PTY 2025/26 at ZF

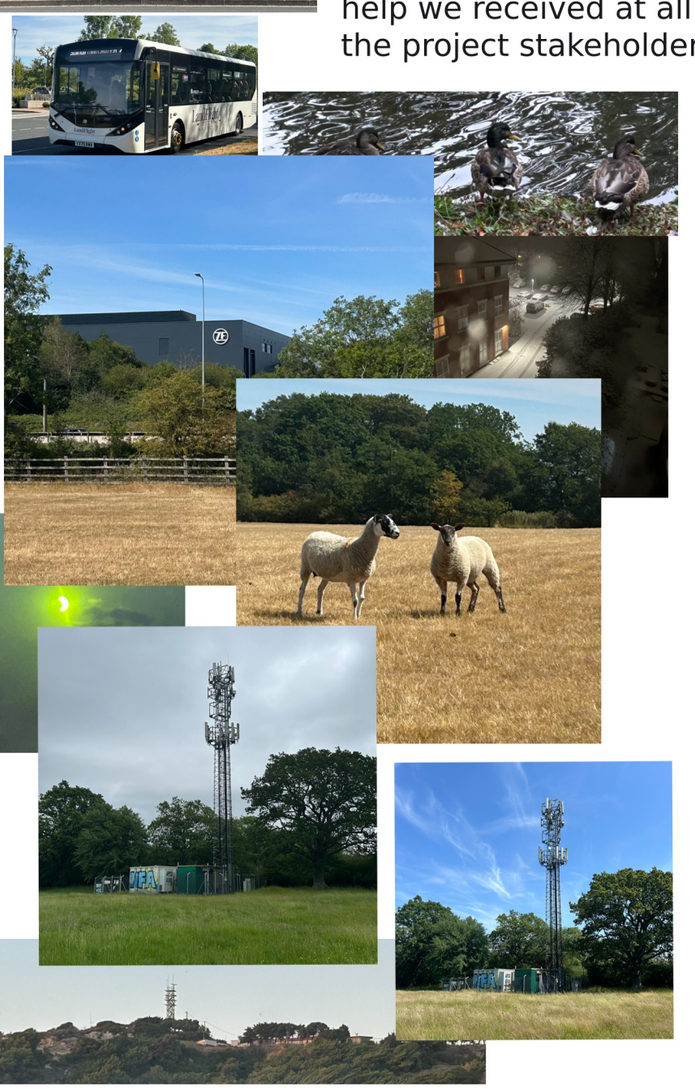
Liam Cracknell

For my placement year I worked at ZF Friedrichshafen AG, also known as ZF Group, at their technical centre in Birmingham. Founded by the inventor of the zeppelin airship, ZF is now a large manufacturer in the automotive sector supplying products such as powertrains and chassis for passenger and commercial vehicles.



As a ‘Student Engineer’ at ZF I worked within the cyber security team, writing code mainly for internal tooling alongside another placement student. We attended cyber team meetings and daily standups to talk about how our work was progressing with those on site and also those from cyber in India.
On Fridays, which were also a half day, I would work with all three other students on one long project throughout the year. This allowed me experience work at scale, with us going through all the stages that a real project at a company would. This involved extensive planning and meeting many people across the whole site to understand their requirements for the project, along with help we received at all stages, through to implementation, presenting our work at the end of the project stakeholders and documentation for its continued future development.



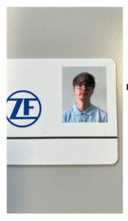


PyDownloader

PyDownloader is an internal ZF tool used for executing custom UDS sequences over a CAN bus, and for downloading firmware such as a bootloader onto ECUs. This is designed for integration with EXAM testing workflows, which allow for further automating the test pipelines for new ECU bootloader versions. On site, we have test rig racks full of mainly steering and braking ECUs connected to a computer. Throughout this work, I would remotely access these rigs, and a mock development rig, to test PyDownloader could send and receive UDS messages with the real hardware.

The goal behind the new version was, other than continuing feature development, to give a new structure to it tool inside a proper ZF tooling repository and DevOps pipelines. Still written in Python with tkinter, we did get the opportunity to use newer tools like uv for dependency management and moon alongside a robust testing strategy.

