SEA:ME

DES02: Instrument Cluster

- Build your own car -


Kwanho Kim
@KKWANH


Ogura Shuta
@Shuta-Syd


Kian Warias
@kianwasabi

2023-08, @SEA-ME-DES-Team01

Index

#1 

About the Project

- Links
- Tech Stacks

#2 

Project Management

- Github & Micromanagement
- Architecture

#3 

Technical Implements

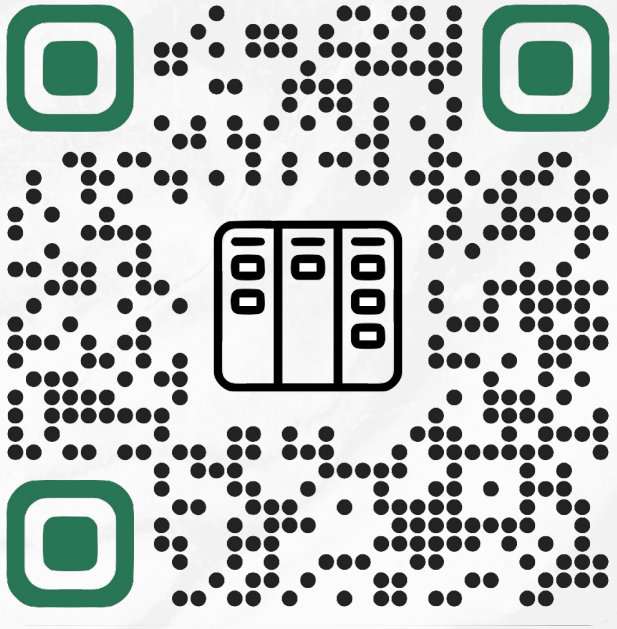
- Data Processing
- Can Communication
- TCP/IP Socket
- D-Bus
- Dashboard Design (QML)
- Startup Routine (Automation)
- Testing Exploration

#4 

Conclusion

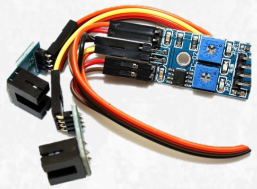
- DEMO
- Q&A
- Thanks

Github Repository



Kanban Board

#1-2 Tech Stacks



Sensors

Linux
(Rasbian)

Multi-Process

Piracer-Py

Calculations

CAN

Can Communication

Raspberry
Pi

D-BUS

D-BUS

Socket

PiRacer

C++

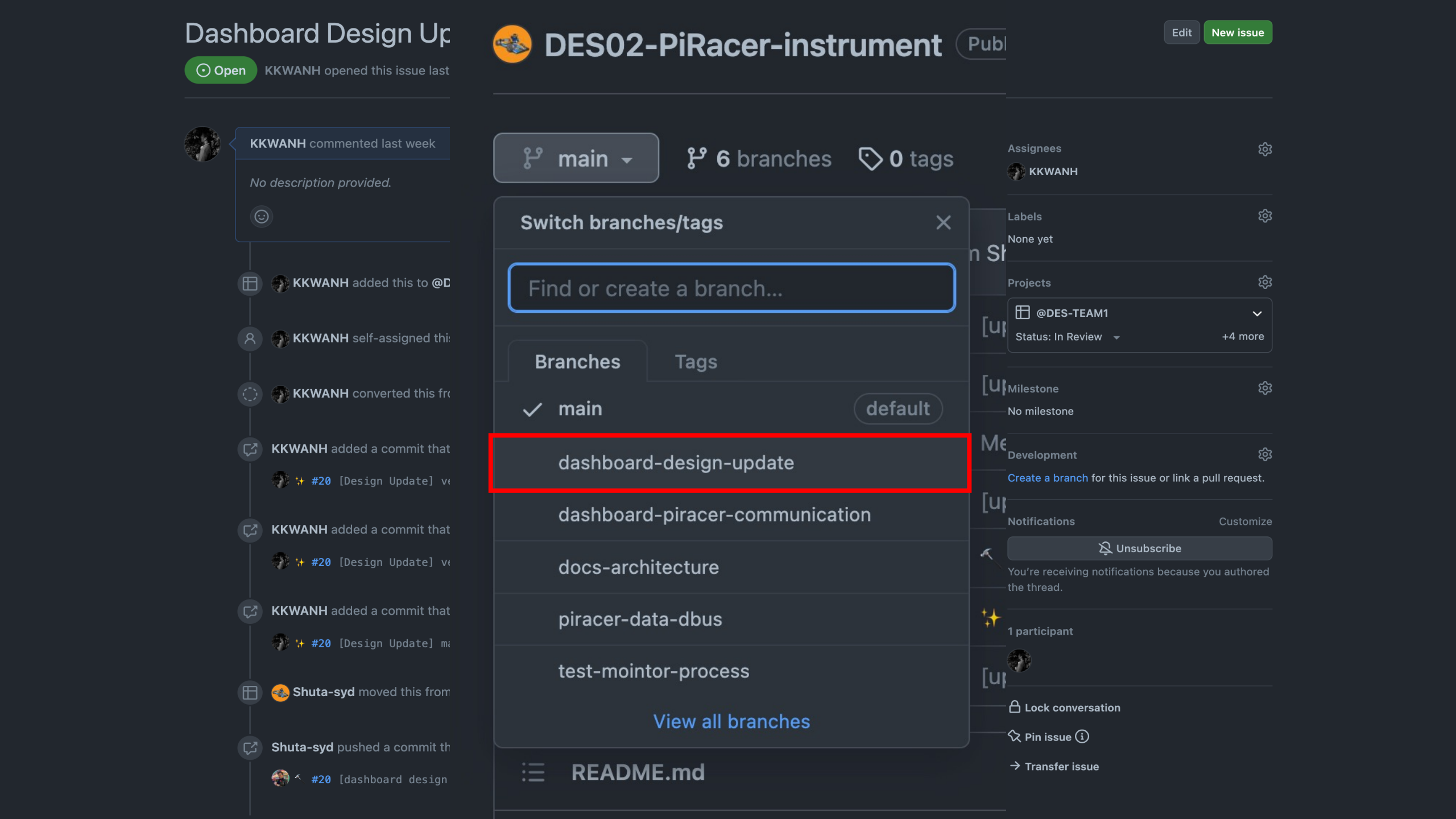
C++

Arduino

Qt

QT/QML Dashboard

Figma


Dashboard Design Up
Open
KKWANH opened this issue last
DES02-PiRacer-instrument
Edit
New issue
KKWANH commented last week
No description provided.
main
6 branches
0 tags
Switch branches/tags
Find or create a branch...
Branches
Tags
main
default
dashboard-design-update
dashboard-piracer-communication
docs-architecture
piracer-data-dbus
test-mointor-process
View all branches
README.md
Assignees
KKWANH
Labels
None yet
Projects
@DES-TEAM1
Status: In Review
+4 more
Milestone
No milestone
Development
Create a branch for this issue or link a pull request.
Notifications
Customize
Unsubscribe
You're receiving notifications because you authored the thread.
1 participant
Lock conversation
Pin issue
Transfer issue
KKWANH added this to @D
KKWANH self-assigned this
KKWANH converted this fr
KKWANH added a commit that
#20 [Design Update] v
KKWANH added a commit that
#20 [Design Update] v
KKWANH added a commit that
#20 [Design Update] ma
Shuta-syd moved this from
Shuta-syd pushed a commit th
#20 [dashboard design

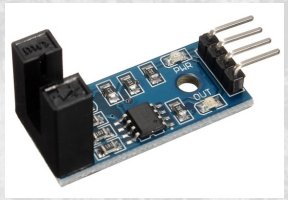
Repository structure

```
├ [branch] main
    ├ docs/
    ├ examples/
    ├ README.md
```

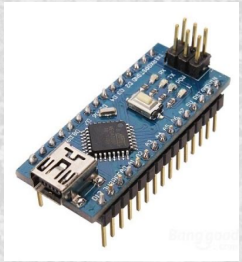
```
├ [branch] dbus-version
    ├ app/
        ├ dashboard/
            ├ asset/
            ├ ...
            ├ build.sh
        ├ piracer_py/
            ├ dbus/
            ├ system/
            ├ ...
            ├ startup_py.sh
            ├ startup_monitor.sh
        ├ startup.sh
    ├ can_modules/
        ├ speedsensor/
            ├ rpm_calculator.ino
        ├ setup_can.sh
    ├ startup.sh
```

```
├ [branch] tcp-version
    ├ app/
        ├ dashboard/
        ├ piracer_py/
            ├ piracer/
            ├ process/
            ├ startup_main.sh
            ├ startup_monitor.sh
    ├ can_modules/
    ├ startup.sh
```

