Experience teaching seL4
seL4 Summit 2022

Sebastian Eckl, HENSOLDT Cyber GmbH

October 12, 2022

HENSOLDT
Detect and Protect.

Agenda

- Course Overview
- Theoretical Phase (lecture units)
- TRENTOS
- Practical Phase (coding units)
  - Homework Assignments
  - Team Projects
- Results
- Future Work

Course Overview

- Since winter term 2020 @TUM
- Participants: max. 15 students per term
- Group work: team size of max. 3 students
- Focus: L4 microkernel development in the area of embedded systems
  - Development of embedded systems
  - Better understanding of components utilized in modern (microkernel) OS
  - OS development based on seL4, CAmkES and TRENTOS
- Structure: from theory to practice
  - Theoretical Part: several lecture units, concerning hardware/bootloader, seL4, CAmkES and TRENTOS
  - Practical Part
    - Hands-on training: homework assignments
    - Team projects focusing on real-world application

Theoretical Phase

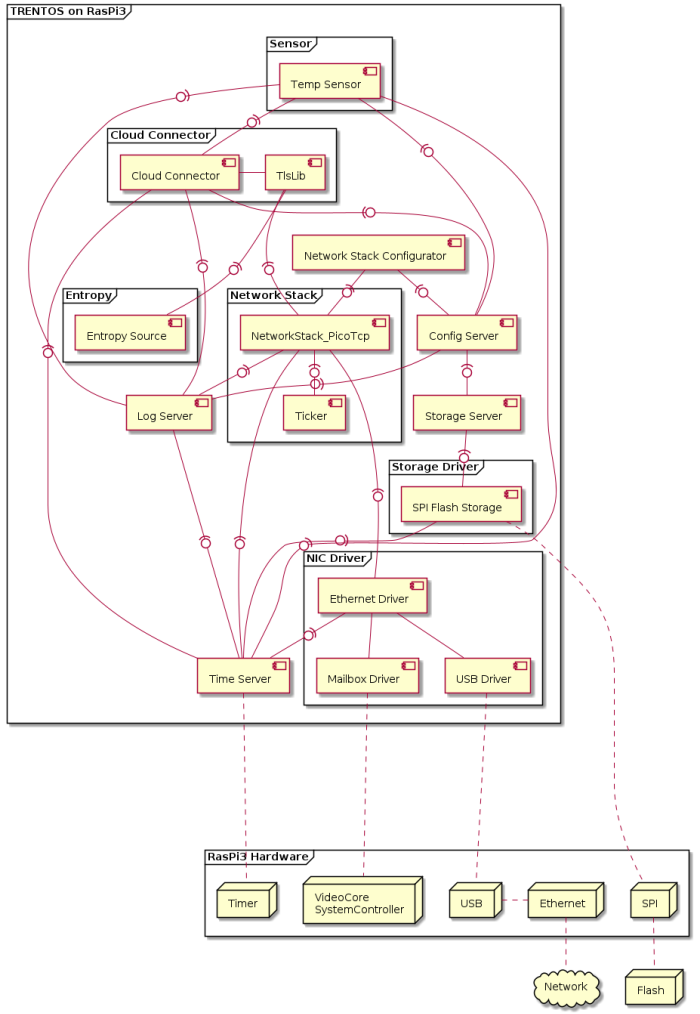
- Embedded Hardware
  - ARM/RISC-V
  - SoC, SBC
  - bootloader
- Kernel: seL4
  - microkernel vs. monolithic kernel
  - L4 history
  - seL4 building blocks (kernel objects, capabilities)
  - formal verification
- CAmkES
  - componentised systems
  - building blocks

Theoretical Phase ctd.

- TRENTOS
  - architecture
  - SDK structure
  - build and test environment
  - building blocks
  - how to write your own TRENTOS application
- TRENTOS Development Kit
  - TRENTOS SDK
  - Raspberry Pi 3 B+ based hardware environment
  - selected peripherals and cabling

TRENTOS – Overview

- A novel secure embedded operating system developed by HENSOLDT Cyber
- Built on top of the seL4 microkernel and the CAmkES framework
- Focused on creating static systems
  - No dynamic reconfiguration (CAmkES)
  - Highly dynamic systems only via virtualization
  - No focus on GUI frameworks
- Ready-to-use components (drivers, middleware, …)
- Standardized, platform-agnostic interfaces
- Support for both ARM and RISC-V
- Suited for formal verification (CAmkES toolchain)

