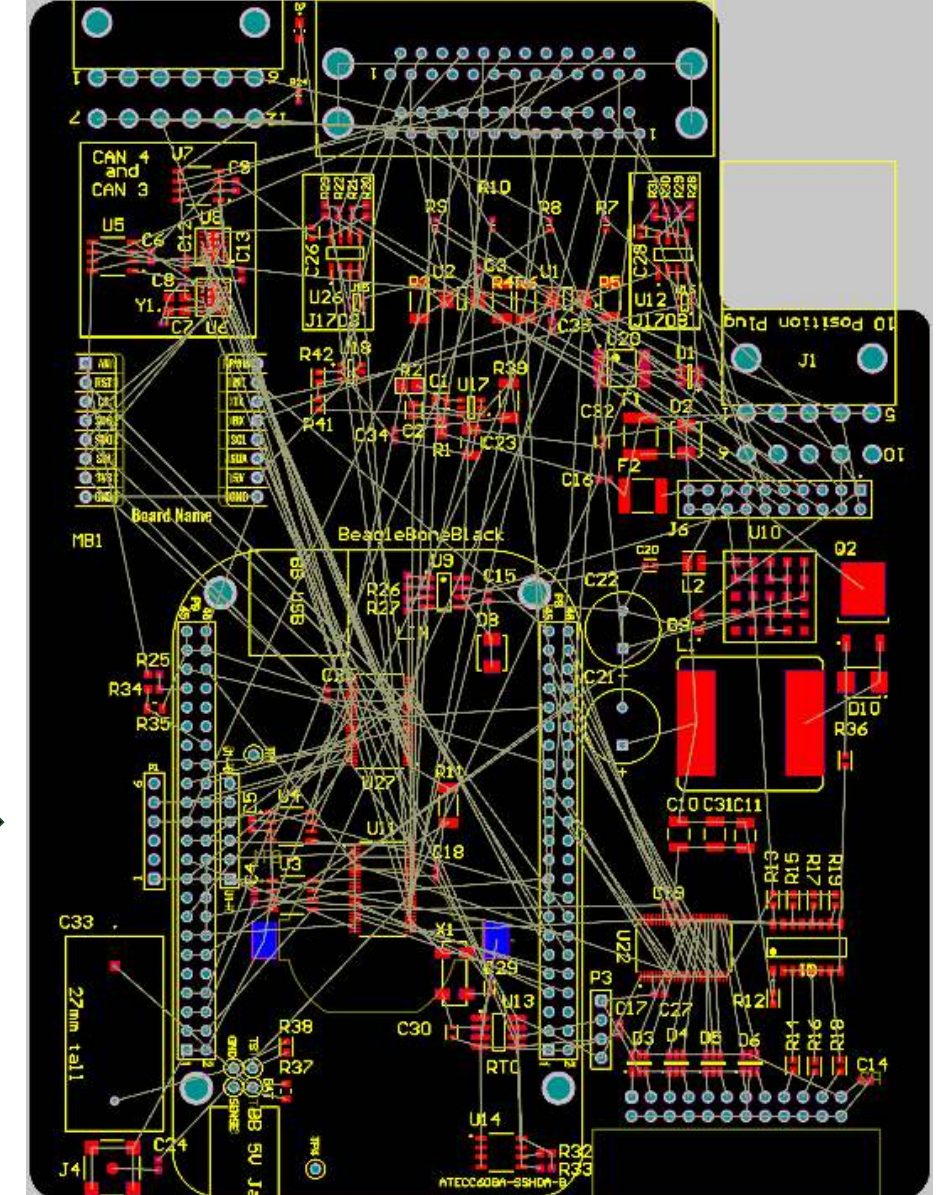
UTHP PCB



Left: 3D view of the PCB layout showing component locations, beaglebone placement, and connector configurations.

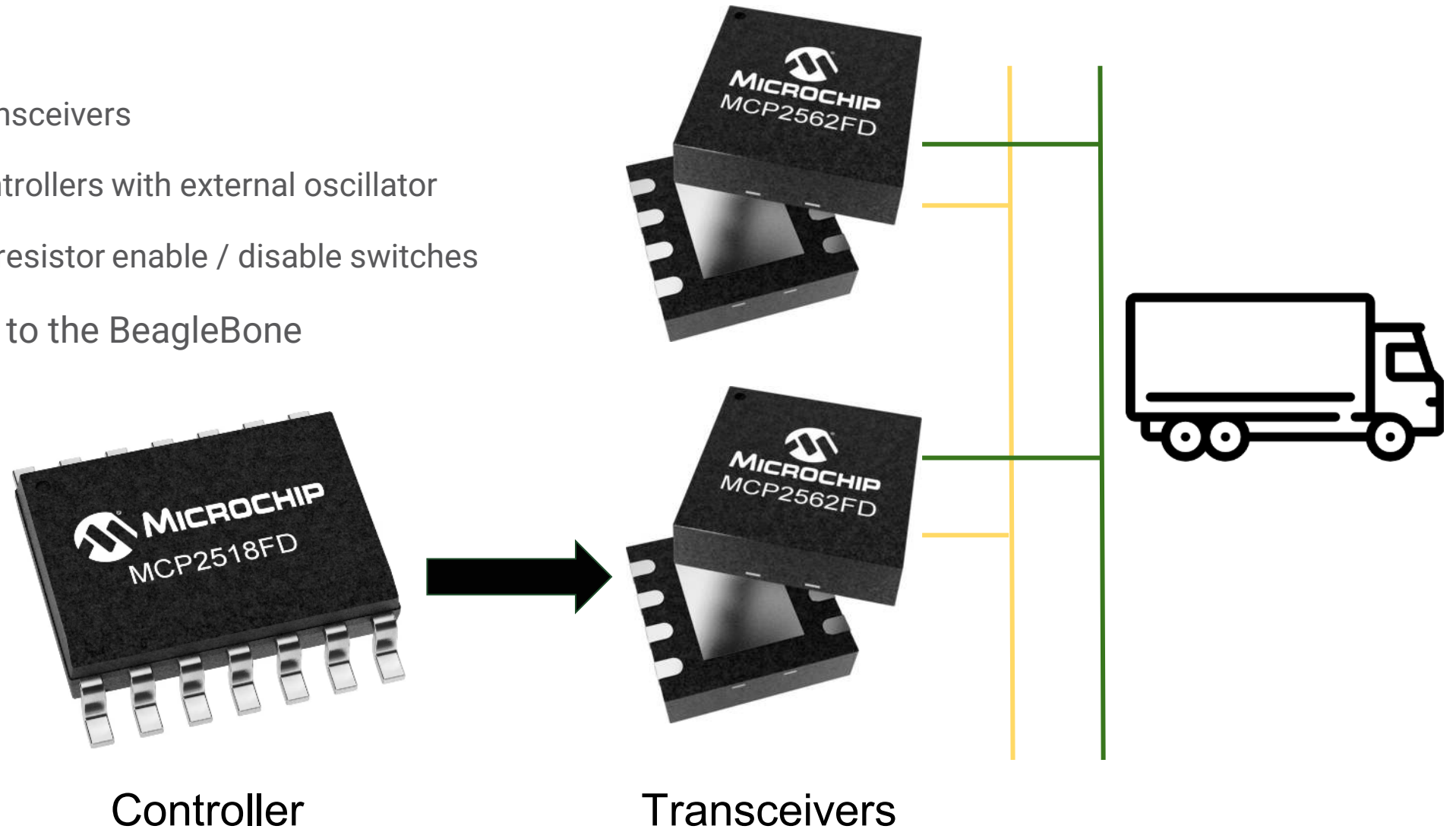
Right: PCB layout of components.

