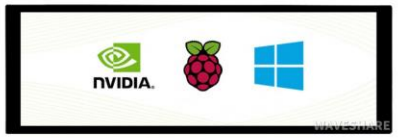
DES - Instrument Cluster

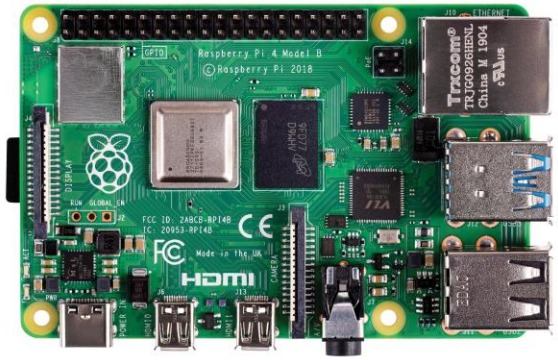
TEAM 07 – MINSEO PARK, CHANGSEOK OH

Table of contents

Specification - Hardware

- 7.9 inch DSI LCD
- Raspberry pi 4B
- Arduino uno rev3
- Speed Sensor (LM393)
- CAN-BUS (FD) Shield (MCP2518FD)
- CAN-BUS Shield V2.0 (MCP2515)
- Piracer Kit
- Shanwan Gamepad (PS3)





Specification – Software

Tools

- Qt6.4.2, Arduino IDE, CMake 3.28.3
- Rasbian OS (bookworm), Qt Design Studio 10.0.0

