FreeMASTER Lite

Getting Started with JSON-RPC Protocol
From Scripting to Visual Dashboards with
Python and JavaScript

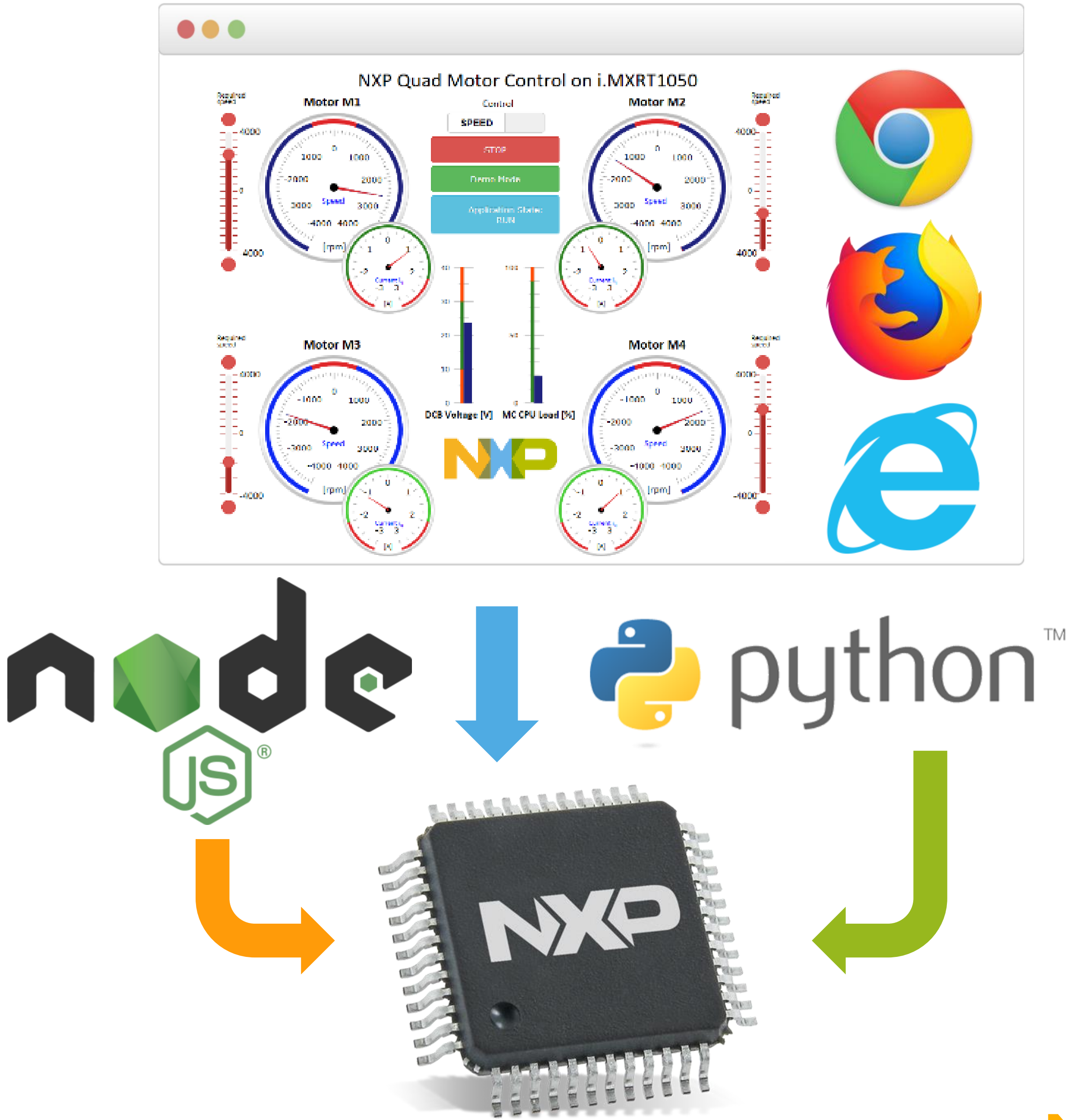
Iulian Stan
Software Engineer, AP System Tools