#2-2 Architecture

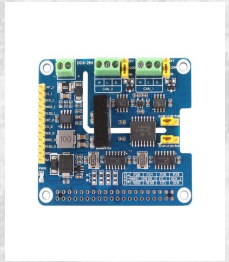


Speed Sensor

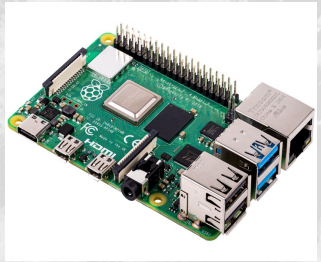


Arduino Nano

MCP2515

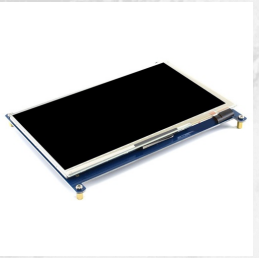


2CH CAN FD HAT



Raspberry Pi

i2C



Display

CAN-BUS

Qt

D-BUS Client or TCP/IP Client

QTimer

QML

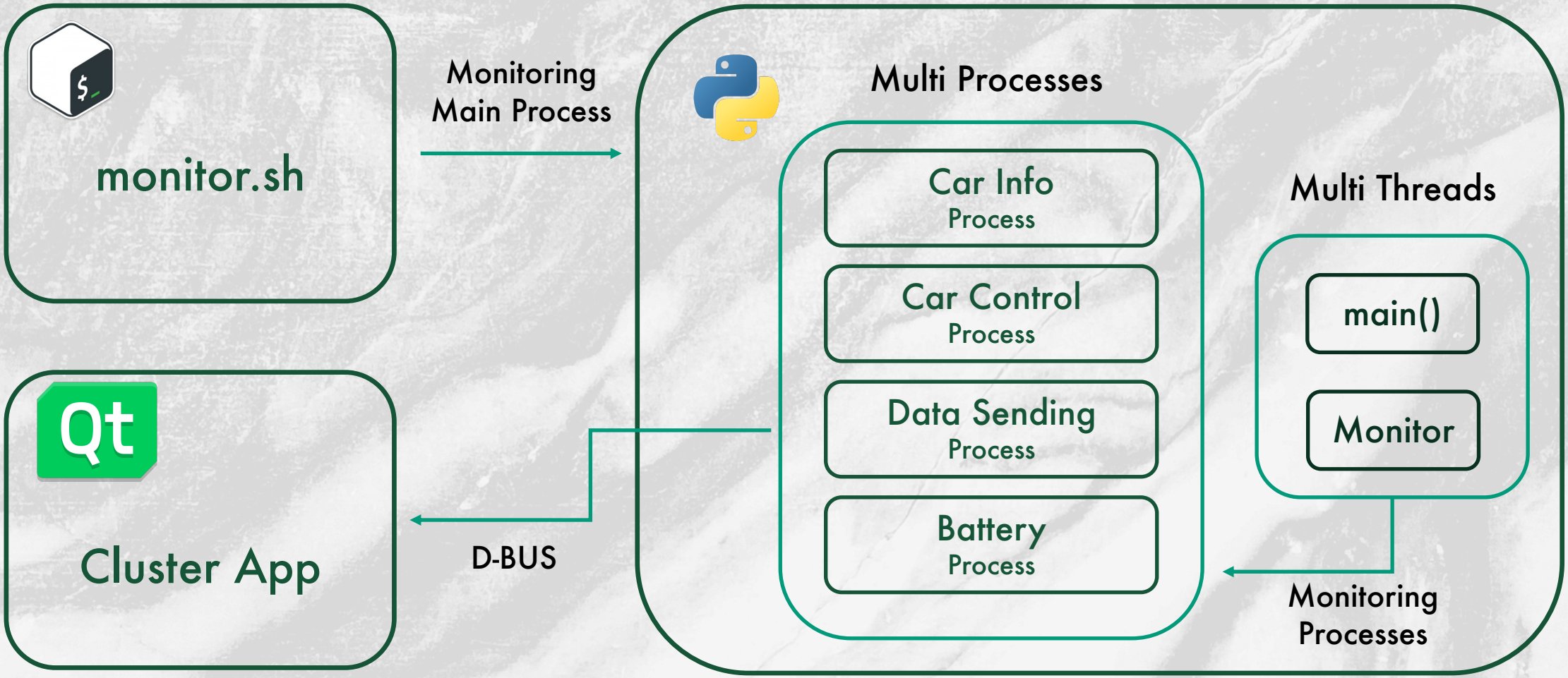
TCP socket or DBUS

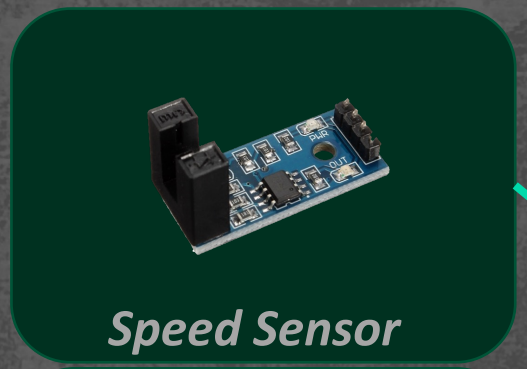
D-BUS Server or TCP/IP Server

Monitor Thread

Read CAN data

#2-2 Architecture





Speed Sensor

PiRacer

CAN

I2C

Steering in %
Throttle in %
Battery Voltage in V
Battery Current in mA
Battery Consumption in W
Wheel Rotation in 1/min
……

$$\frac{1}{n}\sum_{i=1}^{n} x(t-i)$$

Moving Average Filter

$$y^{(n)}(t) = \frac{1}{n}\sum_{i=0}^{n-1} x(t-i) = \frac{1}{n}\left(x(t) + x(t-1) + \ldots + x\left(t-(n-1)\right)\right)$$

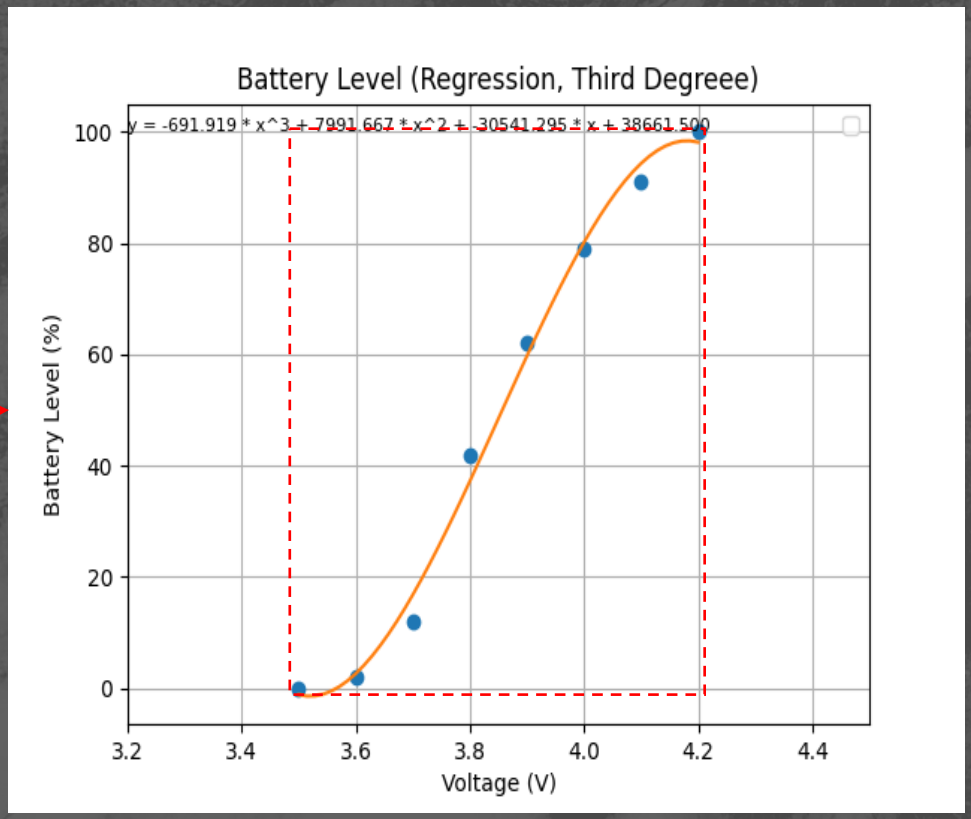
Choose window size (n)

- Smoothing Goal
- Signal-Noise-Ratio
- Data Frequency
- Experimental & Optimization

Example - Battery Level Calculation

Estimated remaining capacity			
Voltage	Sanyo 18650 2600mAh (Red)	Panasonic CGR18650CH 2250mAh	Panasonic NCR18650B 3100/3400mAh
4.2	100%	100%	100%
4.1	91%	93%	94%
4.0	79%	84%	83%
3.9	62%	75%	72%
3.8	42%	64%	59%
3.7	12%	52%	50%
3.6	2%	22%	33%
3.5	0%	9%	15%
3.4	0%	0%	6%
3.3	0%	0%	0%
3.2	0%	0%	0%
Measured 1 hour after discharge min. of 1A & 3A			

https://lygte-info.dk/info/BatteryChargePercent%20UK.html



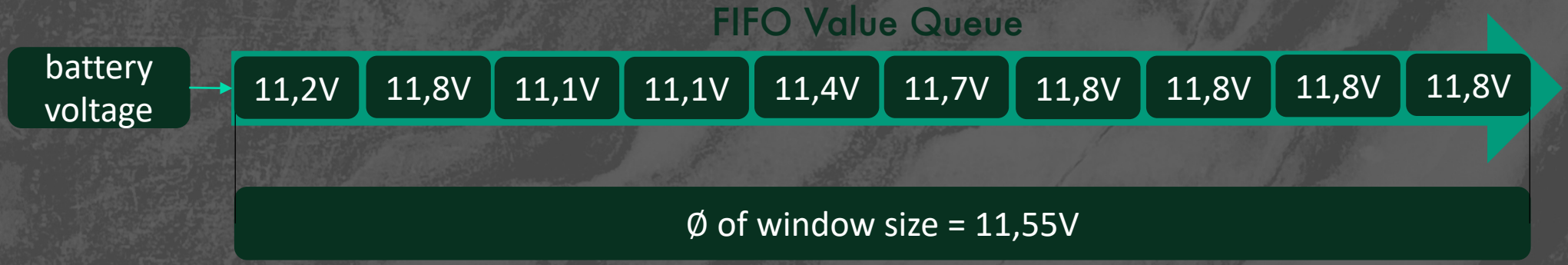
$$y = -691.919 \cdot x^3 + 7991.667 \cdot x^2 + 30541.295 \cdot x + 38661.500$$

y → Battery Level (remaining Capacity) in *%*

x → Voltage per cell in *V*

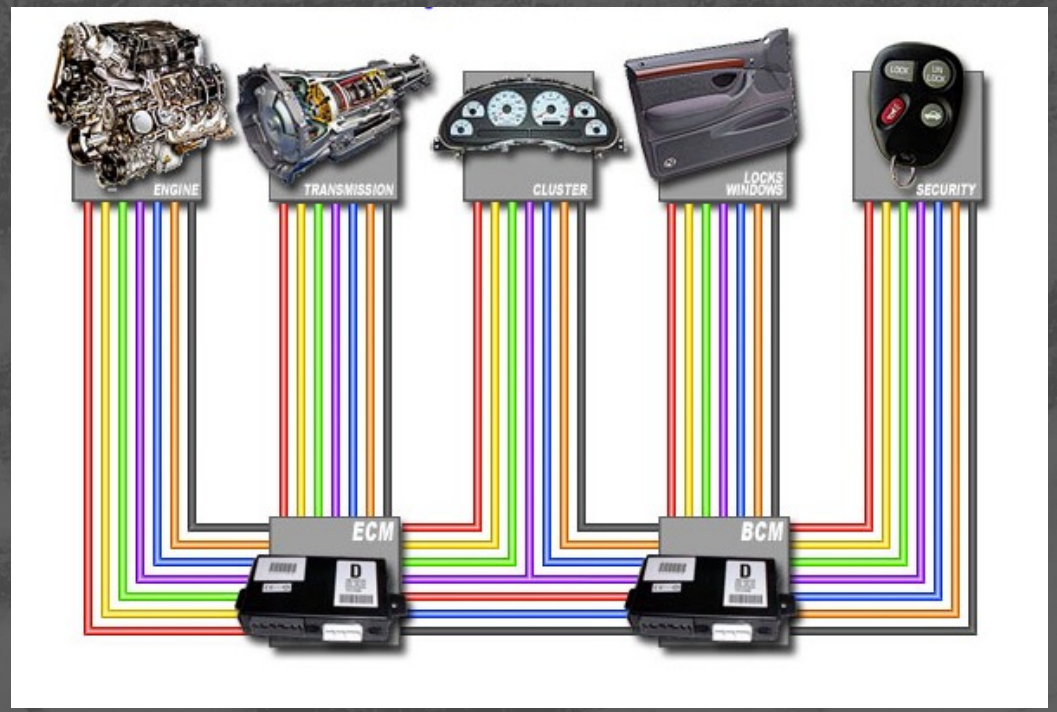
$3{,}2\ \mathrm{V} \leq x \leq 4{,}2\ V$

