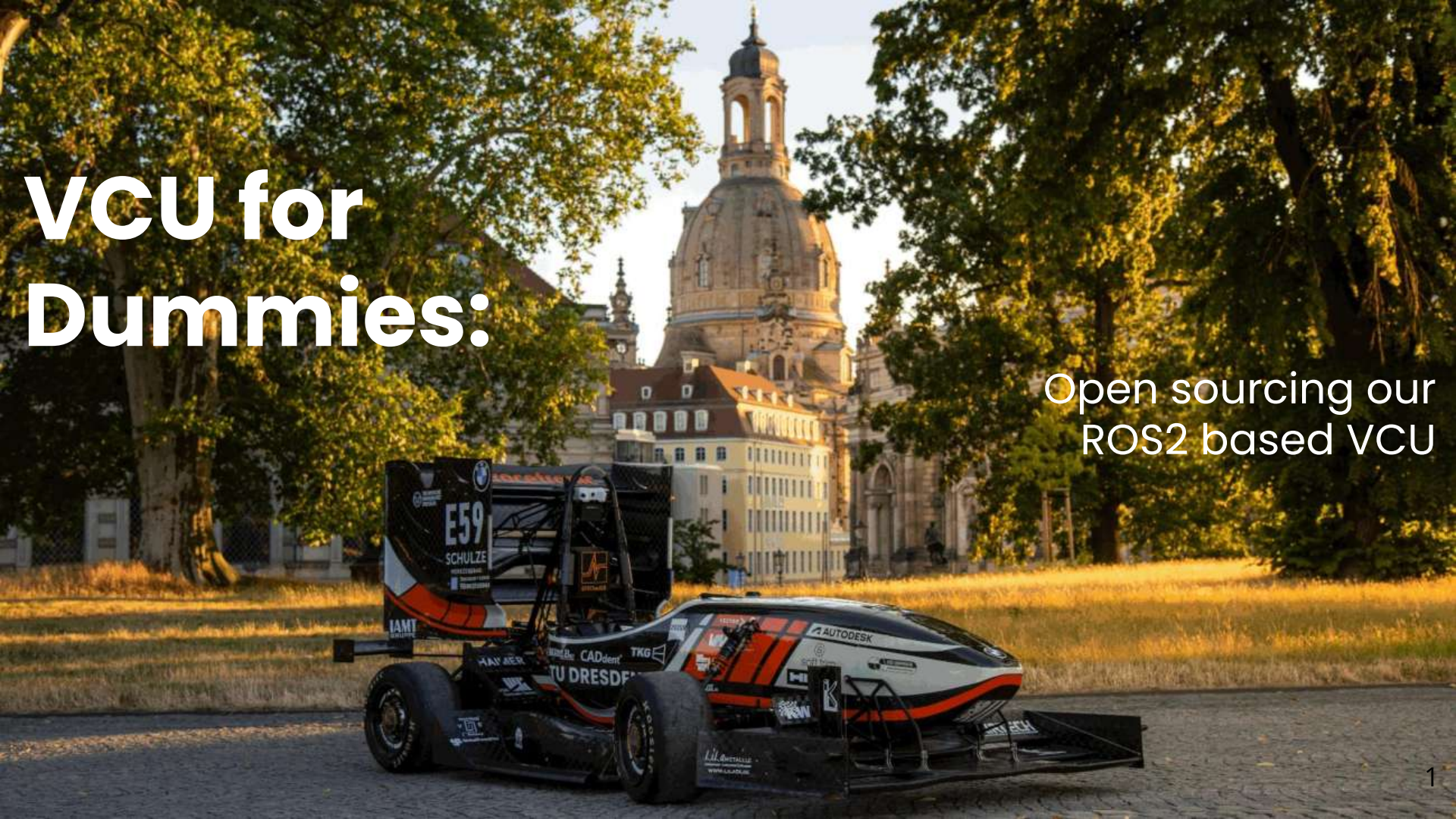
VCU for Dummies:

Open sourcing our ROS2 based VCU



Content

1. Recap of last ARWo and the following integration
2. What will be published
3. Experiences with Development and Integration in terms of technical and human aspects
4. Highlights of our VCU Implementation
5. Simple startup Guide for our open source VCU
6. Guidance for your new VCU features
7. Contributing and Legal stuff ;)

ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

1.a Man of action

Niklas Leukroth:
~4 years at EFR

Head of Autonomous System in EFR16ed04

finished master of computer science

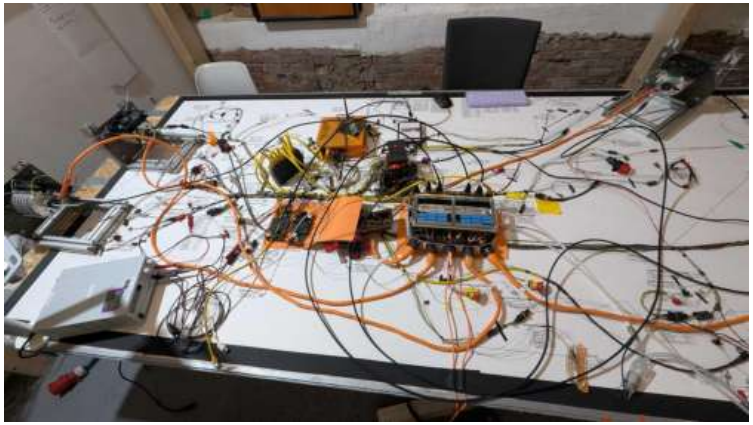
Laurin Heßlich:
~5 years at EFR

Head of Autonomous System in EFR15ed03

working on finishing diploma of computer science; internship

ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

1.b What is VCU

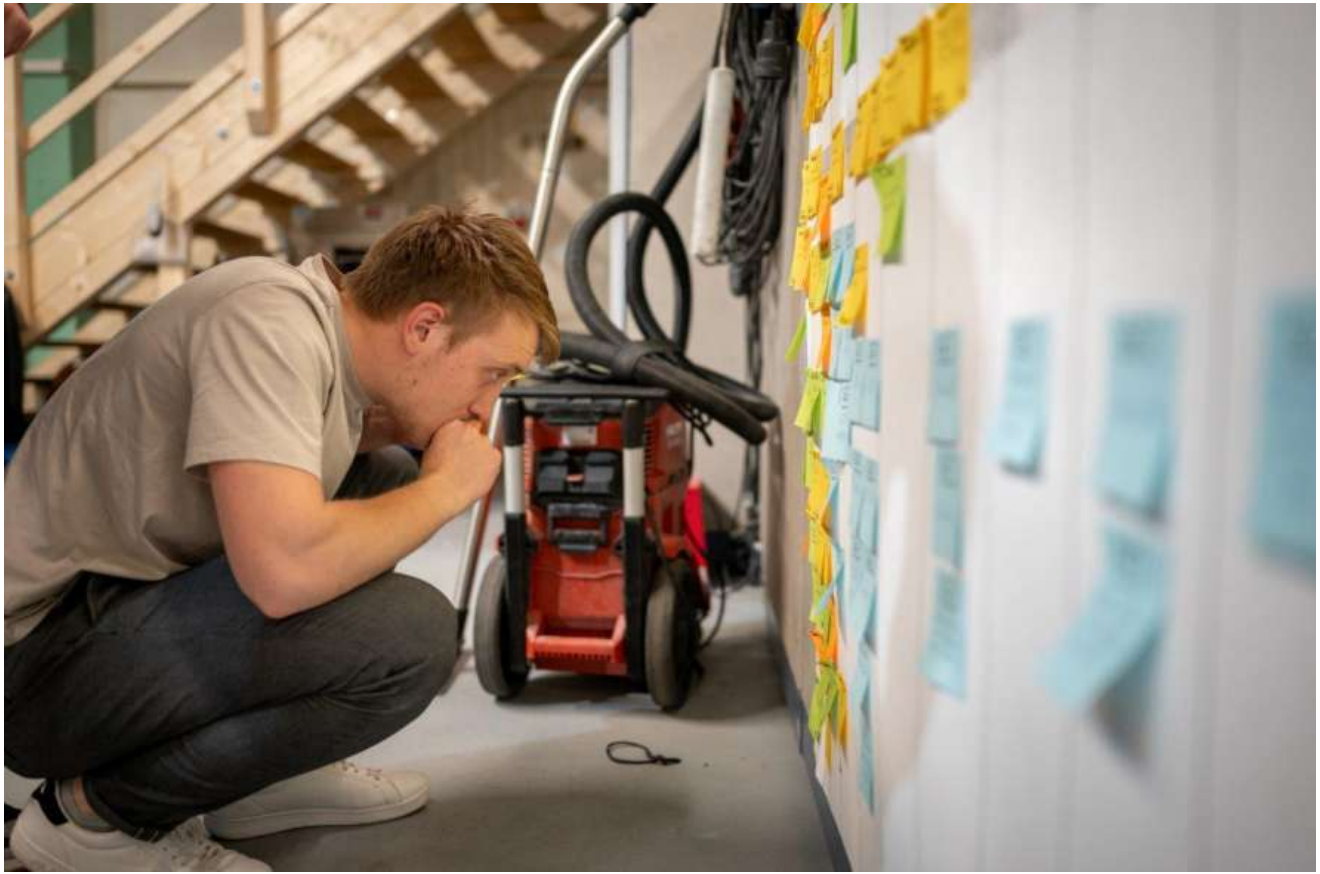


Vehicle **C**ontrol **U**nit:

- Meant is the software stack that:
  - Implements rules and safety features
  - Connects with control algorithms and driving assistance code
  - Adding comfort features
  - Deployed on one central compute unit