MAY 2020

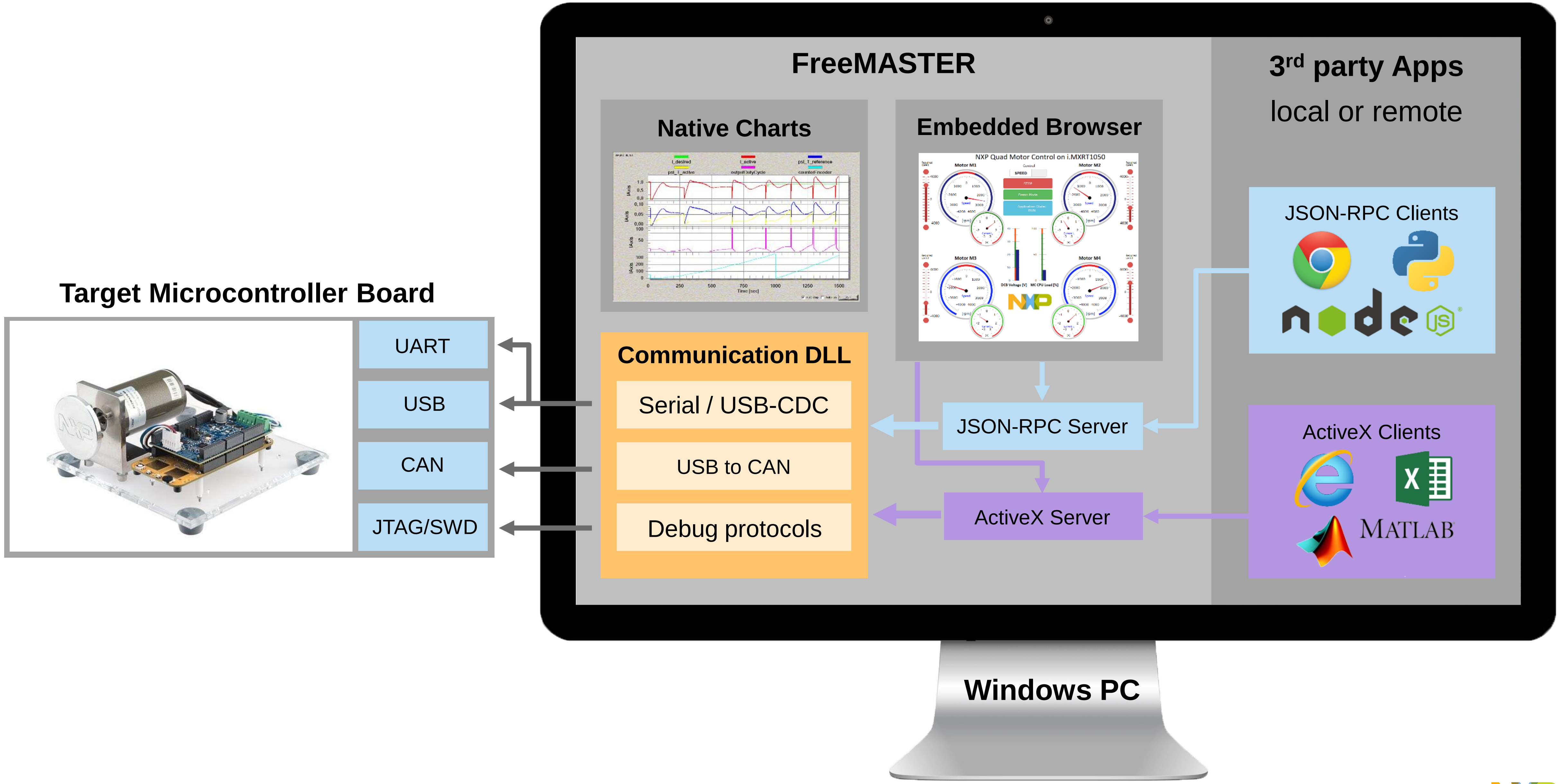
SECURE CONNECTIONS
FOR A SMARTER WORLD

AGENDA

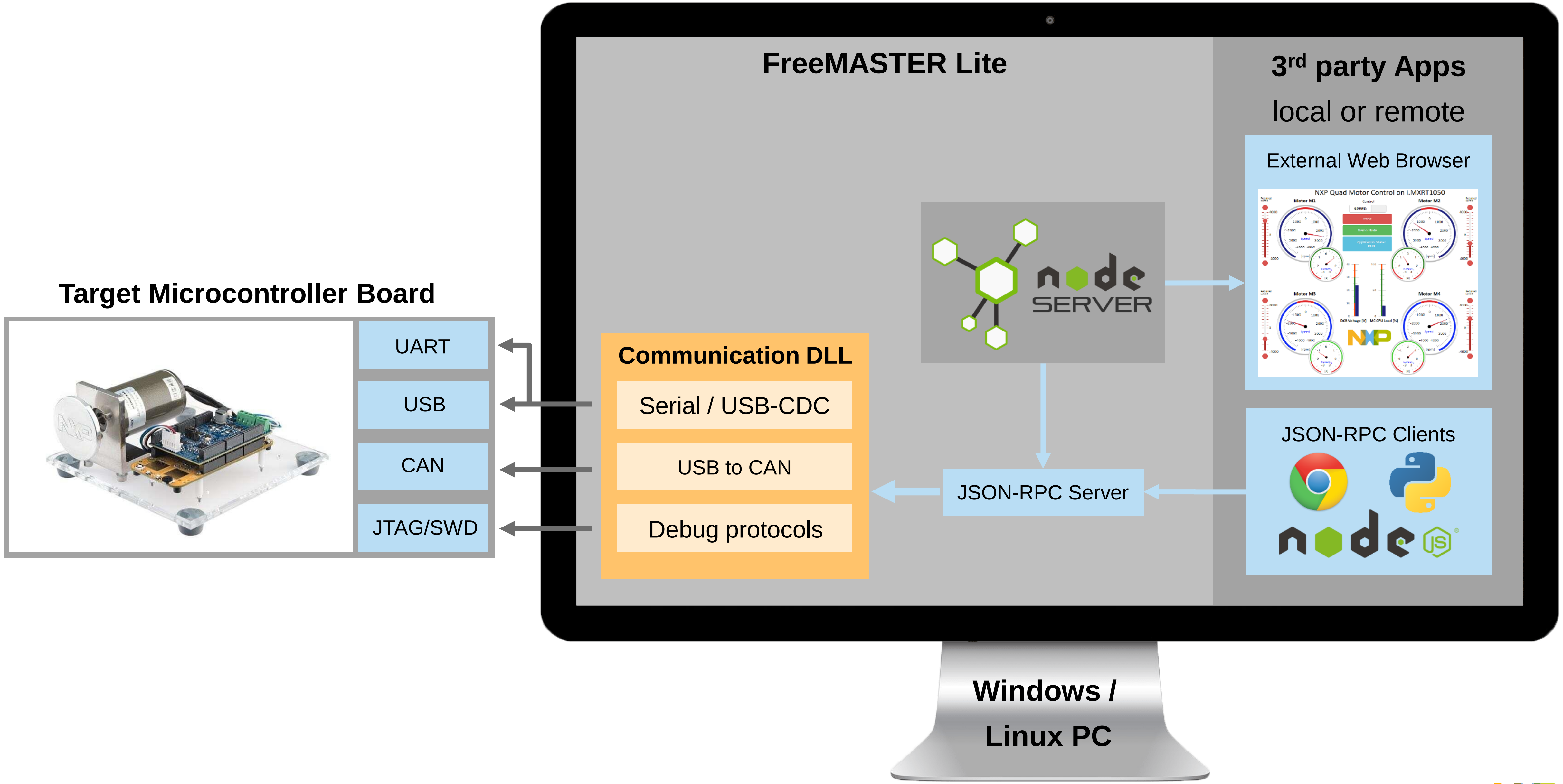
- FreeMASTER 3.0
  - FreeMASTER
  - FreeMASTER Lite
- JSON-RPC
- FreeMASTER Lite
  - Package Content
  - Running the Tool
  - Configuration File
- Coding Examples
  1. Python Scripting
  2. NodeJS Scripting
  3. Web Dashboard