Moving Average - Battery Level Calculation

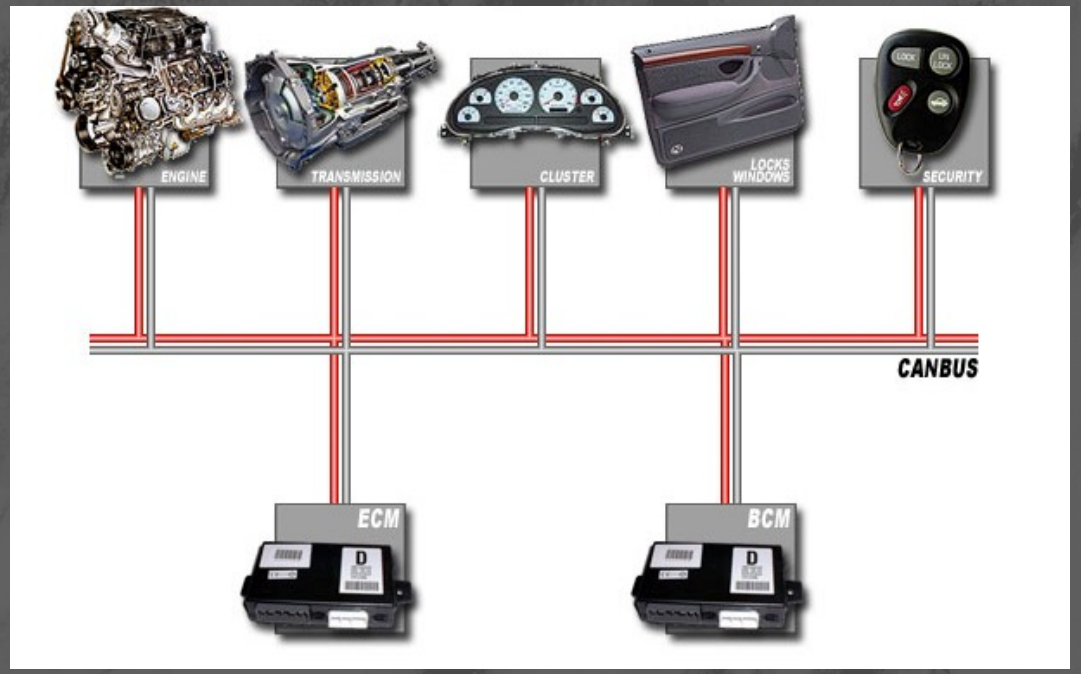


$$y(11{,}5/3) = -691.919 \cdot x^3 + 7991.667 \cdot x^2 + 30541.295 \cdot x + 38661.500 \approx 48\ \%$$

Without CAN BUS (Before)



WithCAN BUS (After)

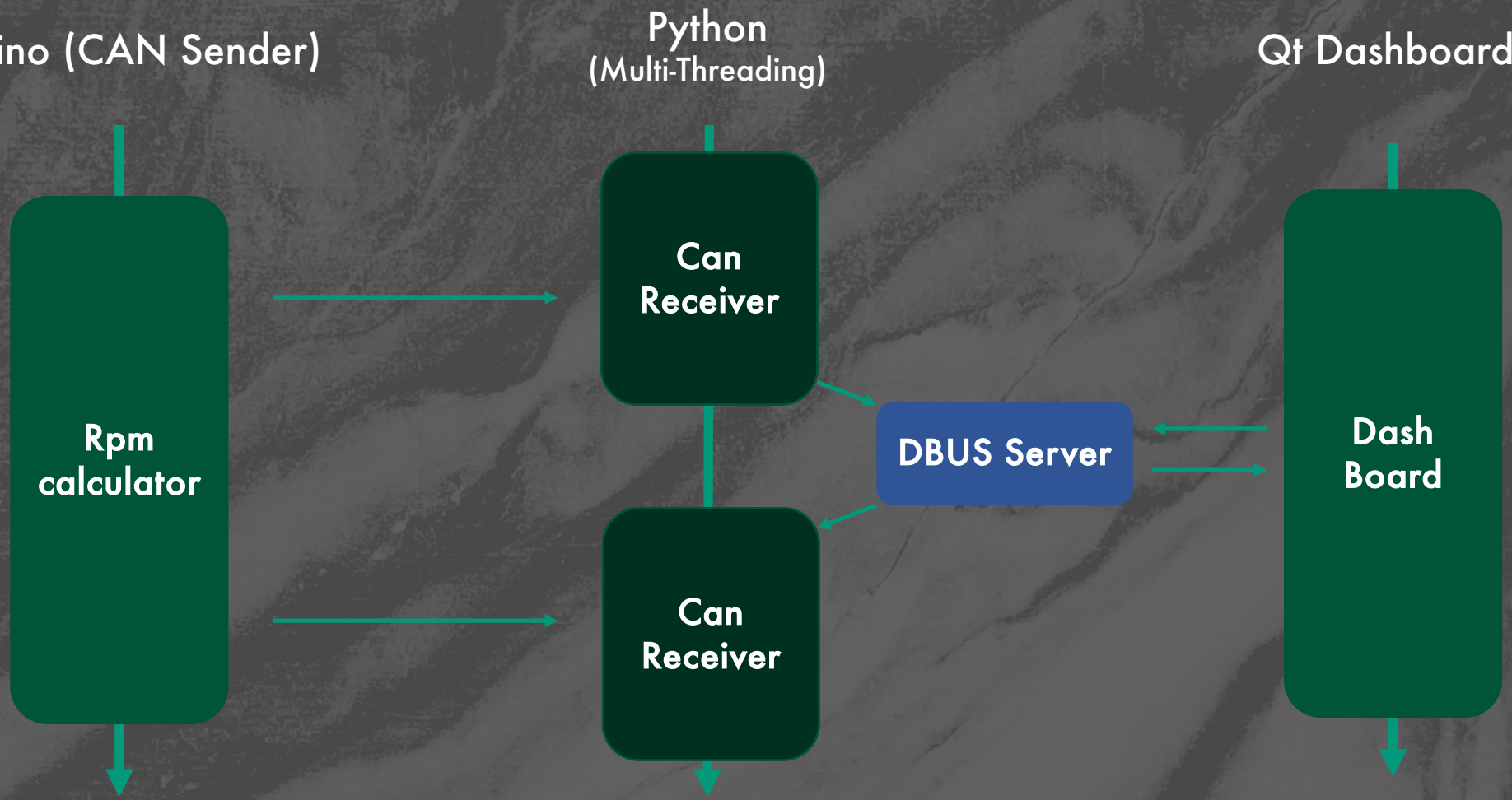


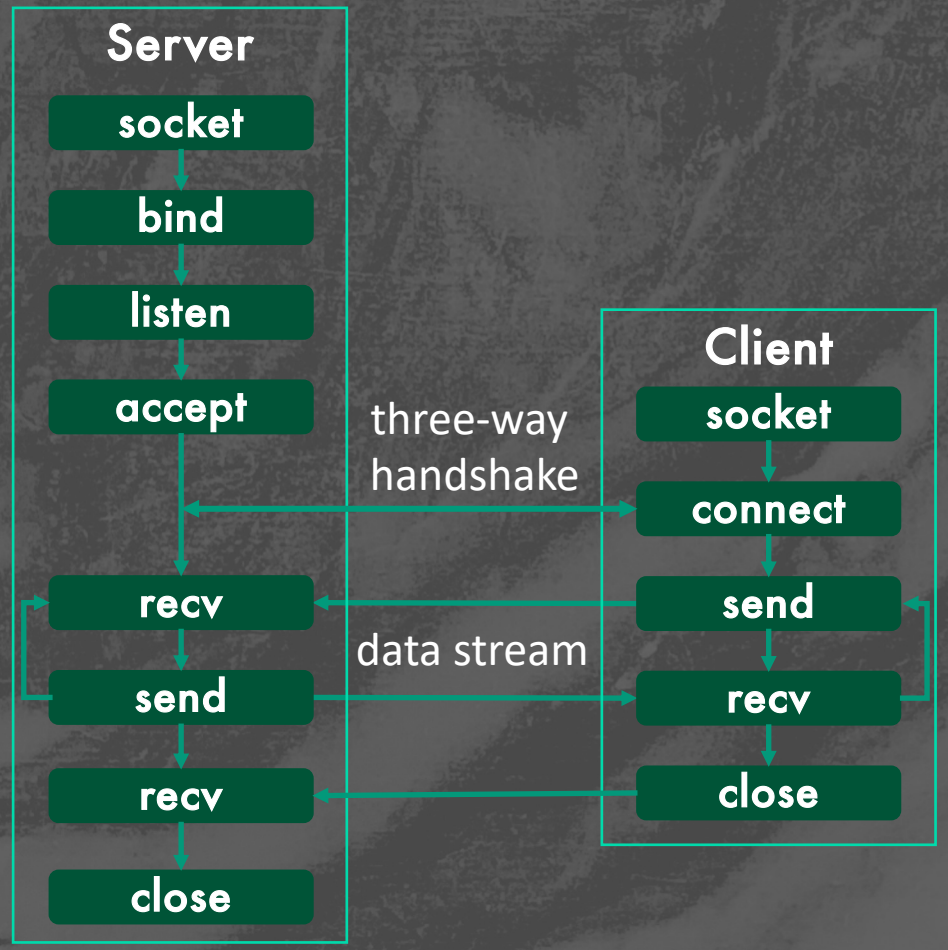
Reference: http://automotiveinterviews.blogspot.com/2015/01/controller-area-network-can.html