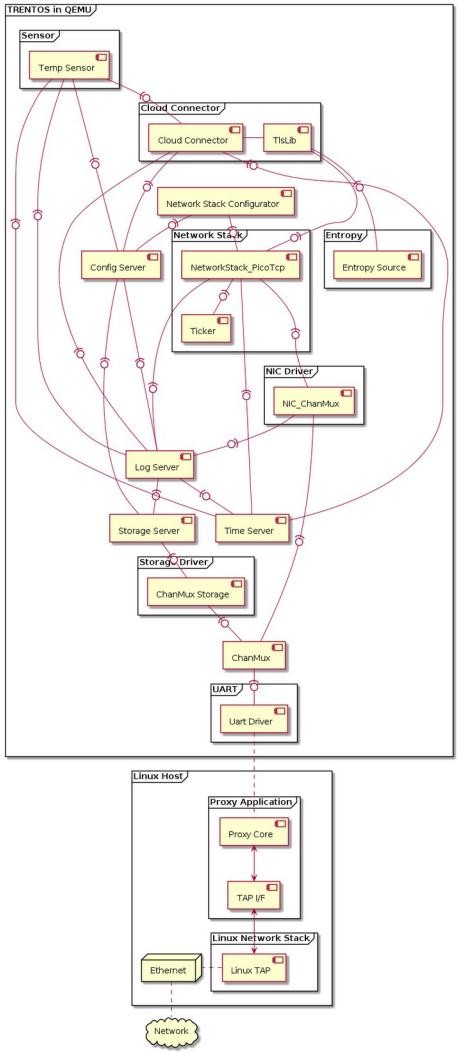


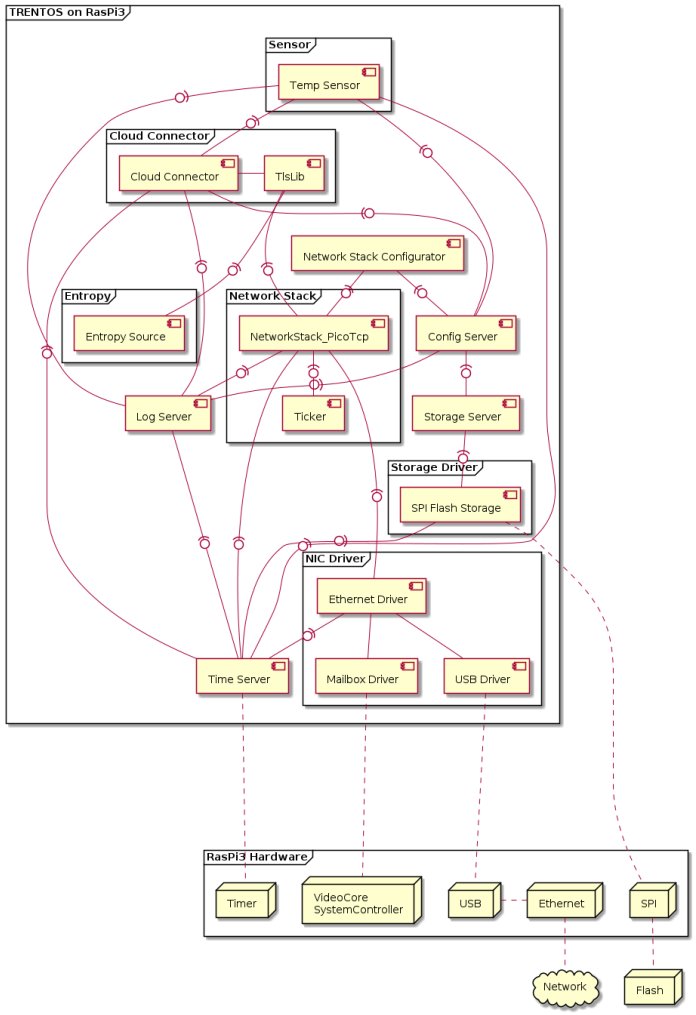
HENSOLDT
Detect and Protect.