⇒ Is the software (and electrical) center of the race car

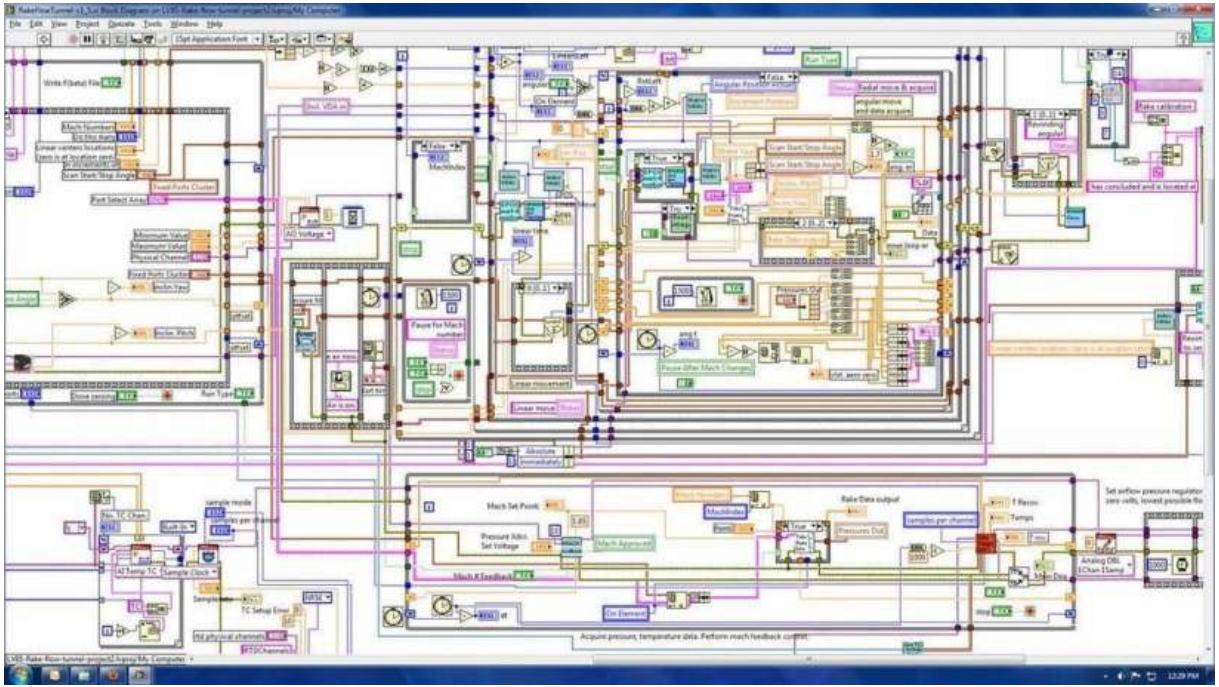
1. Recap of last ARWo



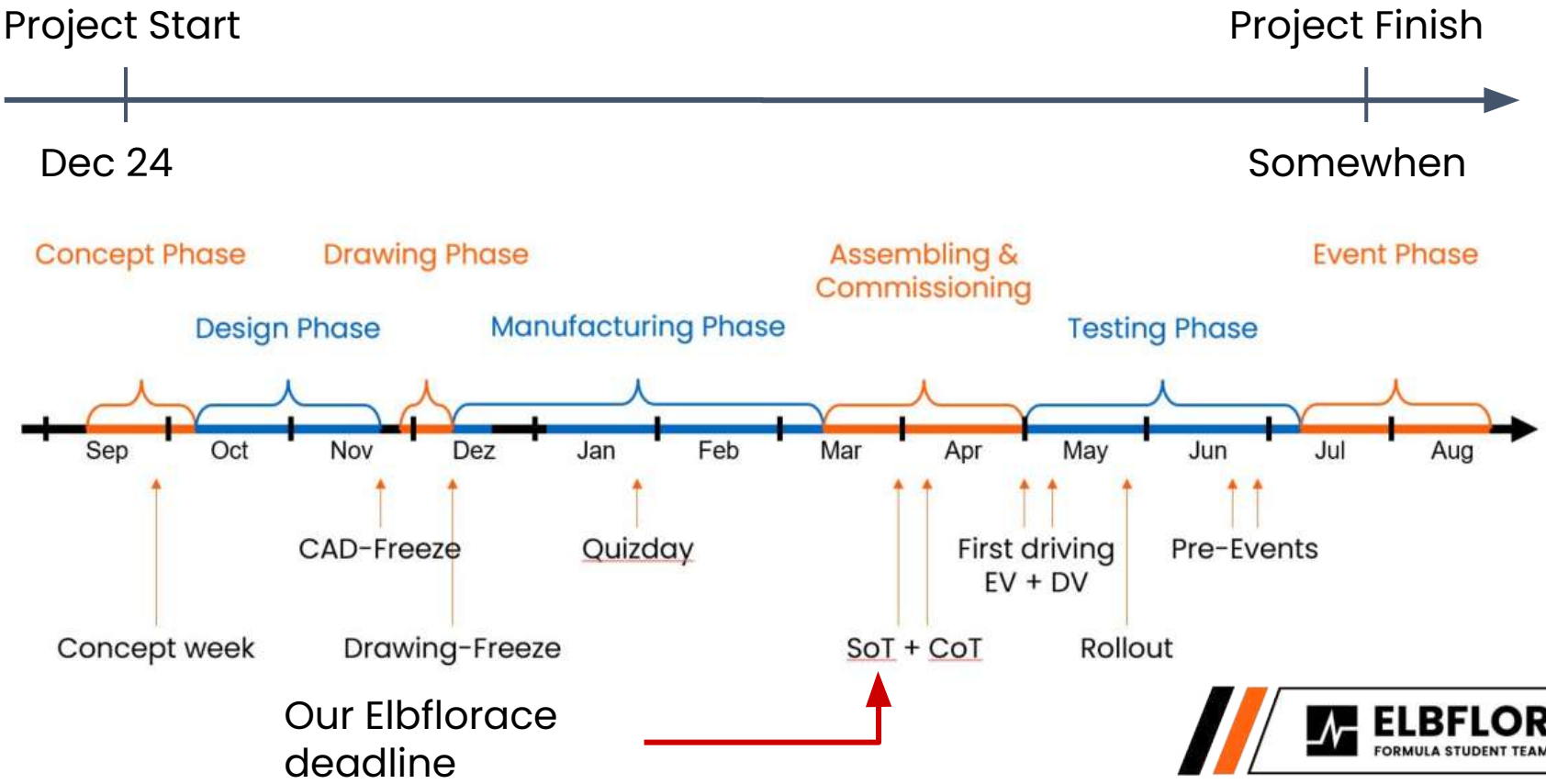
ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

1.b Old System

- Labview VCU
- Nobody really understands what happened in there
- No real version control
- An old grown software stack
- Not able to implement more complex algorithms



1.c Time plan



ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

1.d Task distribution

Niklas:

- Mailman
- Vehicle Control Unit
- Watchdog
- Dummy EV and DV Controls
- Statistics and logging
- Miscellaneous stuff

Laurin:

- State Machine
- EBS State Machine
- Safety Features
- Cooling
- Introduction Task
- Miscellaneous stuff

1.d Task distribution

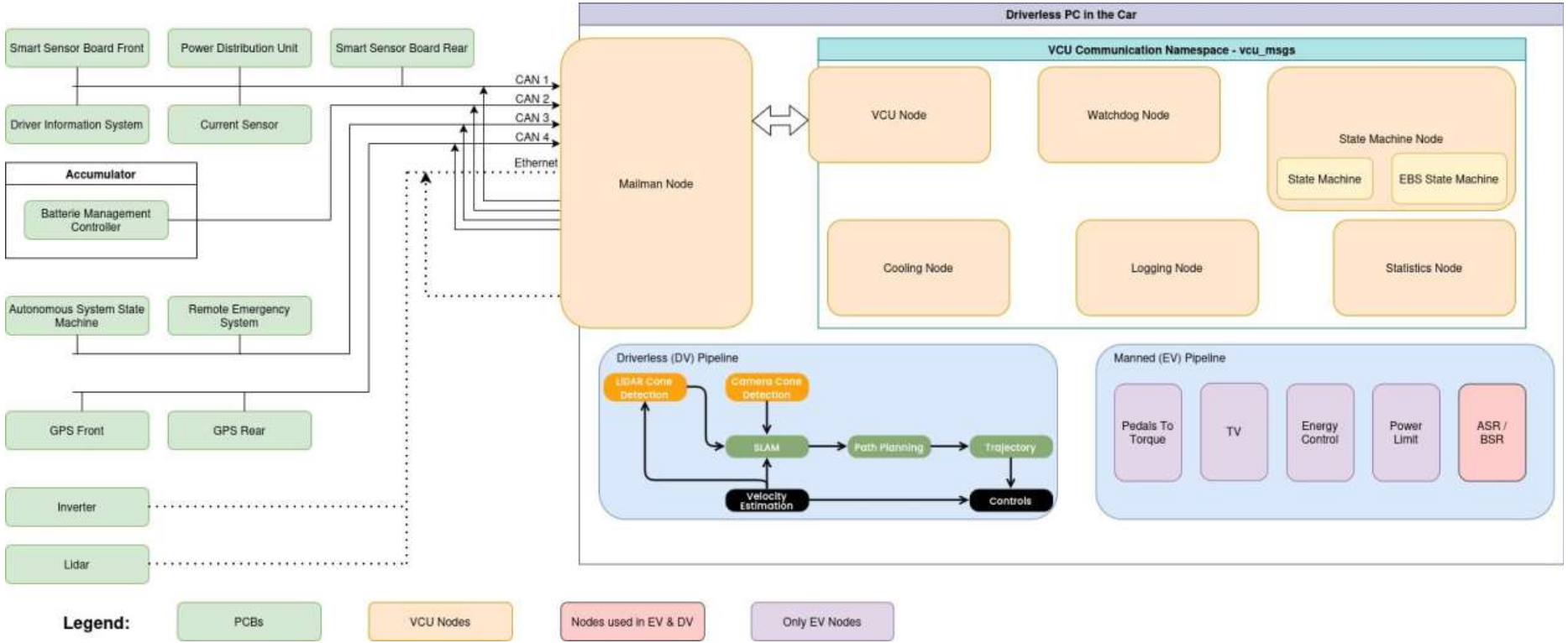
Niklas:

- Bottom up approach
- Start with rough idea, figure it out as you go
- Visualizations during the development process

Laurin:

- Top down approach
- Analysis of existing solution
- Start of with diagrams and get early feedback
- Test driven development

1.e Top Level Design



ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

1.f Status after ARWo

- First good drive after some initial control issues were addressed
- Completed 4km that day without VCU issues or unexpected behaviour

ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

1.g Outlook last ARWo

Team driving experience

- First important step for “in-production” usage
- expected around 500 driven km and uptime of at least 36h

Events with new VCU

- Port of driving assist systems
- Lots of testing prior to events
- Increasing robustness

ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

1.h Driving Experience

Driving Experience boosted the integration:

- In general new VCU was more robust than the car
- In three days the following was developed while doing the driving experience:
  - Adapted “Antriebs-Schlupf-Reglung” (ASR)
  - Ported driving assistance code
  - Tested Live Monitoring
  - Introduces code and handling of the code with the current seasons people

⇒ Sets up the basis for usage on Events

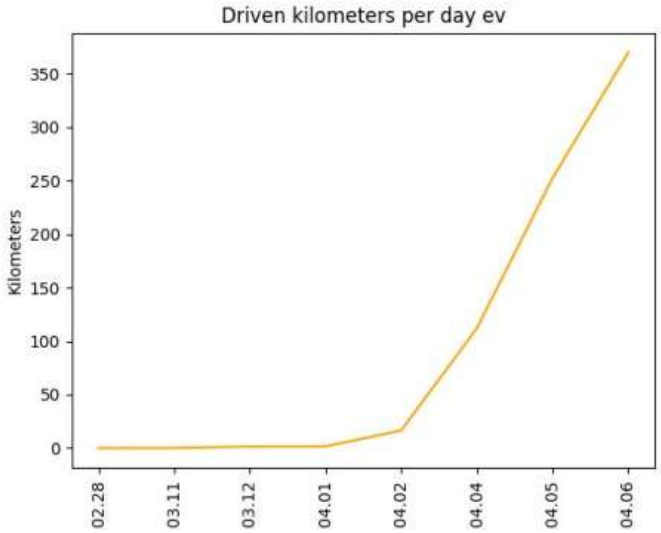
1.h Driving Experience

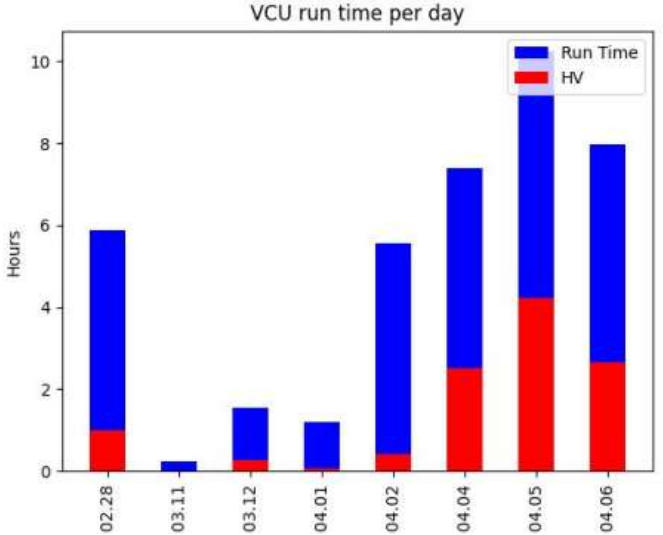
Debugging Life:

ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

1.h Driving Experience

Early/Easy error detection like here with APPs Error





1.i Events

- Drove on all Events with the new VCU
- Reached the same performance as with the old one
  - adopted control algorithms for different disciplines (e.g. TV)
  - way more (live) logdata information
  - way faster debugging
- No one is thinking about the old VCU anymore
- 3. Place Digital Twin Award
- Lots of new ideas