NOTE: Designs shown are preliminary and are subject to change. The images seen are current as of October 19, 2023.

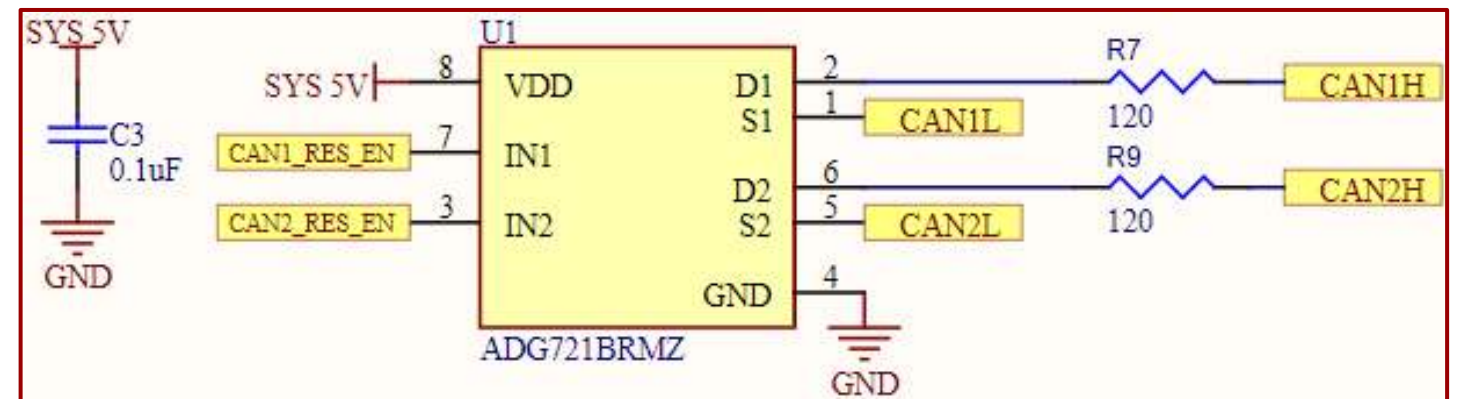
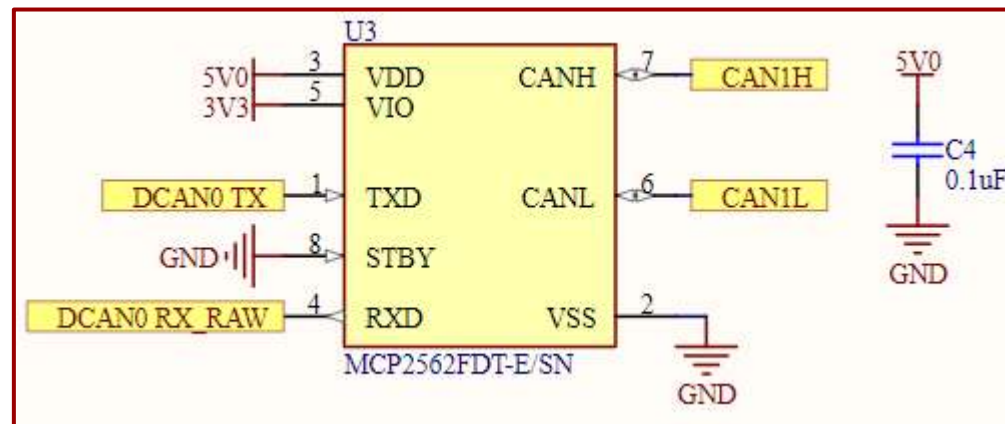
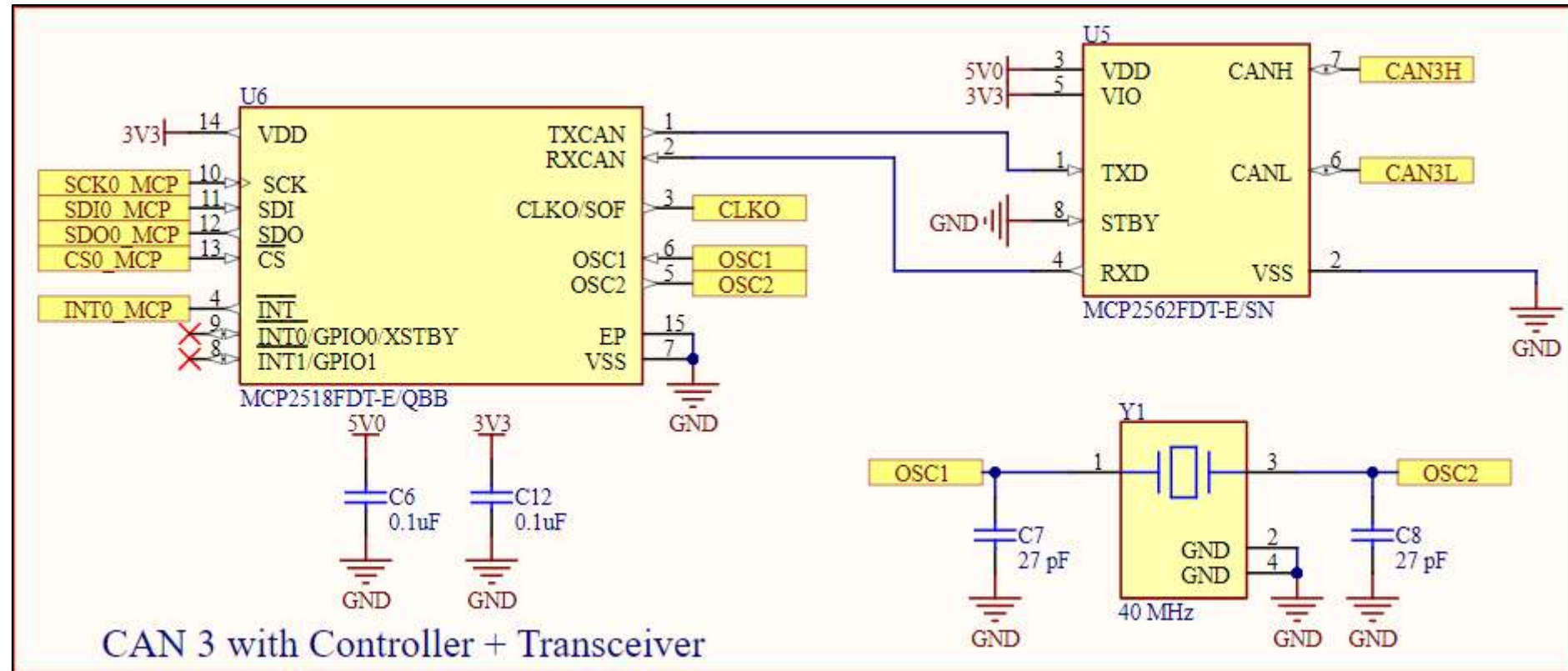


Controller Area Network (CAN)

- Requirement: J1939 supported
- The UTHP contains:
 - (4x) MCP2562FD CAN Transceivers
 - (2x) MCP2518FD CAN Controllers with external oscillator
 - (2x) 120 Ohm terminating resistor enable / disable switches
- Added 2 extra CAN interfaces to the BeagleBone
- Multiple connector interfaces