TRENTOS – SDK Structure

- `GettingStarted.pdf` → Link to `sdk/doc/pdf/TRENTOS_GettingStarted_SDK_V1.3.pdf`
- `docker` → Containers with toolchain to build and test TRENTOS systems
  - `trentos_1.3.sha256sums`
  - `trentos_build_20210503.bz2`
  - `trentos_test_20211025.bz2`
- `sdk`
  - `build-system.sh` → Master build script for building TRENTOS systems
  - `CMakeLists.txt` → Master CMake script for building TRENTOS systems
  - `README.license` → License information
  - `bin` → TRENTOS tools binaries for Linux
    - `cpt` → Configuration Provisioning Tool
    - `proxy_app` → Proxy Application
    - `rdgen` → RamDisk Generator
  - `components` → TRENTOS components sources
    - `CertServer` → Component to offer "read access" to the set of trusted certificates (the "trusted chain")
    - `ChanMux` → Channel Multiplexer component for communication with the Proxy Application
    - `CryptoServer` → CryptoServer+Keystore component
    - `EntropySource` → Dummy TRNG driver
    - `NetworkStack_PicoTcp` → Network stack component based on picoTCP
    - `NIC_ChanMux` → Network driver for a ChanMux NIC channel
    - `NIC_Dummy` → Dummy network driver
    - `NIC_iMX6` → Network driver for the iMX6 platform
    - `NIC_RPi` → Network driver for RPi3B+
    - `RamDisk` → RAM-based storage driver
    - `RPi_SPI_Flash` → Storage driver for SPI-based NOR flash memory for RPi3B+
    - `SdHostController` → Driver that implements the storage interface and allows accessing the SD card peripheral
    - `Storage_ChanMux` → Storage driver for a ChanMux storage channel
    - `StorageServer` → Providing access to storage for various clients
    - `TimeServer` → Providing access to timers for various clients
    - `TlsServer` → TLS client + networking component
    - `UART` → UART driver
  - `demos`
    - `demo_hello_world` → Hello World demo
    - `demo_iot_app` → IoT Demo (QEMU)
    - `demo_iot_app_imx6` → IoT Demo (Nitrogen6_SoloX, BD-SL-i.MX6)
    - `demo_iot_app_rpi3` → IoT Demo (RPi3B+)
    - `demo_network_filter` → Network Filter Demo (QEMU, Nitrogen6_SoloX)
    - `demo_tls_api` → Demo retrieving an HTTPS web page
- `doc`
  - `html` →TRENTOS API Doxygen documentation
  - `pdf` → TRENTOS documentation
    - `TRENTOS_GettingStarted_SDK_V1.3.pdf`
    - `TRENTOS_Handbook_SDK_V1.3.pdf`
    - `TRENTOS_ReleaseNotes_SDK_V1.3.pdf`
    - `TRENTOS_MigrationNotes_SDK_V1.2_to_V1.3.pdf`
    - `3rd_party` → Supplemental documents
- `libs` → TRENTOS libraries and core system sources
  - `3rdParty` → 3rd party libraries used by TRENTOS libraries internally
  - `chanmux` → ChanMux
  - `chanmux_nic_driver` → Network driver based on ChanMux
  - `lib_compiler` → Compiler abstraction utilities
  - `lib_debug` → Debugging facility macros and functions
  - `lib_host` → Host side utilities
  - `lib_io` → I/O operations
  - `lib_logs` → Logging utilities
  - `lib_macros` → Commonly used macros
  - `lib_mem` → Memory management utilities
  - `lib_osal` → Operating System abstraction library
  - `lib_server` → Server components utilities
  - `lib_utils` → Common useful utilities
  - `os_cert` → Certification parser library for x509 certificates
  - `os_configuration` → Configuration server library
  - `os_crypto` → Crypto library
  - `os_filesystem` → File system library (FAT, SPIFFS, LittleFS)
  - `os_keystore` → Keystore library (file based, RAM based)
  - `os_logger` → Log server core library
  - `os_socket` → Socket library
  - `os_tls` → TLS client library
- `os_core_api` → TRENTOS APIs for all subsystems
- `resources`
  - `nitrogen6sx_sd_card` → SD card files with Nitrogen6_SoloX bootloader
  - `openocd_cfgs` → Configuration files for debugging with OpenOCD
  - `rpi3_sd_card` → SD card files with RPi3B+ bootloader
  - `sabre_sd_card` → SD card files with BD-SL-i.MX6 bootloader
- `scripts` → Utility scripts
- `sdk-sel4-camkes` → seL4 and CAmkES
- `tools` → TRENTOS tools source code
  - `cpt` → Configuration Provisioning Tool
  - `proxy` → Proxy Application
  - `rdgen` → RamDisk Generator
  - `rpi3_flasher` → TRENTOS helper system for flashing an image on the RPi3B+ SPI-based NOR flash memory

TRENTOS – Componentised Setup





HENSOLDT
Detect and Protect.

TRENTOS – How to use

```
StorageServer_DeclareCAmkESComponent(
    StorageServer
)

DeclareCAmkESComponent(
    TestApp
    SOURCES
        components/TestApp/src/TestApp.c
    C_FLAGS
        -Wall
        -Werror
    LIBS
        system_config
        lib_debug
        os_core_api
        os_filesystem
)
```

```
import <std_connector.camkes>;

#include "system_config.h"

//------------------------------------------------------------------------
// Storage
//------------------------------------------------------------------------

#include "RamDisk/RamDisk.camkes"
RamDisk_COMPONENT_DEFINE(RamDisk)

#include "StorageServer/camkes/StorageServer.camkes"
StorageServer_COMPONENT_DEFINE(StorageServer)

import "components/TestApp/TestApp.camkes";

//------------------------------------------------------------------------
// System
//------------------------------------------------------------------------

assembly {
    composition {
        //------------------------------------------------------------------------
        // Storage
        //------------------------------------------------------------------------
        component   RamDisk             ramDisk;

        //------------------------------------------------------------------------
        // StorageServer
        //------------------------------------------------------------------------
        component   StorageServer       storageServer;

        StorageServer_INSTANCE_CONNECT(
            storageServer,
            ramDisk.storage_rpc, ramDisk.storage_port
        )

        StorageServer_INSTANCE_CONNECT_CLIENTS(
            storageServer,
            testApp.storage_rpc_1, testApp.storage_dp_1,
            testApp.storage_rpc_2, testApp.storage_dp_2
        )
```

TRENTOS – Create your own application

- Each application consists of
  - Build/test specific parts (*CMakeLists.txt*, *run_demo.sh*)
  - A central CAmkES application description (main *.camkes* file), which is used for declaration and instantiation of all required components and connections
    - re-using TRENTOS standard components (by utilizing specific macros)
    - creating own components (e.g. making use of the TRENTOS API)
  - A folder „components“ which contains a folder for each own component offering
    - a *.camkes* component description + optional interface definitions
    - own logic in C code + usage of TRENTOS API
  - Optional: a global configuration (*system_config.h*)