Part of my work was based around UDS commands. Using the udsoncan python library and an internal addon for it to work best with PyDl, this would allow the tool to create a successful connection and send specific commands, such as disabling and enabling Diagnostic Session Control for flashing purposes. Challenges arose with this around supporting both Vector and PCAN adapters , which needed to be handled differently to interface correctly with udsoncan. Learning for this system afforded me the pleasure to read 'ISO 14229-1, Road Vehicles - Unified Diagnostic Services (UDS), 2020-02 Third Edition’ to understand, among others, specific UDS message frames with their service and subfunction ids.



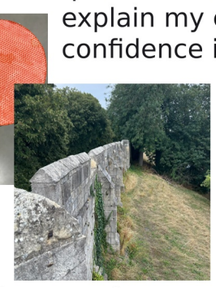
TARA RRR Generation

Coming again with a relevant standards document ‘ISO 21434, Road Vehicles — Cybersecurity engineering, 2021-08 First Edition’, I worked on a long pre-existing script to parse ITEMis Secure XSAM into an RRR excel spreadsheet. Currently a very manual task, this aims to bridge the gap for automatic RRR generation working upon code created by the TARA team and previous placement students. TARA is a risk management framework followed by these processes at ZF to look at handling intentional attacks on systems. After understanding the current system, my work centred on calculating residual risk scores directly from .mps files. instead of relying upon the full TARA library. This expands the current, also Python, script to calculate scores and assign residual treatments of ‘accepting’, ‘sharing’ or reducing’ to specific threat scenarios.

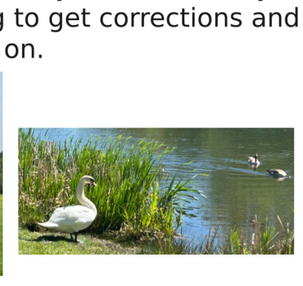


I found this very difficult initially, with understanding the current codebase alongside TARA as a new process to me. However, this did provide many opportunities to speak about and patch holes in my understanding with those more experienced than me on this task. I would be able to ask questions, or go about it the other way around and try to explain my current understanding to get corrections and confidence in what I was working on.



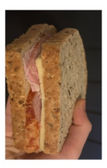












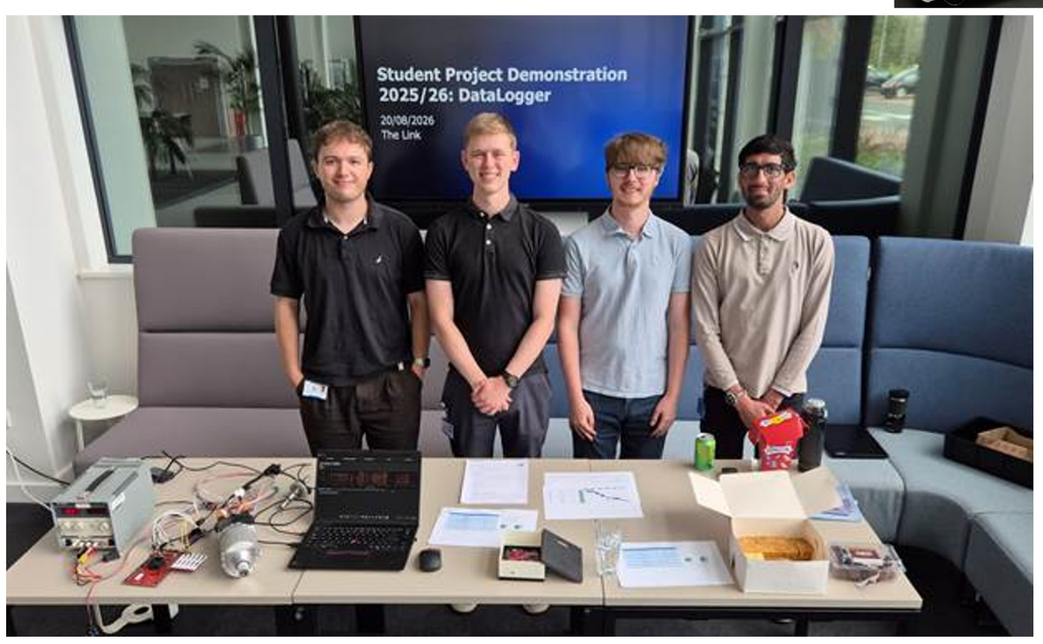
DataLogger Project



Reflection

I have improved at so much during my time on placement with the support from everyone at ZF. Going into a ‘real’ workplace for the first time and getting to immediately work on real projects with people is so valuable. I see the non-technical ‘soft skills’ as a large reason for doing a work placement during university. Gaining confidence throughout and knowing how to go about communicating with everyone is very important

