



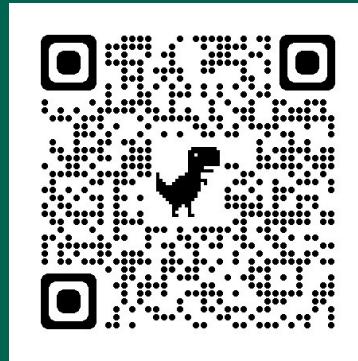
software engineering in automotive
and mobility ecosystems

Team 4: Daekyung La, Jaehyeong Park, Minchan Jung

DES02: Instrument Cluster



Team4 Organization



DES02 repository



Kanban Board

INDEX

01. Specification

02. Display

03. Cross Compile

04. CAN Communication

05. Get Sensor Data

06. D-Bus

07. Connect C++ to QML

08. GUI(QML)

09. Exception Handling

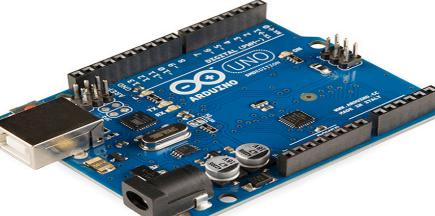
10. Demo

Specification: Hardware

Raspberry Pi

Raspberry Pi 4B	CAN-BUS(FD) Shield(MCP2518FD)	7.9inch DSI LCD
		

Arduino UNO

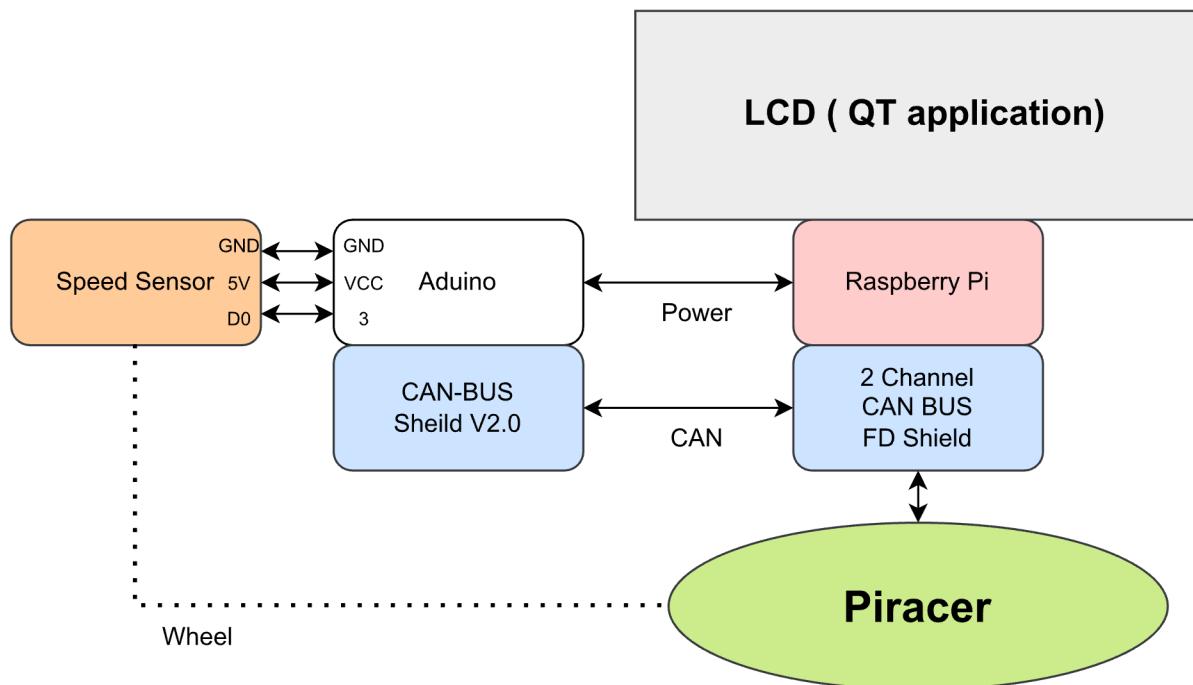
Arduino UNO	CAN-BUS Shield V2.0 (MCP2515)	Speed Sensor (LM393)
		

PiRacer AI Kit

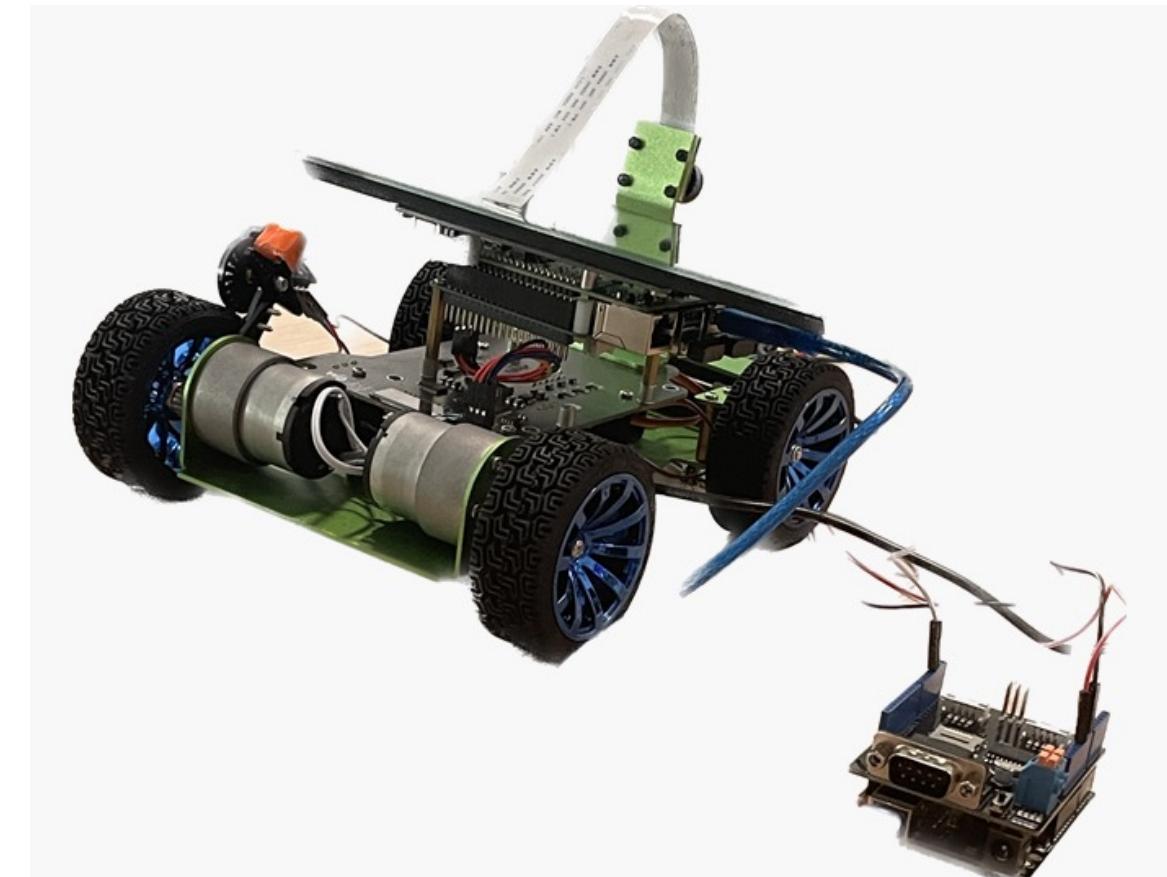
Piracer Kit	Joystick
	

Specification: Hardware connection

Circuit diagram



Physical hardware

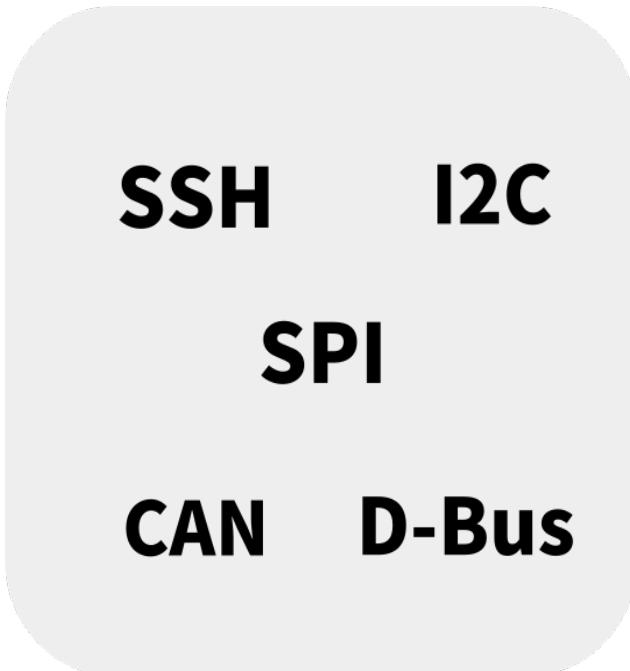


Specification: Software

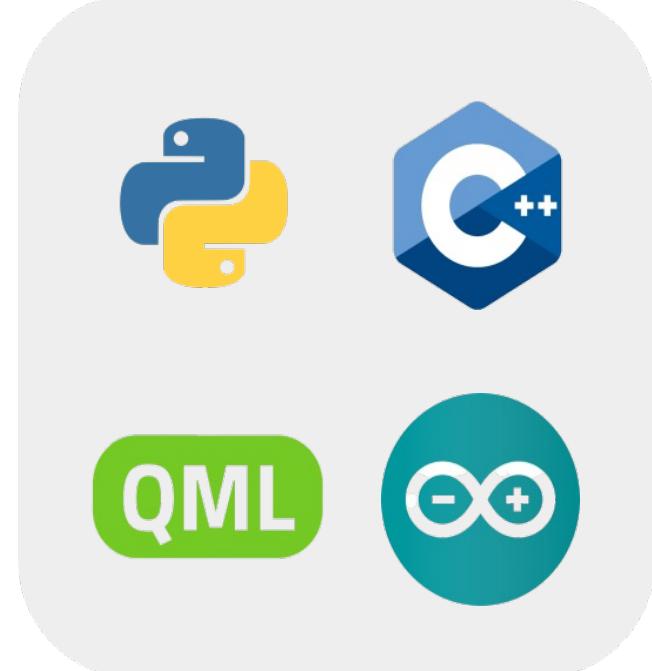
Tool



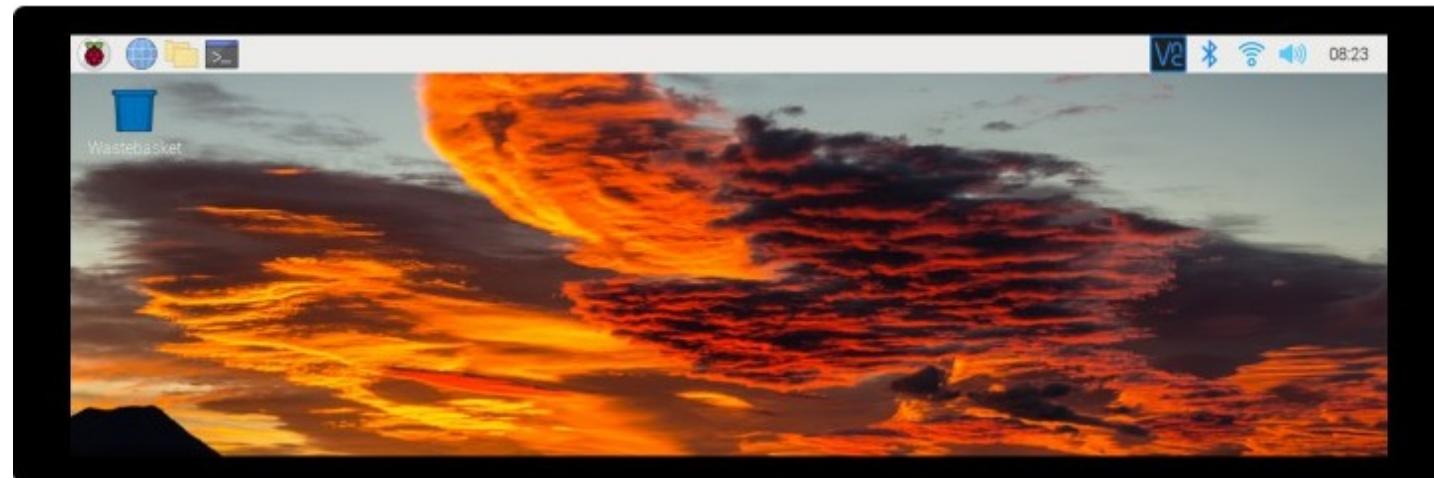
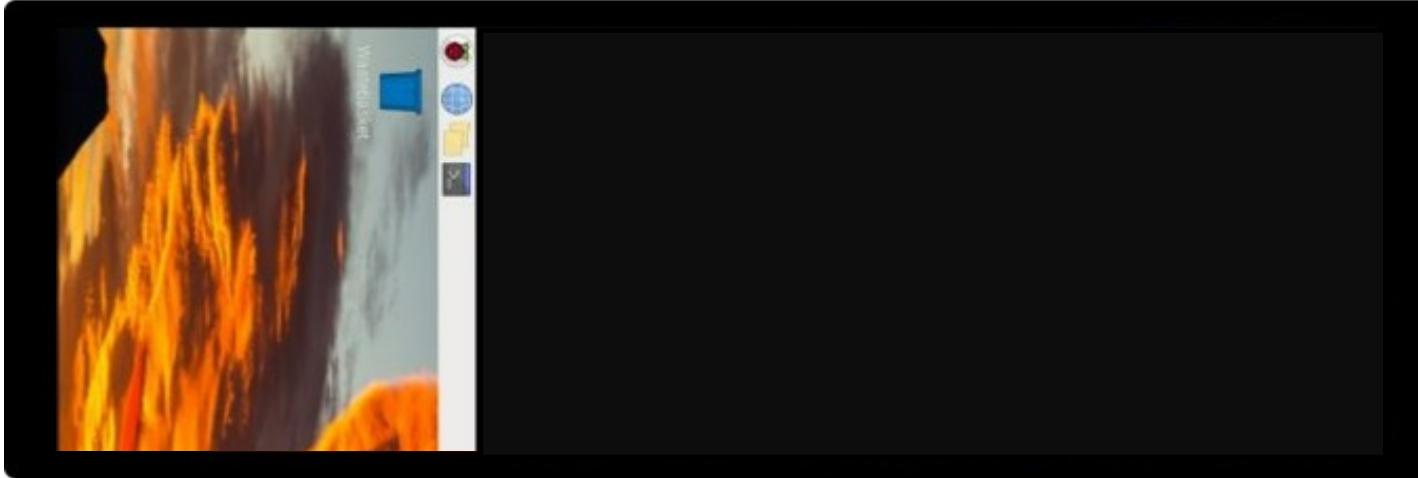
Technology



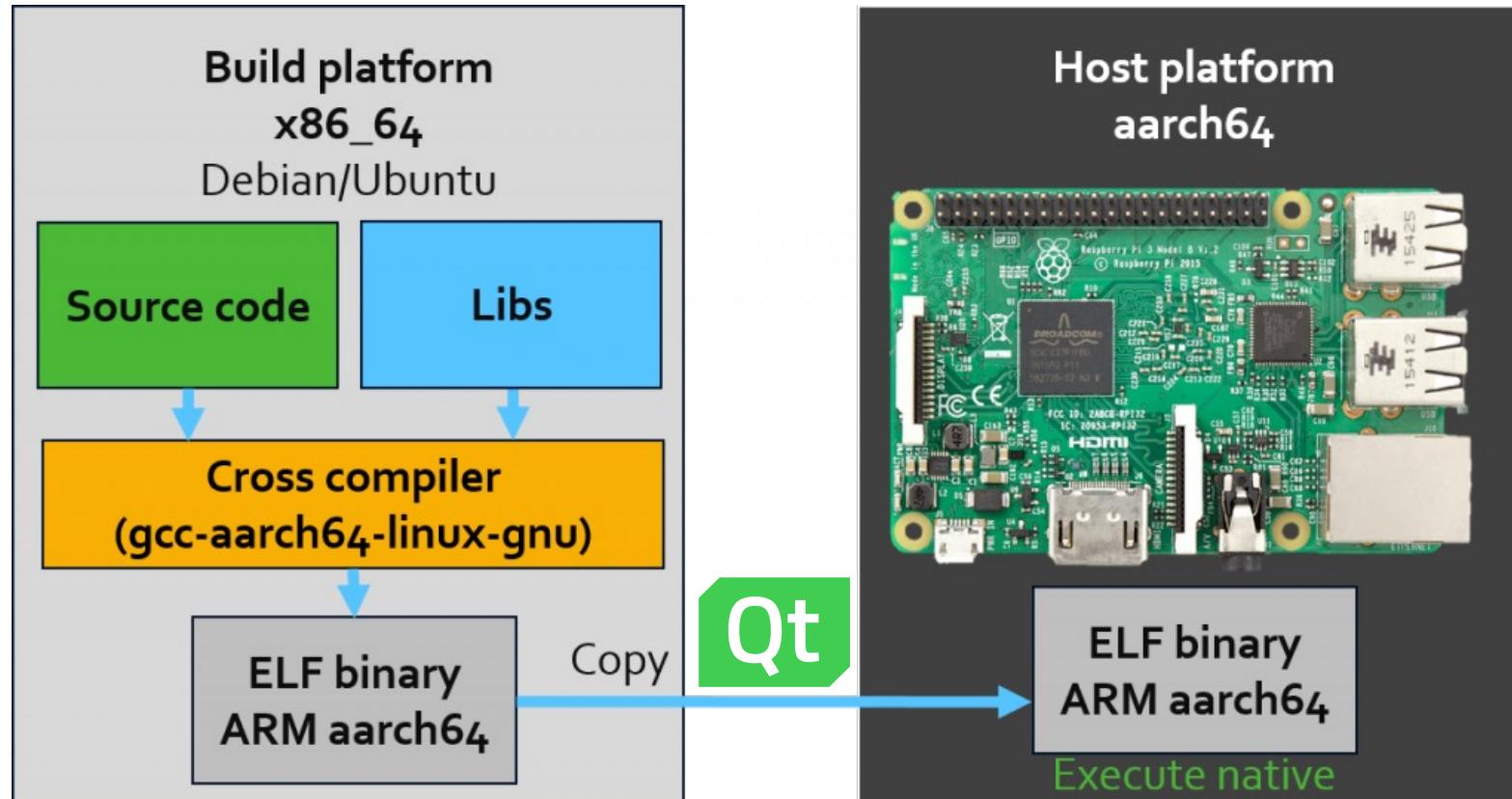
Language



Display

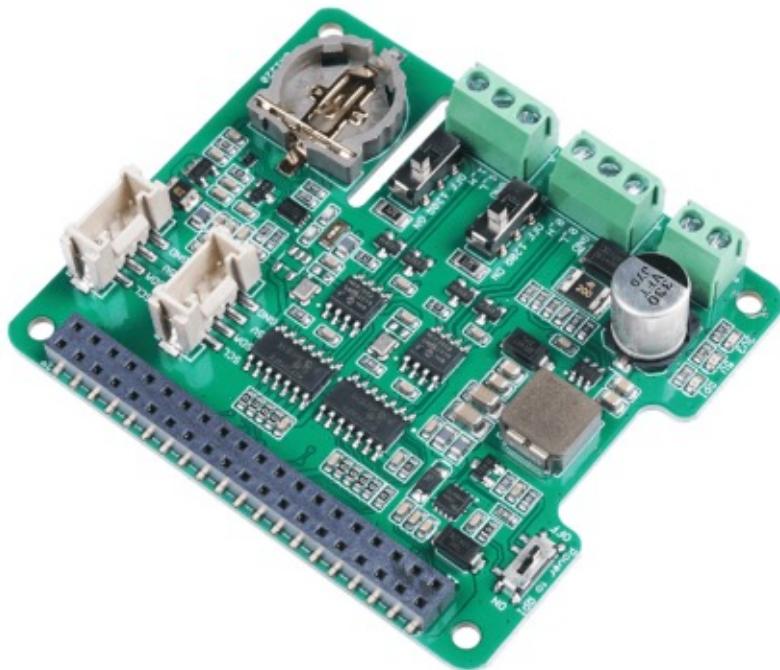


Cross Compile

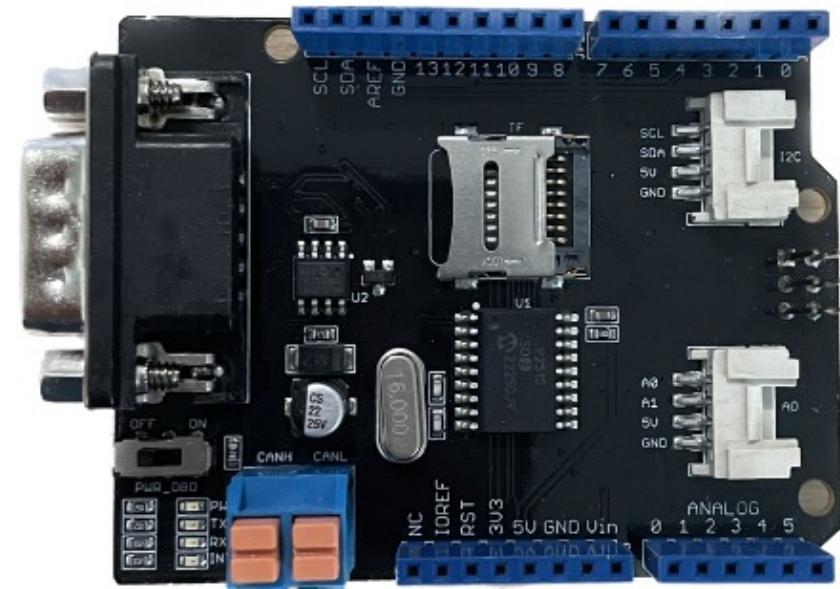


CAN Communication

2 Channel CAN BUS FD Shield



CAN-BUS Shield



CAN Communication

sender code

```
#include <SPI.h>
#include "mcp2515_can.h"

/*SAMD core*/
#ifndef ARDUINO_SAMD_VARIANT_COMPLIANCE
    #define SERIAL SerialUSB
#else
    #define SERIAL Serial
#endif

const int SPI_CS_PIN = 9;
mcp2515_can CAN(SPI_CS_PIN); // Set CS pin

void setup() {
    SERIAL.begin(115200);
    while(!Serial){};

    while (CAN_OK != CAN.begin(CAN_500KBPS)) { // init can bus : baudrate = 500k
        SERIAL.println("CAN BUS Shield init fail");
        SERIAL.println(" Init CAN BUS Shield again");
        delay(100);
    }
    SERIAL.println("CAN BUS Shield init ok!");
}

unsigned char stmp[8] = {0, 0, 0, 0, 0, 0, 0, 0};
void loop() {
    // send data: id = 0x00, standrad frame, data len = 8, stmp: data buf
    stmp[7] = stmp[7] + 1;
    if (stmp[7] == 100) {
        stmp[7] = 0;
        stmp[6] = stmp[6] + 1;

        if (stmp[6] == 100) {
            stmp[6] = 0;
            stmp[5] = stmp[6] + 1;
        }
    }

    CAN.sendMsgBuf(0x00, 0, 8, stmp);
    delay(100); // send data per 100ms
    SERIAL.println("CAN BUS sendMsgBuf ok!");
}
```

receiver code

```
#include <SPI.h>
#include <mcp_can.h>

MCP_CAN CAN(9); //spi cs
void setup() {
    // put your setup code here, to run once:
    Serial.begin(115200);
    while(CAN_OK != CAN.begin(CAN_500KBPS, MCP_8MHz))
    {
        Serial.println("CAN BUS Init Failed");
        delay(100);
    }
    Serial.println("CAN BUS Init Success!");
}

void loop() {
    // put your main code here, to run repeatedly:
    unsigned char len =0;
    unsigned char buf[8];
    if(CAN_MSGAVAIL == CAN.checkReceive())
    {
        CAN.readMsgBuf(&len, buf);
        unsigned long canId = CAN.getCanId();
        Serial.print("\nData from ID: 0x");
        Serial.println(canId, HEX);
        for(int i=0; i<len; i++)
        {
            Serial.print(buf[i]);
            Serial.print("\t");
        }
    }
}
```



CAN Communication

sender code

```
#include <SPI.h>
#include "mcp2515_can.h"

/*SAMD core*/
#ifndef ARDUINO_SAMD_VARIANT_COMPLIANCE
    #define SERIAL SerialUSB
#else
    #define SERIAL Serial
#endif

const int SPI_CS_PIN = 9;
mcp2515_can CAN(SPI_CS_PIN); // Set CS pin

void setup() {
    SERIAL.begin(115200);
    while(!Serial){};

    while (CAN_OK != CAN.begin(CAN_500KBPS)) { // init can bus : baudrate = 500k
        SERIAL.println("CAN BUS Shield init fail");
        SERIAL.println(" Init CAN BUS Shield again");
        delay(100);
    }
    SERIAL.println("CAN BUS Shield init ok!");
}

unsigned char stmp[8] = {0, 0, 0, 0, 0, 0, 0, 0};
void loop() {
    // send data: id = 0x00, standrad frame, data len = 8, stmp: data buf
    stmp[7] = stmp[7] + 1;
    if (stmp[7] == 100) {
        stmp[7] = 0;
        stmp[6] = stmp[6] + 1;

        if (stmp[6] == 100) {
            stmp[6] = 0;
            stmp[5] = stmp[6] + 1;
        }
    }

    CAN.sendMsgBuf(0x00, 0, 8, stmp);
    delay(100); // send data per 100ms
    SERIAL.println("CAN BUS sendMsgBuf ok!");
}
```

receiver code

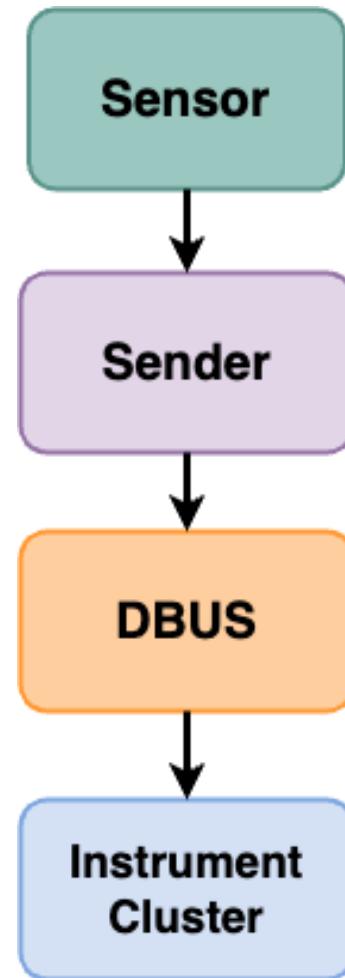
```
#include <SPI.h>
#include <mcp_can.h>

MCP_CAN CAN(9); //spi cs
void setup() {
    // put your setup code here, to run once:
    Serial.begin(115200);
    while(CAN_OK != CAN.begin(CAN_500KBPS, MCP_8MHz))
    {
        Serial.println("CAN BUS Init Failed");
        delay(100);
    }
    Serial.println("CAN BUS Init Success!");
}

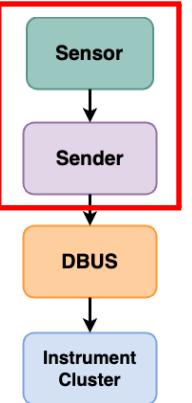
void loop() {
    // put your main code here, to run repeatedly:
    unsigned char len =0;
    unsigned char buf[8];
    if(CAN_MSGAVAIL == CAN.checkReceive())
    {
        CAN.readMsgBuf(&len, buf);
        unsigned long canId = CAN.getCanId();
        Serial.print("\nData from ID: 0x");
        Serial.println(canId, HEX);
        for(int i=0; i<len; i++)
        {
            Serial.print(buf[i]);
            Serial.print("\t");
        }
    }
}
```



Code Flow



Sender: Speed Sender



Arduino (speed_sender.ino)

- Calculate RPM, Speed
- Split Speed Data
- **Send CAN Message**
- Transmission Interval: **0.1s**

Raspberry Pi (speed_sender.py)

- Check **CAN** Connection
- Receive **CAN** Message
- Decode data to Decimal
- Data Processing(weighting)
- Send data to **D-Bus**

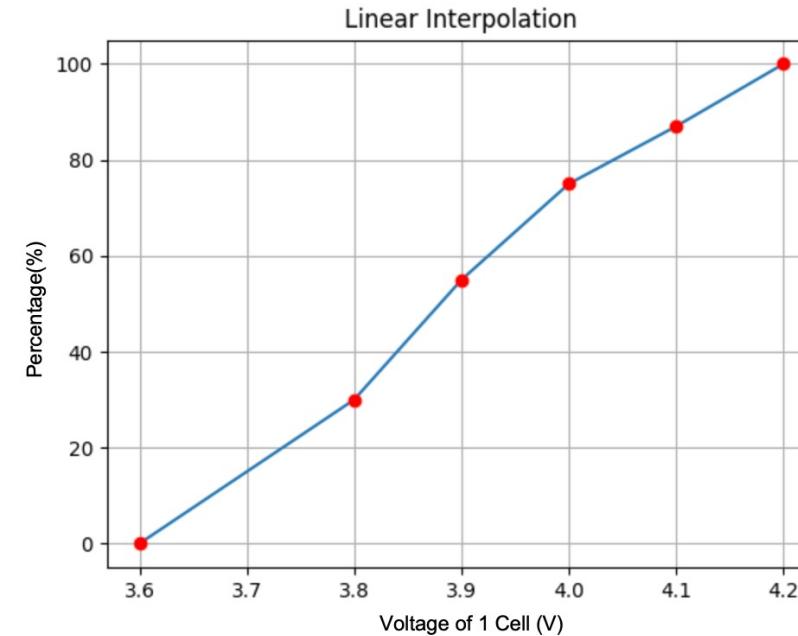
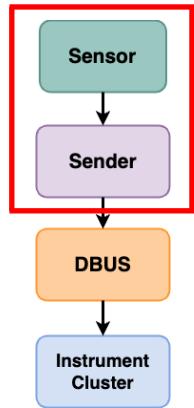
ex) speed: **12.34** cm/s

	(int)speed/256	(int)speed%256	round((speed – (int)speed/256)*100)						
canMsg.data	00	0C(12)	22(34)						
	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	

Sender: Battery Sender

Raspberry Pi (battery_sender.py)

- Get Battery Voltage(i2c)
- Check Battery Percentage
- Data Processing (Maximum value)
- Send data to D-Bus
(Transmission Interval: **0.01s**)



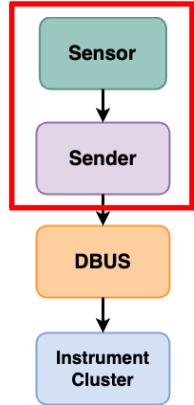
Sender: Brake Sender

Raspberry Pi (brake_sender.py)

- Get Gamepad Input(left stick)
(If input is backward, **True**
else, **False**)
- Timeout Decorator
- Send data to **D-Bus**
(Transmission Interval: **0.01s**)



Brake: **True**

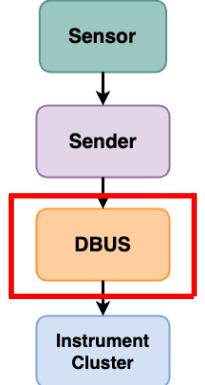


D-Bus: Server Registration (in register_server)

- Define D-Bus Object(carinformation.cpp, carinformation.h)
- Compile C++ to XML
- Add XML to Project
- **Register Server and Expose Object to D-Bus**

```
CarInformation *carinfo = new CarInformation();
new CarInformationAdaptor(carinfo);

QDBusConnection connection = QDBusConnection::sessionBus();
connection.registerObject("/CarInformation", carinfo);
connection.registerService("org.team4.Des02");
```



D-Bus: "org.team4.Des02"

Object: "/CarInformation"

Interface:
"org.team4.Des02.CarInformation"

signals

methods

D-Bus: Send data to D-Bus

sender.py

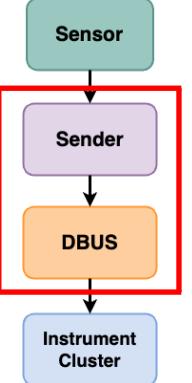
- Connect with SessionBus
- Connect with Object
- Send data to D-Bus by calling **setVariable** method of Interface

D-Bus: "org.team4.Des02"

Object: "/CarInformation"

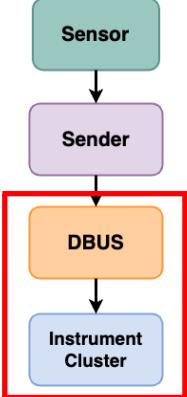
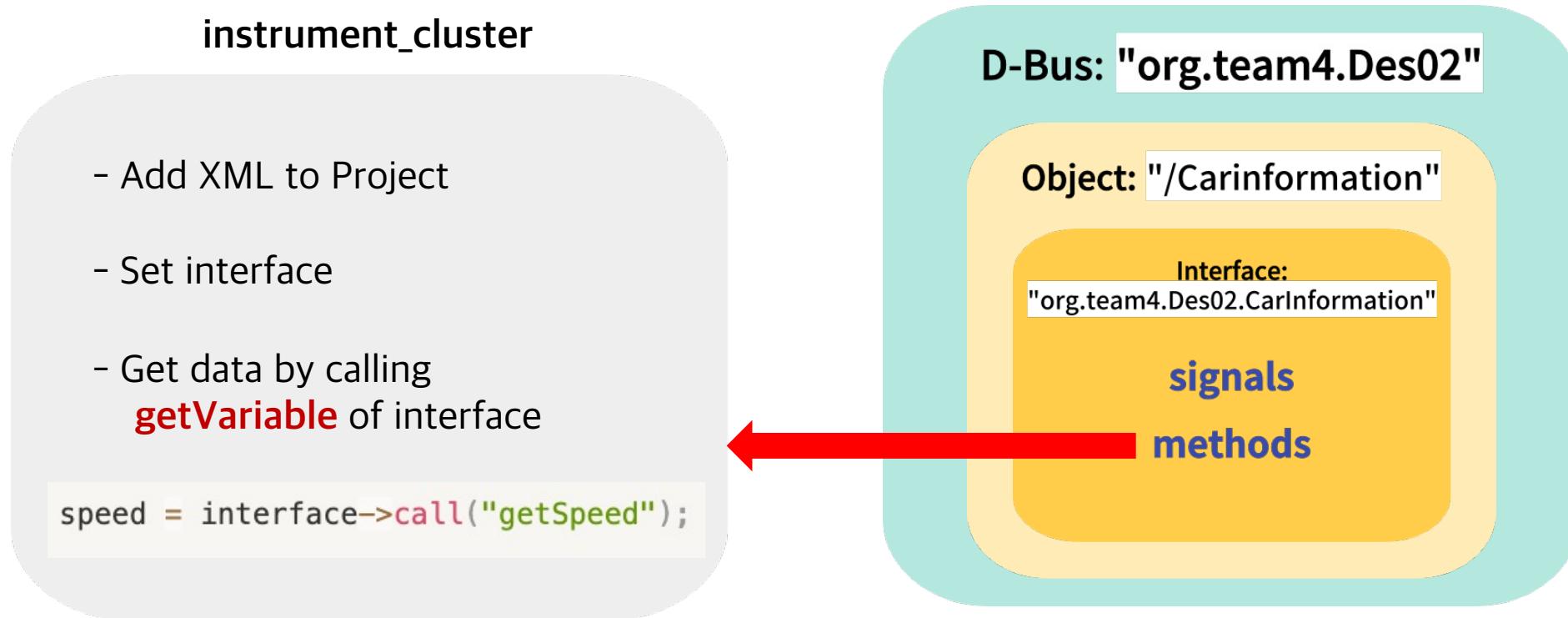
Interface:
"org.team4.Des02.CarInformation"

signals
methods



```
bus = dbus.SessionBus()  
service = bus.get_object("org.team4.Des02", "/CarInformation")  
car_interface = dbus.Interface(service, "org.team4.Des02.CarInformation")  
  
car_interface.setSpeed(speed_data)
```

D-Bus: Get data from D-Bus (in instrument_cluster)



```
// car2qml.h
org::team4::Des02::CarInformation *interface;

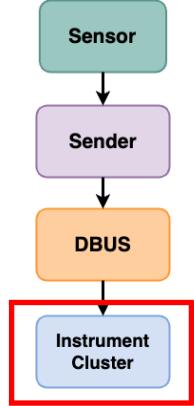
// car2qml.cpp
interface = new org::team4::Des02::CarInformation("org.team4.Des02", "/CarInformation", QDBusConnection::sessionBus());
```

Connect C++ to QML (in instrument_cluster)

```
car2qml.h  
Q_PROPERTY(qreal speed READ getSpeed)
```

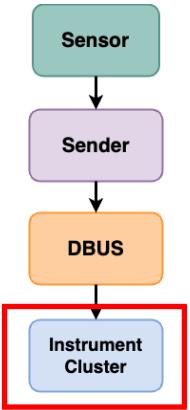
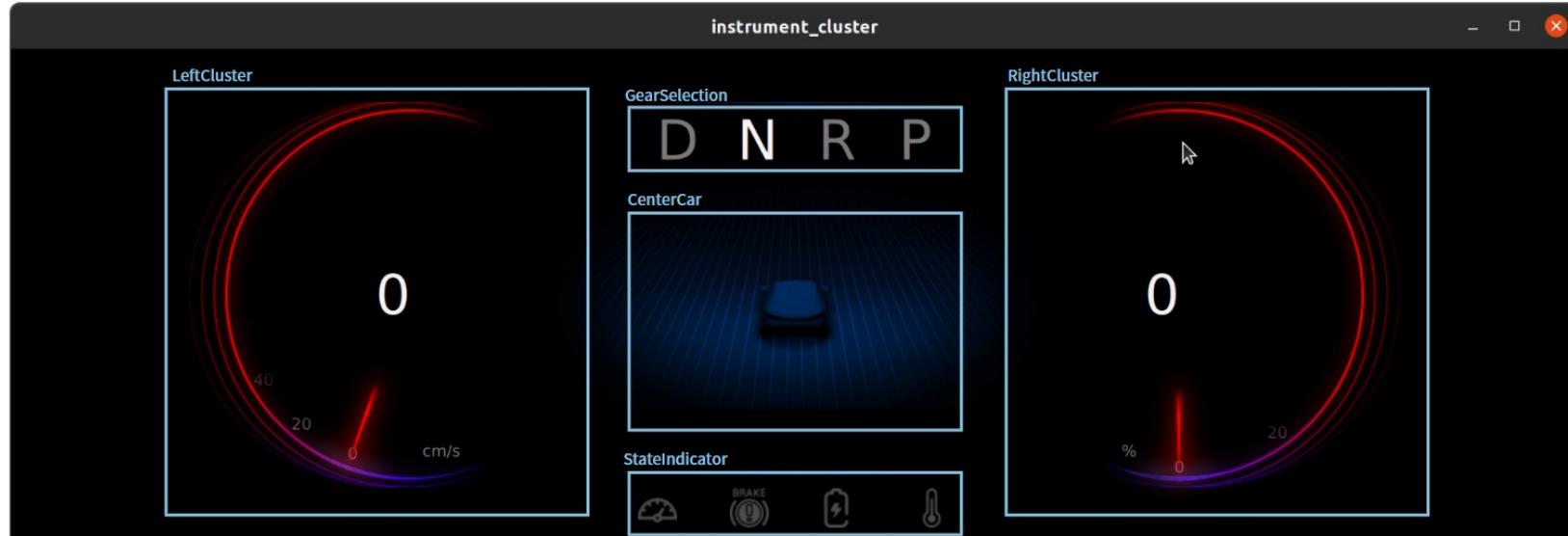
```
car2qml.cpp  
void Car2Qml::setSpeed()  
{  
    speed = interface->call("getSpeed");  
    emit speedChanged(); 1  
}  
  
4 qreal Car2Qml: getSpeed()  
{  
    return speed;  
}
```

```
instrumentCluster.qml  
Car2Qml{  
    id: carinfo  
  
    2 onSpeedChanged: {instrumentcluster.speed = carinfo.speed}
```

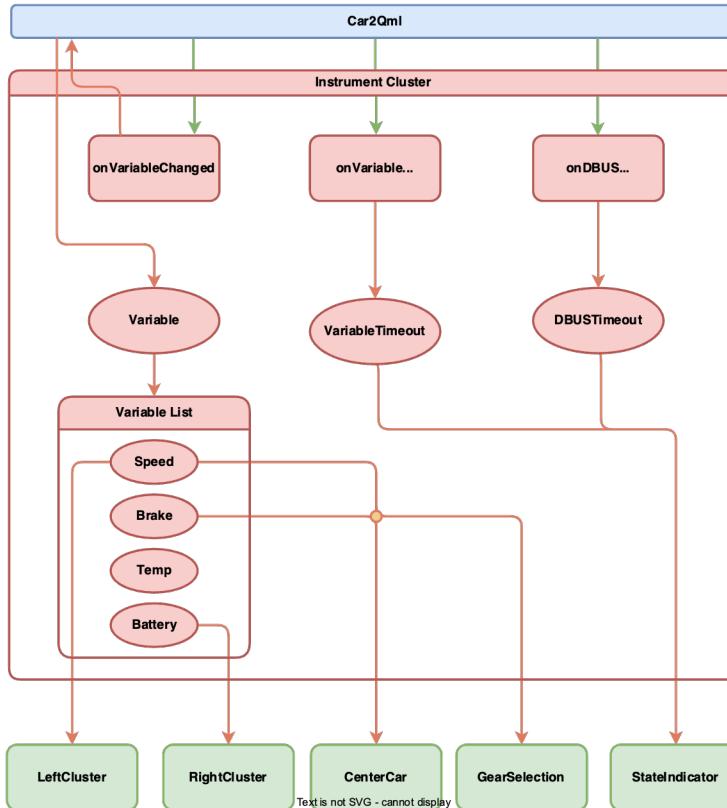


GUI(QML)

instrument_cluster



GUI(QML)



instrumentCluster.qml

```

Car2Qml{
    id: carinfo

    onSpeedChanged: [instrumentcluster.speed] = carinfo.speed
}

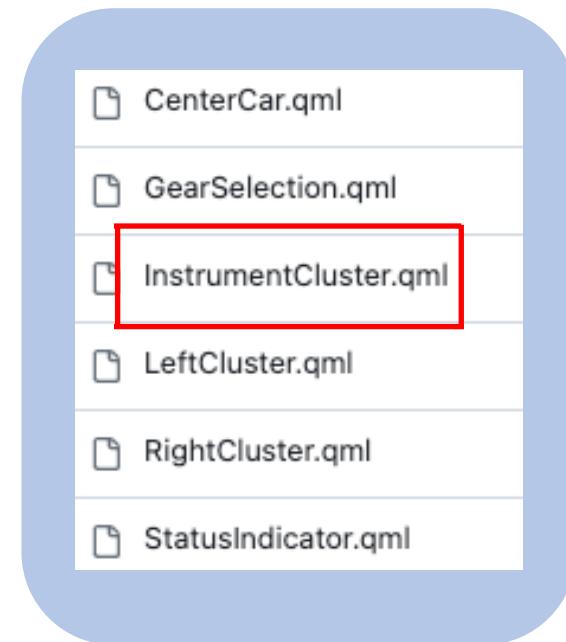
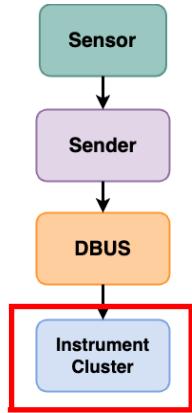
```

LeftCluster.qml

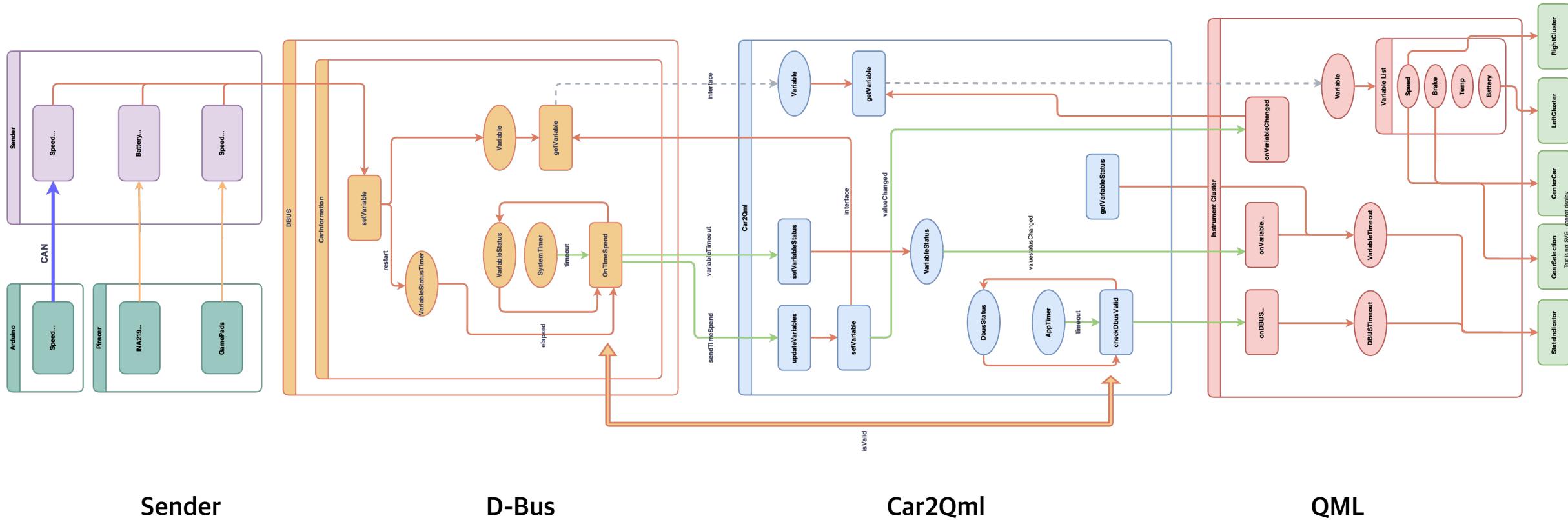
```

Text {
    x: 166
    y: 131
    color: leftcluster.textcolor
    text: Math.abs parent.parent.speed
    font.pixelSize: 45
    font.family: "Sarabun"
    font.bold: false
    width: leftcluster.width
    horizontalAlignment: Text.AlignHCenter
    verticalAlignment: Text.AlignVCenter
}

```



Exception Handling



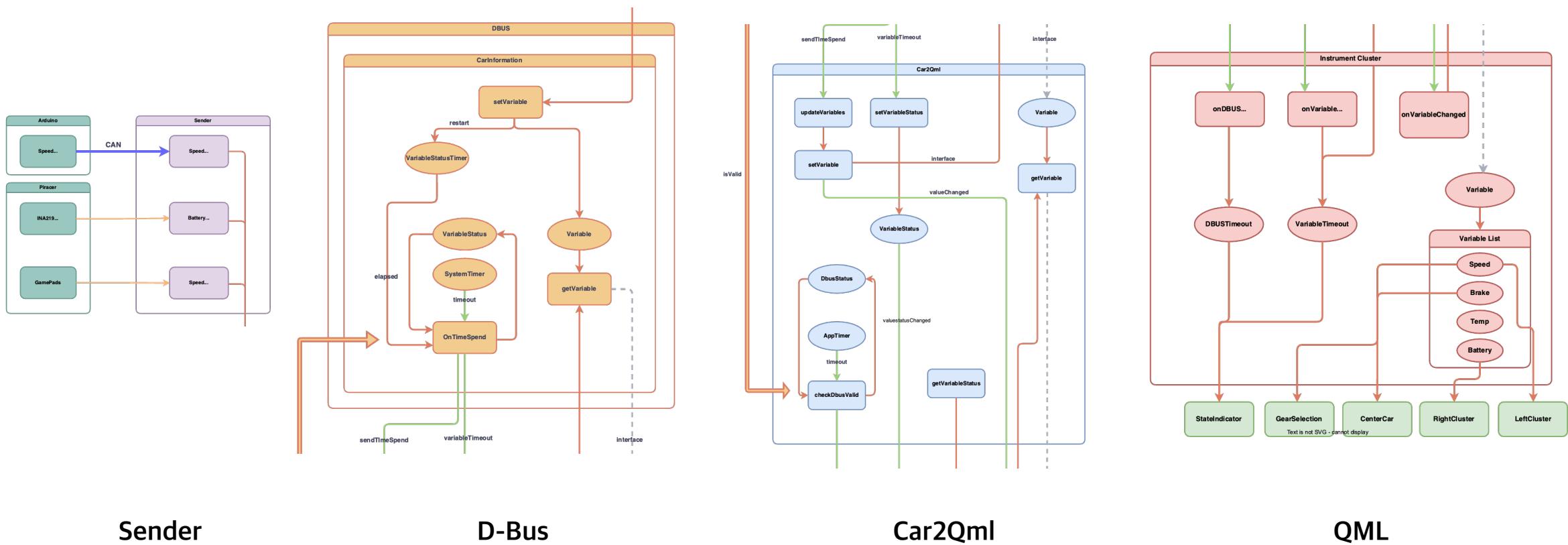
Sender

D-Bus

Car2Qml

QML

Exception Handling



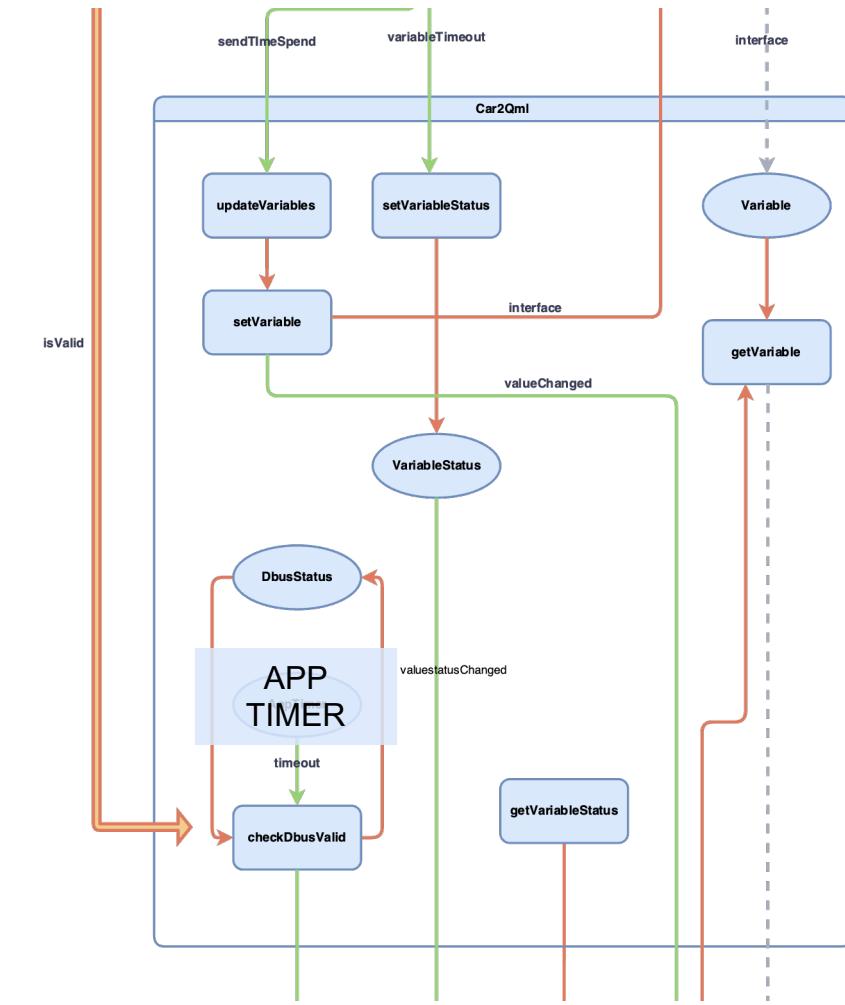
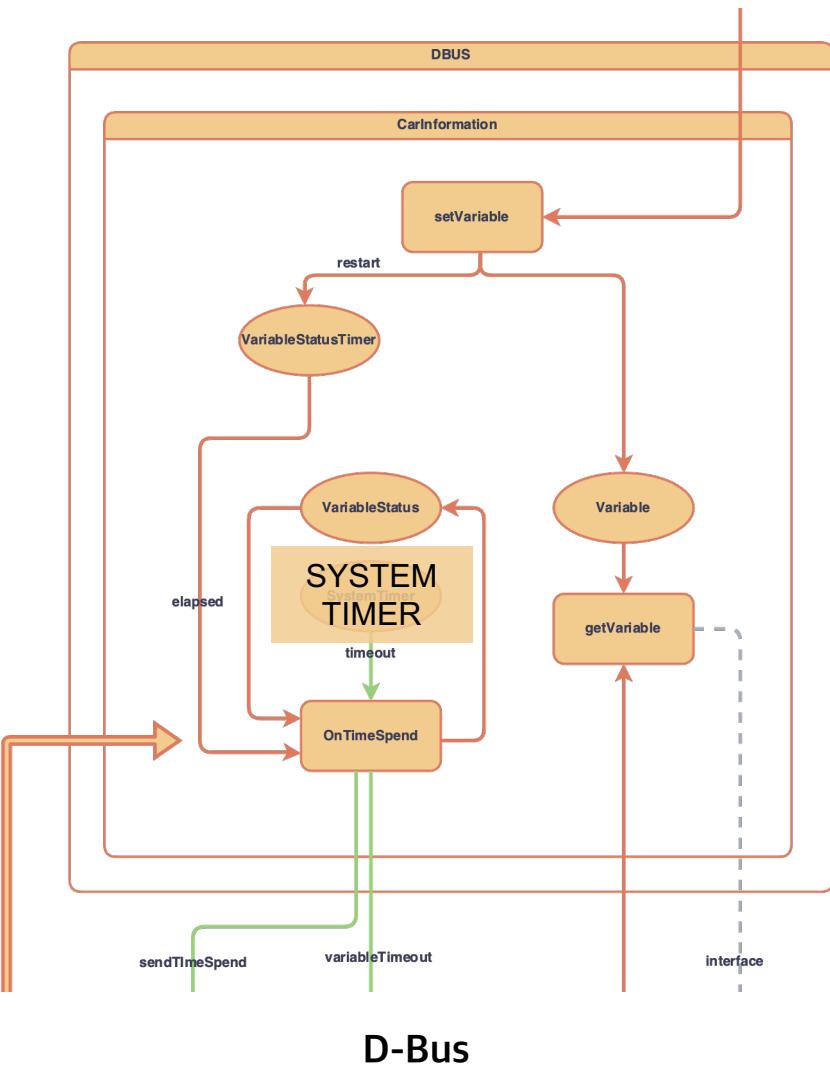
Sender

D-Bus

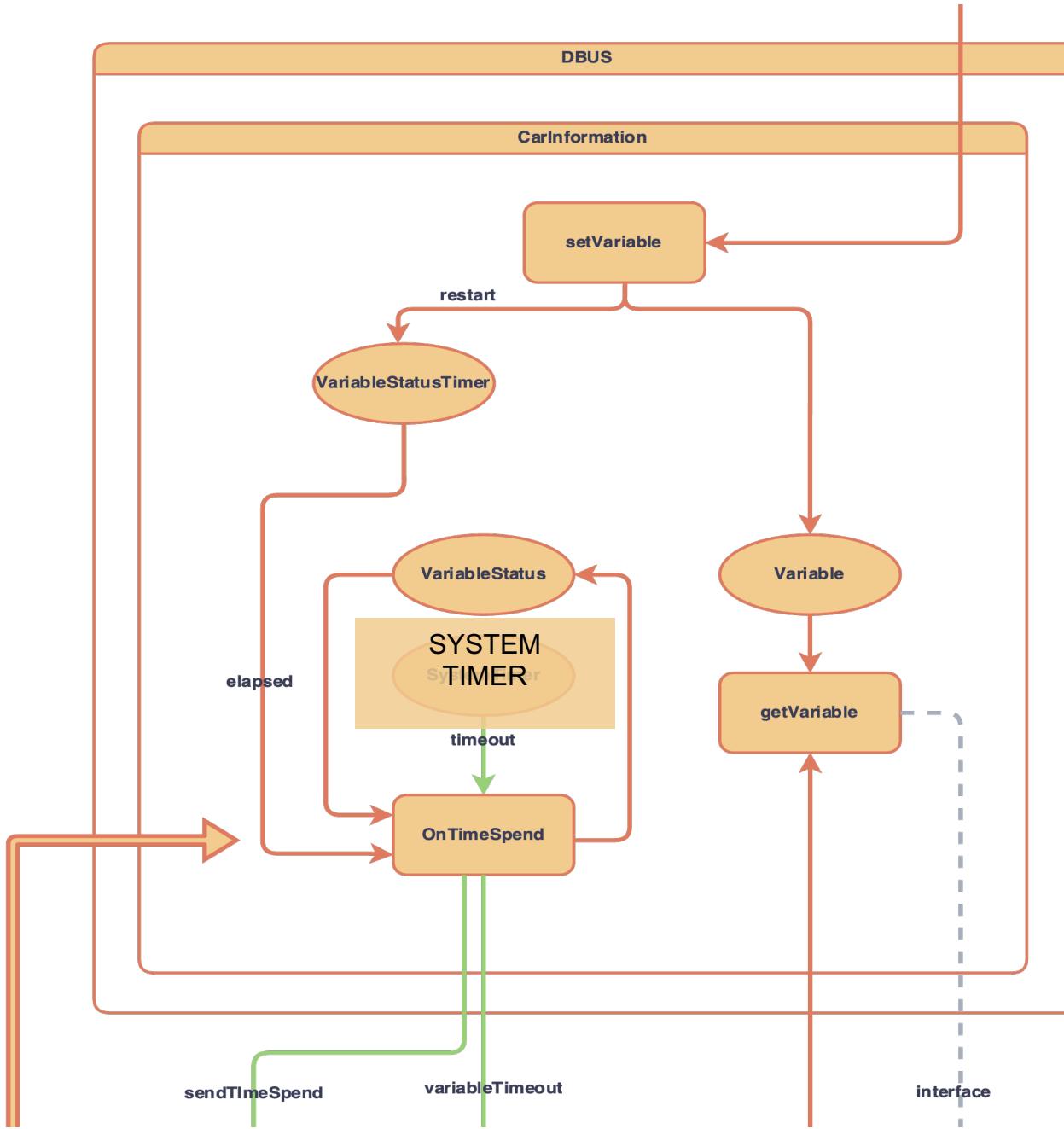
Car2Qml

QML

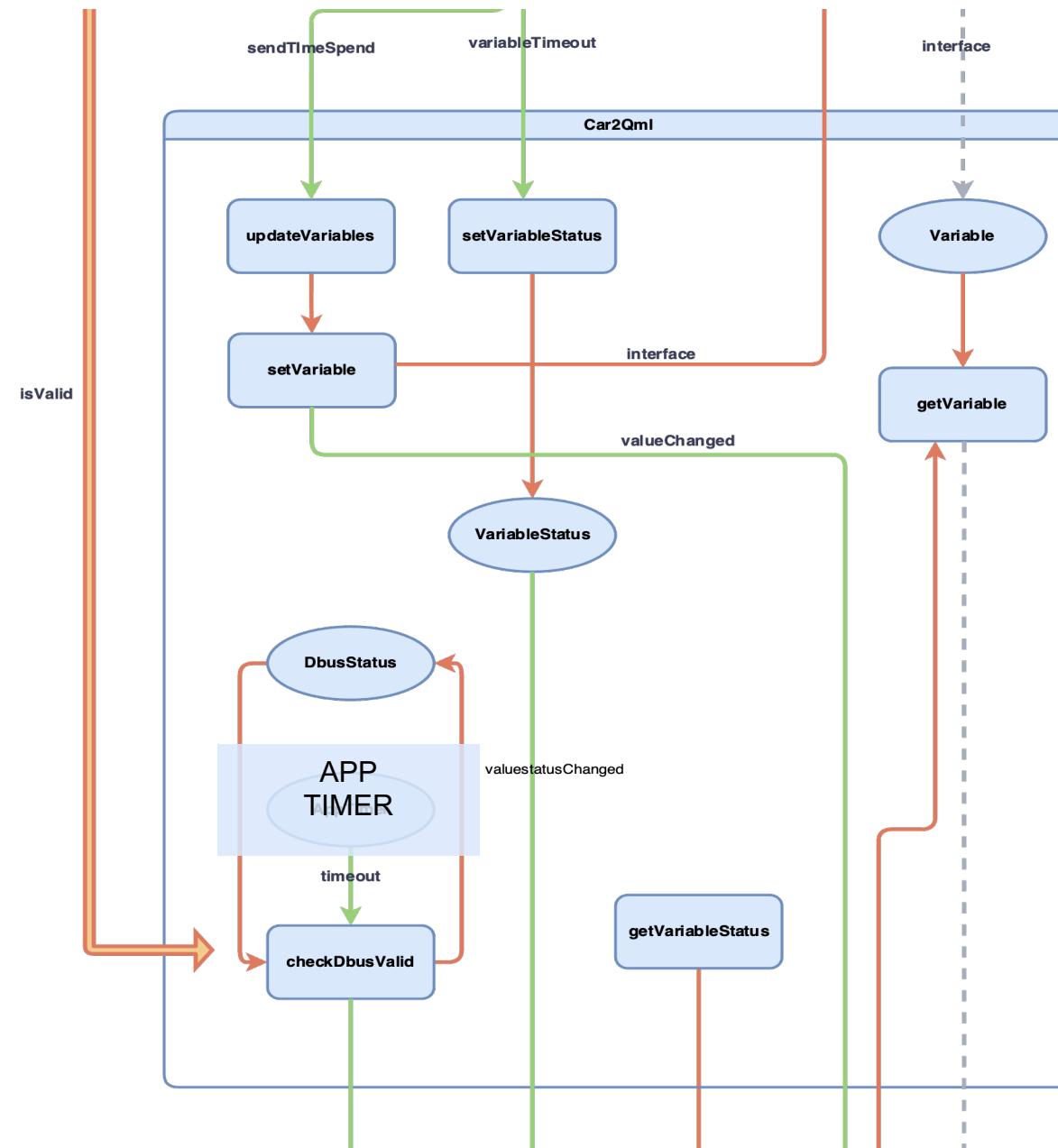
Exception Handling



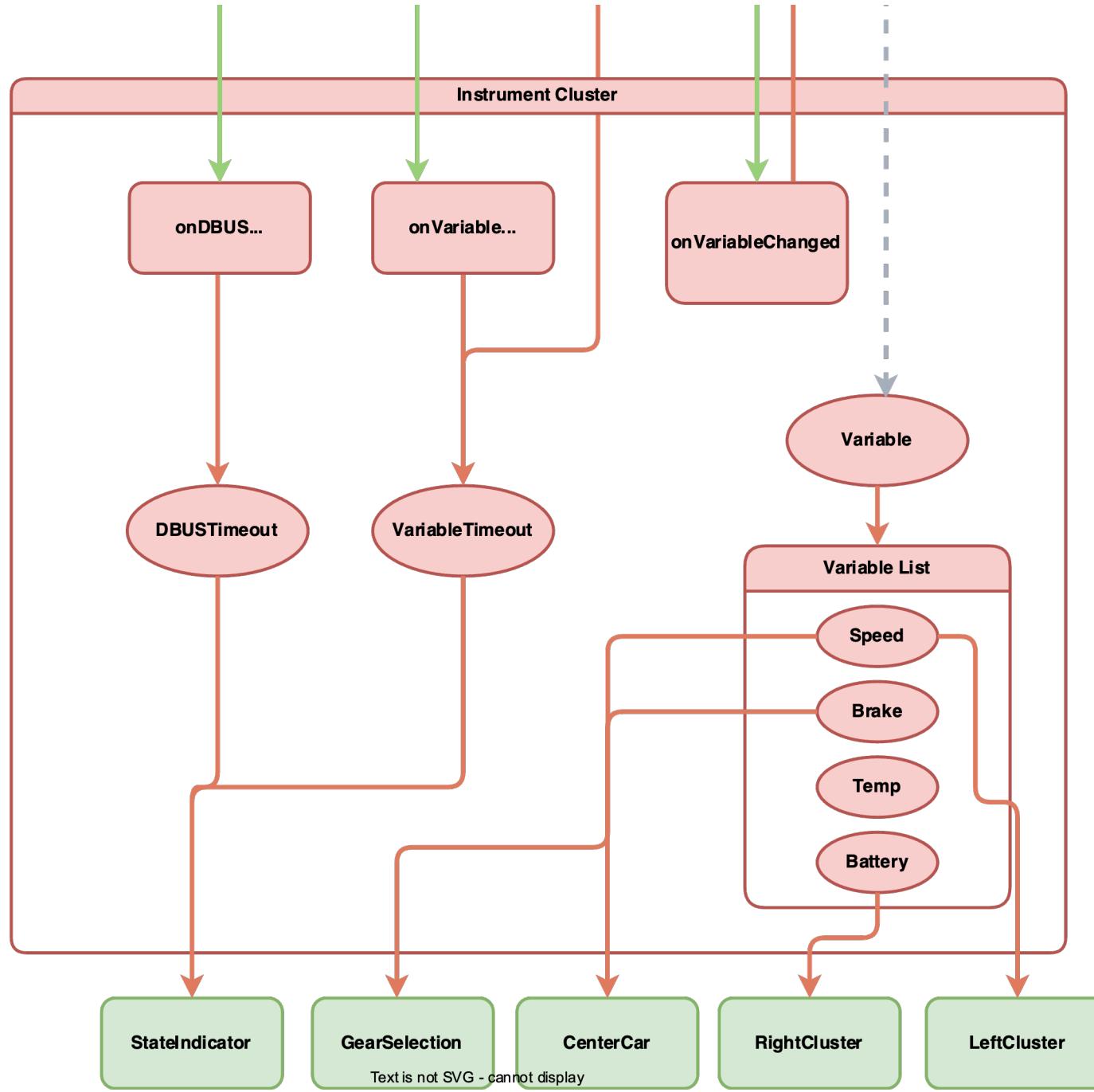
Exception Handling



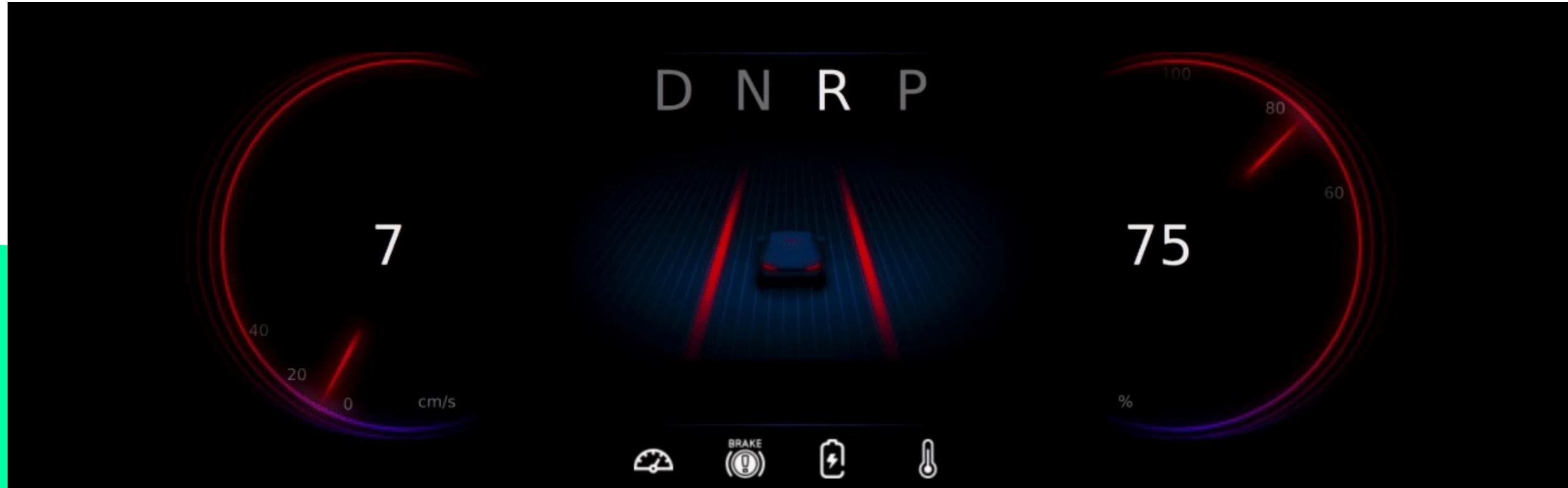
Exception Handling



Exception Handling



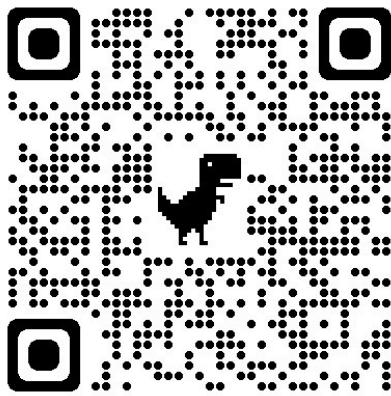
Let's See How It Works!



sea;me

software engineering in automotive
and mobility ecosystems

DES02 Repository



Thank you!

Q&A

Team 4's Evaluation