TRENTOS – Evaluation Kit

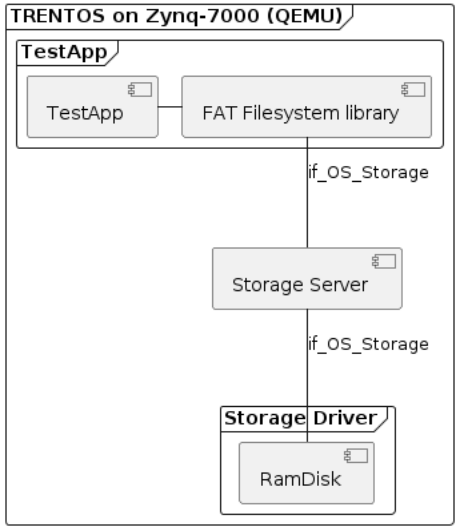
Practical Phase

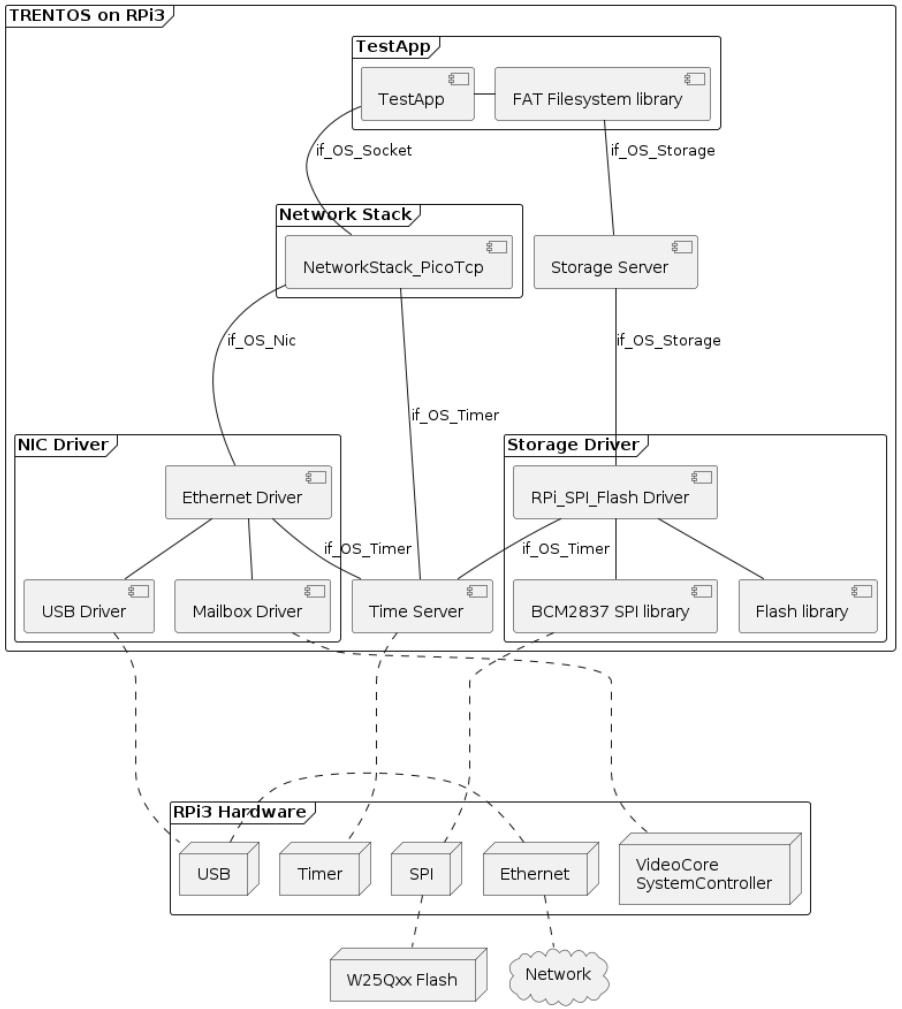
- Homework Assignments (each participant)
  - Based on the OS tutorials presented/explained in the course
  - Small project, iteratively developing (e.g. starting with a simple storage application, which gets extended via networking capabilities)
- Team Projects (exemplarily from summer term 2022)
  - Simulator Case
    - Porting of a driver for a specific SPI/I2C-based peripheral
    - Development of a demo application for a simulator based automotive/robotics/drone setup
  - Security Case
    - Porting of a driver for a SPI-based TPM peripheral
    - Development of a demo application for a crypto/key store related use case, e.g. an encrypted file system

Practical Phase – Homework Assignments

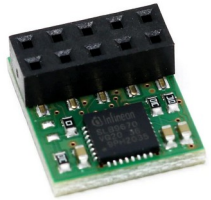
- Base Setup: QEMU-based setup, utilizing TRENTOS storage facilities (RamDisk) in combination with the TRENTOS FileSystem API on two partitions
- Task 1: Switch of file system type (from FAT to SPIFFS)
- Task 2: Switch from QEMU to Raspberry Pi 3 B+
- Task 3: Switch from RamDisk to SPI NOR flash memory
- Task 4: Extension via network facilities, using TRENTOS on the Raspberry Pi as server (receiving a String from a connected PC)
- Task 5: Transition of the network setup from server to client, allowing TRENTOS on the Raspberry Pi to get a small webpage from a webserver running on a connected PC