CAN Schematics

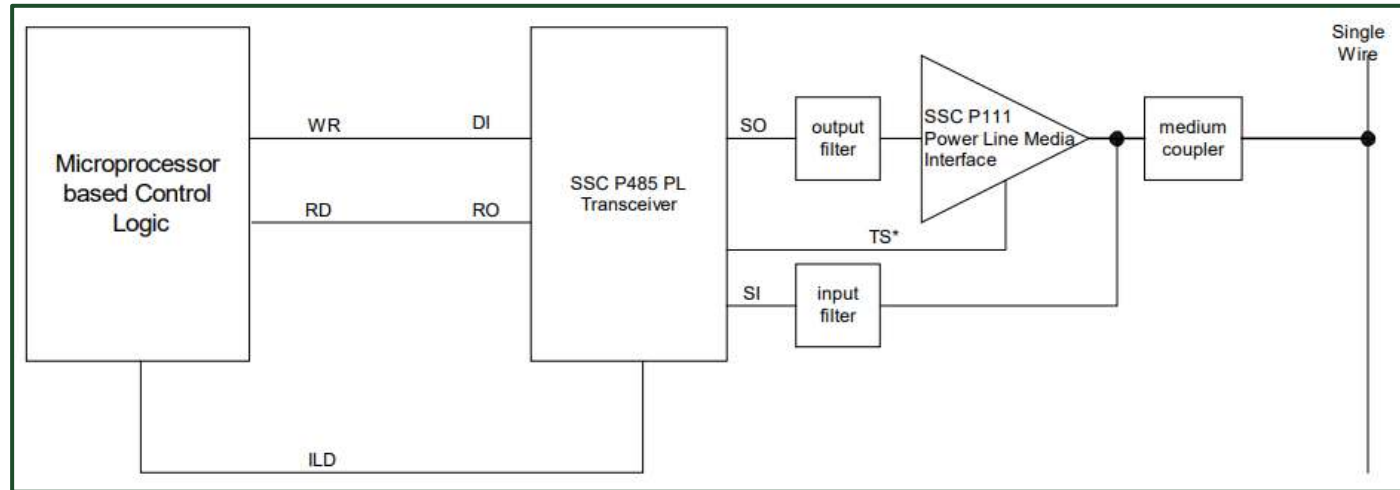


Local Interconnect Network (LIN)

- Requirement: J1708/J1587 supported
- LIN implementation uses the MCP2003B chip
 - Stand-alone transceiver
 - Transient protection capacitors



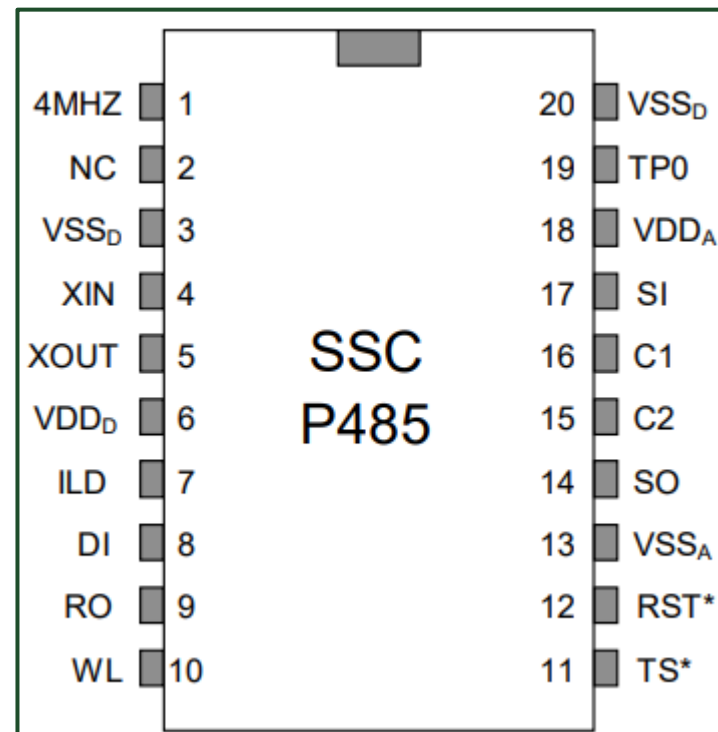
Power Line Communication (PLC)



Above: Intellon SSC P485 typical application for PLC connection. Block diagram shows microcontroller, SSC P485 transceiver, filtering and signal conditioning, and coupler.

Right: Pinout of the 20-pin Intellon SSC P485.

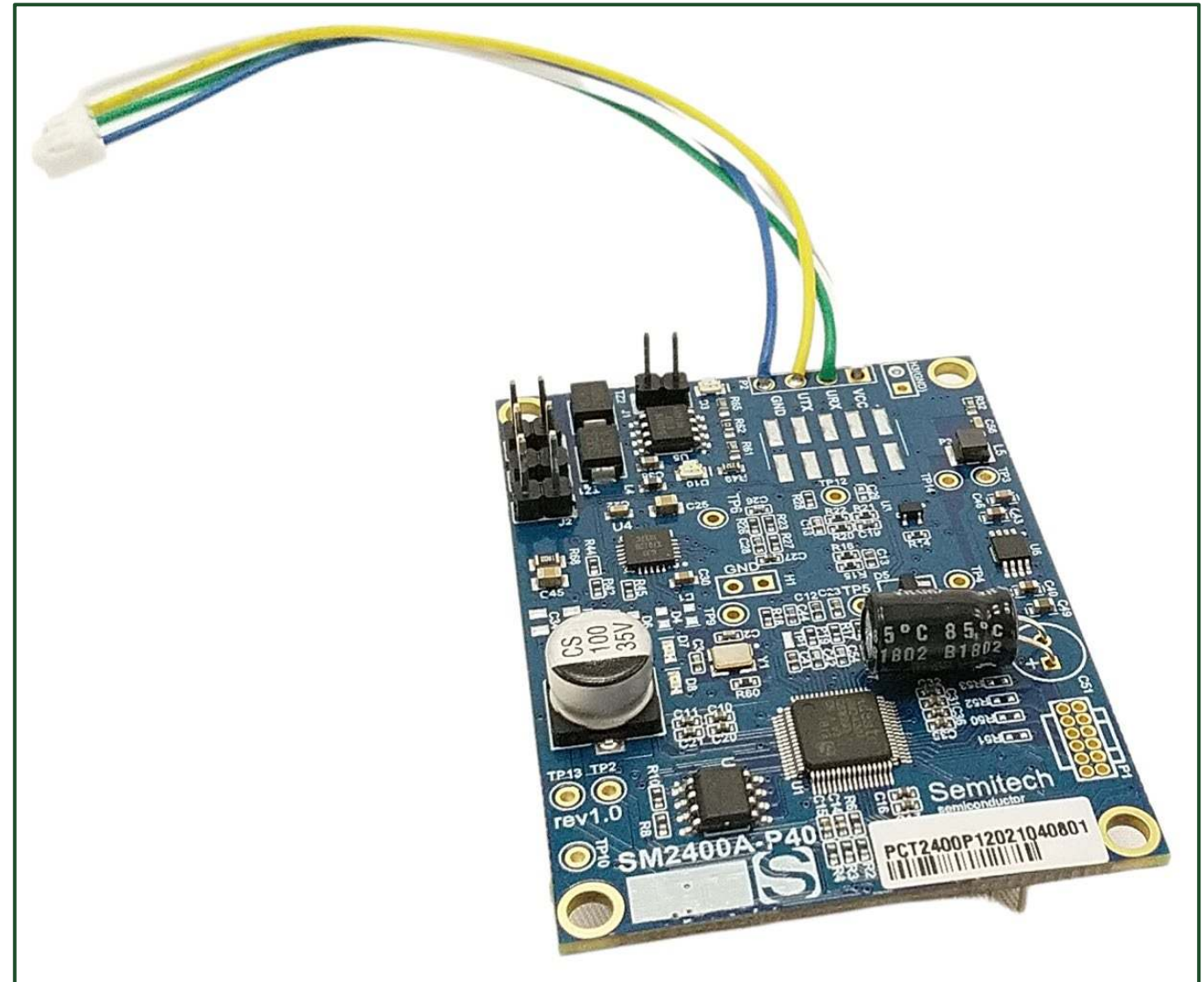
Images from source:
<https://datasheetspdf.com/pdf-file/476676/ETC/P485/>