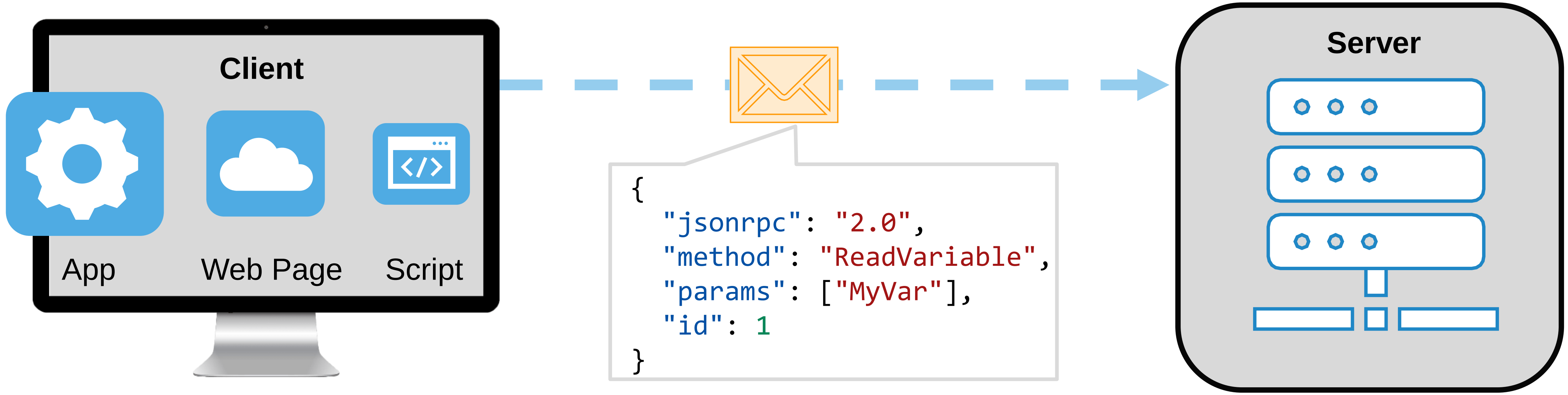
FREEMASTER 3.0



FREEMASTER 3.0



Remote **P**rocedure **C**all encoded as JSON objects

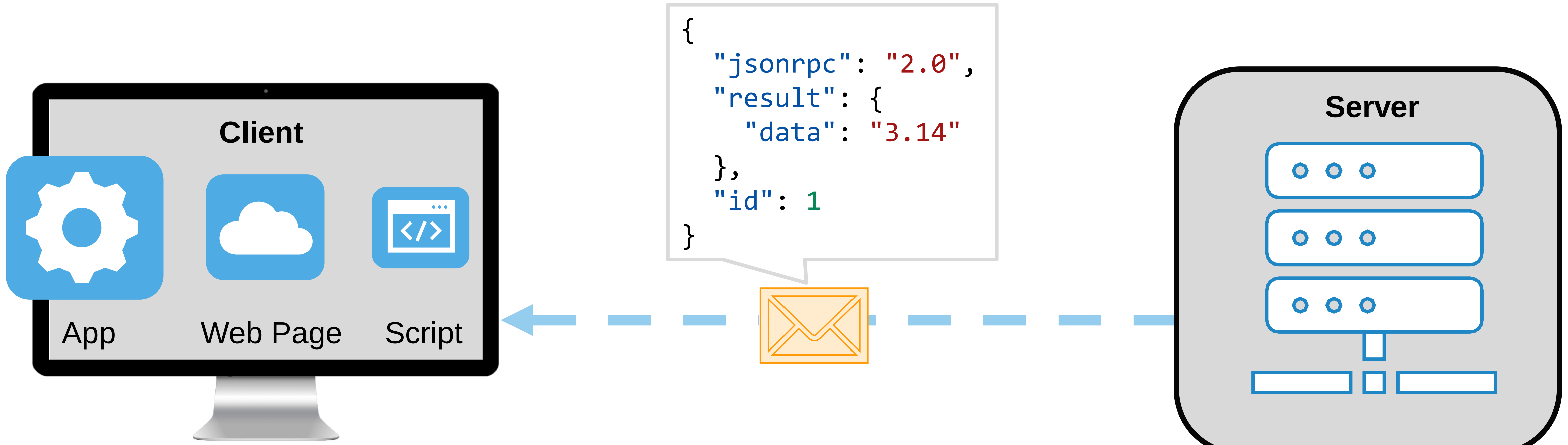


```
{
  "jsonrpc": "2.0",
  "method": "ReadVariable",
  "params": ["MyVar"],
  "id": 1
}
```

Client specifies method name and input arguments to be invoked on the server side

JSON-RPC

Remote **P**rocedure **C**all encoded as JSON objects



```
{
  "jsonrpc": "2.0",
  "result": {
    "data": "3.14"
  },
  "id": 1
}
```

Server replies with method invocation returned data (if any), or error code

FREEMASTER LITE - PACKAGE CONTENT

COM LIBRARY & SERVICE APP

mcbcom.dll

- *Fully compatible with FreeMASTER desktop application*
- *Same com. plugins on Windows*
- *Serial communication on Linux*

&

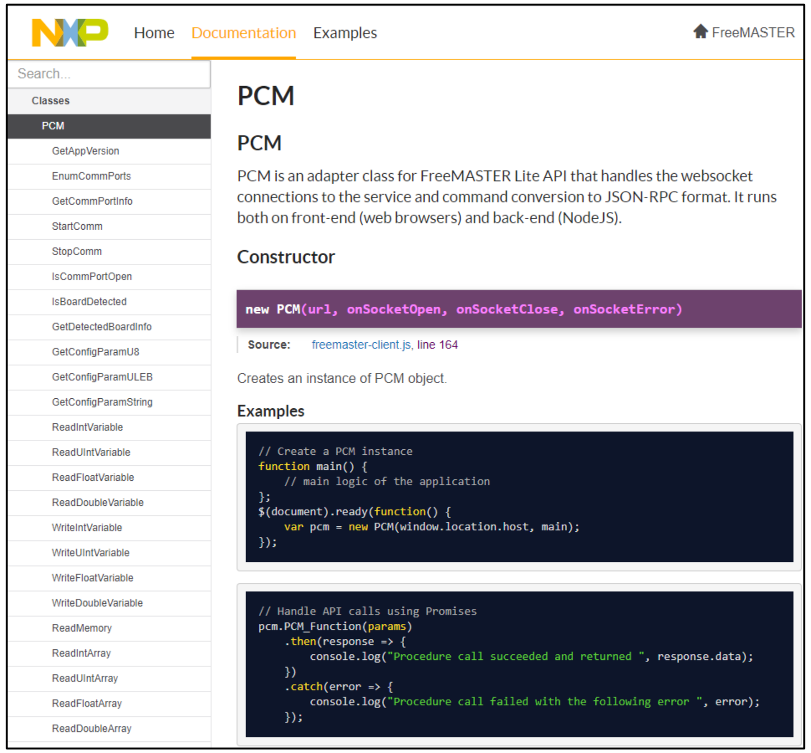
node.exe

- *NodeJS powered service*
- *Web server (static content & custom web applications)*
- *WebSocket server (JSON-RPC API)*

DOCUMENTED CLIENT LIBRARY

freemaster-client.js

- *Vanilla JavaScript library running both on front-end and back-end*
- *JSDoc documentation*



PCM

PCM

PCM is an adapter class for FreeMASTER Lite API that handles the websocket connections to the service and command conversion to JSON-RPC format. It runs both on front-end (web browsers) and back-end (NodeJS).