2. What we will publish

ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

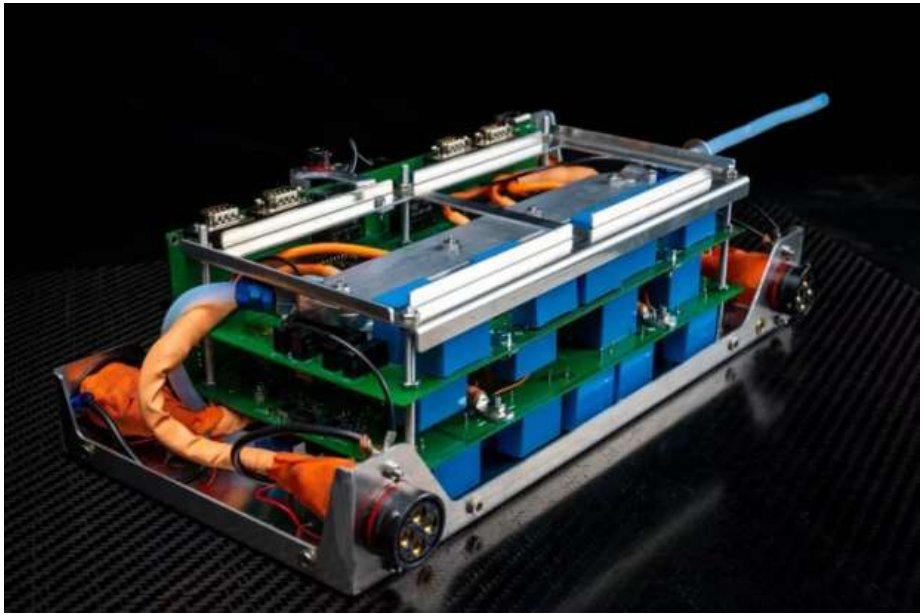
2.a What we will publish

- A VCU that includes all safety features & rules requirements:
  - State Machine (including EBS State Machine)
  - Watchdog
  - Example VCU node implementations
  - Simple cooling, logging and analytics node

- A (very) simple example controls node for EV
  - Only mapping accelerator pedal to wheel speeds & torque

2.a What we will publish

- A driverless inspection mission node
- CAN Message Generator with dummy .dbc file
- Inverter Ready Communication with Electrophorus Inverters
  - Inverter Ethernet message already implemented
    ⇒ Faster Time to Race



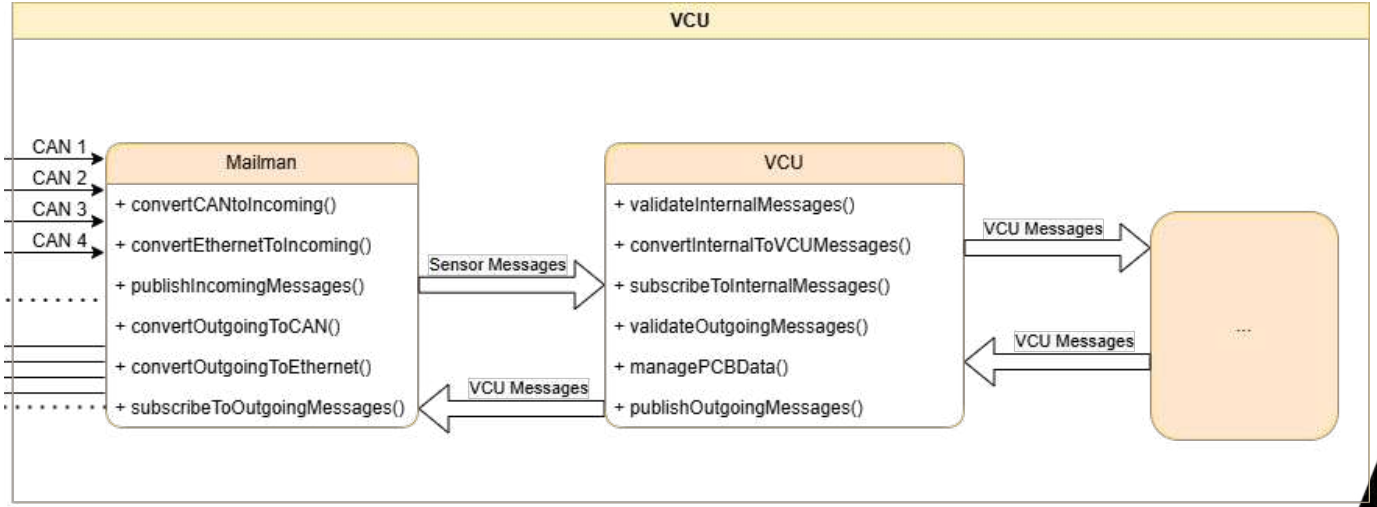
2.b What you must do

- Generate sensor messages with your .dbc file
- Implementing VCU node:
  - Conversion sensor_msgs to vcu_msgs (next slide)
  - Custom PCB control (e.g. steering actuator)
- Implement/Port your Controls for EV/DV
- Implement more useful features:
  - Custom Cooling Node
  - E.g. more analytics, other features
    ⇒ Eventually share them ;)

2.b What you must do

Converting sensor_msgs to vcu_msgs:

- sensor_msgs generated from your .dbc file
- vcu_msgs internal messages for the watchdog & state machine
- sensor_msgs can have more data that is not needed for the vcu



ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

3. Experiences with Development

ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

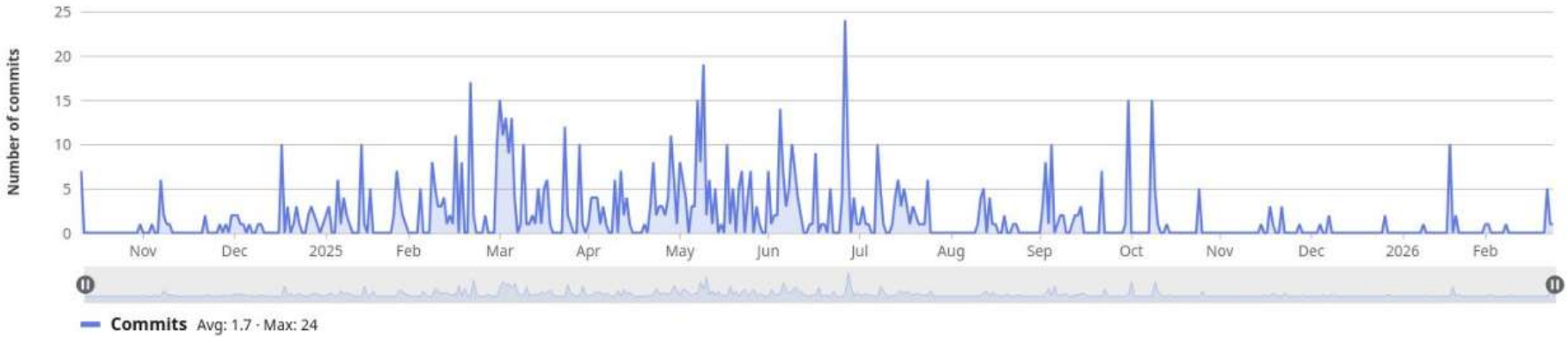
3. Development Continued

Contributor analytics

main | History

Commits to `main`

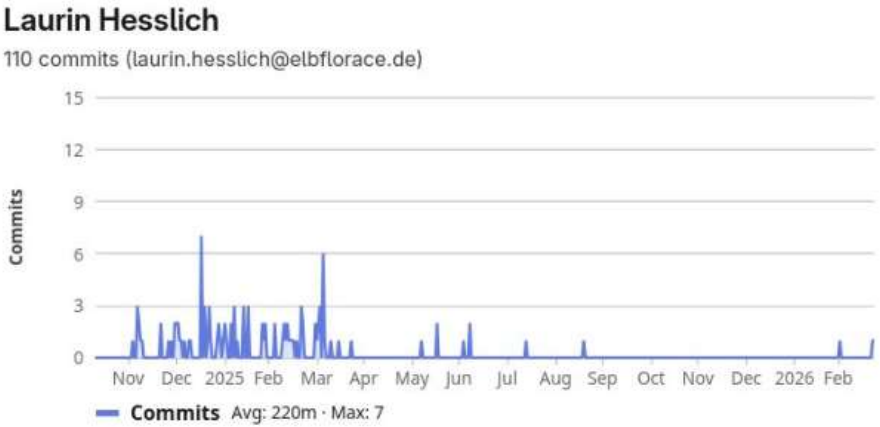
Excluding merge commits. Limited to 6,000 commits.

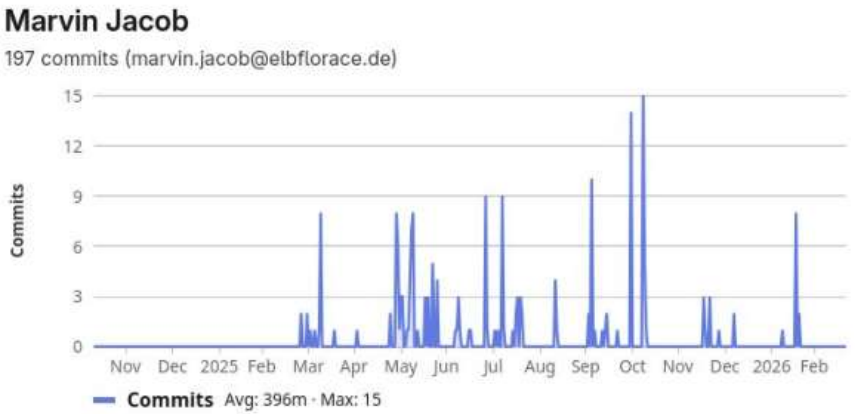


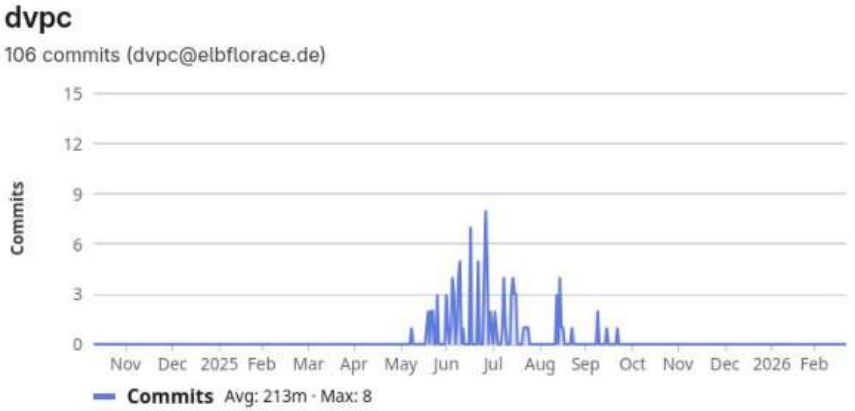
ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

3. Development Continued

- Only small contributions by Niklas & Laurin after April
- Shifted responsibilites to the new staff







ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

3.a New Developments

- (Port of) Torque Vectoring
- Port of Power Limit
- Different ports of Energy Control
- LTE-Monitoring
- Video Recorder
- Porting of various electrical control stuff
- Various adaptations on existing stuff
- Miscellaneous stuff

3.a New Developments

- (Port of) Torque Vectoring
  → Porting of old version as well as entire new version implemented
- Port of Power Limit
- Different ports of Energy Control
- LTE-Monitoring
- Video Recorder
- Porting of various electrical control stuff
- Various adaptations on existing stuff
- Miscellaneous stuff

- done by non AS member; - done by experienced member; - done by AS member

ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

3.b Experiences

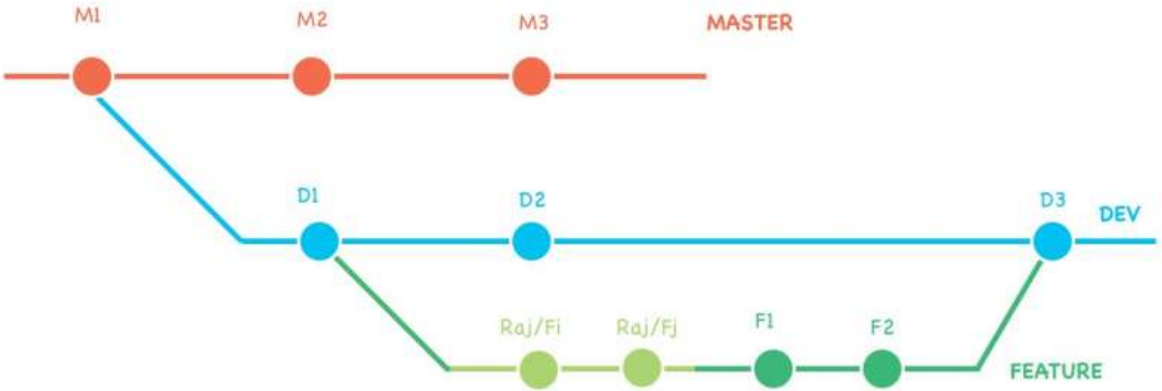
- Onboarding of team took a lot of time:
  - Several multiple hour phone calls
  - Lots of on site help during System on the Table (SOT) and commissioning
  - There was not much time
- People were motivated:
  - new and older members tried the introduction task
  - some non AS members implemented new stuff
    - needed support but worked
- **Development got faster and enabled new workers**