Although much of my work at ZF was new to me with it being more electrical orientated, certain university modules really stood out. In first year working with C for controlling a 7-segment display for morse code decoding gave a strong baseline for understanding how the intermediary board state machine functioned for when I controlled this via the interface PC side. Learning unit testing in java throughout software engineering modules was transferable to the BDD approach taken throughout PyDownloader development lead to me helping explaining test standards and mock objects to others. Going back into university, I feel I will do better in my final year than had I not completed a work placement. Working in large codebases with others is a much different style of software development, leading to code becoming more readable and better documented. Irrespective of simply an extra year of learning and technical knowledge, working collaboratively on anything for a long period of time (sometimes across teams, and on work done years before) is something you dont quite get at university.

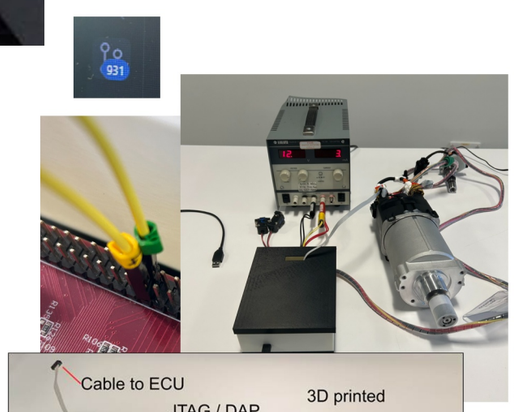


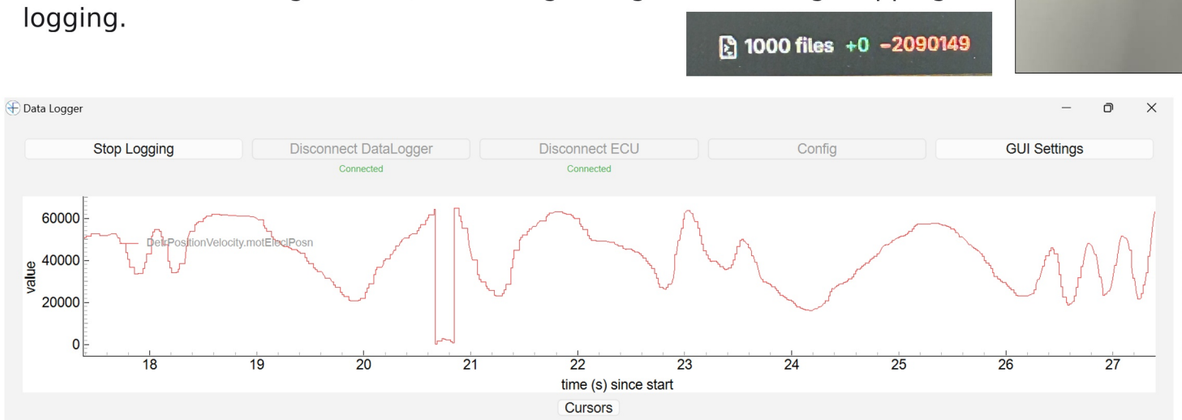


On Fridays I would work with all the other placement students on a ‘Multi-Lane High Throughput Data Logger’ for logging live data from ECUs, lasting throughout the whole year.

The given brief was to ‘develop a data logging solution that supports system development and condition monitoring, with a particular focus on multi-lane systems. The logger will be built on existing ZF electronics hardware and will unify data at a high acquisition rate into a common, standardised format for streamlined analysis and diagnostics.’

Our prototype system can read a signal from an ECU through the JTAG board over DAP (using ASCLIN instead) and to the DataLogger intermediary board using UART over ASCLIN with our custom header cable. This data is read from the serial monitor onto the interface to be displayed on a graph (image below). I mainly worked on the PC side code, making the GUI in Python using PyQt components and building more complex ones like graph panels and scrollable checklists. Handling the connection to the DataLogger board required reading the serial monitor, and sending signals to the board to change states, for setting configs and starting/stopping logging.







UKESF

My scholarship with the UK Electronic Skills Foundation also involves a professional development course they hold for that year's students at the end of their placements on the University of York campus. This was an incredible experience to conclude with, getting to meet the other scolars from across the country and for talks given from guest speakers. Some speakers stayed for our formal dinner on the Wednesday so was nice to get to speak to them more.