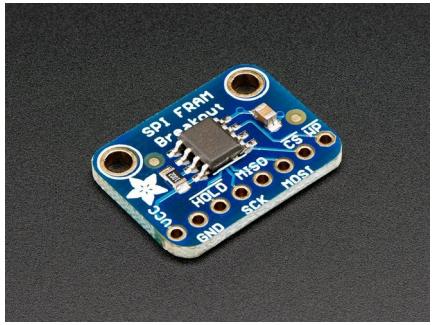
Practical Phase – Homework Assignments

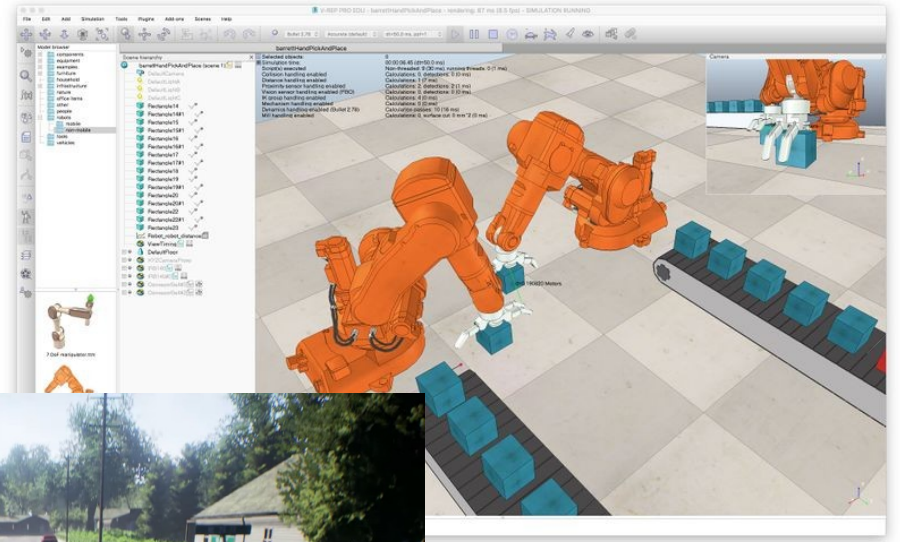




Practical Phase – Team Projects: Utilized Environments







HENSOLDT
Detect and Protect.

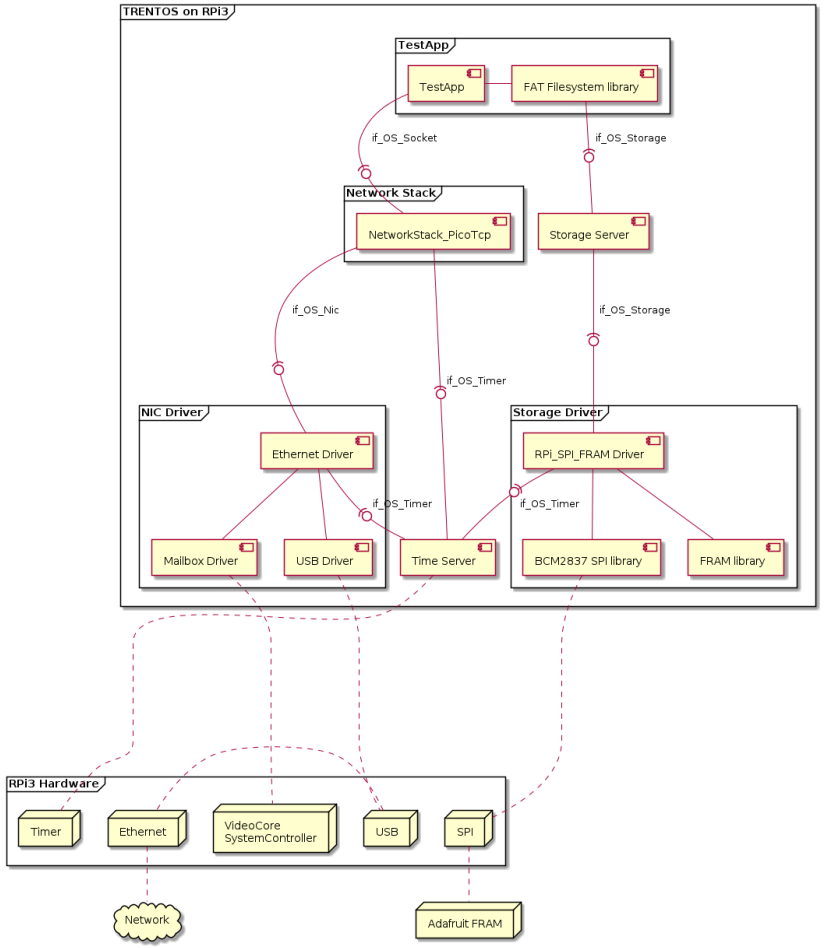
Practical Phase – Example 1: Drone Simulator Setup