Technology

- Dbus (Dbus-daemon 1.14.10), I2C, SPI, CAN

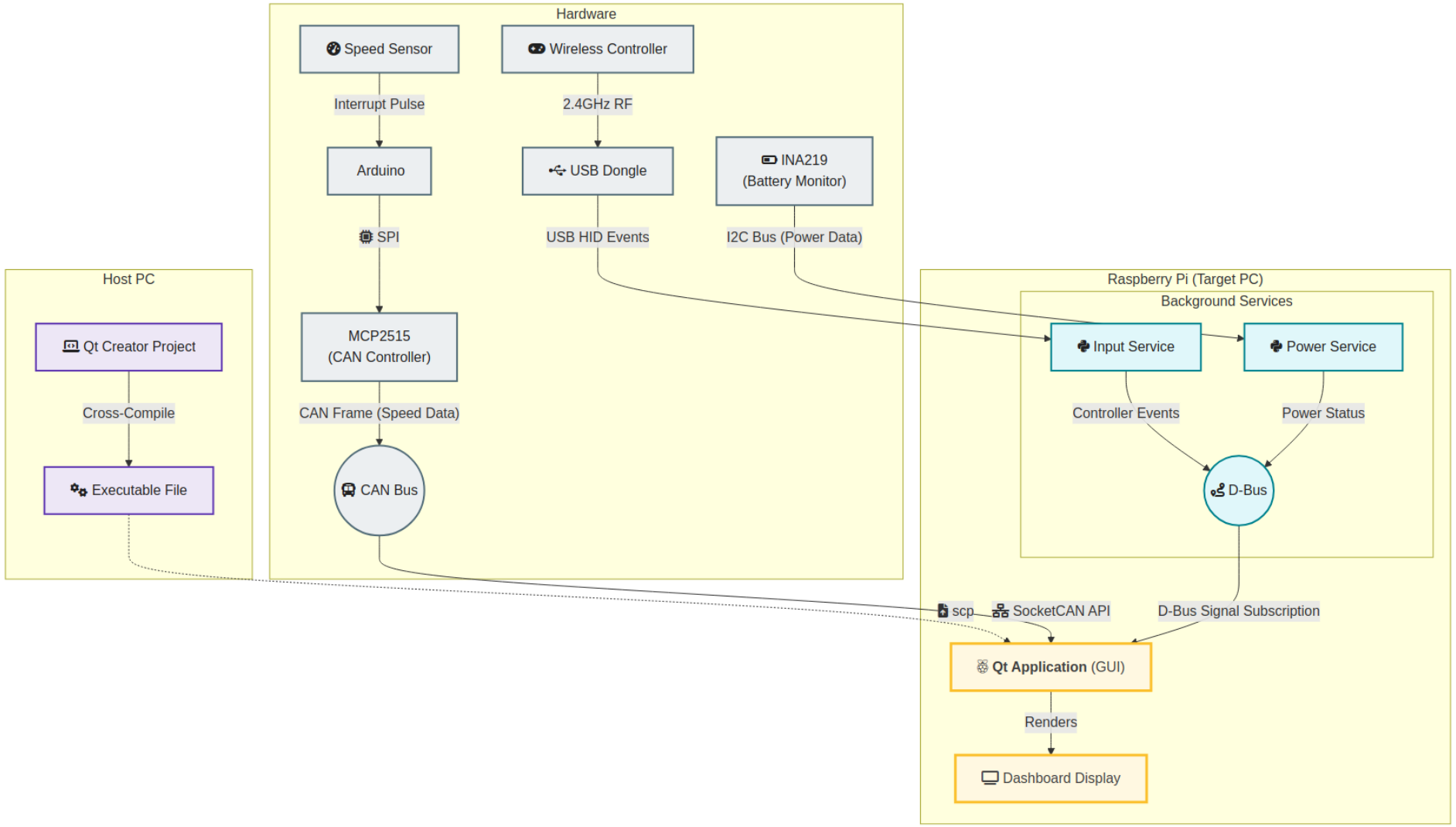
Languages

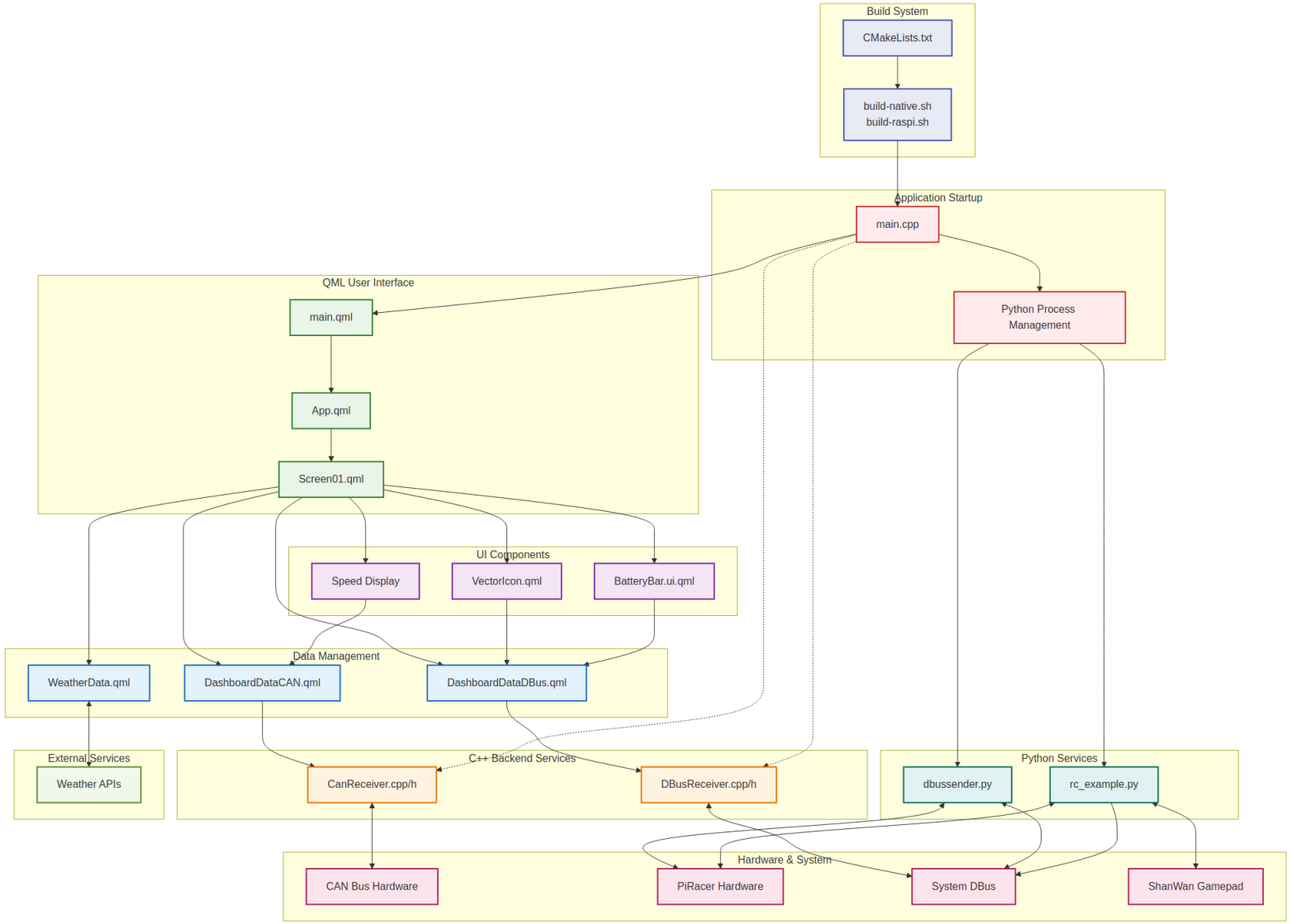
- Python 3.11.2, C++ 17, QML

Features & Setup

- Driving & Steering
- Speed calculation & rpm
- Battery capacity & charging indicator
- Location tracking with IP
- Weather and temperature indicator
- Left and Right turn signal