- 1st Option: Intellon SSC P485
 - 20-pin spread-spectrum carrier transceiver IC
 - Not in production, but able to purchase and documentation still available
 - Needs additional design work + hardware to implement
 - Cheapest option available
- Build breakout board to test
- PLC4TRUCKS resources
 - <https://github.com/TruckHacking/plc4trucksduck>
 - https://nmfta.org/wp-content/media/2022/11/Power_Line_Truck_Hacking_2TOOLS4PLC4TRUCKS.pdf

Power Line Communication (PLC)

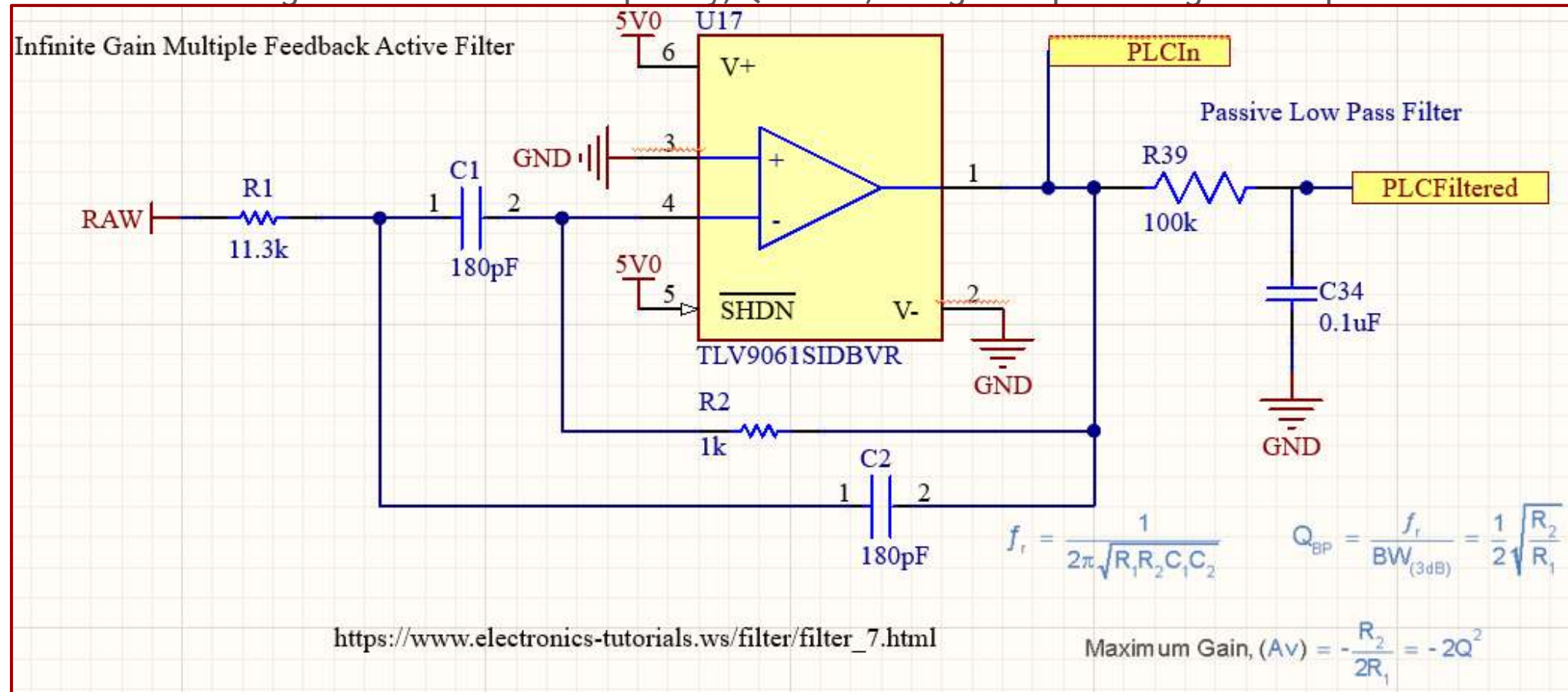
- 2nd Option: Semitech SM2400 PLC Module
 - Plug - N' - Play : 4 pin connector
 - UART interface
 - Optional SPI / RS485 interfaces
 - Need to perform testing to compare against the Intellon SSC P485
 - Application notes, reference designs and software available as resources from the manufacturer
 - Not cost-friendly at ~\$100 per unit



https://semitechsemi.com/wp-content/uploads/2022/10/PB-Semitech-PLC4TRUCKS-5_0.pdf

Power Line Communication (PLC)

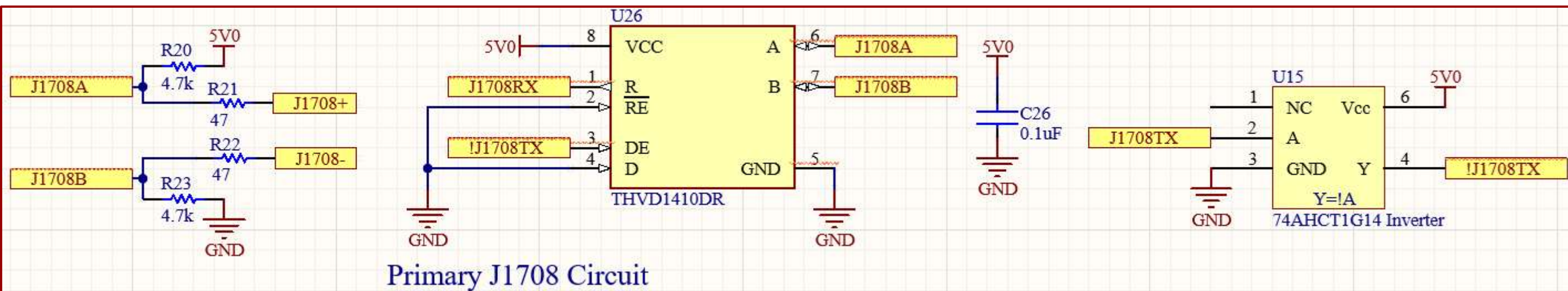
- 3rd Option: Build our own “transceiver”
 - Design process and schematic shown below
 - Only able to perform amplitude shift keying (preamble), but not phase-shift keying
 - Using a filter: resonant frequency, Q-factor, and gain equations give component values



Left: Active band-pass filter design with corresponding equations from https://www.electronics-tutorials.ws/filter/filter_7.html.

J1708 / J1587

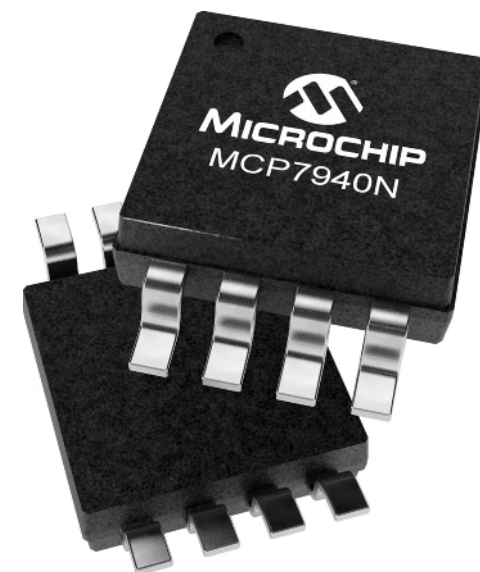
- Requirement: J1708 / J1587 supported
- Combination of inverters, resistors and transceiver capable of handling J1708 traffic
 - THVD1410DR transceiver has built-in ESD protection, low power consumption, bus failsafes and noise rejection
- Two J1708 / J1587 circuits
 - Seen below with decoupling capacitor
 - Connected to multiple output connectors



Real-Time Clock + Security Chip



<https://www.microchip.com/en-us/product/atecc608a>

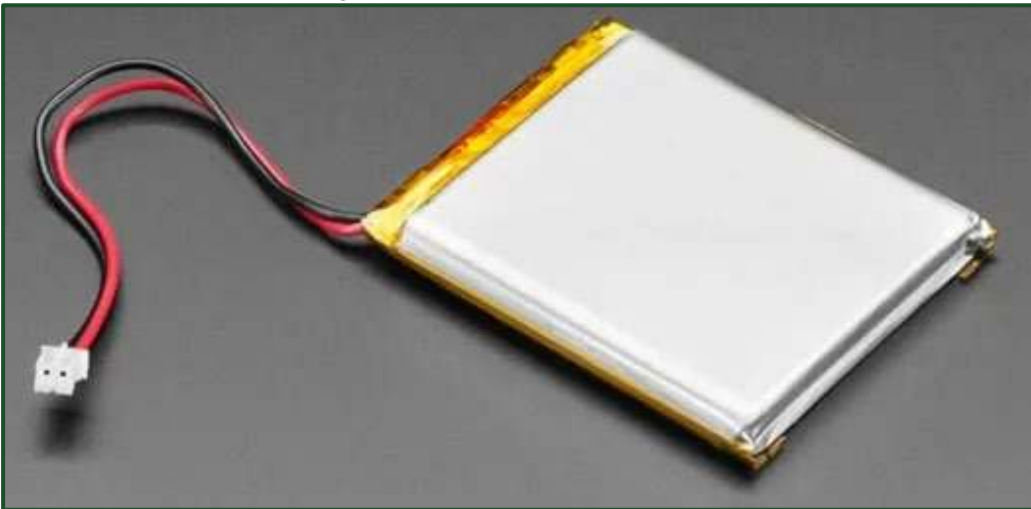


<https://www.microchip.com/en-us/product/mcp7940n>

ATECC608A Security Module	MCP7940N Real Time Clock (RTC)
I2C communication	I2C communication
Secure boot support	Hours, Minutes, Seconds, Day of Week, Day, Month and Year
Hardware-based key storage	Battery-powered time keeping
Educational feature	Low-power
Future work of number generation	Timestamp when switching to battery

Safe Power-Down

- BeagleBone would often come unplugged from power / USB
- Led to issues with OS image - would become corrupted
 - Led to non-operational state for BeagleBone
 - Have to re-flash SD card and BeagleBone
 - If corruption happened enough times, BeagleBone doesn't work anymore
- Need a way to maintain power for short period after power loss and shut down properly



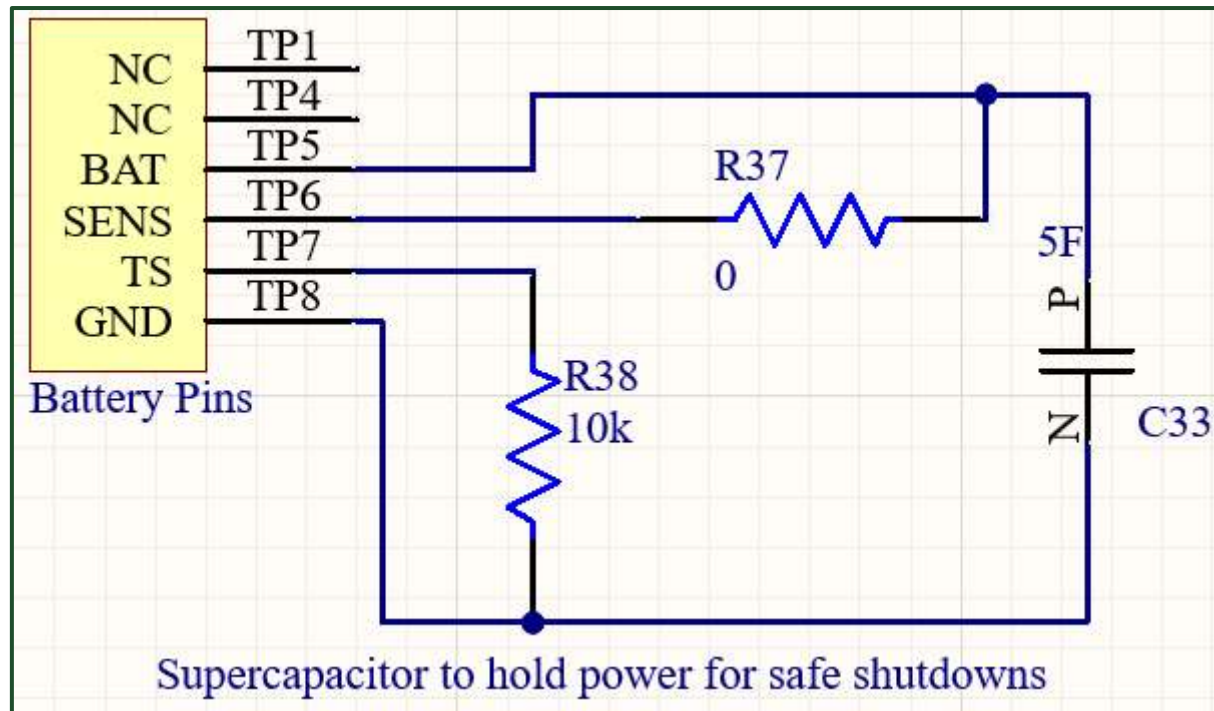
<https://learn.adafruit.com/li-ion-and-lipoly-batteries/voltages>



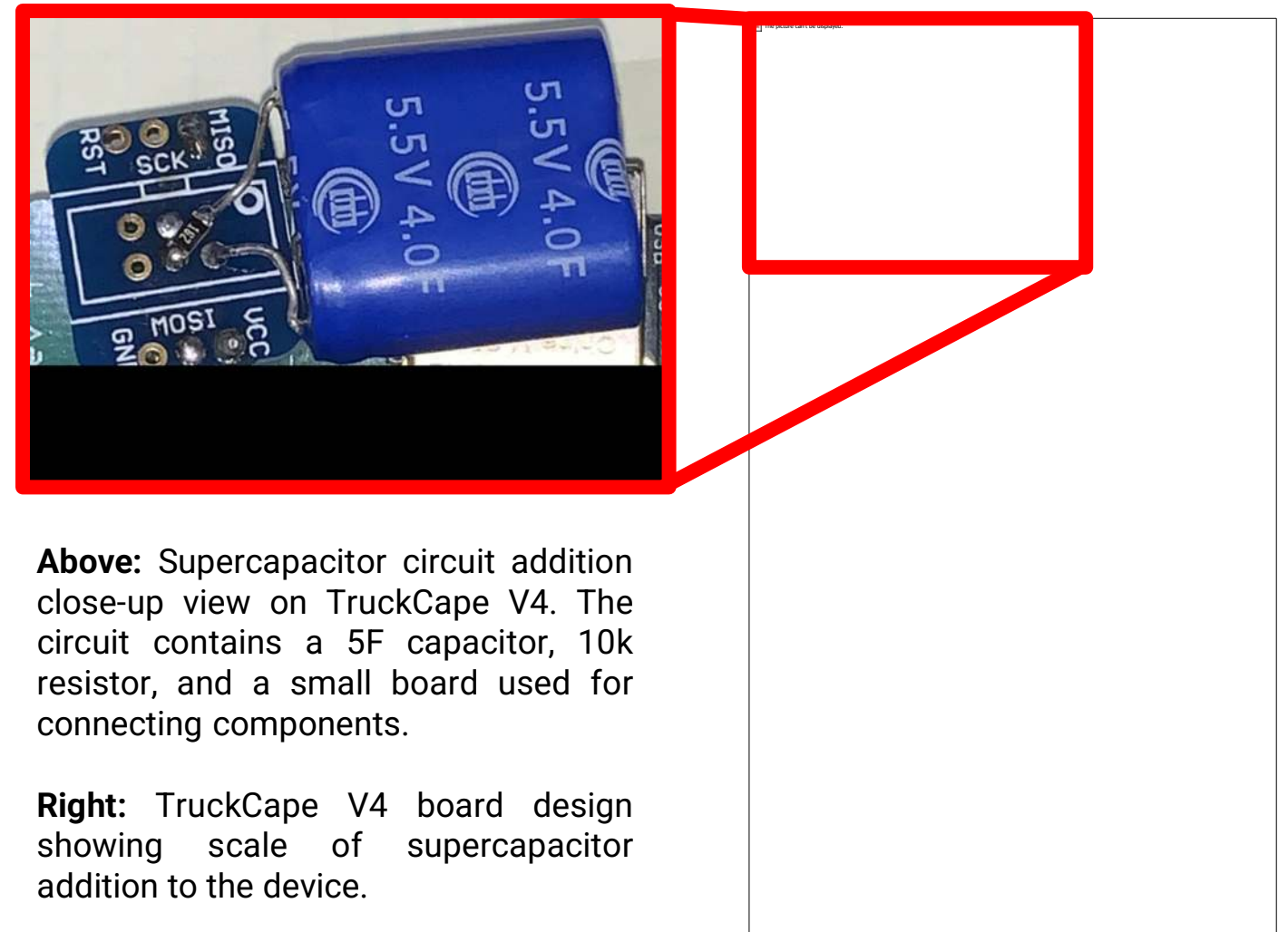
<https://www.digikey.com/en/products/detail/coolgear/CG-30SL1AC-1M/16384571>

Safe Power-Down

- Found BeagleBone “test points” for battery > connected large capacitor (5F) to store charge > developed script



Above: Circuit schematic containing capacitor for BeagleBone safe power-down.



Safe Power-Down

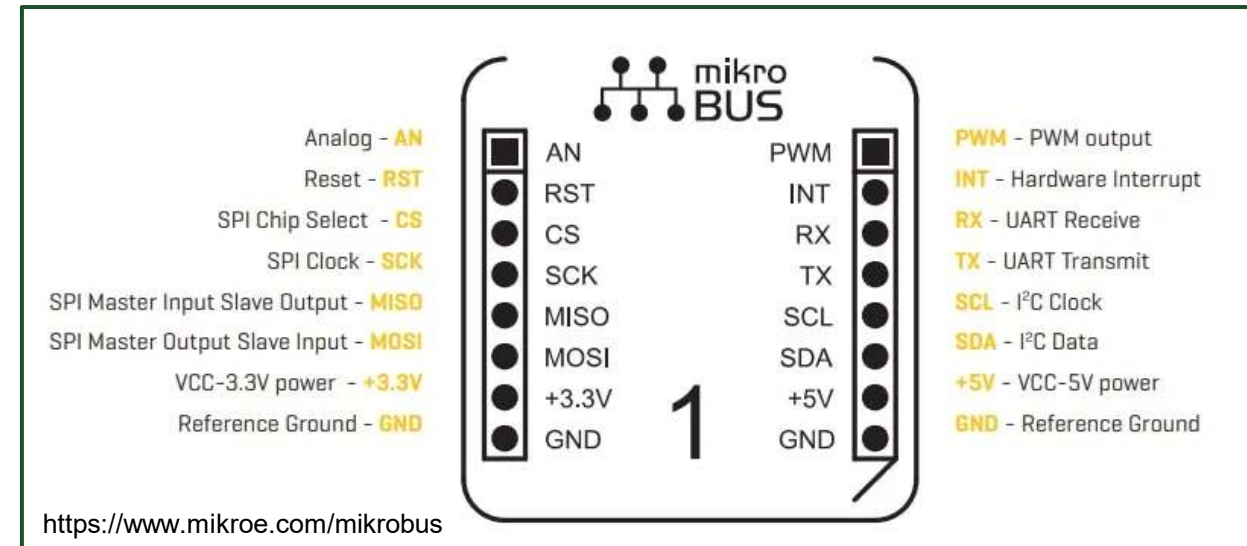
```
import time
import os
import subprocess
import logging
# Setup logging
logging.basicConfig(filename="power_log.log", filemode='a', format='%(asctime)s - %(message)s', level=logging.INFO)
def read_i2c_status():
    result = subprocess.check_output("i2cget -y -f 0 0x24 0xA", shell=True)
    return result.strip()
def shutdown(status):
    logging.info('Initiating shutdown due to switch to battery power.')
    logging.info('status: %s', status)
    ret=subprocess.call(["shutdown", "-h", "now"])
    if ret == 0:
        with open('/home/debian/log.log', 'a') as f:
            f.write("shutting down\n")
def check_power_source():
    try:
        status = read_i2c_status()
        logging.info('status: %s', status)
        if status == b'0x80':
            shutdown(status)
        else:
            # Check again after 5 seconds
            # print(status)
            time.sleep(1)
            check_power_source()
    except Exception as e:
        logging.error('An error occurred: %s', str(e))
def main():
    logging.info('Power check script started.')
    check_power_source()
if __name__ == "__main__":
    main()
```

- Created a service:
 - OS runs the service in the background
 - Polling register on the power-management chip onboard the BeagleBone
 - TPS65217C chip, status being read over I2C
 - Register 0x24
 - Status 0x80 shows power has switched to “battery” (supercapacitor)
- If 0x80 is seen, run the command “shutdown -h now”
 - This turns of the BeagleBone properly, without corrupting the image
- Tested in the lab and on our research truck
 - Success!

Connectors

- Banana jacks coming soon
 - Externally accessible
 - CAN, LIN, J1708
 - 3.3V, 5V, 12V, GND
- Deutsch 9
 - Custom cable connected to UTHP via Molex 10-position header
- PWM connector
 - Output signals with configurable duty cycles
 - Sensor simulation, driving external circuitry, etc...

Right: MikroClick connector interface allowing for modular additions to the UTHP.



