Mid Level Controller

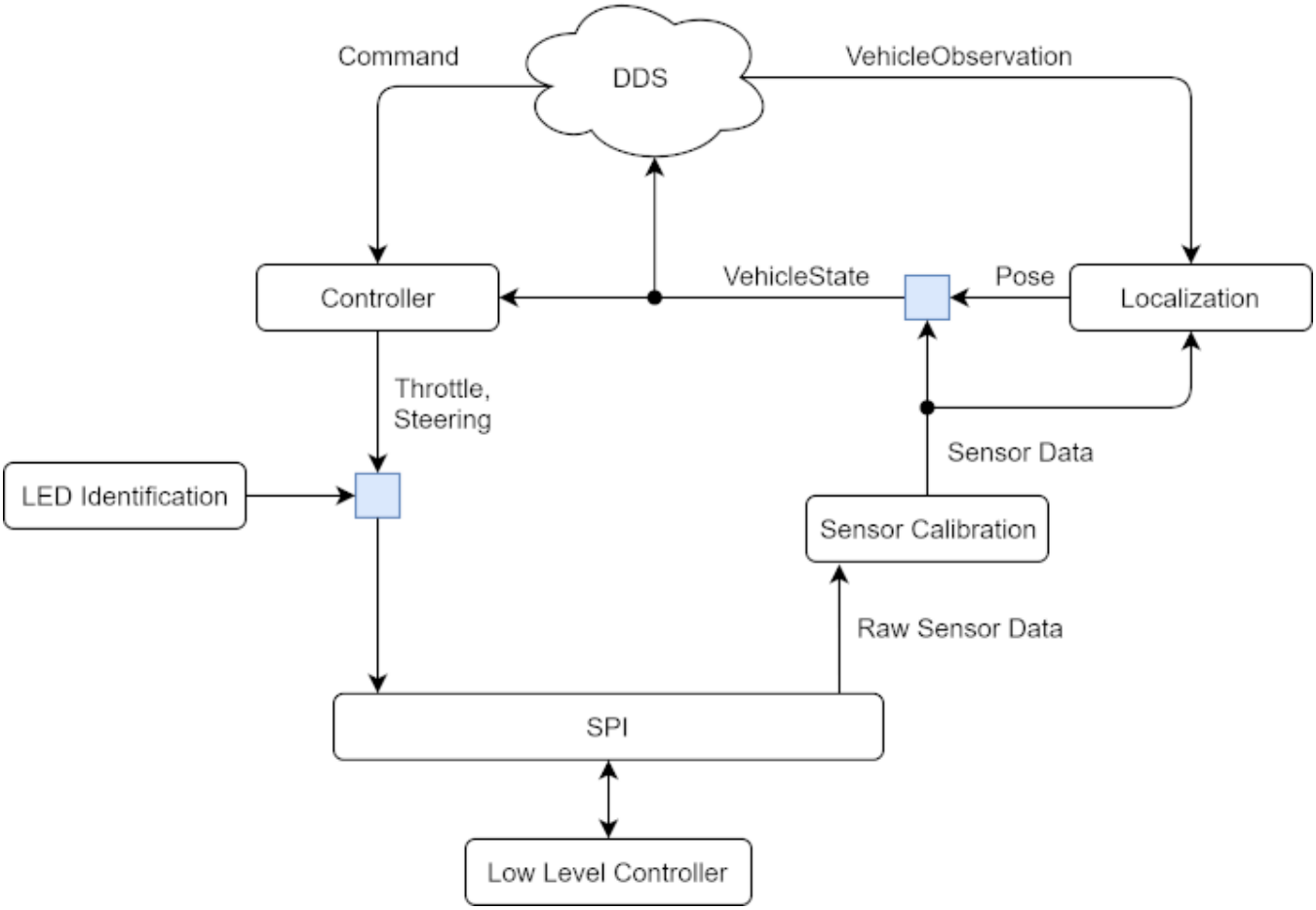
The Mid Level Controller is based on the Raspberry Pi Zero W. It is part of the vehicle on-board electronics.

Its purposes are:

- Provide a wireless network connection for the vehicle.
- Provide a synchronized clock on the vehicle, via the Linux NTP implementation.
- Provide a control system for accurate trajectory following of the vehicle. The control system runs locally on the vehicle to eliminate the network latency from the control loop.
- Send vehicle state information and receive trajectory commands via the network.

The setup and software build for the Raspberry Pi is explained under Raspberry Pi Setup.

The following data-flow diagram shows the MLC architecture.



Trajectory Controller

A model predictive controller runs on-board on the Raspberry Pi. It can steer the vehicle to follow a given trajectory.

To see how to use it to control a vehicle, go to Vehicle Commands.

To see how it works, go to https://github.com/embedded-software-laboratory/cpm-lab/blob/master/matlab_scripts/vehicle_dynamics_identification_and_mpc/documentation/main.pdf

Localization

The localization combines the IPS vehicle observation with local vehicle sensor data to determine the vehicle pose. If the IPS vehicle observation is delayed or temporarily unavailable, the localization can continue to give accurate poses through dead reckoning. However, the dead reckoning will fail when driving aggressively, i.e. with high wheel slip.

Sensor Calibration

The Low Level Controller provides its data as low-resolution integers. The LLC units are chosen to match the required value range and quantization error. The sensor calibration converts the LLC data to floating point values with SI units.

LED Identification

See Indoor Positioning System.

SPI

The MLC / Raspberry Pi is the SPI master. It exchanges data with the LLC / ATmega2560 slave. The communication is synchronous, i.e. the LLC waits for the MLC to initiate the exchange. The data exchange repeats at 50 Hz. SPI is full duplex, the input and output data are exchanged simultaneously.

In each time step (20 ms), two data packets are transmitted between the MLC and LLC, one in each direction. The data integrity is checked on both sides using the cyclic redundancy check. The LLC will acknowledge receiving a correct message from the MLC through a status bit. The MLC will repeat the transmission within a time step, until a correct message from the LLC is received and the LCC's acknowledge bit is set. Thus, every transmission will be made at least twice. The MLC signals the end of the (re)transmission phase to the LLC through a separate "chip select" / "slave select" line.

https://github.com/embedded-software-laboratory/cpm-lab/blob/master/vehicle_raspberry_firmware/src/spi.c

https://github.com/embedded-software-laboratory/cpm-lab/blob/master/vehicle_atmega2560_firmware/vehicle_atmega2560_firmware/spi.c

https://github.com/embedded-software-laboratory/cpm-lab/blob/master/vehicle_atmega2560_firmware/vehicle_atmega2560_firmware/spi_packets.h

Simulation Mode

The MLC contains a simulation of the vehicle, which replaces the LLC and SPI. The switch to the simulation mode is made at compile time. The MLC build script creates multiple variants of the MLC software, for simulation and for normal operation.

Static ARP table

In some cases ARP is unsuccessful and this leads to high latencies. To mitigate this a static ARP table is provided for the vehicles at `software/mid_level_controller/package/arp_table`. See also Vehicle Setup.