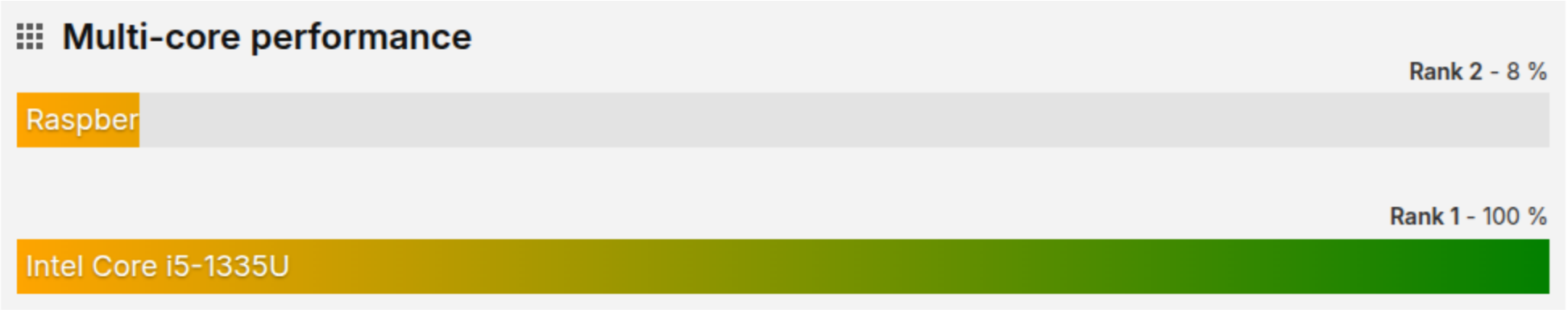

2:23 PM • Wolfsburg, NI
16°C
0
cm/s
0 rpm
84%

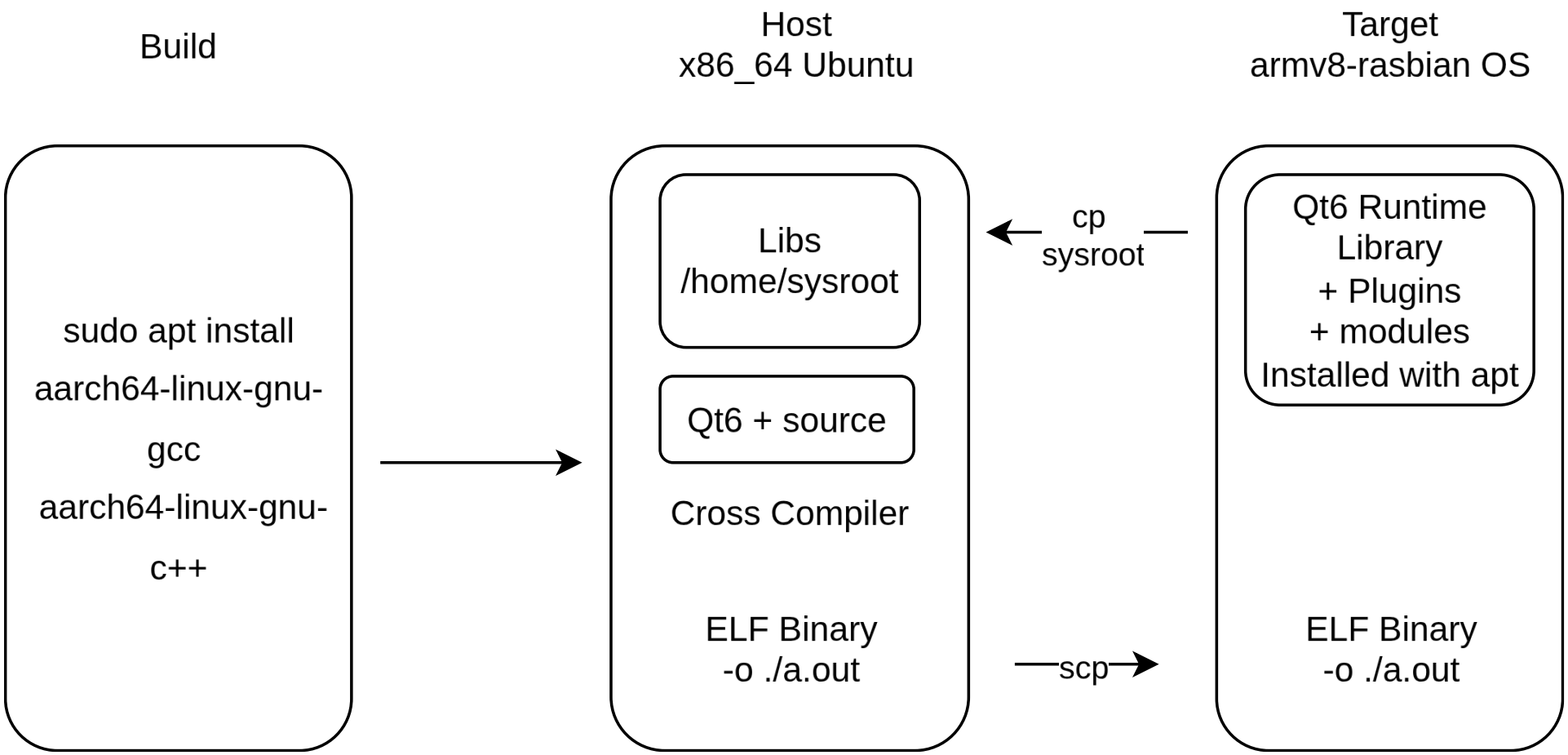

Hardware
Speed Sensor
Wireless Controller
Interrupt Pulse
2.4GHz RF
Arduino
USB Dongle
INA219
(Battery Monitor)
SPI
USB HID Events
I2C Bus (Power Data)
Host PC
Qt Creator Project
Cross-Compile
Executable File
MCP2515
(CAN Controller)
CAN Frame (Speed Data)
CAN Bus
Raspberry Pi (Target PC)
Background Services
Input Service
Power Service
Controller Events
Power Status
D-Bus
scp
SocketCAN API
D-Bus Signal Subscription
Qt Application (GUI)
Renders
Dashboard Display


Build System
CMakeLists.txt
build-native.sh
build-raspi.sh
Application Startup
main.cpp
Python Process Management
QML User Interface
main.qml
App.qml
Screen01.qml
UI Components
Speed Display
VectorIcon.qml
BatteryBar.ui.qml
Data Management
WeatherData.qml
DashboardDataCAN.qml
DashboardDataDBus.qml
External Services
Weather APIs
C++ Backend Services
CanReceiver.cpp/h
DBusReceiver.cpp/h
Python Services
dbussender.py
rc_example.py
Hardware & System
CAN Bus Hardware
PiRacer Hardware
System DBus
ShanWan Gamepad

Cross Compile

- Cross Compile is needed when
  - Architecture
  - OS / ABI
  - Low computing power (e.g. embedded)
- Native Compilation is feasible on Pi for small programs