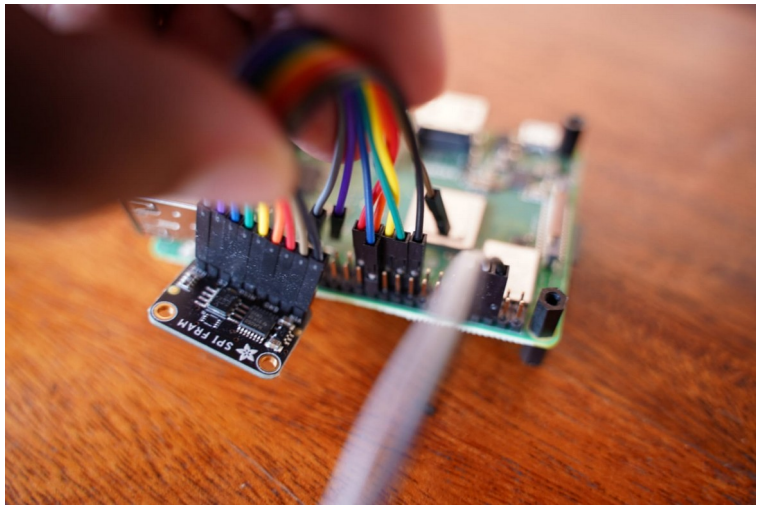
- Connection of a drone simulator (e.g. Microsoft AirSim) environment for controlling a virtual drone via TRENTOS that
  - receives virtual sensor data (e.g. from a Lidar sensor)
  - calculates a flight path to the highest surrounding position
  - offers adequate flight controller logic for steering the drone by utilizing selected virtual actuators
- Implementation of a driver for an SPI-based FRAM breakout board in order to replace the existing SPI-based NOR flash memory
  - An Adafruit SPI FRAM breakout board is used exemplarily
  - An existing open-source driver/library shall be ported to TRENTOS
- Implementation of a driver for an SPI-based Ethernet breakout board in order to replace the existing RPi 3 B+ internal network controller
  - A Microchip ENC28J60 SPI-based Ethernet controller is used exemplarily
  - An existing open-source driver/library shall be ported to TRENTOS

Practical Phase – Example 2: TPM-based security

- Porting of a standardized open-source library for integrating an SPI-based TPM device (e.g. LetsTrust TPM) within TRENTOS to allow for
  - hardware-supported key store functionality
  - hardware-based acceleration of OS crypto functionality
- Development of encrypted file system functionality by using
  - the TPM as key store
  - selected software-based crypto algorithm(s) for encrypting/decrypting files/file content within the existing TRENTOS file system support
  - selected hardware-accelerated crypto algorithm(s) for encrypting/decrypting files/file content within the existing TRENTOS file system support
- Realization of performance measurements to allow for a comparison between both software-based and hardware-accelerated crypto functionality

