# API communication protocol

*IQ Toolkit Calibration Plate Changer*

|  |  |  |
|---|---|---|
| Opdrachtgever | : | Philips Health Tech |
| Date | : | 23-03-2017 |
| Project Number | : | 1 |

Sign client :

### Version

| Version | Date | Author(s) | Changes | Status |
|---|---|---|---|---|
| 1.0 | 22-03-2017 | LJ, LVR | Document creation | In progress |
| 1.1 | 23-03-2017 | LJ, LVR | Implementation | In progress |
| 1.2 | | | | |
| 1.3 | | | | |
| 1.4 | | | | |
| 1.5 | | | | |

### Distribution

| Version | Date | To |
|---|---|---|
| | | |
| | | |
| | | |

# Protocol definition

The protocol uses TCP/IP as a communication layer to send messages between the embedded and the PC platform. A protocol was defined in which functionality is specified. It also defines the parameters, the priority and the sender of the message.

Two ways are defined to describe the protocol: in the XML and JSON formats. In consultation with PTT group 1, we have decided to use the JSON format.

```xml
<?xml version='1.0' encoding='utf-8'?>
<function>
  <priority>1</priority>
  <sender>ip</sender>
  <functionName>myFunction1</functionName>
  <returnType>int</returnType>
  <parameters>
    <parameter>
      <name>param_1</name>
      <dataType>int</dataType>
      <value>123</value>
    </parameter>
    <parameter>
      <name>param_2</name>
      <dataType>std::string</dataType>
      <value>ding</value>
    </parameter>
  </parameters>
</function>
```

Protocol in XML format.

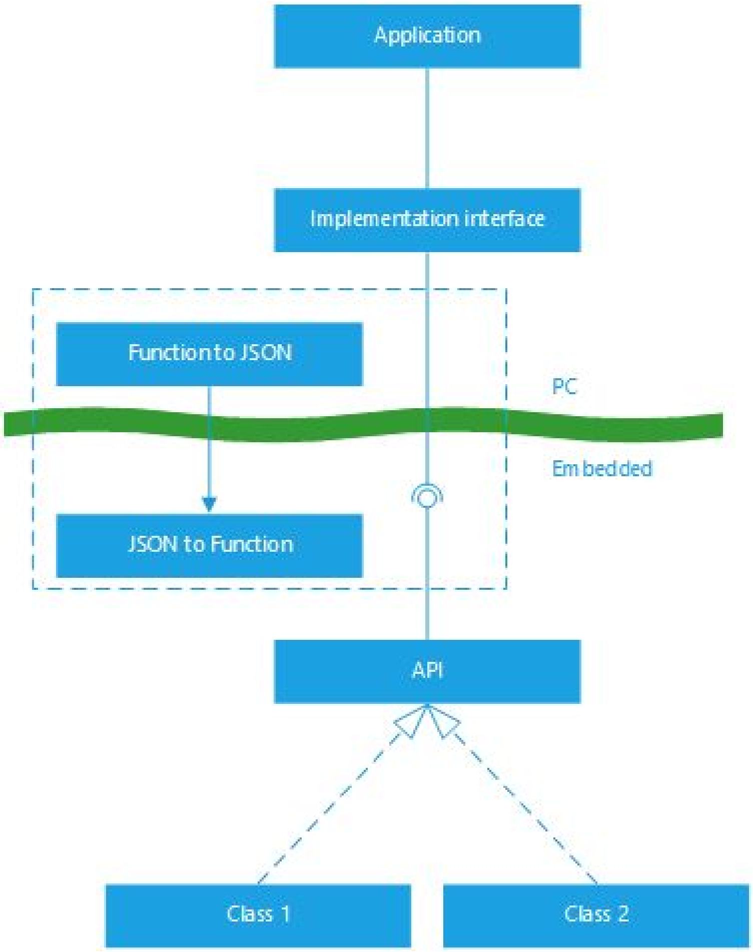
```json
{
  "function": {
    "priority": "1",
    "sender": "ip",
    "functionName": "myFunction1",
    "returnType": "int",
    "parameters": [{
      "name": "param_1",
      "dataType": "int",
      "value": "123"
    }, {
      "name": "param_2",
      "dataType": "std::string",
      "value": "ding"
    }]
  }
}
```

Protocol in JSON format.

# Implementation

To control the calibration system, we have to communicate between the user interface and the hardware. We designed an API which implements the functions offered by the hardware. The implementation interface will contain the functions offered by the API. These functions can be used by the user application.

When a function, provided by the API, is called in the PC application, the implementation interface will generate a JSON object which represents the function call on the PC. This JSON object will then be sent to the embedded platform.



A TCP/IP socket server is set up on the embedded platform. This socket server will be listening to incoming messages from the PC platform. These messages will be in JSON format. When a message is received, it will be decoded and the desired function (provided by the API) will be called. These functions will always have a return value. This value will be parsed to JSON and then sent back to the PC platform. On the PC platform, this JSON message will be parsed back to useful data.