SEA:ME

#3-2 Can Communication

Example for dbus version

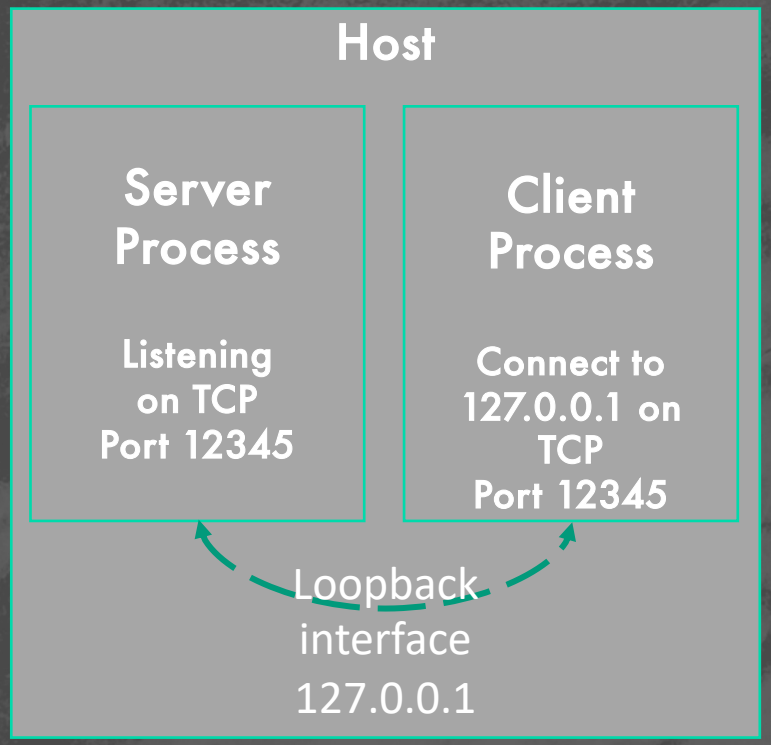




What is TCP/IP Socket Communication

- Communication between Processes over the Network
- Bidirectional Communication
- Platform-Independent API
- Scalability & Flexibility

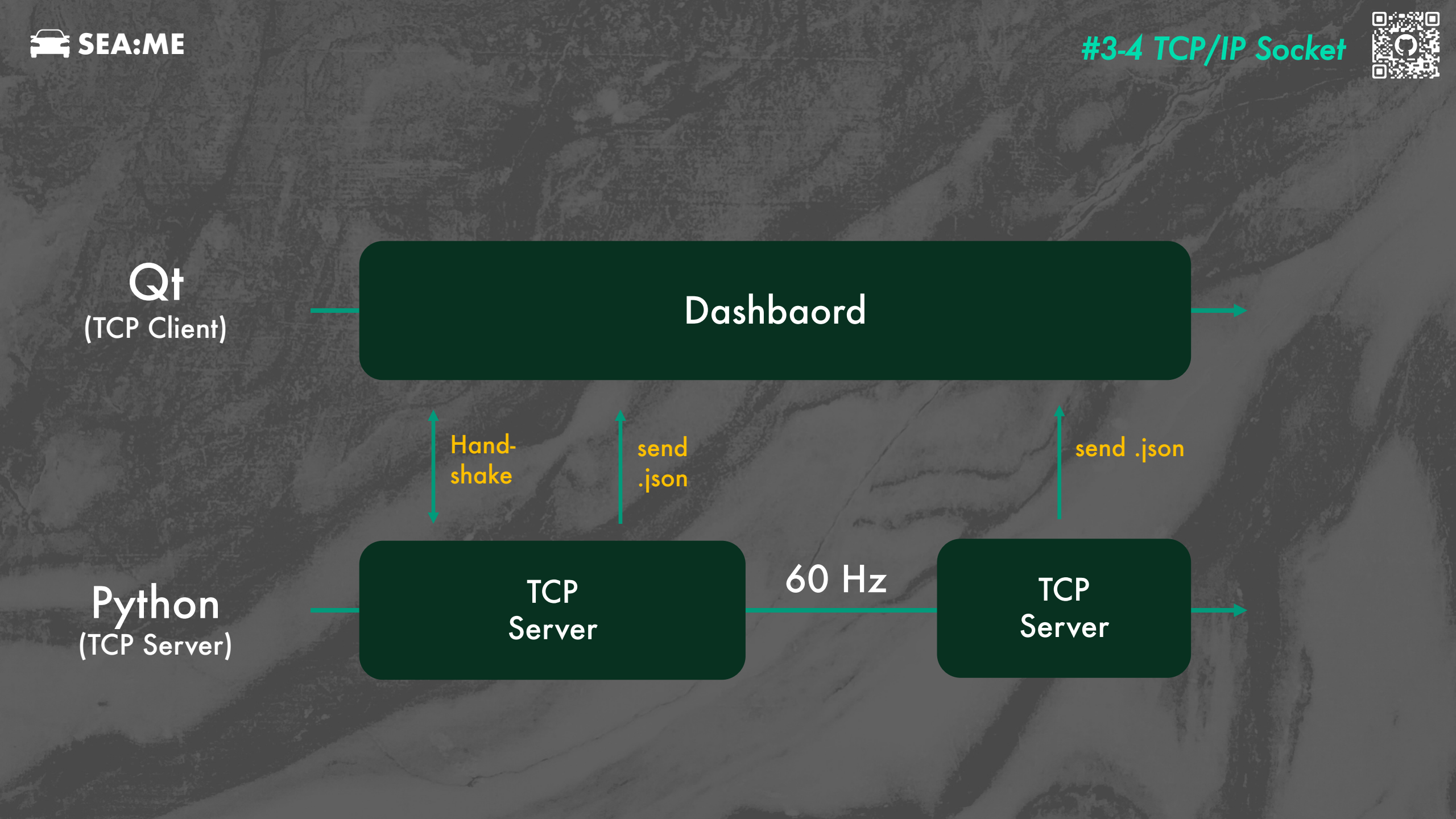
TCP/IP Socket Communication on same Host



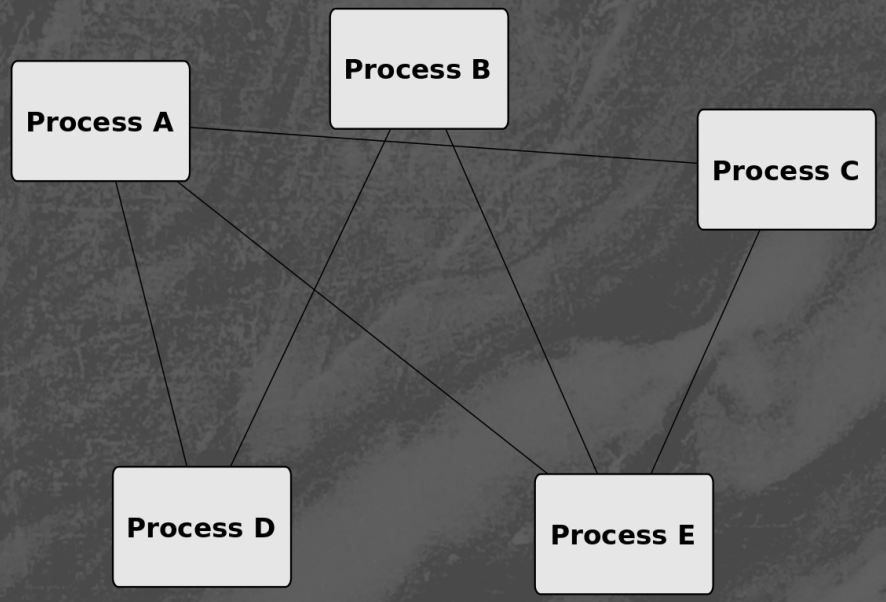
UNIX Domain Sockets vs. TCP/IP-Sockets

DBUS avoids operations like routing, therefore it is faster & lighter than TCP/IP

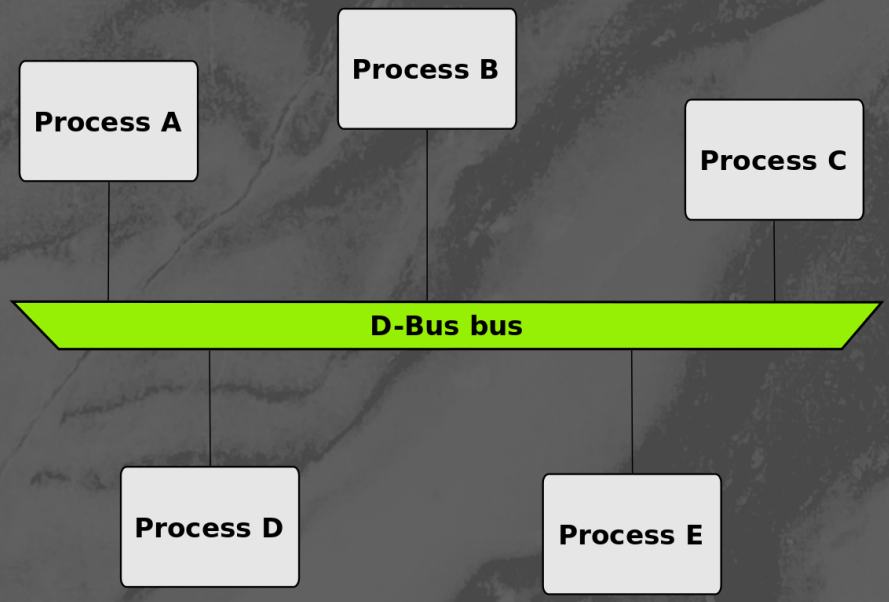
→ **For DES02: UNIX domain sockets > IP sockets**


Qt
(TCP Client)
Dashbaord
Hand-
shake
send
.json
send .json
Python
(TCP Server)
TCP
Server
60 Hz
TCP
Server

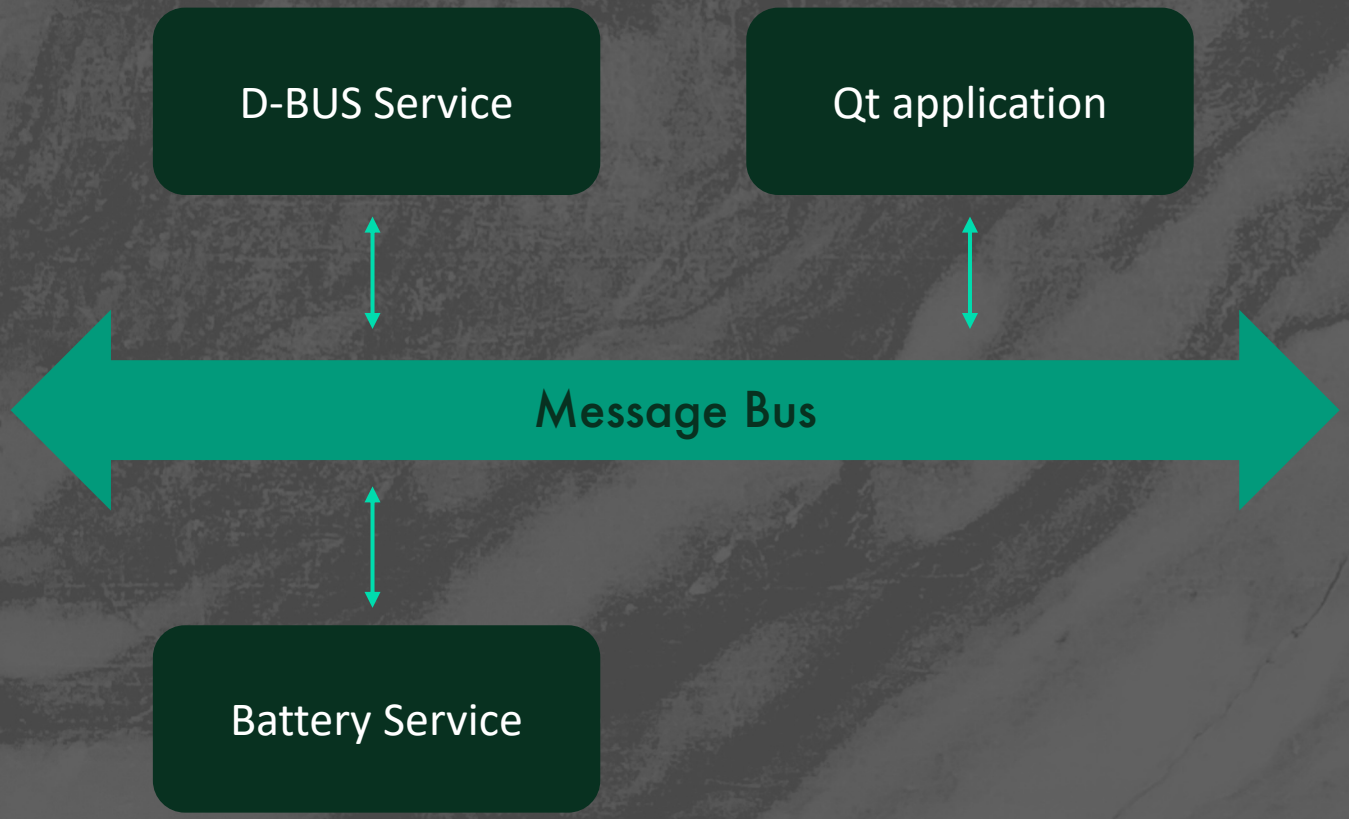
Without D-BUS



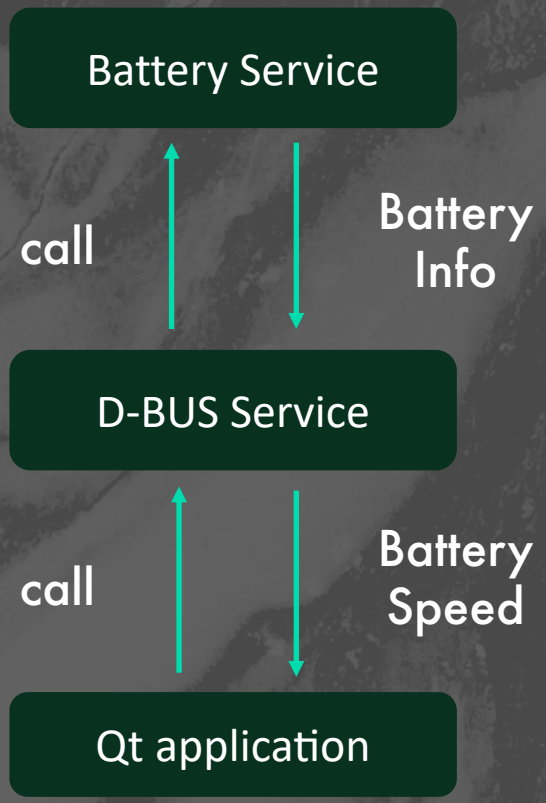
With D- BUS

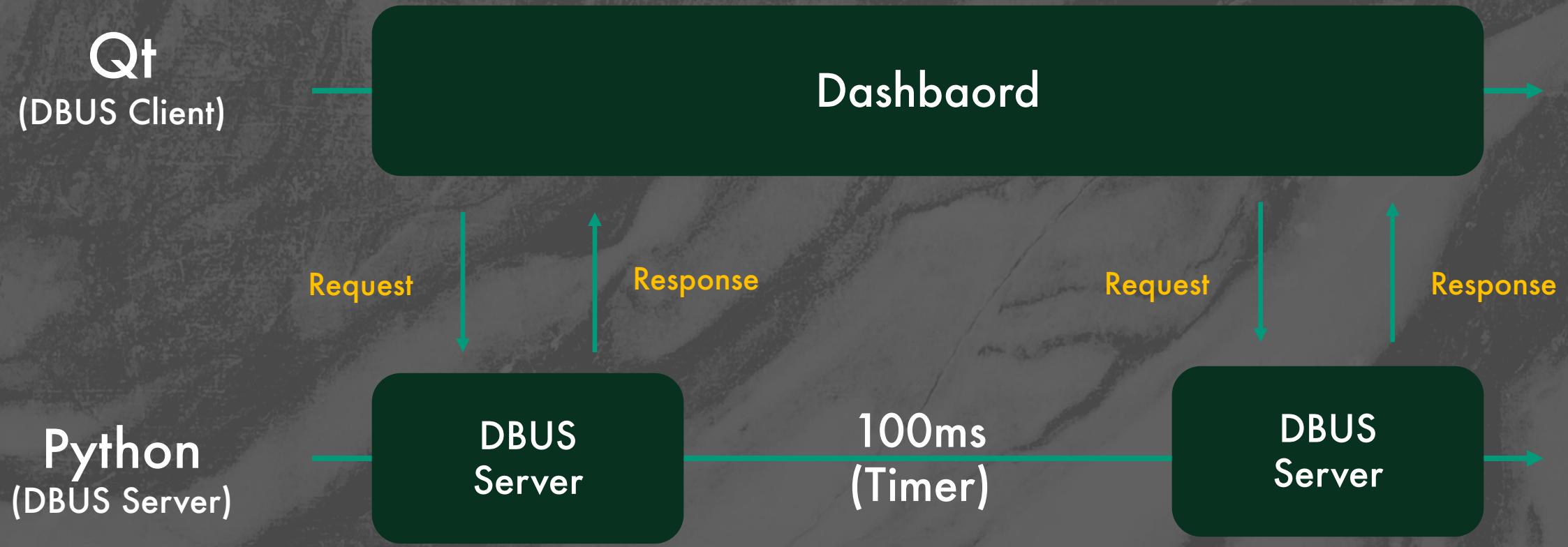


D-BUS Figure



Work Flows




Qt
(DBUS Client)
Dashbaord
Request
Response
Request
Response
Python
(DBUS Server)
DBUS
Server
100ms
(Timer)
DBUS
Server

- About direction 🔽
  - steering
    - float
      - -1 ~ 1
  - throttle
    - float
      - -100 ~ 100
  - indicator
    - int
      - 0: nothign
      - 1: left
      - 2: right
      - 3: warning light

- About battery 🔋
  - battery_voltage
    - float
    - V
  - battery_consumption
    - float
    - W
  - battery_current
    - float
    - mA
  - battery_level
    - float
    - %
  - battery_hour
    - float
    - hour

- About speed 🚤
  - speed
    - unsigned short
    - m/min
  - rpm
    - unsigned short

- etc
  - ip_address
    - string
  - time
    - string
    - hh:mm

1. UI sketch 🎨

- Design concept setting
- by Figma

2. Functional Specification ⚒️

- What data can be collected from backend?
- How are we going to preprocess the data?

3. API Specification ⚡

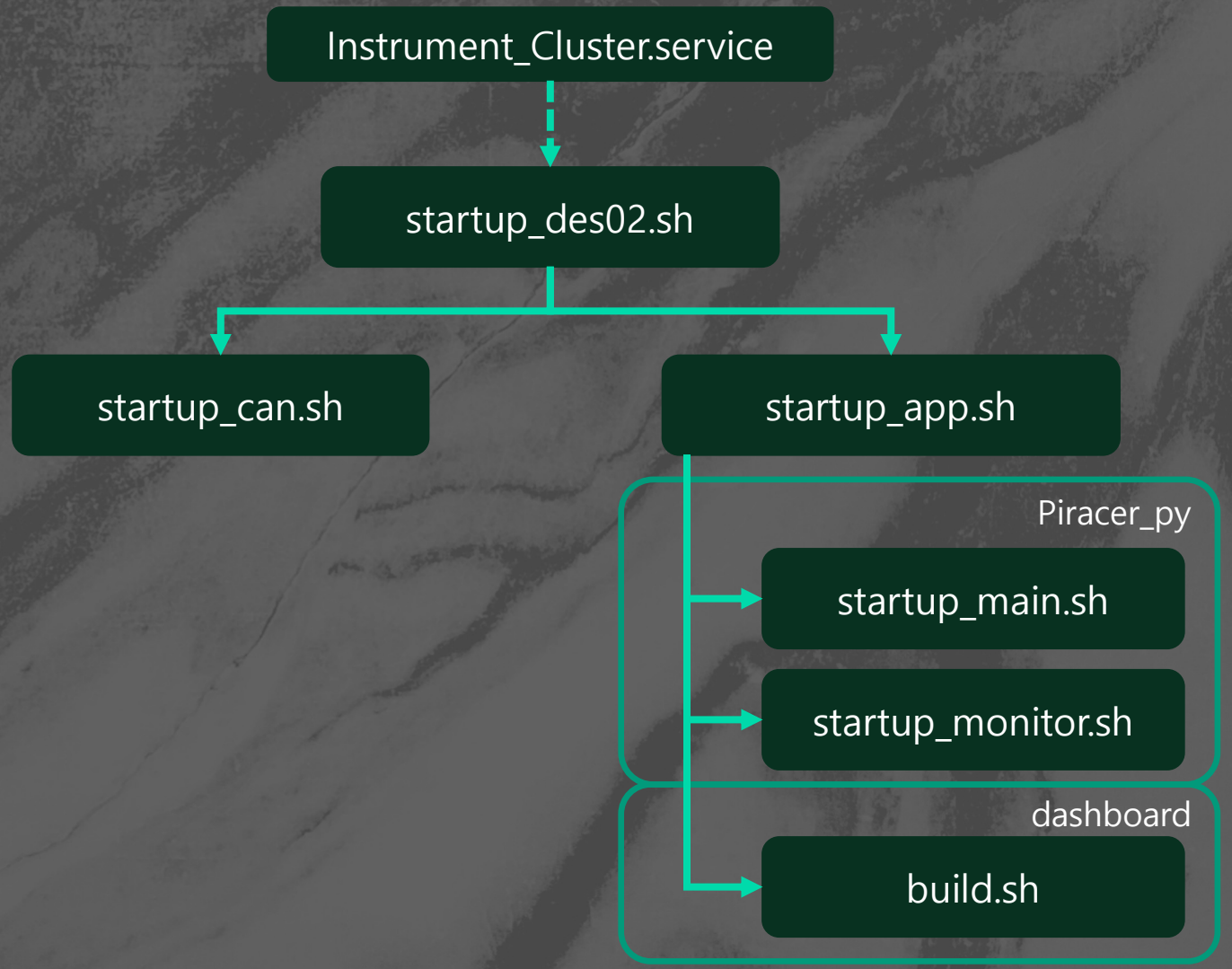
- What data will be sended?
- Which data structure are we going to use?

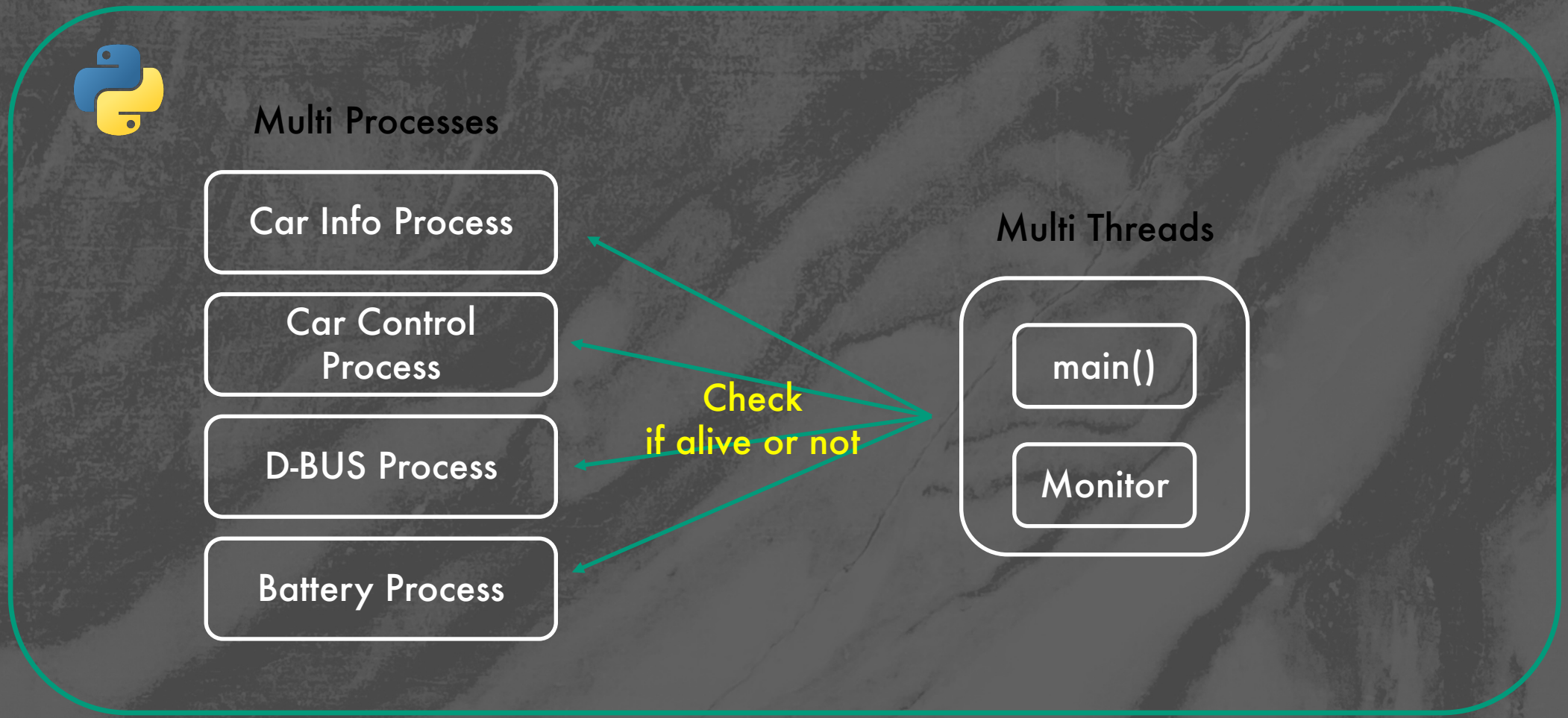
4. Detailed UI Design in Figma 🎇

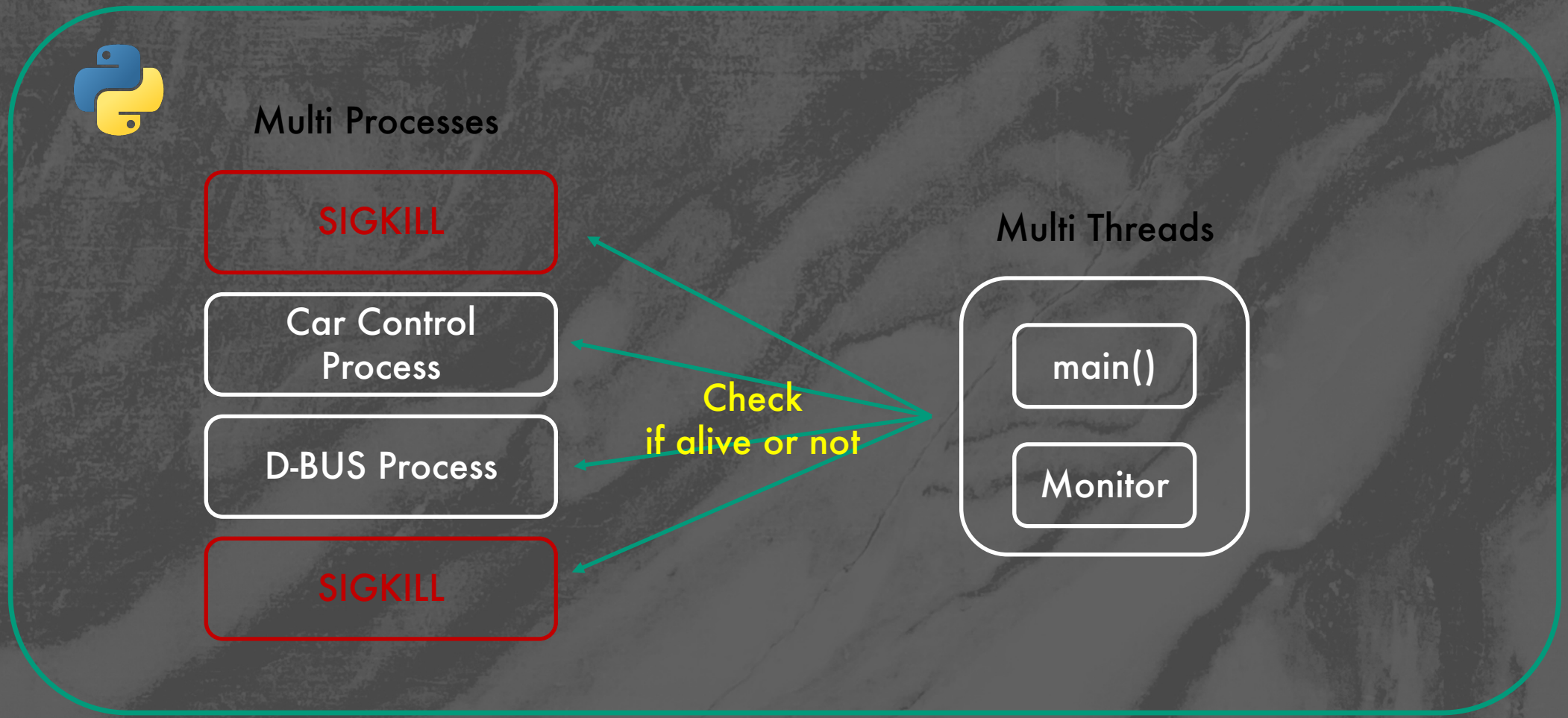
5. Frontend Re-implement in QT&QML 💻

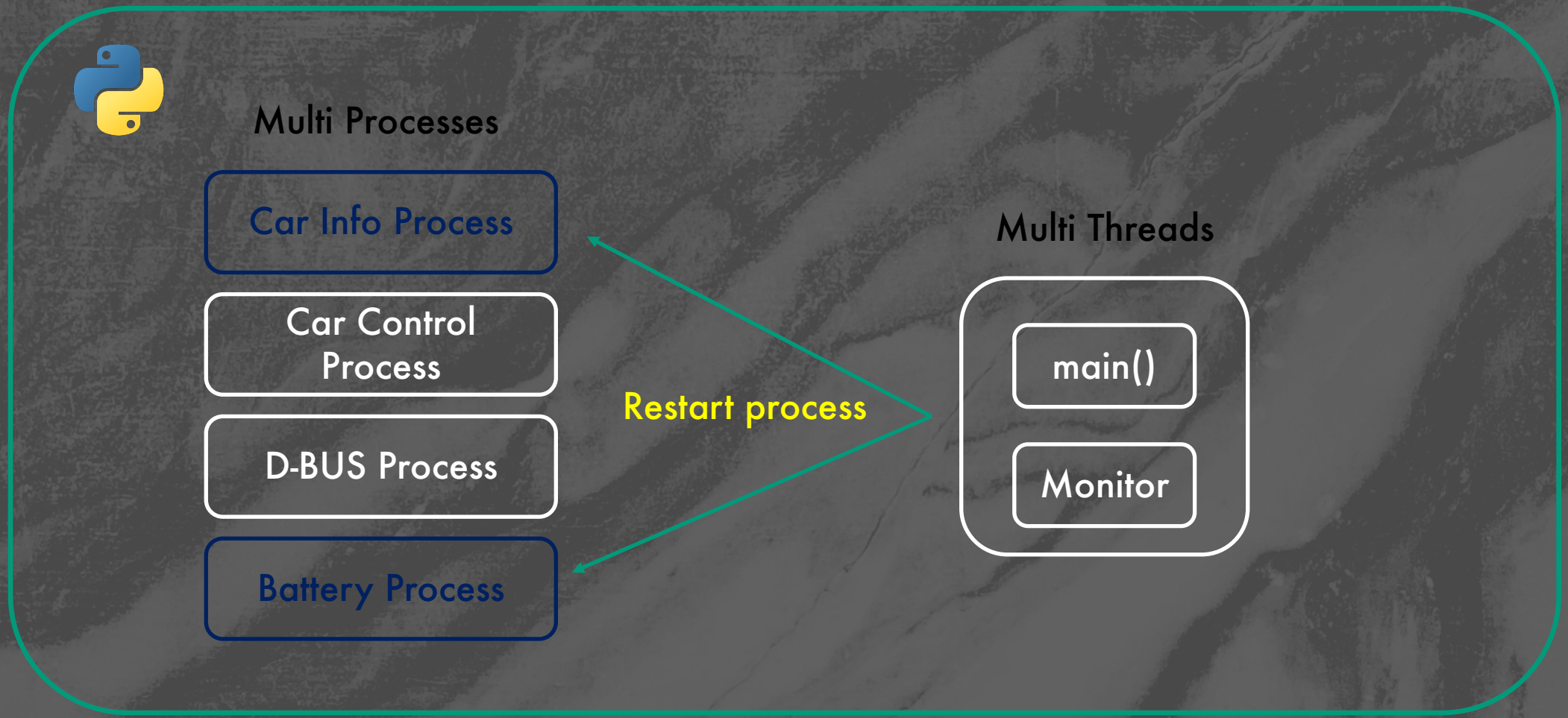
Setup OS

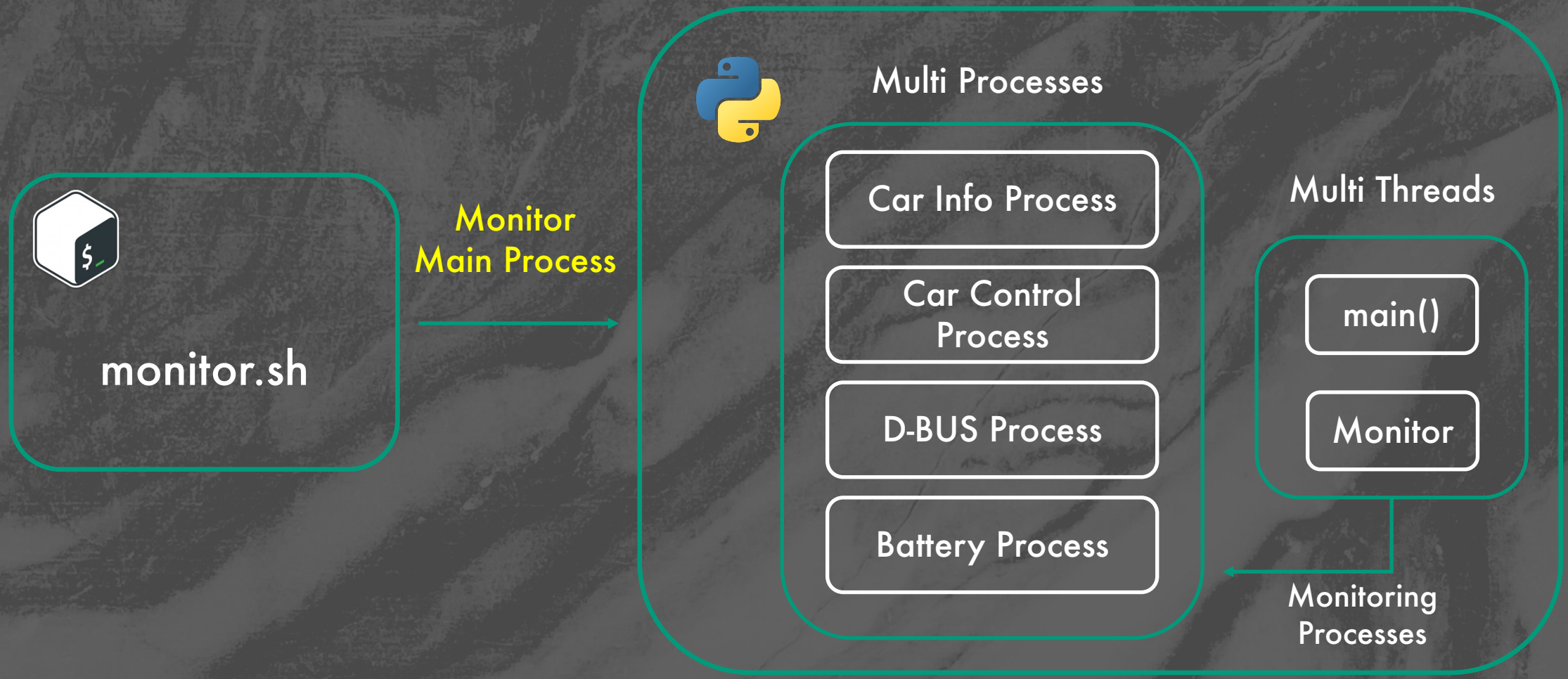
1. Create shell files
2. Modify sudoers
3. Create a new service
4. Enable and Start service

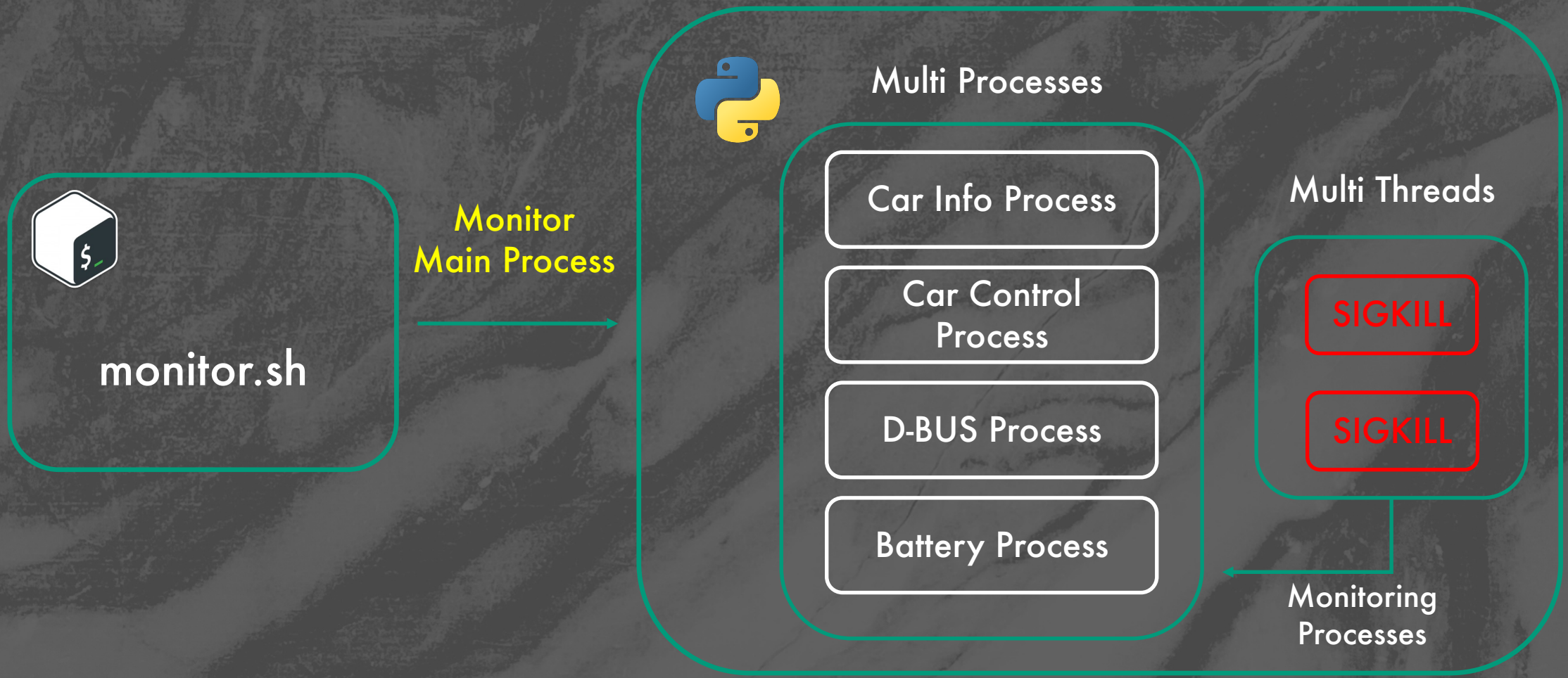


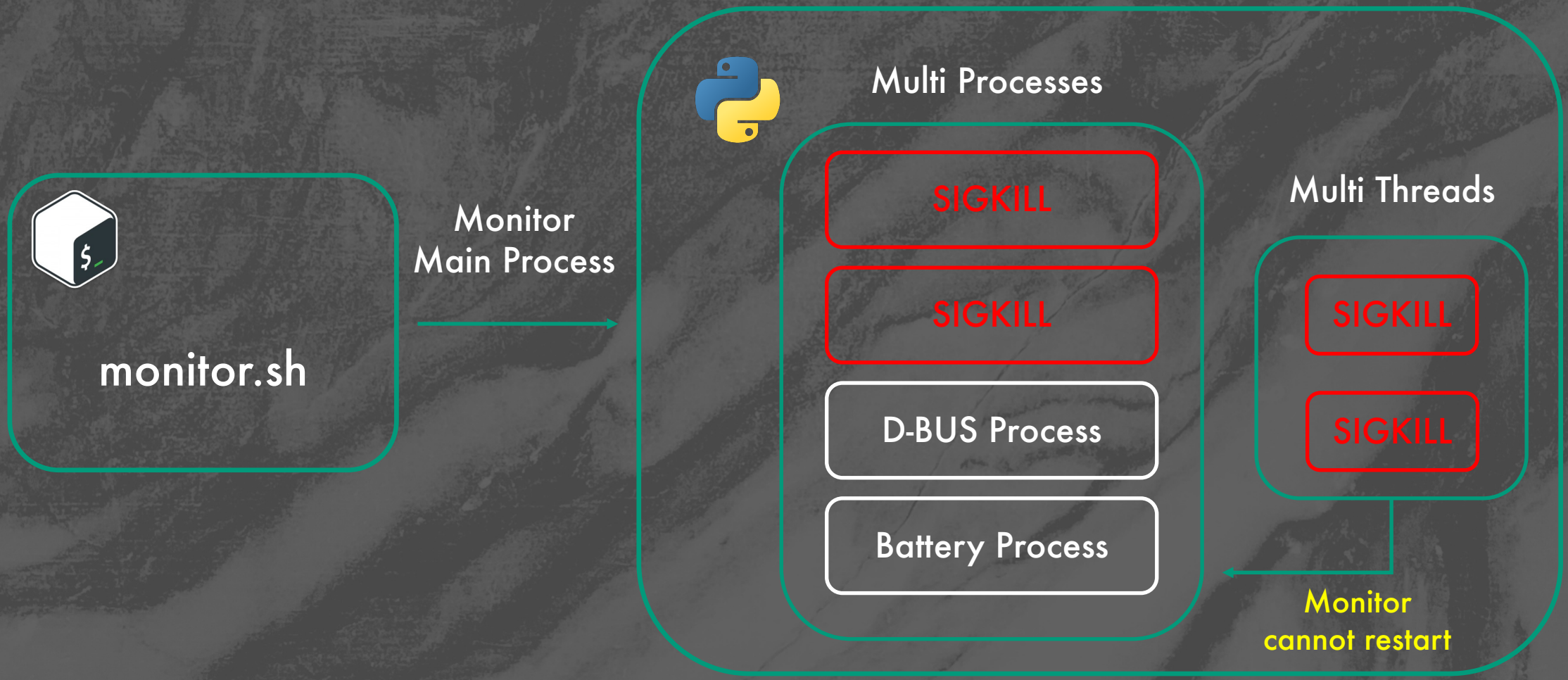

Multi Processes
Car Info Process
Car Control Process
D-BUS Process
Battery Process
Check if alive or not
Multi Threads
main()
Monitor

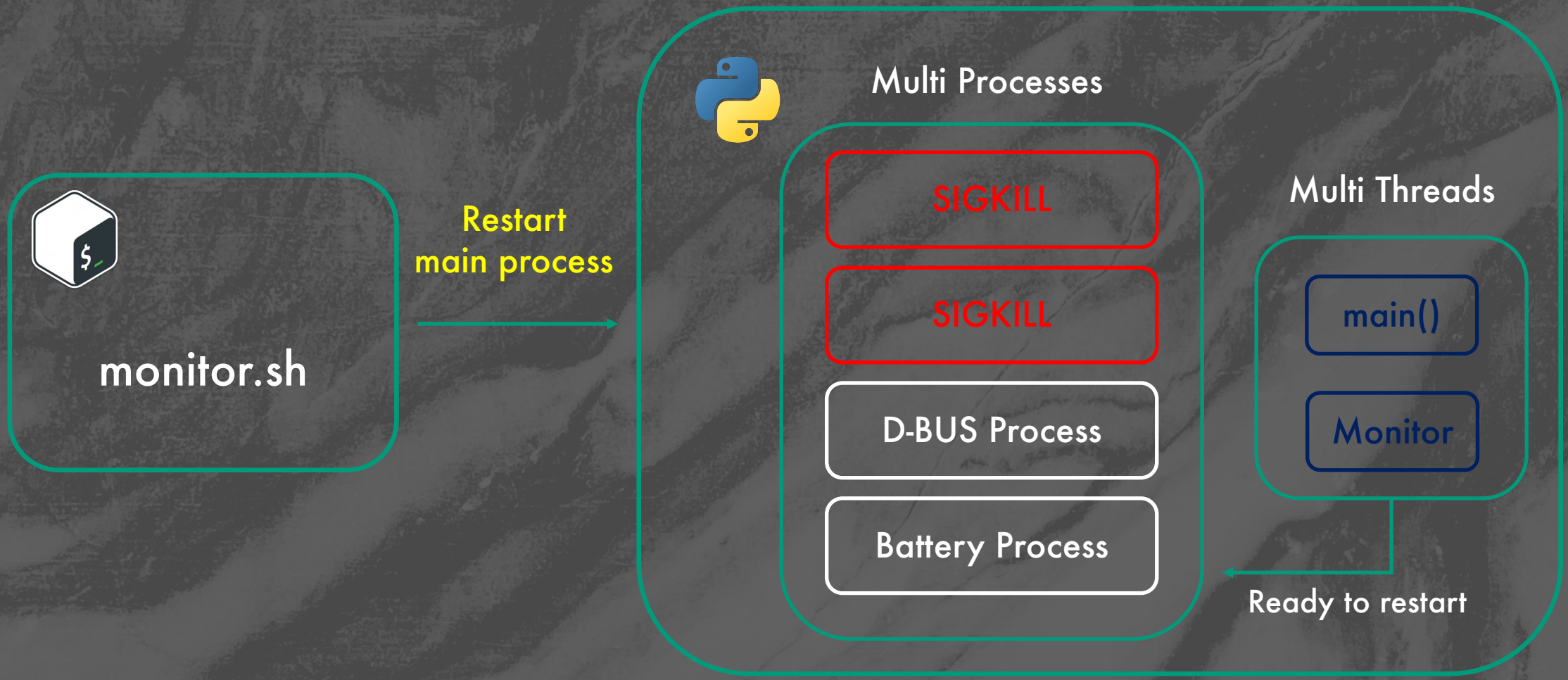

Multi Processes
SIGKILL
Car Control Process
D-BUS Process
SIGKILL
Check if alive or not
Multi Threads
main()
Monitor

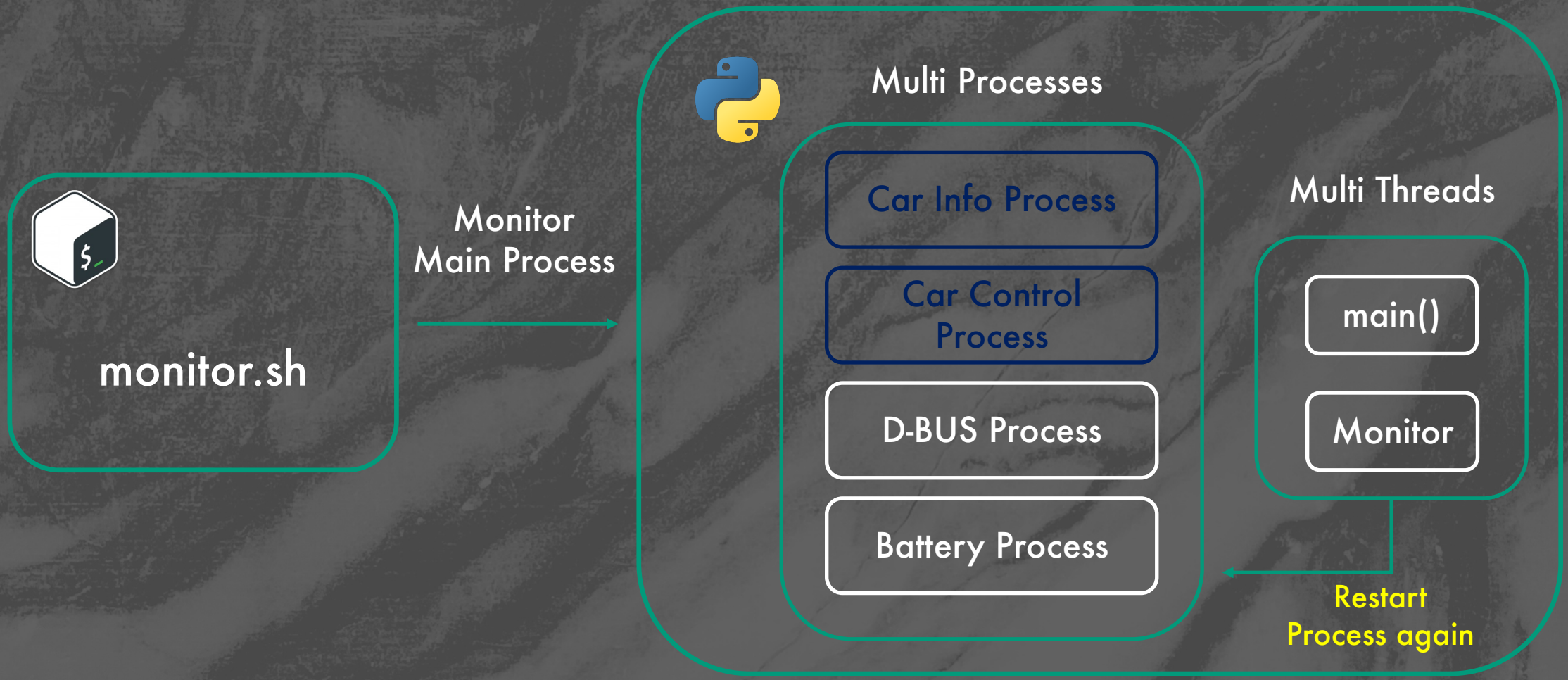

Multi Processes
Car Info Process
Car Control Process
D-BUS Process
Battery Process
Restart process
Multi Threads
main()
Monitor


monitor.sh
Monitor
Main Process
Multi Processes
Car Info Process
Car Control Process
D-BUS Process
Battery Process
Multi Threads
main()
Monitor
Monitoring Processes


monitor.sh
Monitor
Main Process
Multi Processes
Car Info Process
Car Control
Process
D-BUS Process
Battery Process
Multi Threads
SIGKILL
SIGKILL
Monitoring
Processes


monitor.sh
Monitor
Main Process
Multi Processes
SIGKILL
SIGKILL
D-BUS Process
Battery Process
Multi Threads
SIGKILL
SIGKILL
Monitor
cannot restart


monitor.sh
Restart
main process
Multi Processes
SIGKILL
SIGKILL
D-BUS Process
Battery Process
Multi Threads
main()
Monitor
Ready to restart


monitor.sh
Monitor
Main Process
Multi Processes
Car Info Process
Car Control Process
D-BUS Process
Battery Process
Multi Threads
main()
Monitor
Restart
Process again

#4-1 DEMO

Ask Me Anything!

Thanks for listening.


Kwanho Kim
@KKWANH


Ogura Shuta
@Shuta-Syd


Kian Warias
@kianwasabi