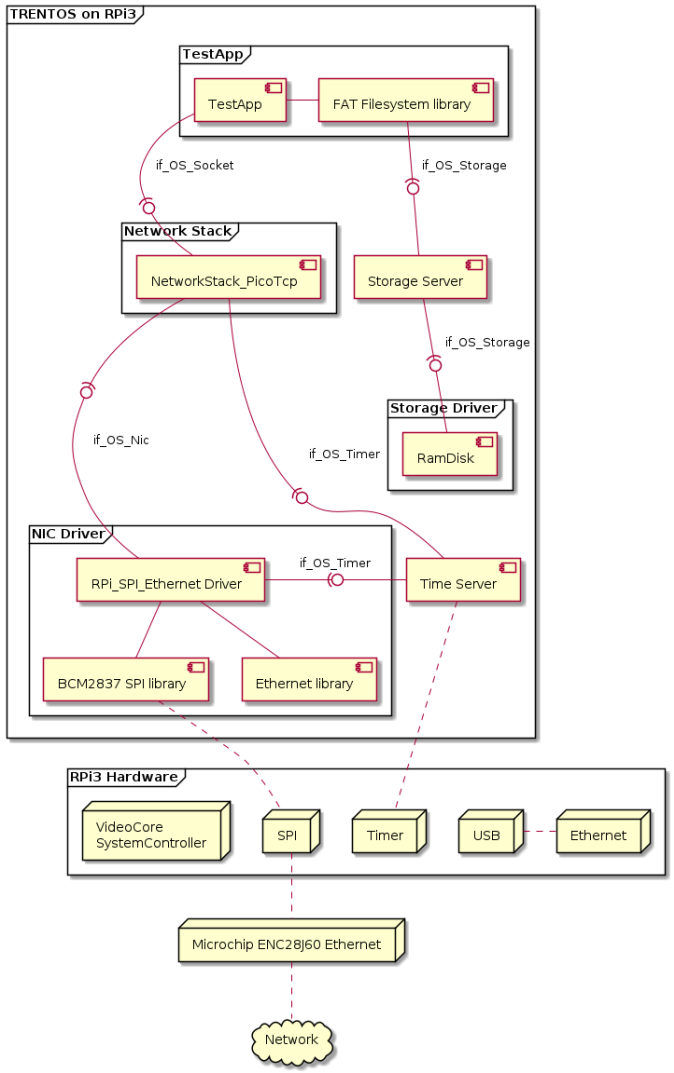
Practical Phase – Results: SPI FRAM

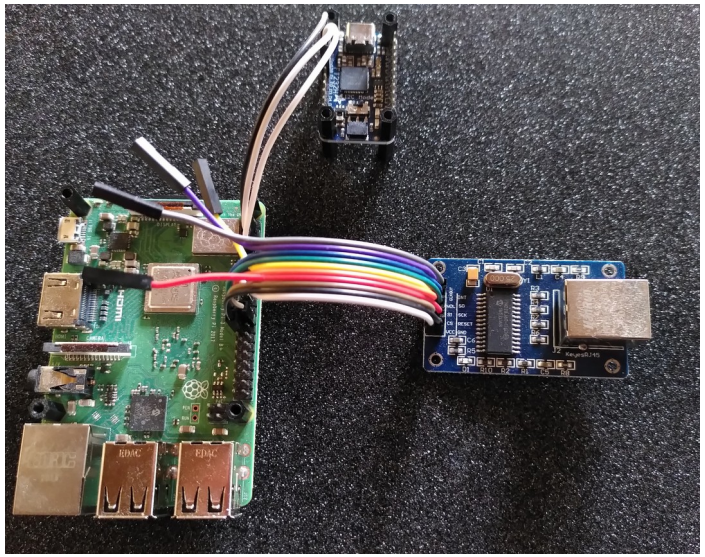




HENSOLDT
Detect and Protect.

Practical Phase – Results: SPI Ethernet

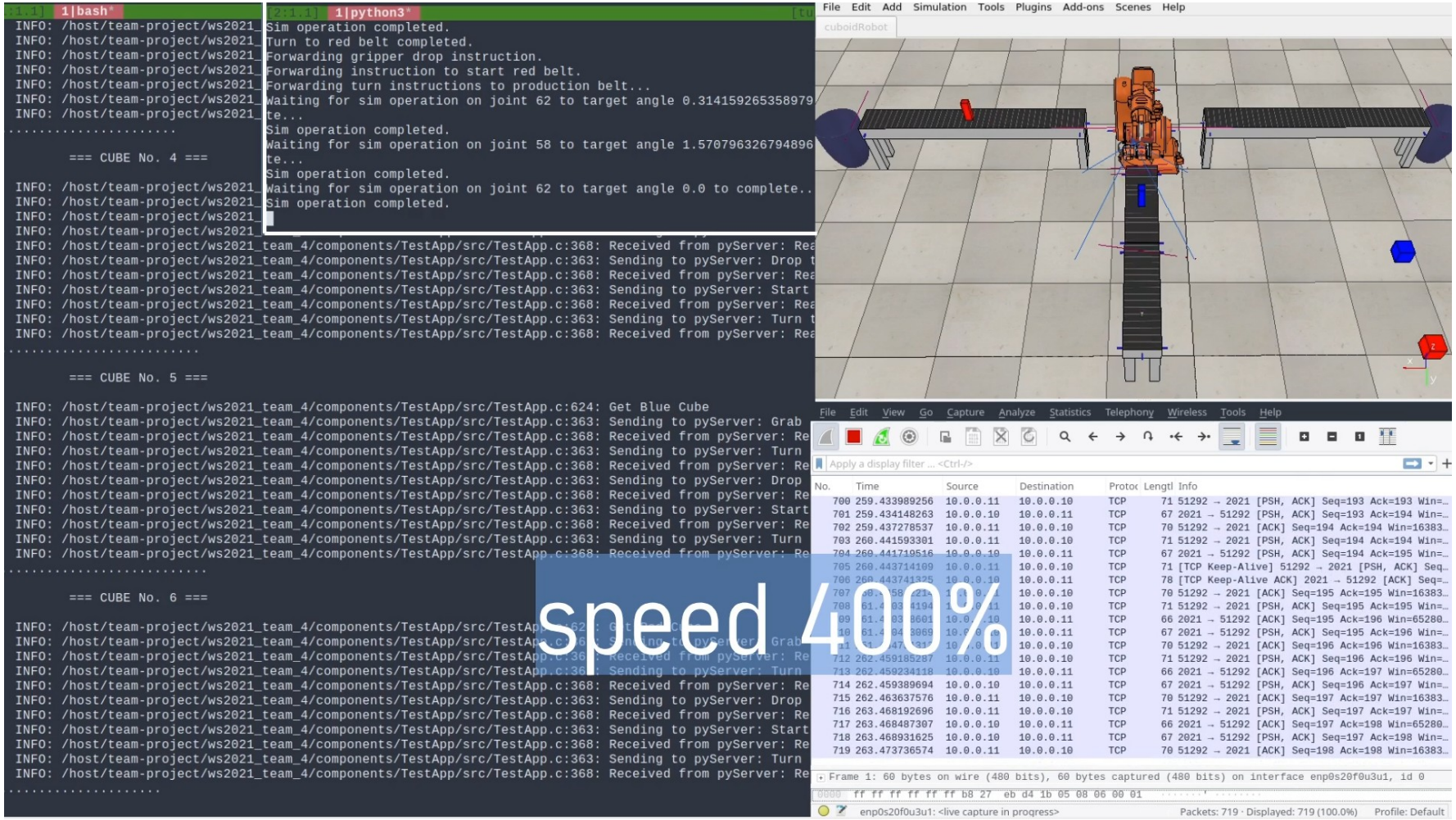




HENSOLDT
Detect and Protect.