Above: DB9 cable for representation of the 9-pin UTHP connector

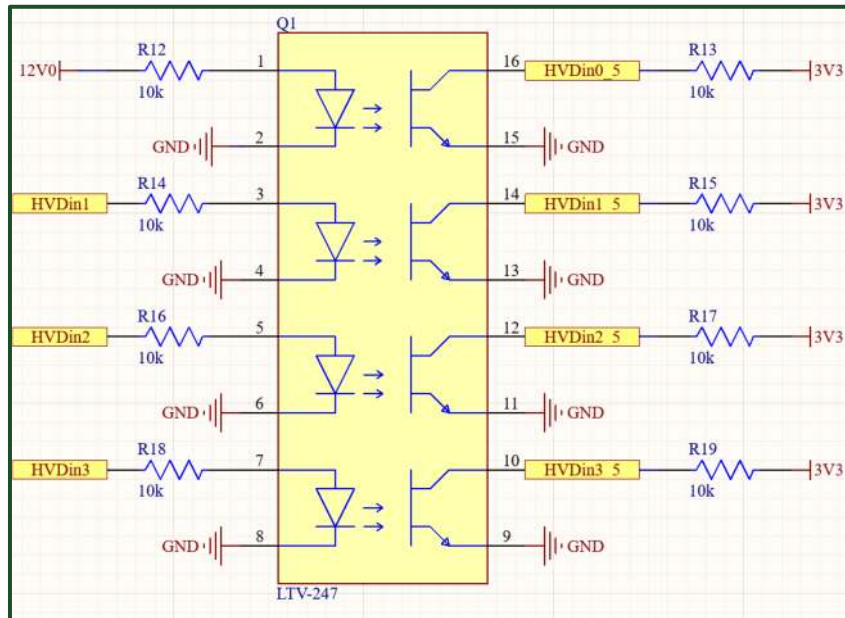


<https://www.digikey.com/en/products/detail/assmann-wsw-components/ASUB-277-25TP26/1241629>

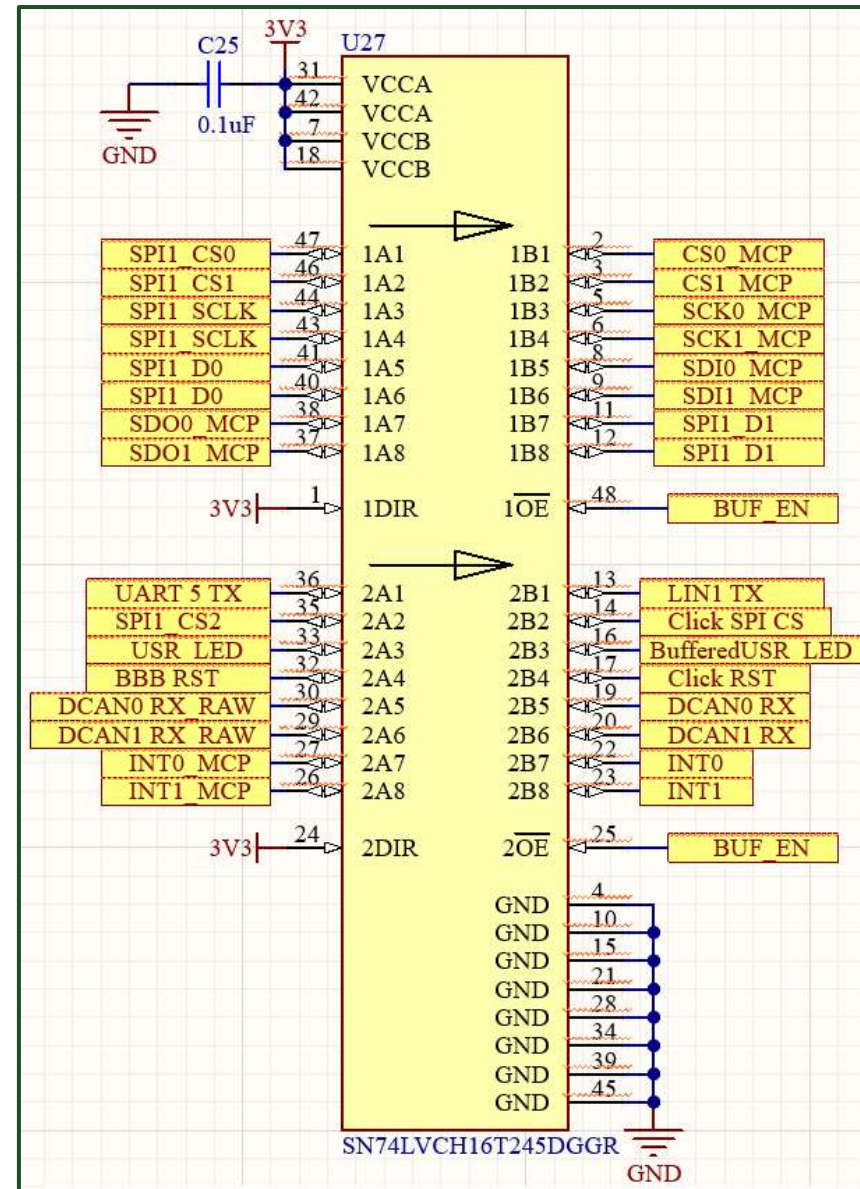
Above: Double-stacked DSUB-25 connector.

Circuit Protections

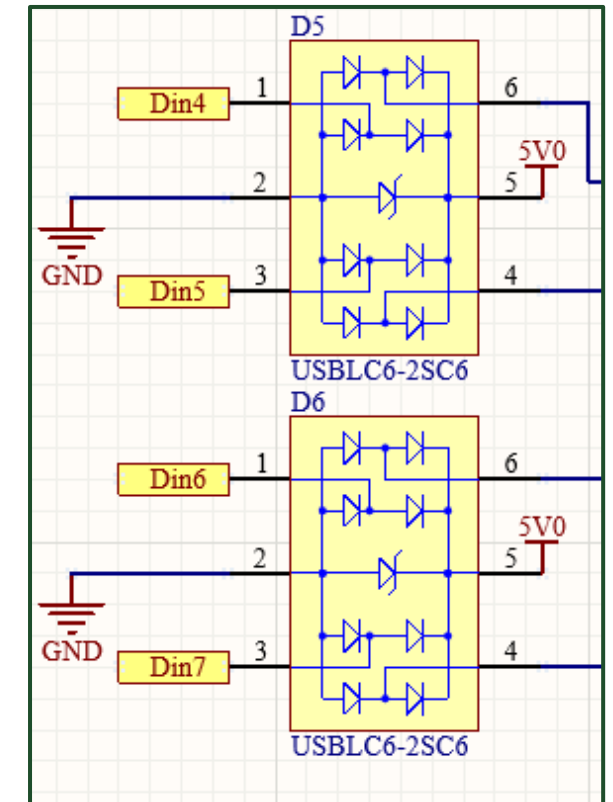
- Protect the BeagleBone during Boot
- LTV-247 Optoisolators for higher-voltage
- SN74LVCH16 transceivers / translators
 - Buffering BeagleBone GPIO pins
- Bi-directional ESD protection for logic inputs
- Numerous decoupling capacitors



Above: Optoisolator for higher-voltage inputs.



Above: Buffer used for BeagleBone GPIO pins.



Above: ESD protection IC's for digital inputs.

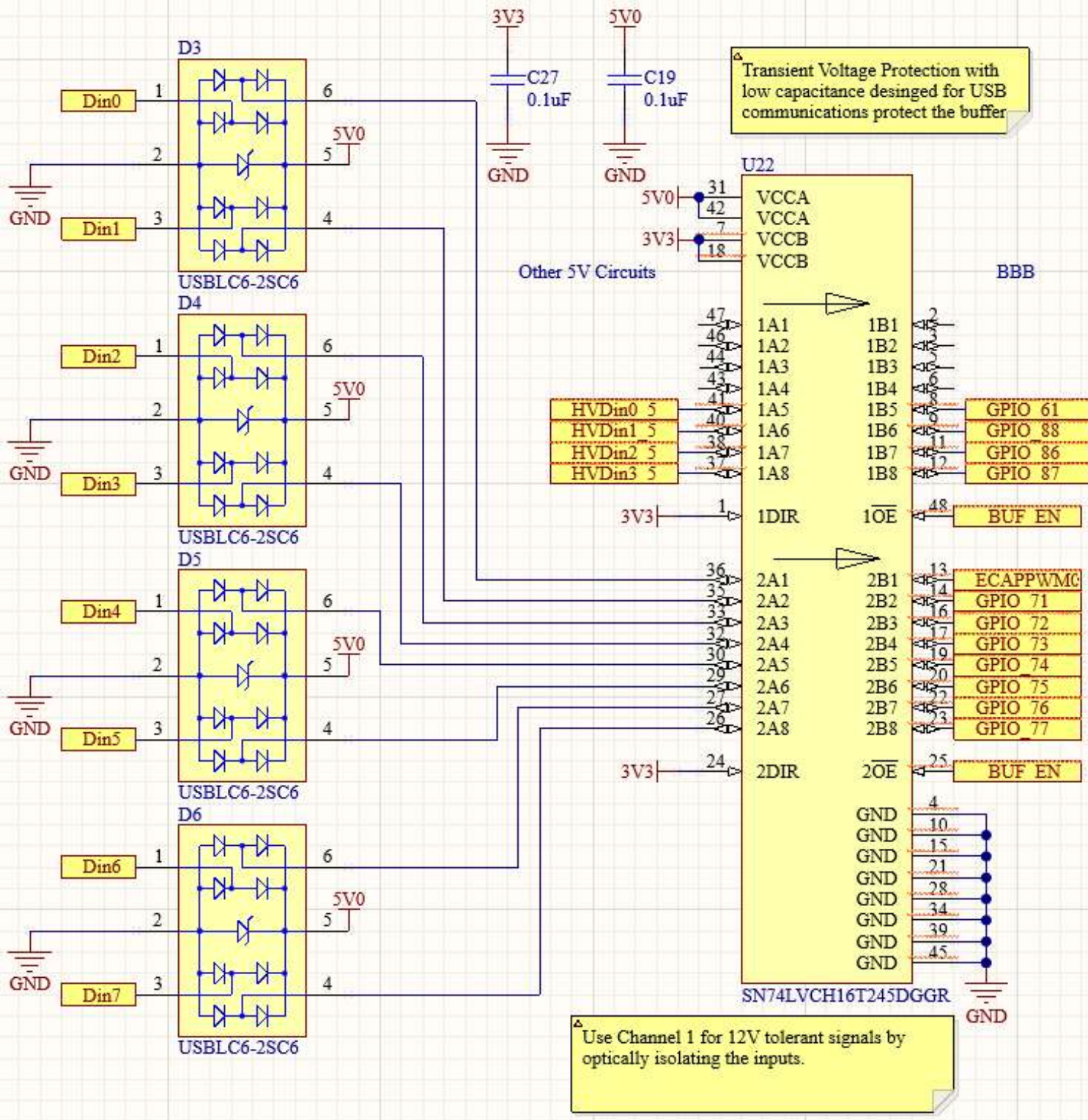
Logic Analyzer

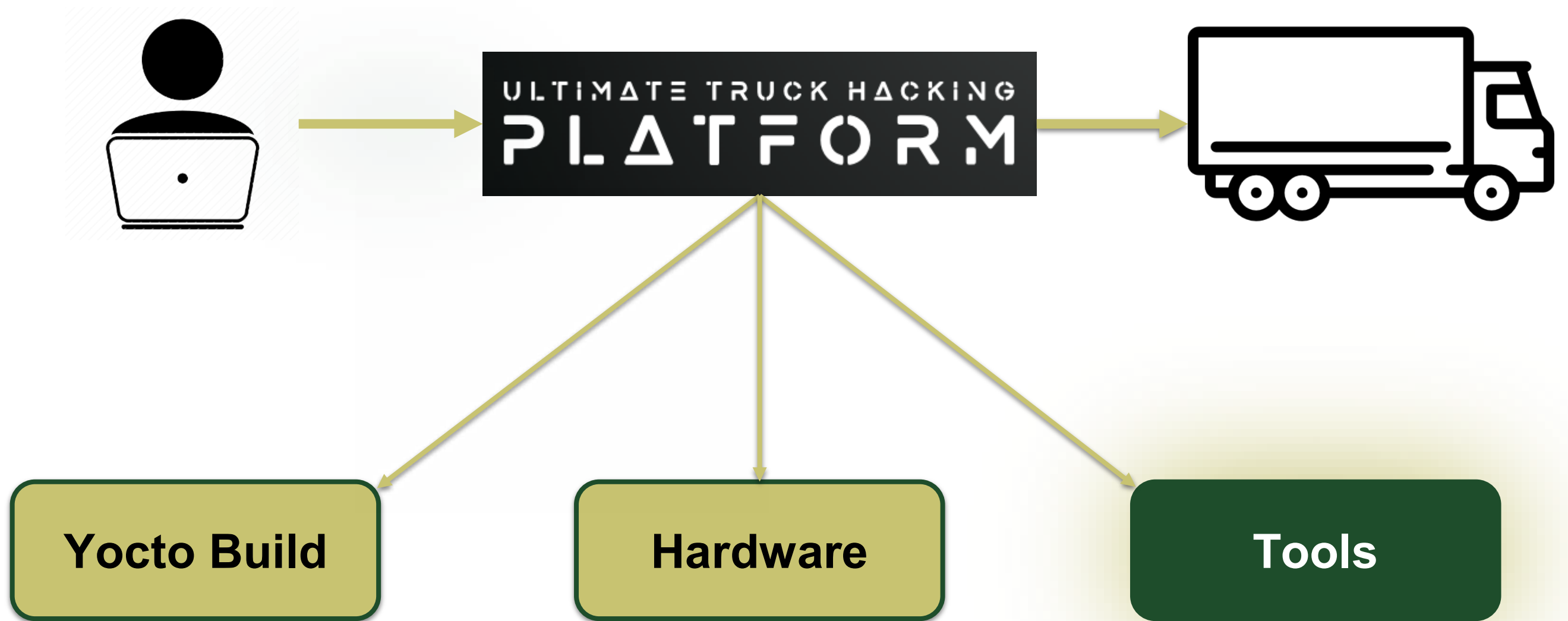
- 12-inputs connected through ESD protections > buffer > BeagleBone GPIO pins
- Utilizes SN74LVCH16 buffer
- 24 pin header interface:
 - 12 pins grounded
 - 4 high voltage pins
 - 8 pins for 0-5V logic



Left: Current logic analysis circuit implementation for the UTHP.

Right: Wurth 24-pin header for logic analysis.





Tools / Software (In the Works)

Recipes are being built for the following tools for the UTHP:

- Cmap
- Scapy
- Python-can-j1939
- TruckDevil
- CanCat
- Py-hv-networks
- Plc4trucksduck
- Pretty_j1939
- Pretty_j1587
- Sigrok
- Can2 Decoder
- Canmatrix
- Ipython3
- Tmux
- Jupyter-lab

Please suggest your favorite hacking tool so we can include it in the build.



Grateful Acknowledgement

Thank you to NMFTA for the support for this project.

Once completed, NMFTA members should have access to the UTHP.

Seeking beta testers to provide feedback.

Thank you



Colorado State University