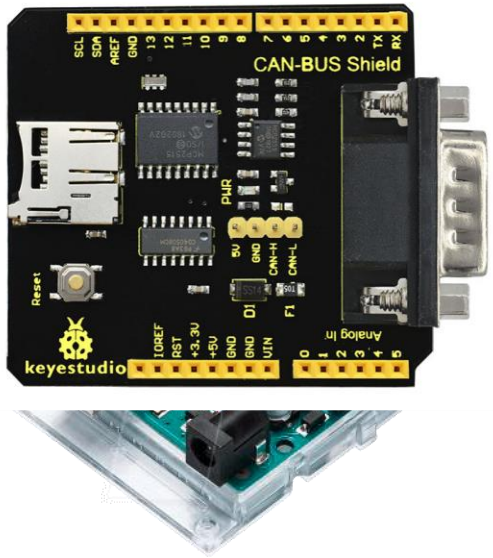

Multi-core performance
Rank 2 - 8 %
Raspber
Rank 1 - 100 %
Intel Core i5-1335U

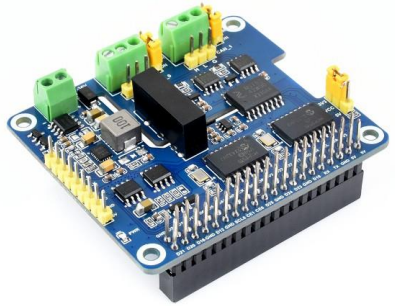

Build
Host
x86_64 Ubuntu
Target
armv8-rasbian OS
sudo apt install
aarch64-linux-gnu-
gcc
aarch64-linux-gnu-
c++
Libs
/home/sysroot
Qt6 + source
Cross Compiler
ELF Binary
-o ./a.out
cp
sysroot
scp
Qt6 Runtime
Library
+ Plugins
+ modules
Installed with apt
ELF Binary
-o ./a.out

CAN

Why used twisted cable? : to cancel out external electrical noise and interference

Twisted Pair Cable





CAN

CAN message format

Speed (High byte)	Speed (Low byte)	Speed (Fractional Part)	RPM (High byte)	RPM (Low Byte)	RPM (Fractional Part)		

Arduino

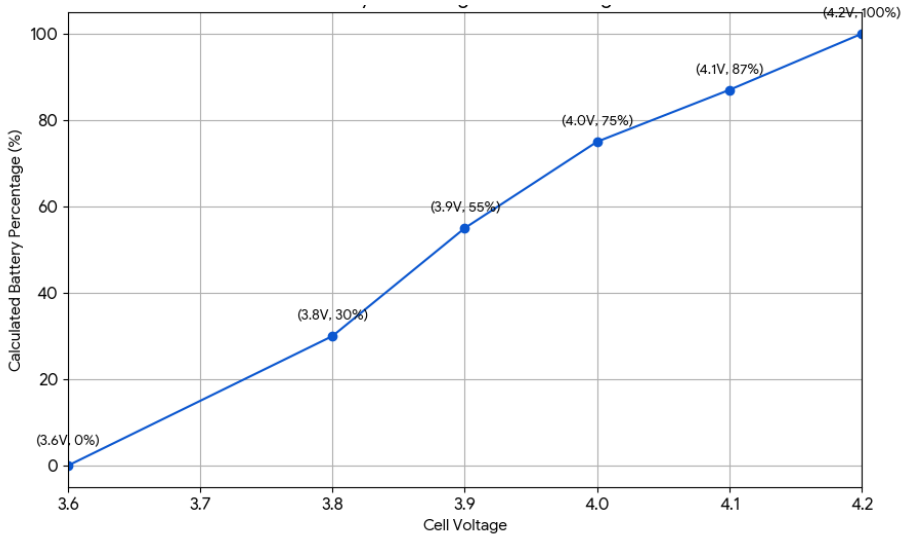
- Calculate Speed, RPM data
- Split speed, RPM datas
- Send CAN messages
- Interval for sending CAN messages : 0.3sec

Raspberry Pi

- CAN connection setup / verification
- Receiving CAN message
- Assigning speed, RPM datas
- Emitting signals to update the UI

Battery

- Used a method from PiRacerStandard library to read the raw battery voltage over I2C bus.
- Used **a sliding window algorithm** and **a low-pass filter** to prevent sudden voltage drops
- Used linear interpolation to calculate battery percentage.
- Charging state : if current is 100mA or more is detected



D-Bus

- What is D-Bus?

  : Standardized IPC system that allows multiple processes on Linux OS to exchange messages with one another.

- Why used Session Bus?

  - Session Bus : communication between applications within a single user's login session

  - System Bus : system-wide communication channel used to interact with the entire OS

  Instrument Cluster is an application that operates within a single user's login session, not a system-wide service!