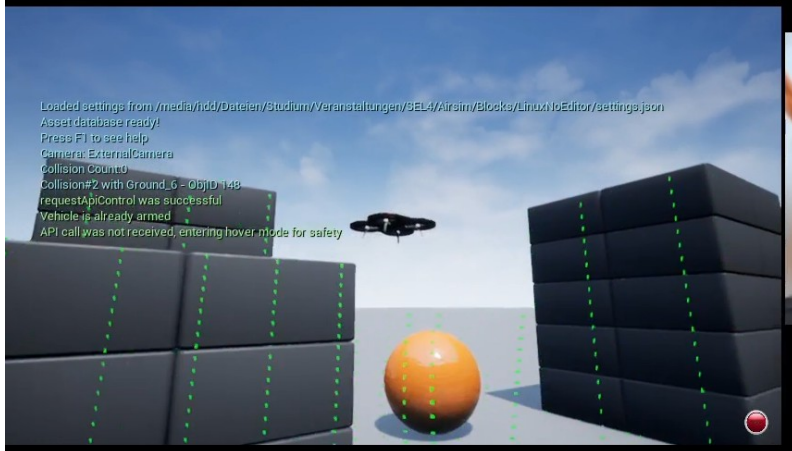
Practical Phase – Results: Autonomous Driving Simulator

Practical Phase – Results: Robotic Simulator



Practical Phase – Results: Drone Simulator + SPI LED

AirSim

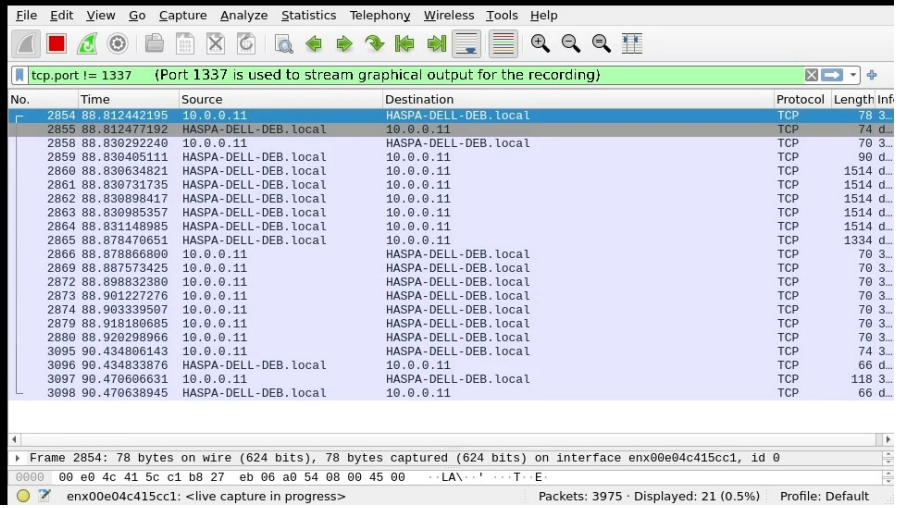


Graphical output

TRENTOS log

```
   INFO: /host/home/haspa/ss2022_team_1_ds/airsim_lidar_lcd/components/TestApp/src/TestApp.c:
409: Block #24: -39.59, 279.08, 0.87

   INFO: /host/home/haspa/ss2022_team_1_ds/airsim_lidar_lcd/components/TestApp/src/TestApp.c:
409: Block #27: -94.32, 265.62, 0.87

   INFO: /host/home/haspa/ss2022_team_1_ds/airsim_lidar_lcd/components/TestApp/src/TestApp.c:
416: Vector to Destination is (24.99, 50.44, -25.88).

   INFO: /host/home/haspa/ss2022_team_1_ds/airsim_lidar_lcd/components/TestApp/src/TestApp.c:
476: Vector of Flightpath 1 is (2.08, 4.20, -4.31).

   INFO: /host/home/haspa/ss2022_team_1_ds/airsim_lidar_lcd/components/TestApp/src/TestApp.c:
477: Vector of Flightpath 2 is (2.08, 4.20, -2.16).

   INFO: /host/home/haspa/ss2022_team_1_ds/airsim_lidar_lcd/components/TestApp/src/TestApp.c:
478: Vector of Flightpath 3 is (2.08, 4.20, 0.00).

   INFO: /host/home/haspa/ss2022_team_1_ds/airsim_lidar_lcd/components/TestApp/src/TestGraphi
cs.c:291: Drawing message "Flying"; it has extent (0, -6, 19, 1)
```

Wireshark



HENSOLDT
Detect and Protect.

Future Work

- Application of further simulators
- Extension to cryptographic use cases
  - via TRENTOS Crypto API
  - via dedicated hardware components
- Application of RISC-V as underlying hardware platform (e.g. via QEMU)
- Utilization of seL4 virtualization facilities from within TRENTOS, running a Linux VM and native TRENTOS components side-a-side

Sebastian Eckl

sebastian.eckl@hensoldt.net

HENSOLDT Cyber GmbH
Willy-Messerschmitt-Straße 3
82024 Taufkirchen
www.hensoldt-cyber.com

HENSOLDT
Detect and Protect.