Constructor

```
new PCM(url, onSocketOpen, onSocketClose, onSocketError)
```

Source: freemaster-client.js, line 164

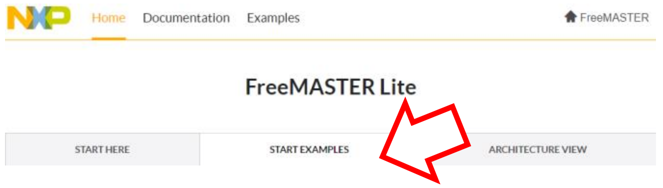
Creates an instance of PCM object.

Examples

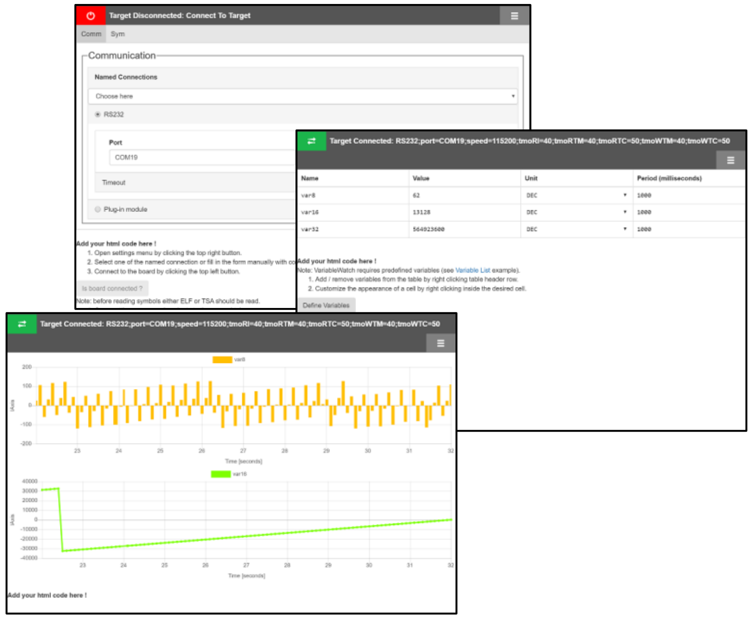
```
// Create a PCM instance
function main() {
    // main logic of the application
};
$(document).ready(function() {
    var pcm = new PCM(window.location.host, main);
});
```

```
// Handle API calls using Promises
pcm.PCM_Function(params)
    .then(response => {
        console.log("Procedure call succeeded and returned ", response.data);
    })
    .catch(error => {
        console.log("Procedure call failed with the following error ", error);
    });
```

READY TO USE EXAMPLES



- *Ready to use widget-like examples*
- *Features: variables read & write, scope & recorder functionalities*



Add your html code here !
1. Open settings menu by clicking the top right button.
2. Select one of the named connection or fill in the form manually with co
3. Connect to the board by clicking the top left button.

Note: before reading symbols either ELF or TSA should be read.

Target Connected: RS232;port=COM19;speed=115200;tmoRI=40;tmoRTM=40;tmoRTC=50;tmoWTM=40;tmoWTC=50

Name	Value	Unit	Period (milliseconds)
var8	62	DEC	1000
var16	13128	DEC	1000
var32	564923600	DEC	1000

Add your html code here !
Note: VariableWatch requires predefined variables (see Variable List example).
1. Add / remove variables from the table by right clicking table header row.
2. Customize the appearance of a cell by right clicking inside the desired cell.

Target Connected: RS232;port=COM19;speed=115200;tmoRI=40;tmoRTM=40;tmoRTC=50;tmoWTM=40;tmoWTC=50

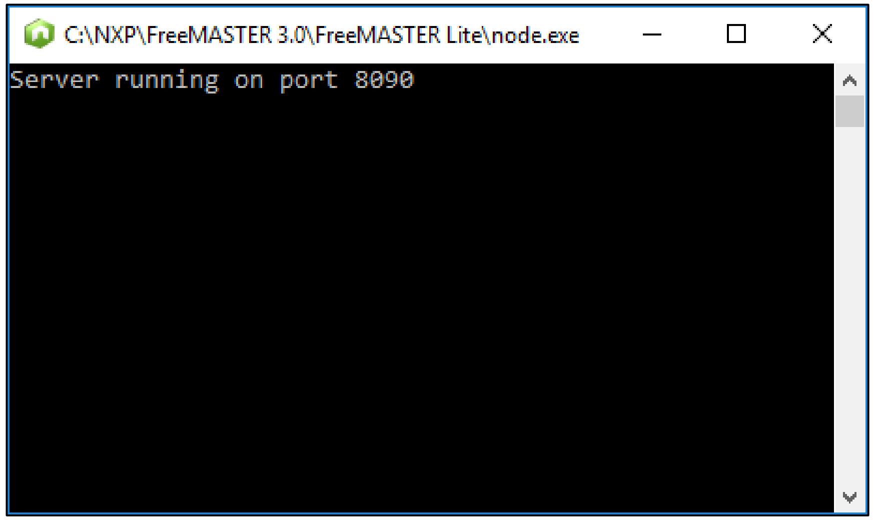
Add your html code here !