3.b Experiences

- Also shift of responsibilities:
  - Niklas & Laurin only consulting services
  - Only implemented nice to have features
- New people now decided:
  - what should be implemented and when
  - organized testing
  - introduced new people
- **BUT:**
  - Not all relevant people worked with it
  - Monitoring now way simpler, but still a task for specific people
  - Niklas & Laurin supported on Events for a faster performance

3.c Git Dev Cycle

- main: Always real-world tested and running
- dev: new SIM-tested code, not real-world tested
- feature-branches:
  - branch from dev or main
  - when ready (and SIM-tested) merged into dev
- Discussions about merge strategies: rebase or normal merge



4. Highlights of our VCU

ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

4.a LTE Monitoring

- We are able see the current status (and all data flowing in the dvpc network) from everywhere in the world
- Easy remote debugging possible
  - old routine: pull Labview log data from car
    → convert to MatLab file
    → load data in MatLab (or Python notebook)
    → start plotting graphs for debugging
    (min. 15min from failure to start of analysis)
  - enormous speed up of debugging from failures
    → seeing errors directly while still driving
- Problem sometimes no LTE connection (e.g. FS Czech)
- System not dependent on e.g. Wifi

4.c AS Integration

Autonomous System Controls are seamlessly integrated:

- AS Mission are like EV missions
- Switch between EV and AS via ASMS
- Integrated EBS State Machine
- Control nodes are easily exchangeable
- Exchange of EV and Driverless control nodes
- Usage of livecam images via AS cameras also in EV mode now possible

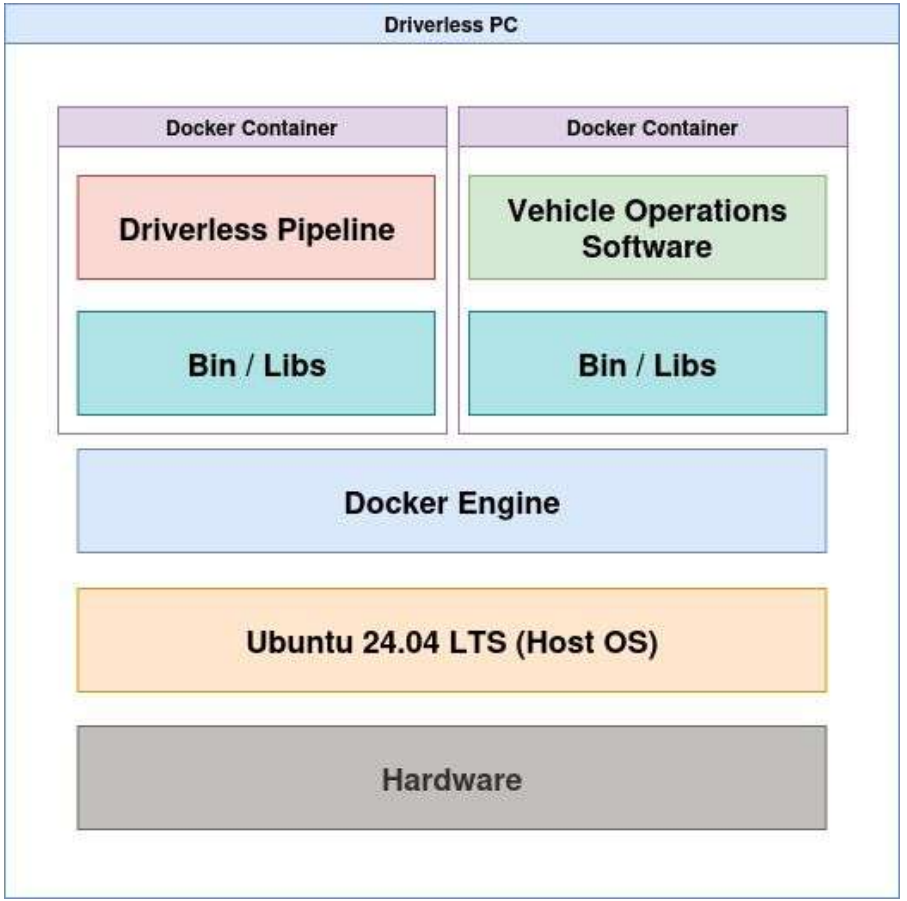
4.d Controls Port

Ported and developed controls for EV

- Integrated a new Torque Vectoring and achieved better times
- Ported Launch Control for Acceleration:
  - Could deeply analyze it
  - Tested new tweaks and adaptions
- Porting of not understandable Energy Control
  - Did not drove it due to
- Integrated Driver Communication that worked better than a mobile phone

4.e Docker Setup

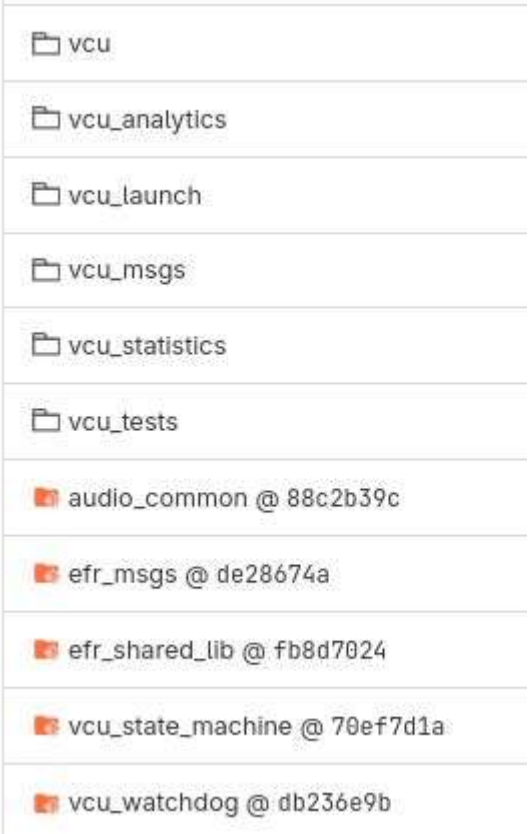
- Separation of VCU, EV pipeline and DV pipeline
- Also separated repositories:
  - one repo per season
  - submodules for persisting modules



ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

4.f Repo Setup

- Every season has an own top-level repository
- overarching packages are structured in sub-modules
  - less changes expected
  - changes possibly concerning all cars
- packages changing every season are just forked and adapted



ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

5. Start up Guide

ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

5.a What you need

- Software:
  - Linux Operating System
  - Docker
  - Recommended: VS Code DevContainer and SSH setup
- Hardware:
  - at least 8GB of RAM
  - enough CPU power for your algorithms (recommended i5-13th)
  - SSD/HDD for log files (recommended 1TB)
  - CAN Hardware (e.g. M.2 CAN Card)
- Note:
  - Only the VCU code could also be able to run on a Raspberry Pi (Building with less than 8GB of RAM is not possible but there are other solutions)

5.b Our Hardware

- **LattePanda Sigma**
  - Nvidia RTX 4060 GPU
  - Intel Core i5-1340P
  - 32 GB RAM
  - 2TB SSD
  - PCAN M.2 CAN card

⇒ Costs about 1500€,
but runs also all the control code
(EV & DV)

ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

5.c Get Started

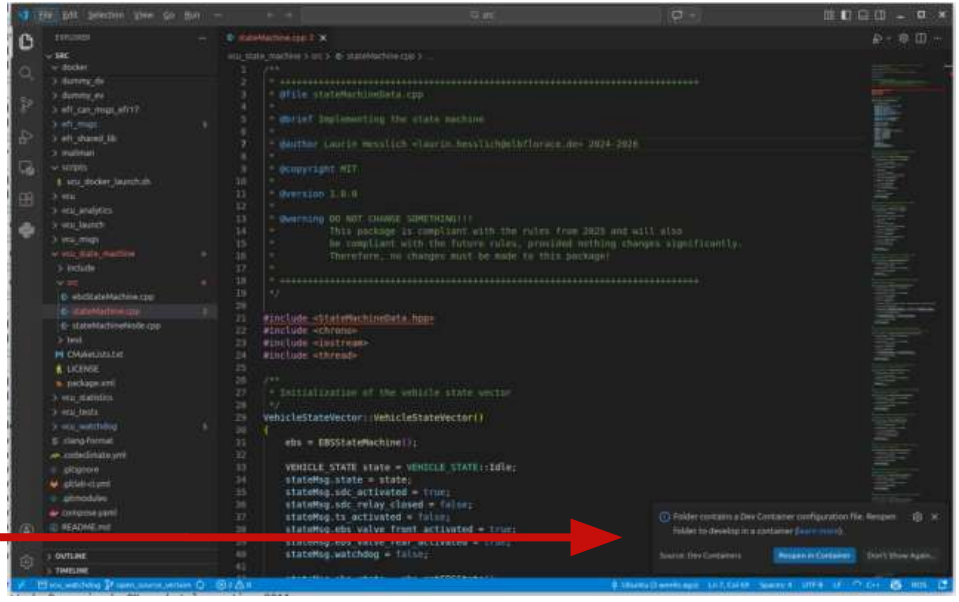
1. Pull
2. Build Container

Command:

```
docker build --build-arg USERNAME=devuser -t vcu:dev -f Dockerfile --target dev
.
```

With DevContainer:

Prior to install:
Using Ubuntu
DevContainer Extentions



5.c Get Started

1. Pull
2. Build Container
3. Build Code
   3.1. Create can_msgs with CAN generator from .dbc file and implement the vcu_node

5.c Use the CAN Gen.

When using Docker (recommended):

- Run command “generate-can”
- More information in src/can_communication/vcu_can_generator/README.md
- Expected output similar to:

```
$ generate-can
Using /home/devuser/vcu_ws/src/./efr_can_msgs_efr17/can-dbc-efr17 for DBC file
Generating for season efr17
Found venv at /home/devuser/vcu_ws/src/efr_can_generator/venv
.........................................
Generating for ['CAN1_Vehicle.dbc', 'CAN2_Accu.dbc', 'CAN3_DV.dbc', 'CAN4_GPS_IMU.dbc', 'FSG_DBC_DV.
dbc']
.........................................
Replacing efr_can_msgs_efr17
Replacing efr_mailman_efr17
Replacing proto files in efr_monitoring_backend
```

5.c Get Started

1. Pull
2. Build Container
3. Build Code
   - 3.1. Create can_msgs with CAN generator from .dbc file and implement the vcu_node
   - 3.2. Run tests
4. Setup Autostart

5.c Run Tests & Autostart

When using Docker (recommended):

Run tests:

- Run command “colcon test”
- Output shows if tests failed
- Analyze test results with:
  - Run command: colcon test-result --verbose
  - (there are currently failing tests :/)

Autostart:

- We used systemd service to start the and run the docker

5.c Get Started

1. Pull
2. Build Container
3. Build Code
   - 3.1. Create can_msgs with CAN generator from .dbc file and implement the vcu_node
   - 3.2. Run tests
4. Setup Autostart
5. Reboot and Start Driving
6. Have fun!
7. Optional 1: Implement EV controls
8. Optional 2: Integrate your driverless pipeline

6. Guidance for own features

ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

6. Guidance for own Features

- Simplest way is to start by copying an existing simple package (e.g. cooling)
- Adapt all the names and dependencies
- Implement your own business logic
- When new messages are needed:
  - internal VCU node communication (not controls):
    → use of vcu_msgs package recommended
  - between VCU and Controls:
    → use of ix_msgs package recommended
- Reuse code between different cars: use git submodules

6. Guidance for own Controls

It is possible to integrate different controls for different missions

- Advices for this:
  - Set up defined topics and do not change them in the code
  - Remap the topics in the launch file
  - Make control nodes small and modular
  - Define multiple launch files with different controls

7. Contributing and Legal stuff ;)

7.a Contribution Guide

- Use auto formatting
- Use common sense when writing code
- Write tests which cover your code
- Do pull requests and issues
- Make good descriptions on Github
- Add contact information for questions in the own team and other stakeholders

⇒ See contributing guidelines on GitHub

7.b Legal stuff ;)

We accept no liability for damage to persons or property.

The software is provided free of charge to the best of our knowledge and belief and can be modified and adapted as desired.

We would be delighted if you would contribute to its improvement.

7. Feedback

ARWo Feedback Link:

Presenter contact:

Niklas Leukroth
niklas.leukroth@elbflorace.de

Laurin Heßlich
laurin.hesslich@elbflorace.de

General contact:
vcu@elbflorace.de
Slides at
www.elbflorace.de/publications

ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN