Driver Second Generation – Tests

(Code 0400022)

Author:

Alessandro Incandela

Version:
1.0

Date:

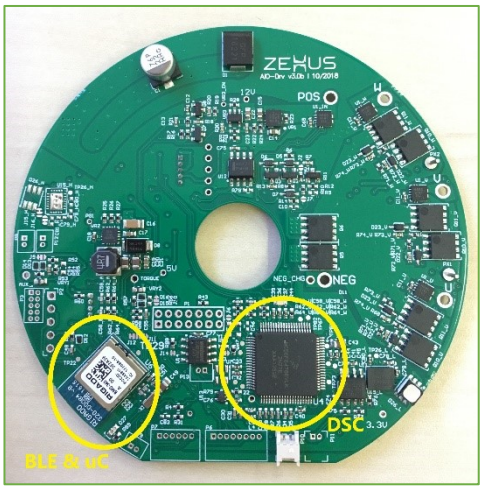
08.04.2019

Please, contact alessandro.incandela@zehus.it for any further questions.

Table of Contents

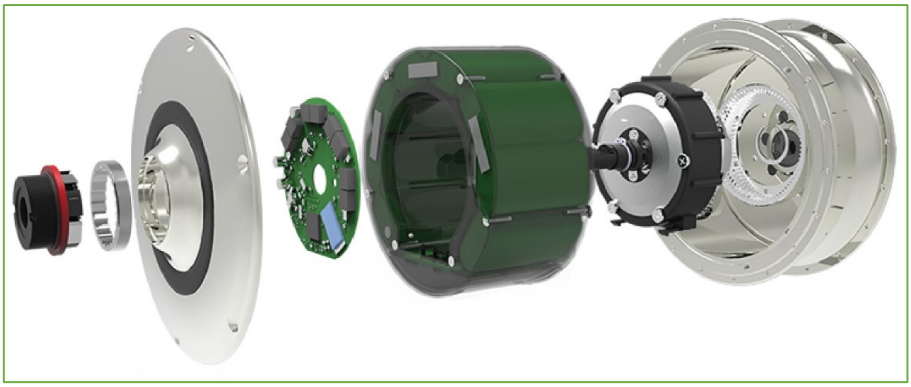
Introduction

This document provides the specification of the *driver second generation* in order to develop PC tool for testing functional requirements. Below, the PCB top and bottom sides are illustrated.



In the driver board, there are two uC assembled: Bluetooth Low Energy (BLE & uC) and Digital Signal Controller (DSC).

The Driver second generation is one of the components that the Zehus All In One (AIO) product is composed of.



AIO product is composed by three main parts:

1. **Smart Motor**, which includes:
    - Driver
    - Motor
2. **Battery Pack**, which includes:
    - BMS
    - Battery cells
3. **Hub Chassis**

Driver Tests configuration

The Driver tests are divided into:

- In-circuit Test: electrical points in the Driver PCB must be tested.
- Firmware Upload: FW files are downloaded from Zehus server.
- Functional Test: Zehus must provide specification to PCB in order to develop PC tool for testing functional requirements.

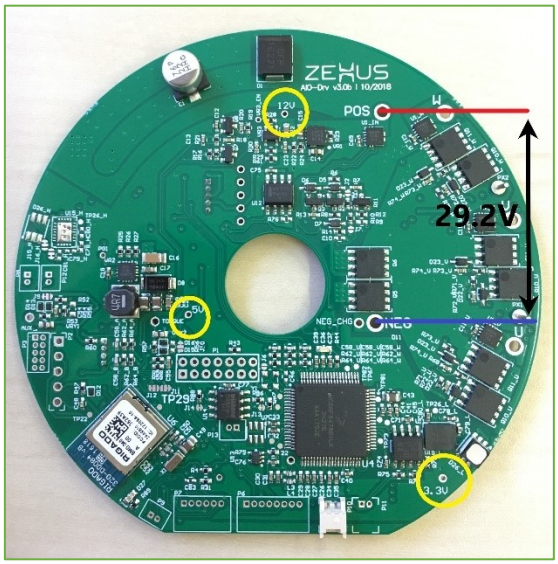
1. PCB Board on Test Bench

Firstly, the driver board must be inserted in the test bench. After that, the driver PCB must be supplied with a voltage equal to 29.2V (see figure below) in order to carry out all driver tests.

2. In-circuit Test

In the driver PCB there are three voltage levels: 3.3V, 5V and 12V.

The position of the test points where to find the voltage levels are shown in the figure below:



As first step, all the voltage levels should be tested. Therefore, a voltage tester is necessary to check all the voltage levels, which are shown in the above figure. The following requirements must be respected:

- 3.3V ± 5%
- 5V ± 5%
- 12V ± 5%

The driver should absorb a current equal to 18 mA ±25%.

3. Firmware Upload

If the test shown in the step#2 is passed, the firmware upload is performed.

In this phase, the Firmware should be uploaded on both microcontrollers (DSC and BLE), which are assembled on the hardware board.

Firmware is downloaded from Zehus Server. The procedure is explained in chapter #5.

There are two JTAG connectors on Driver board:

1. DSC JTAG
2. BLE JTAG

The following picture shows the location of the JTAG connectors on the Driver Board.

Firmware are uploaded using two JTAG programmers.

- For DSC: USB Multilink Universal FX (PE micro)
- For BLE: J-Link BASE (8.08.00) and J-Link 9-Pin Cortex-M Adapter (8.06.02)

Firmware can be uploaded by using two PC tools:

- **DSC Firmware**
  PROGDSC Flash/EEPROM Programmer Software for DSC devices
  http://www.pemicro.com/products/product_viewDetails.cfm?product_id=15320158&productTab=1

- **BLE Firmware**
  nRF5-Command-Line-Tools
  https://www.nordicsemi.com/Software-and-Tools/Development-Tools/nRF5-Command-Line-Tools

After the firmware is uploaded, the driver should absorb a current equal to 40 mA ±25%.

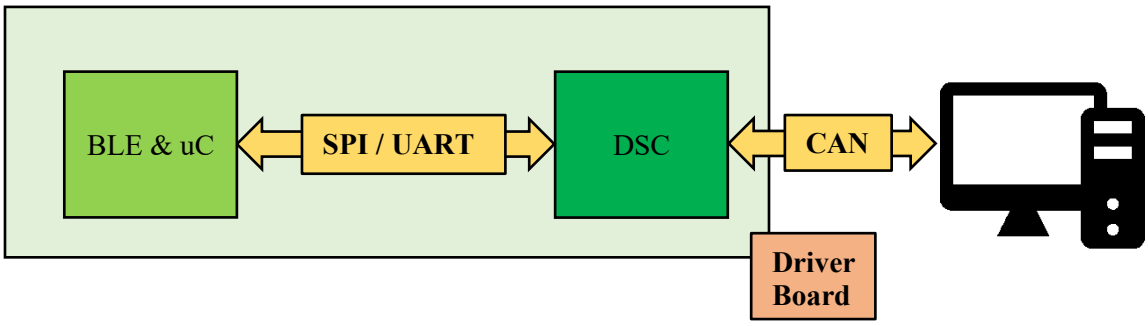
4. Functional Tests Description

All driver functional tests start using a specific command via CAN (ID 0x354, Byte 0: 0x08).

The CAN bus connector's location is shown in the following picture.

CAN packets involved in functional tests are:

- Debug IDs: 0x109, 0x1A2, 0x1A3, 0x1A4, 0x1A6, 0x1A8, 0x101, 0x102, 0x103



The tests are divided in:

- **Static** tests, which *are based on static value check.*
- **Dynamic** tests, which *are based on variable value check.*

The following CAN packet must be sent at the start of all functional tests:

Signals to write

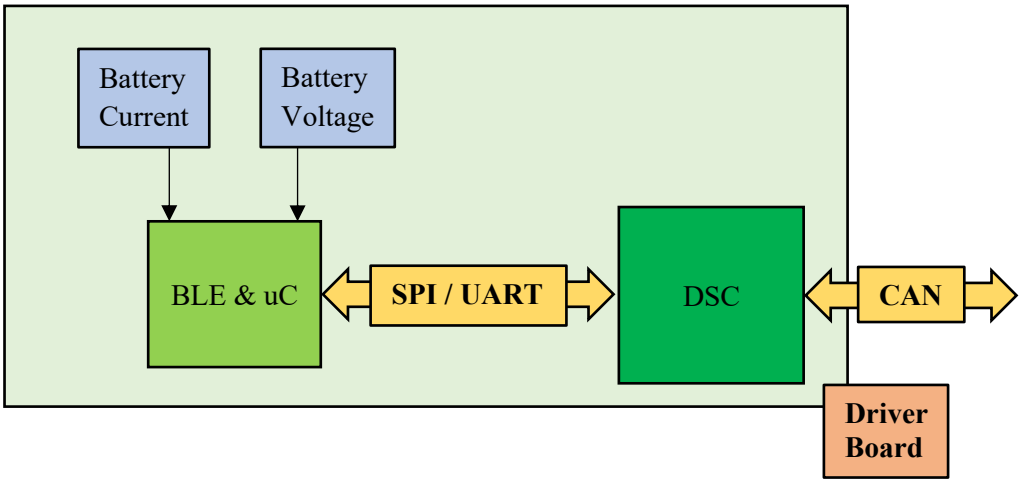
Action Command	CAN packet ID	CAN packet bytes
Start Quality Test Data	0x354	Byte 0: 0x08 Byte 1-7: 0x00

4.1 Static Tests

List of atomic tests based on a signal check.

4.1.1 Battery Current & Voltage Sensors Test

The aim of this test is to check the functioning of battery current and battery voltage sensors, which are assembled on the driver board and connected to the BLE & uC.



Test Description

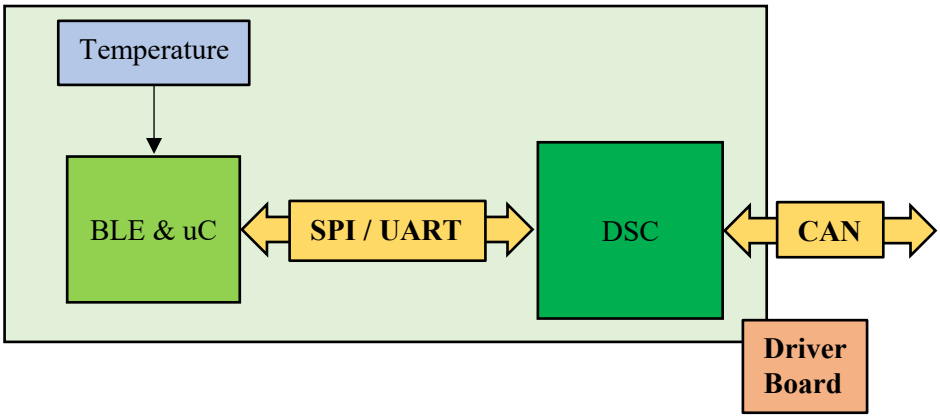
Test Type	Description
Battery Static Test	It checks that the battery current is between a certain range [-0.5A; 0.5A] for 2s.
Voltage Static Test	It checks that the battery voltage is equal to 29.2V ± 15% for 2s.

Signals to read

Data	CAN Packet ID	CAN Packet Bytes Data Type Data Conversion
Battery Current	0x101	Byte 3 (LSB) & Byte 4 (MSB) Int16 *0.01
Battery Voltage	0x101	Byte 0 (LSB) & Byte 1 (MSB) Int16 *0.01

4.1.2 Temperature Sensor test

The aim of this test is to check the functioning of the temperature sensor, which is assembled on the driver board and connected to the BLE & uC.



Test Description

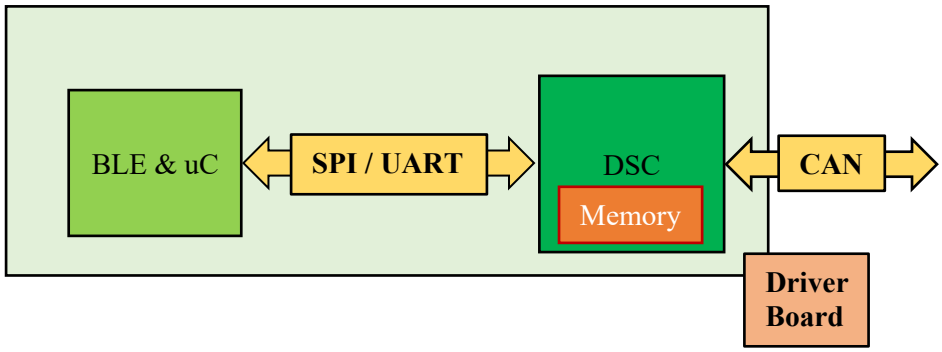
Test Type	Description
Driver Temperature Test	It checks if the value is between +15°C and +40°C.

Signals to read

Data	CAN packet ID	CAN Packet Bytes Data Type Data Conversion
Driver Temperature	0x109	Byte 6 Int8 *1

4.1.3 Total & Partial Kilometres Test

This test checks the values of Total Kilometres (Odometer) and Partial Kilometres (Trip) counters. These values are stored in an internal flash memory on the DSC.



Test Description

Test Type	Description
Total & Partial Kilometres Check	Total and partial kilometres should be equal to zero.

Signals to read

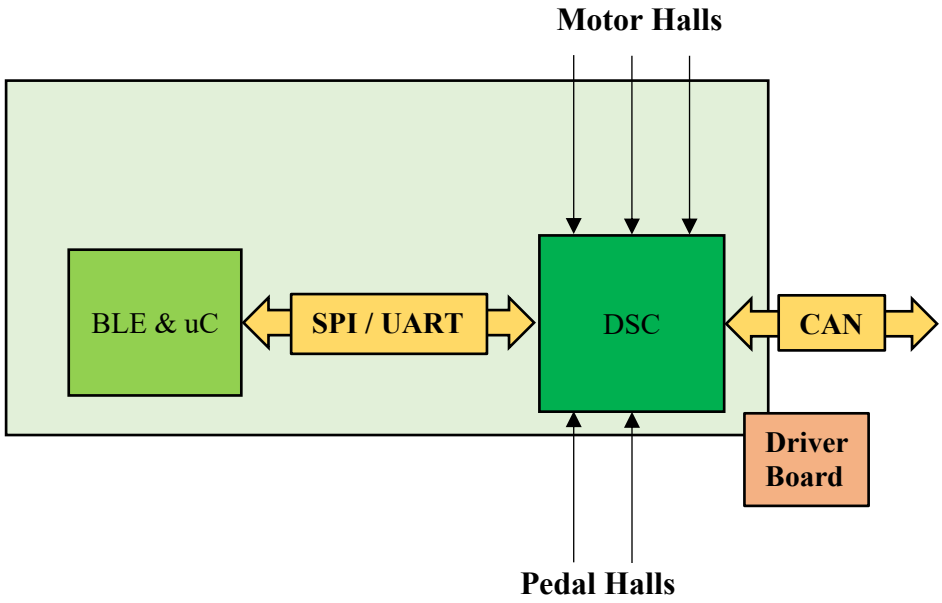
Data	CAN packet ID	CAN Packet Bytes Data Type Data Conversion
Total Kilometres	0x1A2	Byte 5 (MSB) and byte 4 (LSB) Uint16 *0.1
Partial Kilometres	0x1A2	Byte 7 (MSB) and byte 6 (LSB) Uint16 *0.1

4.2 Dynamic Tests

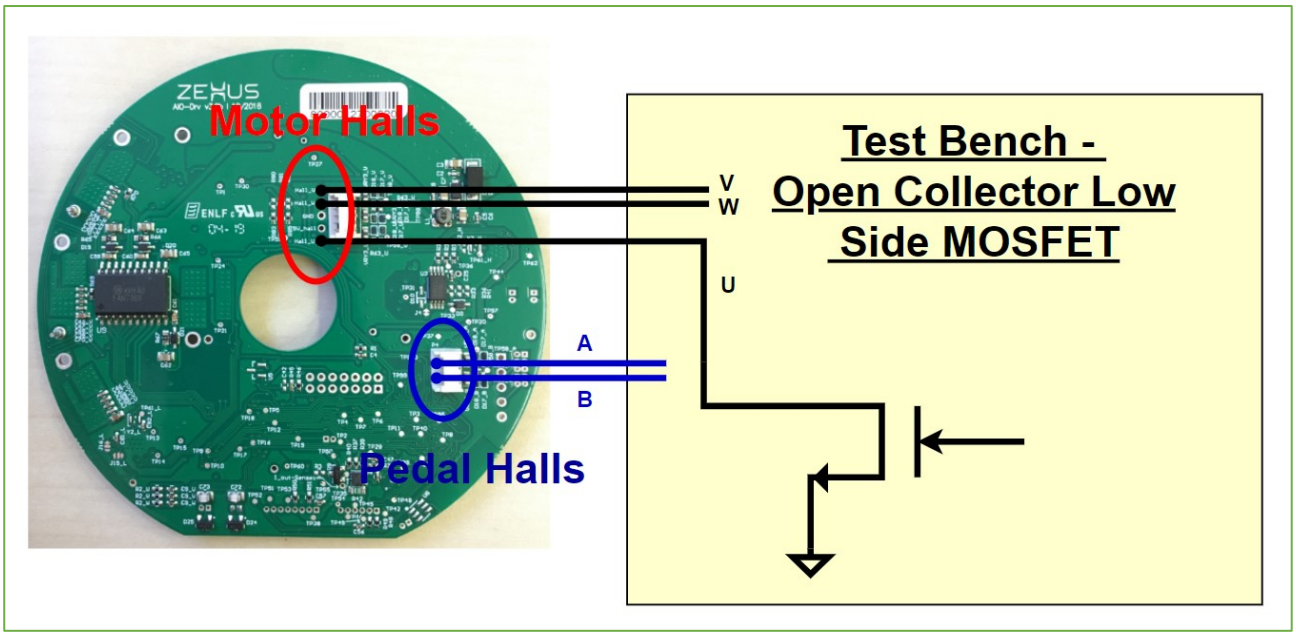
List of tests based on variable signal check.

4.2.1 Motor & Pedal Halls Test

This test is done in order to check the functioning of the Motor halls and pedal halls sensors that are assembled on the Driver board.



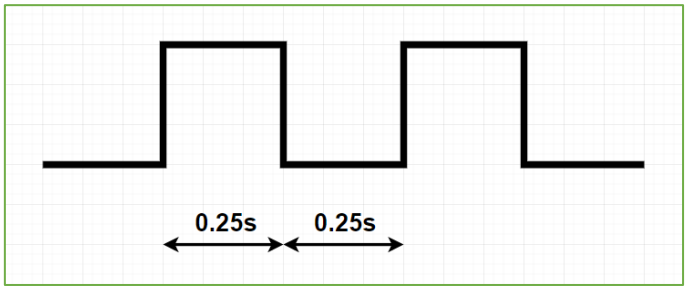
As shown in the following picture, the motor halls (U, V and W) and pedal halls (A and B) should be connected to a test bench that consists of an open collector low side MOSFET.



Test Description

Test Type	Description
Dynamic Test	Low side MOSFET switches (on-off-on)

For the dynamic test, the on-off-on switches should follow the following pattern, for each hall pin:



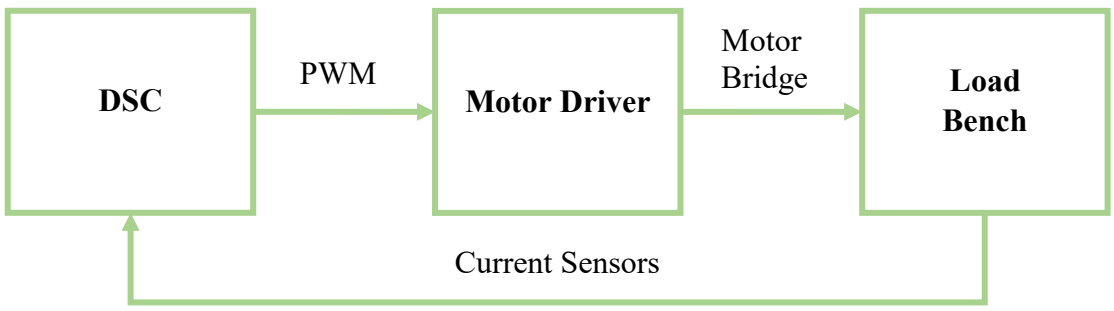
Signals to read

Data	CAN packet ID	CAN Packet Bytes Data Type Data Conversion
Motor hall U	0x101	Byte 2, bit 0 Uint8 *1
Motor hall V	0x101	Byte 2, bit 1 Uint8 *1
Motor hall W	0x101	Byte 2, bit 2 Uint8 *1
Pedal Hall A	0x103	Byte 6, bit 0 Uint8 *1
Pedal Hall B	0x103	Byte 6, bit 1 Uint8 *1

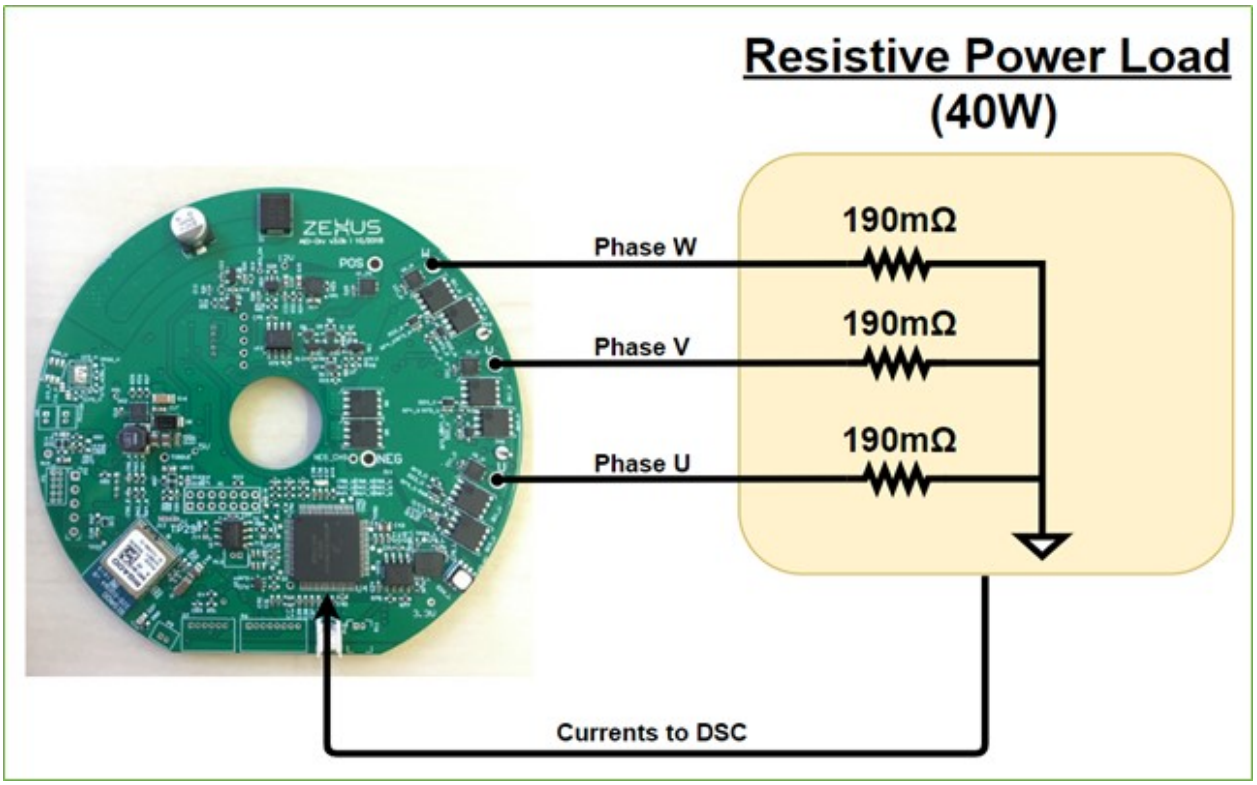
4.2.2 Half-Bridge Test

This test checks motor phase currents (U, V, W) and battery current after setting a specific current set-point.

This test also includes Battery Current Sensor test.



The load bench should be connected as follows:



Test Description

Test Type	Description
Motor Phase Current Dynamic Test	It checks if the value follows the switching table described below.
Battery Current Dynamic Test	It checks if the value follows the switching table described below.

Signals to write

Action Command	CAN packet ID	CAN packet bytes
Start Power Chain Test	0x354	Byte 0: 0x0A Byte 1-7: 0x00
Stop Power Chain Test	0x354	Byte 0-7: 0x00

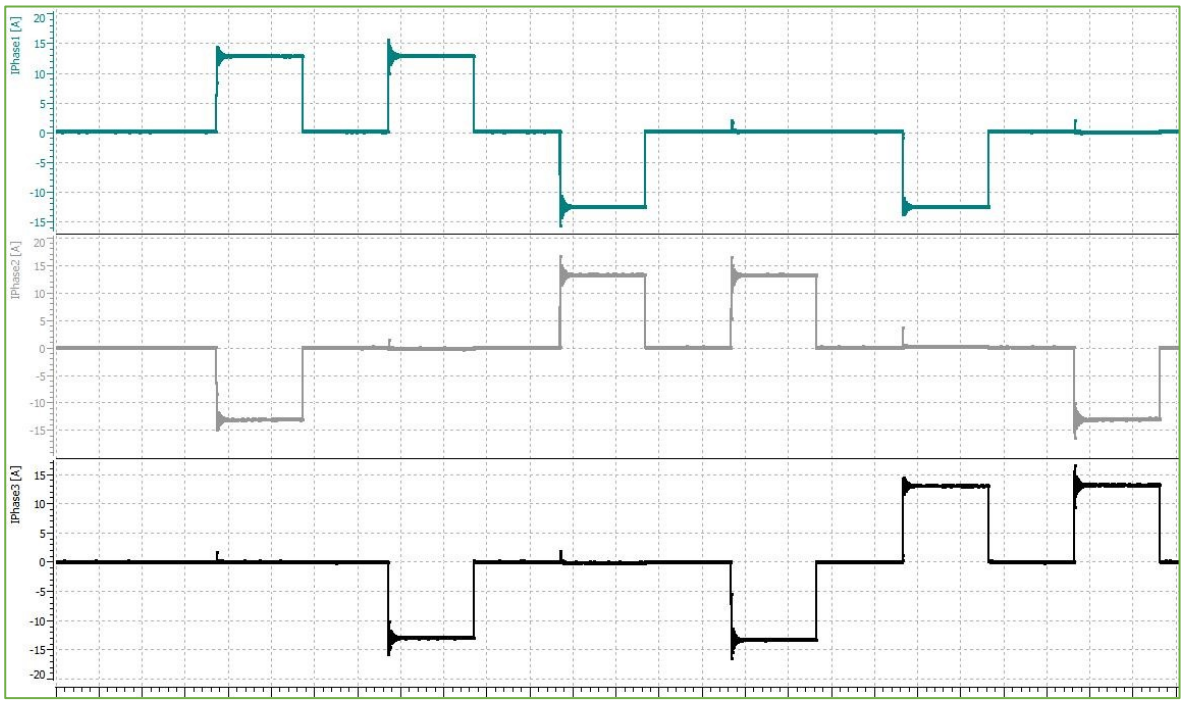
These commands enable/disable the Half-Bridge Test with Current sensor check (see table How to read signals Section for details).

Signals to read

Data	CAN packet ID	CAN Packet Bytes Data Type Data Conversion
Iphase1_A_U	0x103	Byte 0 (LSB) & Byte 1 (MSB) Int16 *0.00223796
Iphase2_B_V	0x103	Byte 4 (LSB) & Byte 5 (MSB) Int16 *0.00223796
Iphase3_C_W	0x103	Byte 4 (LSB) & Byte 5 (MSB) Int16 *0.00223796
Battery Current	0x101	Byte 3 (LSB) & Byte 4 (MSB) Int16 *0.01

How to read signals:

During the Half-Bridge test, the phase currents and the hall state must follow the patterns shown in the following figures.



During the execution, test checks phase current signs according to the following pattern:

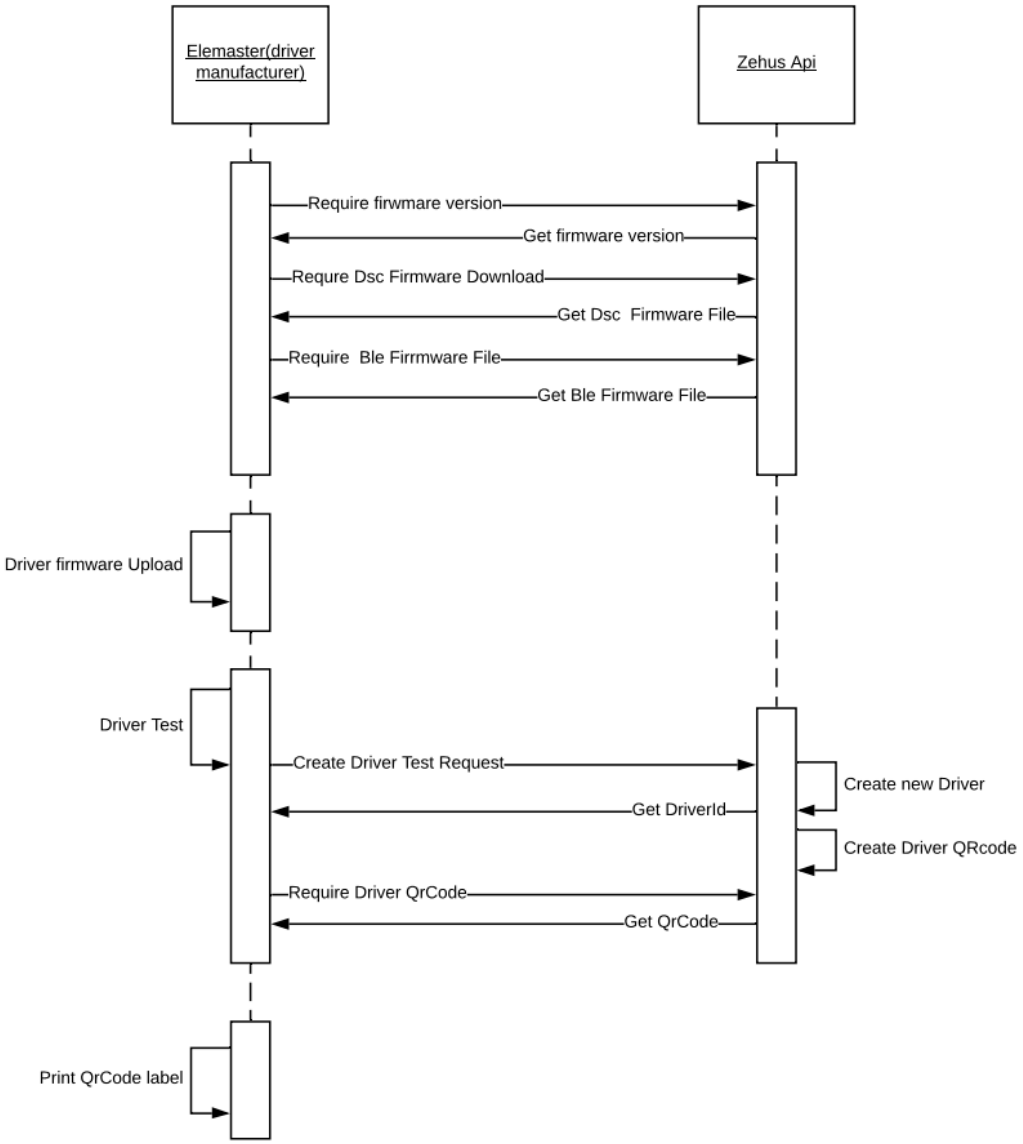
Hall State	Current 1 or U	Current 2 or V	Current 3 or W	Battery Current
3	+13A ± 15%	-13A ± 15%	0	+13A ± 15%
1	+13A ± 15%	0	-13A ± 15%	+13A ± 15%
4	-13A ± 15%	+13A ± 15%	0	+13A ± 15%
5	0	+13A ± 15%	-13A ± 15%	+13A ± 15%
6	-13A ± 15%	0	+13A ± 15%	+13A ± 15%
2	0	-13A ± 15%	+13A ± 15%	+13A ± 15%

5. Application Programming Interface (API) Specifications

Base URL for developing: https://zehusdrivermanufacturer-integr.azurewebsites.net

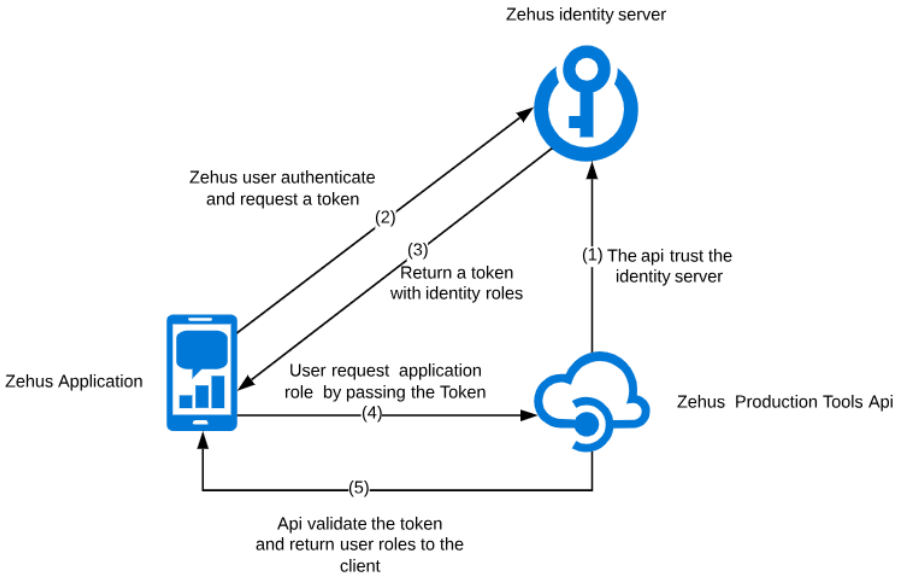
API documentation: https://zehusdrivermanufacturer-integr.azurewebsites.net/Documentation/Index

5.1 API sequence



5.2 Authentication

The API protected by the **OAuth 2.0** protocol with **OpenId Connect**. Therefore, to call them it is necessary to authenticate the user on the **Zehus Identity Provider**.



At the following URL the endpoints for implementing the protocol : Discoveryurl

It will be necessary to configure the client application that connects to our services. If it is a web application, it is necessary to provide a **redirect_uri** where the client will be sent to after the successful account authorization.

5.3 Get Firmware information

5.3.1 Get FirmwareProductType

Calling the following API will return the information about the firmware Product types:

End Point	Type
/api/v{version}/FirmwareProductTypes?page={page}&rows={rows}	GET

Http Status Code 200:

The information returned visible in the following json snippet, the most important are:

- **FirmwareProductTypeId**: identifier required for create a new Driver
- **FirmwareProductTypology:**
    1. **Virgin** if the firmware contains only the test logic**.**
    2. **Application** if the firmware contains the final product logic
- **FirmwareTypeId:** identifier required for download the firmware file
- **ComponentType:**
    1. **DriverDsc**
    2. **DriverBle**
- **LastAppFirmwareVersion:** the current application firmware version
- **LasBootloaderFirmwareVersion:** the current bootloader firmware version
- **OtherFirmwareVersion:** the current softdevice firmware version (only if the **ComponentType** is ***DriverBle***)


```
{
  "PaginationViewModel": {
    "Rows": 0,
    "Page": 0,
    "Sort": "string",
    "TotalPages": 0
  },
  "Data": [
    {
```

```
      "FirmwareProductTypeId": "00000000-0000-0000-0000-000000000000",
      "FirmwareProductTypeName": "string",
      "LastFirmwareVersions": [
        {
          "FirmwareTypeId": "00000000-0000-0000-0000-000000000000",
          "FirmwareTypeName": "string",
          "ComponentType": "string",
          "LastAppFirmwareVersion": "string",
          "LastBootloaderFirmwareVersion": "string",
          "OtherFirmwareVersion": "string",
          "LastFirmwareFileName": "string",
          "UpdateChannelTypes": [
            "string"
          ],
          "OtherFirmwareVersionDetails": "string"
        }
      ],
      "FirmwareProductTypology": "string"
    }
  ]
}
```


Http Status Code 401 - Unauthorized request

Http Status Code 403 - User not allowed to call the api.

Http Status Code 500:

ErrorKey	Description
InternalServerErrorKey	Internal API error

5.3.2 Get Firmware file

Calling the following API, you can download the firmware file :

End Point	Type
/api/v{version}/FirmwareType/{id}/Firmware	GET

Http Status Code 200:

Response Hader

```
{
  "date": "Wed, 17 Apr 2019 14:55:45 GMT",
  "content-type": "application/octet-stream"
  "content-disposition": "attachment; filename={filename}",
  "content-length": "1796",
}
```

Http Status Code 401 - Unauthorized request

Http Status Code 403 - User not allowed to call the api.

Http Status Code 500:

ErrorKey	Description
InternalServerErrorKey	Internal API error

5.4 Save test result

Calling the following API, a driver test result will be stored in our system:

End Point	Type
/api/v{version}/Driver/Test	POST

5.4.1 New Driver

By passing the following parameters to the API a new driver will be created before saving the test result. The firmware information can be retrieved by calling the firmware api

```
POST /api/v1/Driver/Test HTTP/1.1
 {
  "BleMacAddress": "string",
  "FirmwareProductTypeId": "00000000-0000-0000-0000-000000000000",
  "DscApplicationFirmwareVersion": {
    "Major": 1,
    "Minor": 0,
    "Fix": 0
  },
  "DscBootloaderFirmwareVersion": {
    "Major": 1,
    "Minor": 0,
    "Fix": 0
  },
  "BleApplicationFirmwareVersion": {
    "Major": 1,
    "Minor": 0,
    "Fix": 0
  },
  "BleBootloaderFirmwareVersion": {
    "Major": 1,
    "Minor": 0,
    "Fix": 0
  },
  "BleSoftDeviceFirmwareVersion": {
    "Major": 1,
    "Minor": 0,
    "Fix": 0
  },
```

```
  "Success": true,
}
```

How to retrieve MAC Address from BLE device

- Read DEVICE Address from memory using JTAG of BLE microcontroller:
  - Read Device Address Part 0 using nrfjprog –memrd command

```
> nrfjprog --memrd 0x100000A4 --n 4
0x100000A4: 63A60992                                |...c|
```

  - Read Device Address Part 1 using nrfjprog –memrd command

```
> nrfjprog --memrd 0x100000A8 --n 4
0x100000A8: 1F7485B6                                |..t.|
```

- Generate MAC address from DEVICE address
  - Concatenate Device Address Part 1 and Part 0

    1F7485B6 63A60992

  - Remove first 2 byte (4 chars)

    1F7485B6 63A60992 => 85B663A60992

  - Do "Or" with C00000000000

    85B663A60992 |
    C00000000000 =
    C5B663A60992

- MAC Address is: C5B663A60992

5.4.2 Existing Driver

By passing the following parameters a new test result will be stored for the provided Driverid.

```
POST /api/v1/Driver/Test HTTP/1.1
{
```

```
    "DriverId": "00000000-0000-0000-0000-000000000000"  --Existing DriverId,
    "Success": true
}
```

After calling the previous API a driver ID will be returned as a successful response.

Http Status Code 200:

```
{
  "Data": [
    {
      "Id": "00000000-0000-0000-0000-000000000000"
    }
  ]
}
```

In case of any error the possible responses are the following:

Http Status Code 400:

ErrorKey	Description	Action
DriverNotFoundKey	If the passed DriverId does exist in the db	Add new test on existing Driver
ModelValidationErrorKey	If the passed model is not valid	Create new Driver
DriverAlreadyCreatedKey	If the passed bleMacAddres exist in the db	Create new Driver
FirmwareProductTypeNotFoundKey	If the passed FirmwareProductTypeId does exist in the db	Create new Driver

Http Status Code 401 - Unauthorized request

Http Status Code 403 - User not allowed to call the api.

Http Status Code 500:

ErrorKey	Description
InternalServerErrorKey	Internal API error

5.5 Get QRCode label

Calling this API will be possible to download the QRCode file for the given DriverId

End Point	Type
/api/v{version}/Driver/{id}/Label?mimeType ={type}	GET

Type
image/png
application/pdf

Http Status Code 200:

```
{
  "content-type": "application/pdf",
  "content-disposition": "attachment; filename=DR00000003.pdf",
  "content-length": "1796",
}
```

Http Status Code 401 - Unauthorized request

Http Status Code 403 - User not allowed to call the api.

Http Status Code 500:

ErrorKey	Description
InternalServerErrorKey	Internal API error

5.6 Get list of Drivers

Calling this API will be returned a pagination with the date of the created drivers.

End Point	Type
/api/v{version}/Drivers? page={page}&rows={rows}	GET

Optional Parameter

- **DriverId**: entering this value, the list will only return the information of the relative driver
- **SerialNumber**: entering this value, the list will only return the information of the relative driver. This value can be retrieved by scanning the QrCode label generated during component creation

Http Status Code 200:


```
{
  "PaginationViewModel": {
    "Rows": 0,
    "Page": 0,
    "Sort": "string",
    "TotalPages": 0
  },
  "Data": [
    {
      "DriverId": "00000000-0000-0000-0000-000000000000",
      "SerialNumber": "string",
      "MacAddress": "string",
      "DscApplicationFirmwareVersion": "string",
      "DscBlFirmwareVersion": "string",
      "BleApplicationFirmwareVersion": "string",
      "BleBlFirmwareVersion": "string",
      "BleSoftDeviceFirmwareVersion": "string",
      "CreationDateUtcMillis": 0,
      "LastTestDateUtcMillis": 0,
      "FirmwareProductTypeId": "00000000-0000-0000-0000-000000000000",
      "FirmwareProductTypology": "string"
    }
  ]
}
```

Http Status Code 401 - Unauthorized request.

Http Status Code 403 - User not allowed to call the API.

Http Status Code 500:

ErrorKey	Description
InternalServerErrorKey	Internal API error