- What part did it used?

  : Battery percentage, Turn indicator signal

D-Bus

- Json Format

  : {

    "battery_capacity" : float type

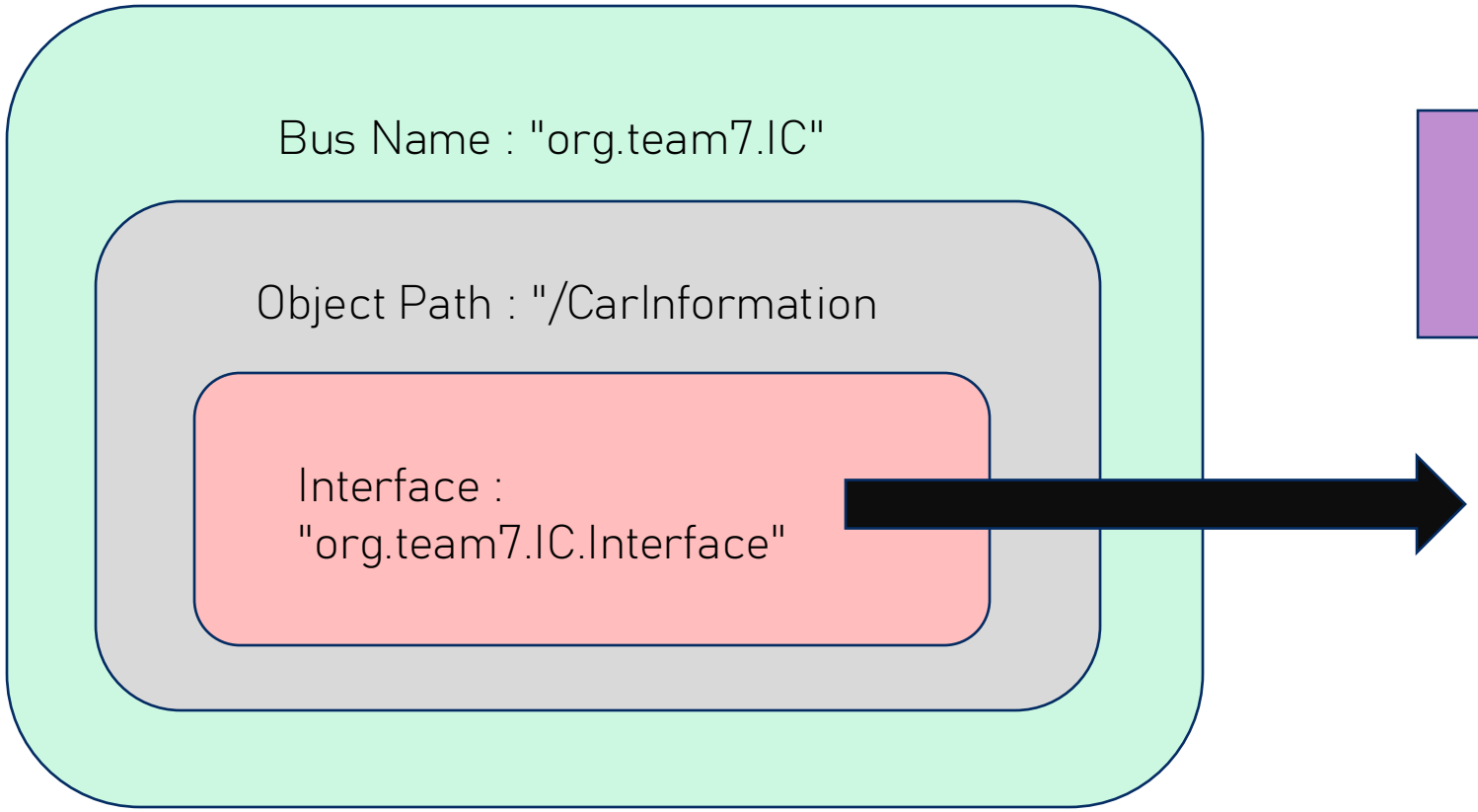
    "charging_current" : float type

    "left_turn_signal" : Bool type

    "right_turn_signal" : Bool type

  }

D-Bus Sending Structure



Signal-based push model for D-Bus

Methods :
SetBattery, setCurrent, setTurnSignals

Signal : DataReceived

Q & A

Thank you