FREEMASTER LITE - RUNNING THE TOOL

```
C:\NXP\FreeMASTER 3.0\FreeMASTER Lite\node.exe
```

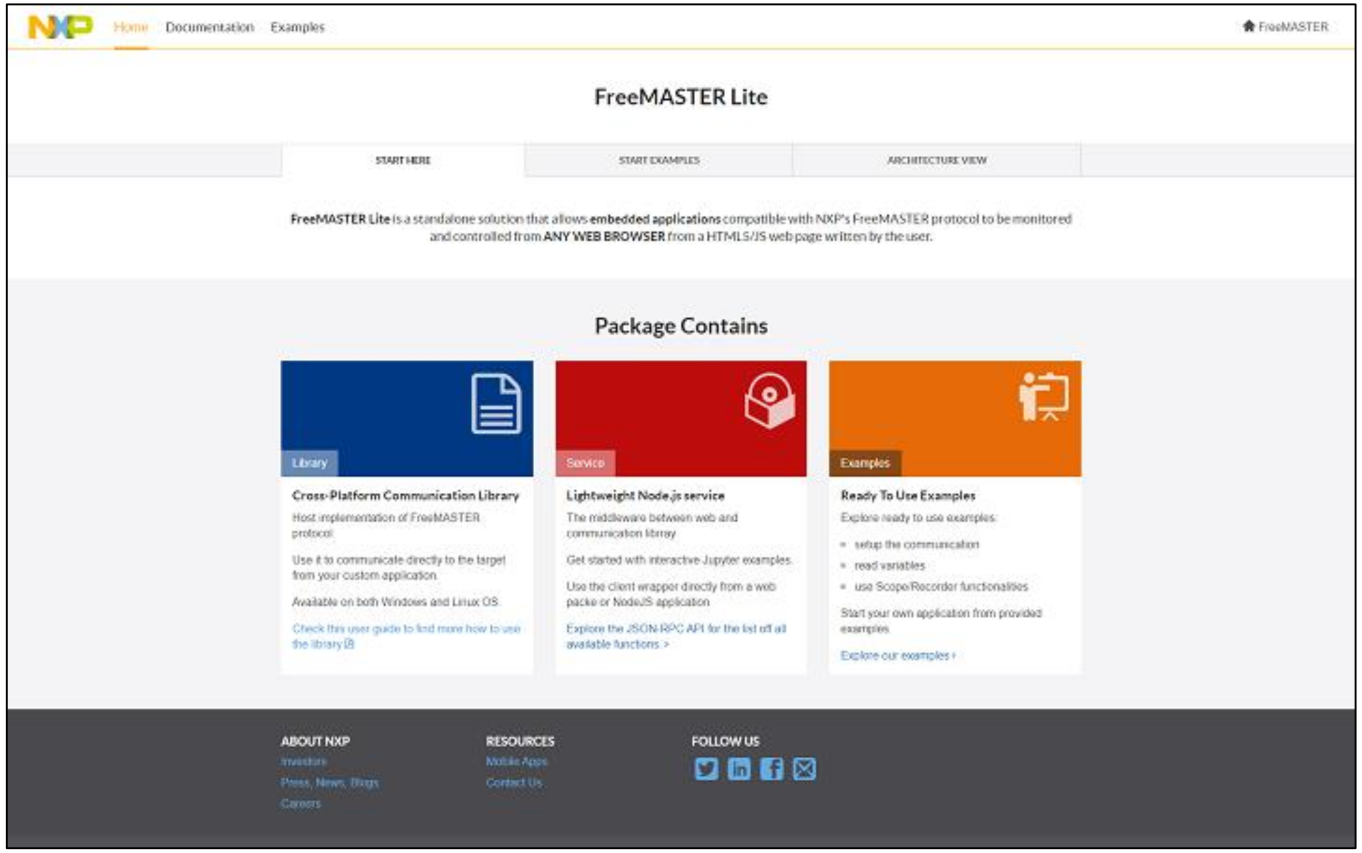
http://localhost:8090/

node.exe



C:\NXP\FreeMASTER 3.0\FreeMASTER Lite\node.exe

```
Server running on port 8090
```



FreeMASTER Lite

FreeMASTER Lite is a standalone solution that allows **embedded applications** compatible with NXP's FreeMASTER protocol to be monitored and controlled from **ANY WEB BROWSER** from a HTML5/JS web page written by the user.

Package Contains

Cross-Platform Communication Library

Host implementation of FreeMASTER protocol

Use it to communicate directly to the target from your custom application.

Available on both Windows and Linux OS.

Check this user guide to find more how to use the library

Lightweight Node.js service

The middleware between web and communication library

Get started with interactive Jupyter examples.

Use the client wrapper directly from a web packe or NodeJS application

Explore the JSON-RPC API for the list off all available functions >

Ready To Use Examples

Explore ready to use examples:

- setup the communication
- read variables
- use Scope/Recorder functionalities

Start your own application from provided examples.

Explore our examples >

ABOUT NXP
Investors
Press, News, Blogs
Careers

RESOURCES
Mobile Apps
Contact Us

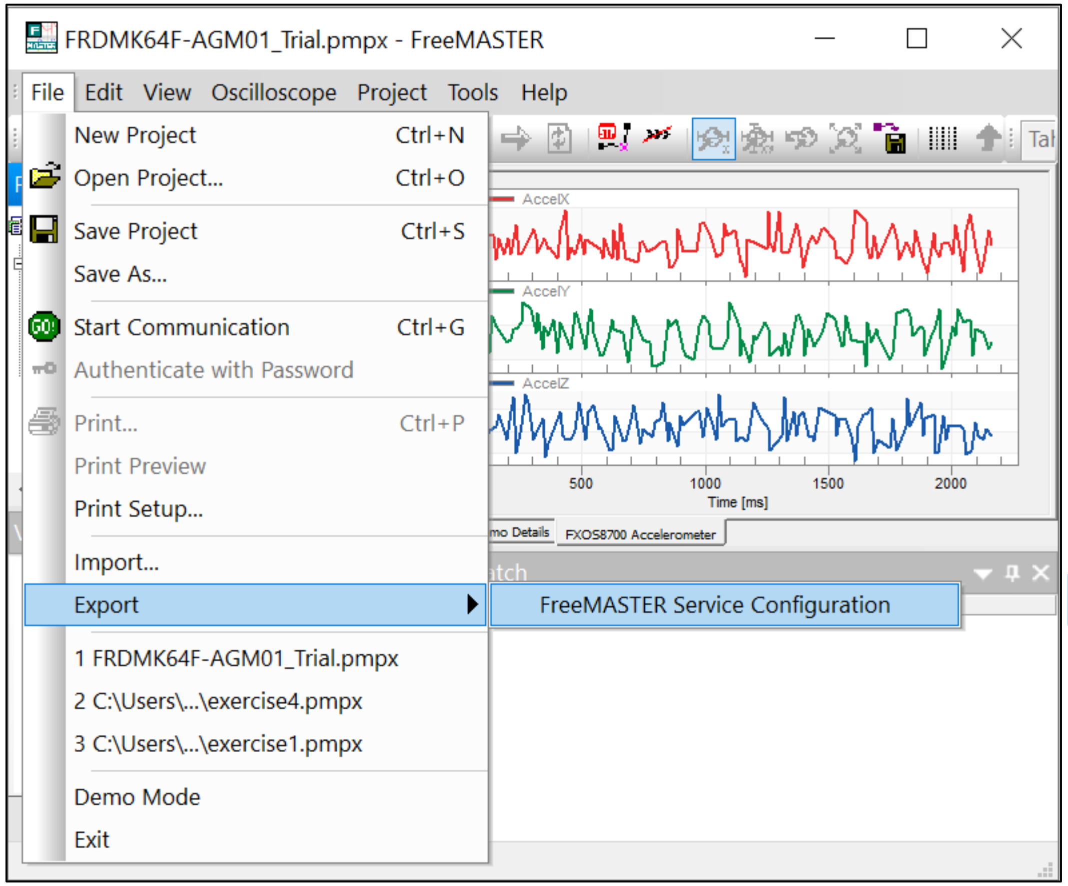
FOLLOW US

Server starts with default config

FREEMASTER LITE - CONFIGURATION FILE

Exporting project configuration from FreeMASTER

File → Export → FreeMASTER Service Configuration



```
{
  "port": 41000,
  "web_root": "path_to_static_web_content",
  "dirs": [
    {
      "path": "C:\\Temp",
      "opts": "r",
      "exts": [
        ".txt",
        ".json"
      ]
    }
  ],
  "connections": [
    {
      "name": "S32K144 UART",
      "description":  "UART over USB (Open SDA)",
      "connection_string": "RS232;port=COM6;speed=115200",
      "elf": ""
    }
  ],
  "variables": [
    {
      "name": "Potentiometer",
      "addr": "potmtr",
      "size": 4,
      "type":  "uint"
    }
  ]
}
```

Service config

File IO config

Target connection config

Target project config

EXAMPLE 1
PYTHON SCRIPTING

```
C:\NXP\FreeMASTER 3.0\FreeMASTER Lite\
scripting examples\Python
```

- Install Prerequisites:

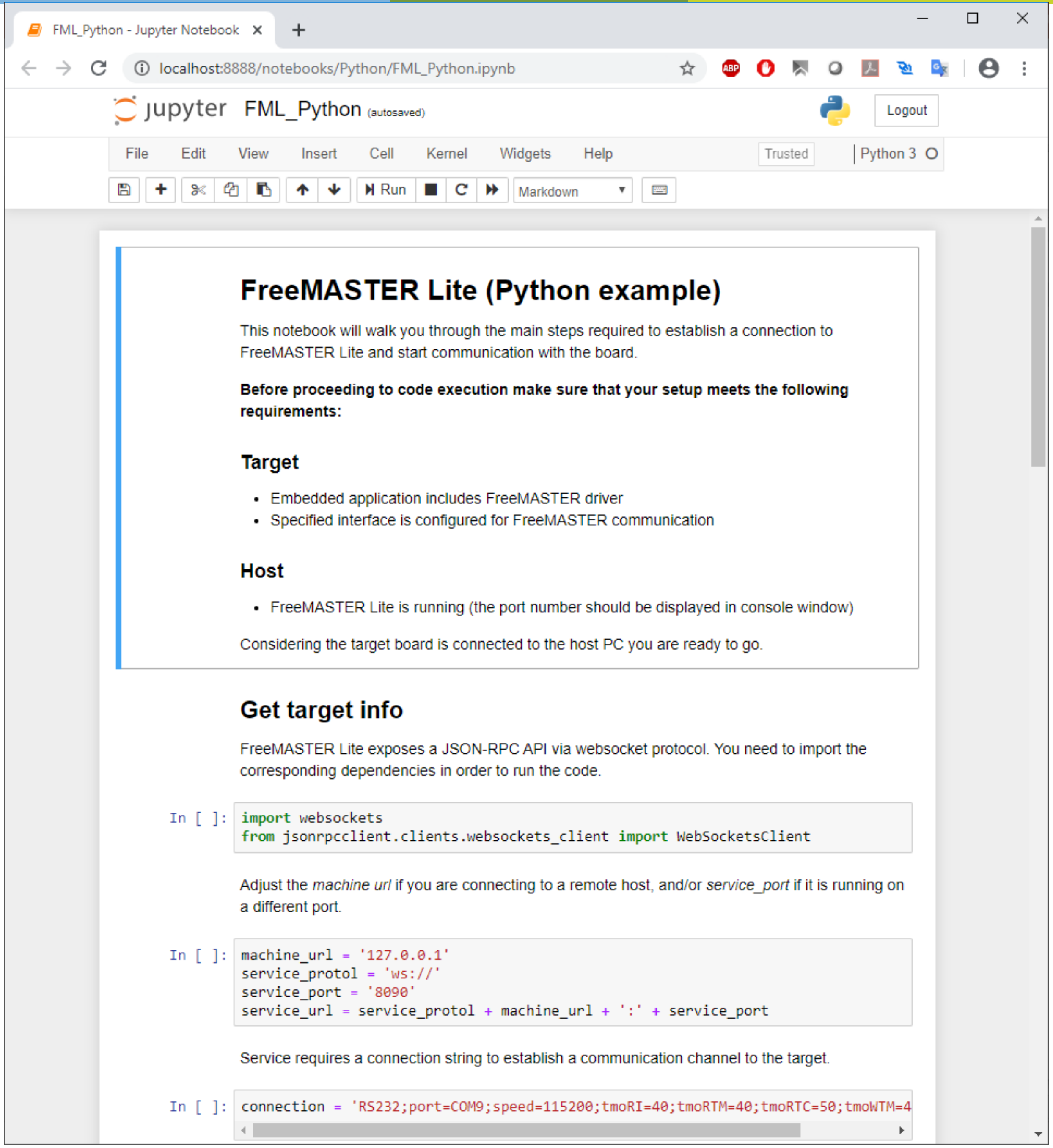
```
_>pip install -r
requirements.txt
```

- Start Jupyter notebook

```
_>jupyter notebook
```

- Use browser based IDE

to interact with the board


FML_Python - Jupyter Notebook
localhost:8888/notebooks/Python/FML_Python.ipynb
jupyter FML_Python (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

FreeMASTER Lite (Python example)

This notebook will walk you through the main steps required to establish a connection to FreeMASTER Lite and start communication with the board.

Before proceeding to code execution make sure that your setup meets the following requirements:

Target

- Embedded application includes FreeMASTER driver
- Specified interface is configured for FreeMASTER communication

Host

- FreeMASTER Lite is running (the port number should be displayed in console window)

Considering the target board is connected to the host PC you are ready to go.

Get target info

FreeMASTER Lite exposes a JSON-RPC API via websocket protocol. You need to import the corresponding dependencies in order to run the code.

```
In [ ]: import websockets
        from jsonrpcclient.clients.websockets_client import WebSocketsClient
```

Adjust the *machine url* if you are connecting to a remote host, and/or *service_port* if it is running on a different port.

```
In [ ]: machine_url = '127.0.0.1'
        service_protol = 'ws://'
        service_port = '8090'
        service_url = service_protol + machine_url + ':' + service_port
```

Service requires a connection string to establish a communication channel to the target.

```
In [ ]: connection = 'RS232;port=COM9;speed=115200;tmoRI=40;tmoRTM=40;tmoRTC=50;tmoWTM=4
```

EXAMPLE 2
NODEJS SCRIPTING

```
C:\NXP\FreeMASTER 3.0\FreeMASTER Lite\
scripting examples\NodeJS
```

- Install Prerequisites:

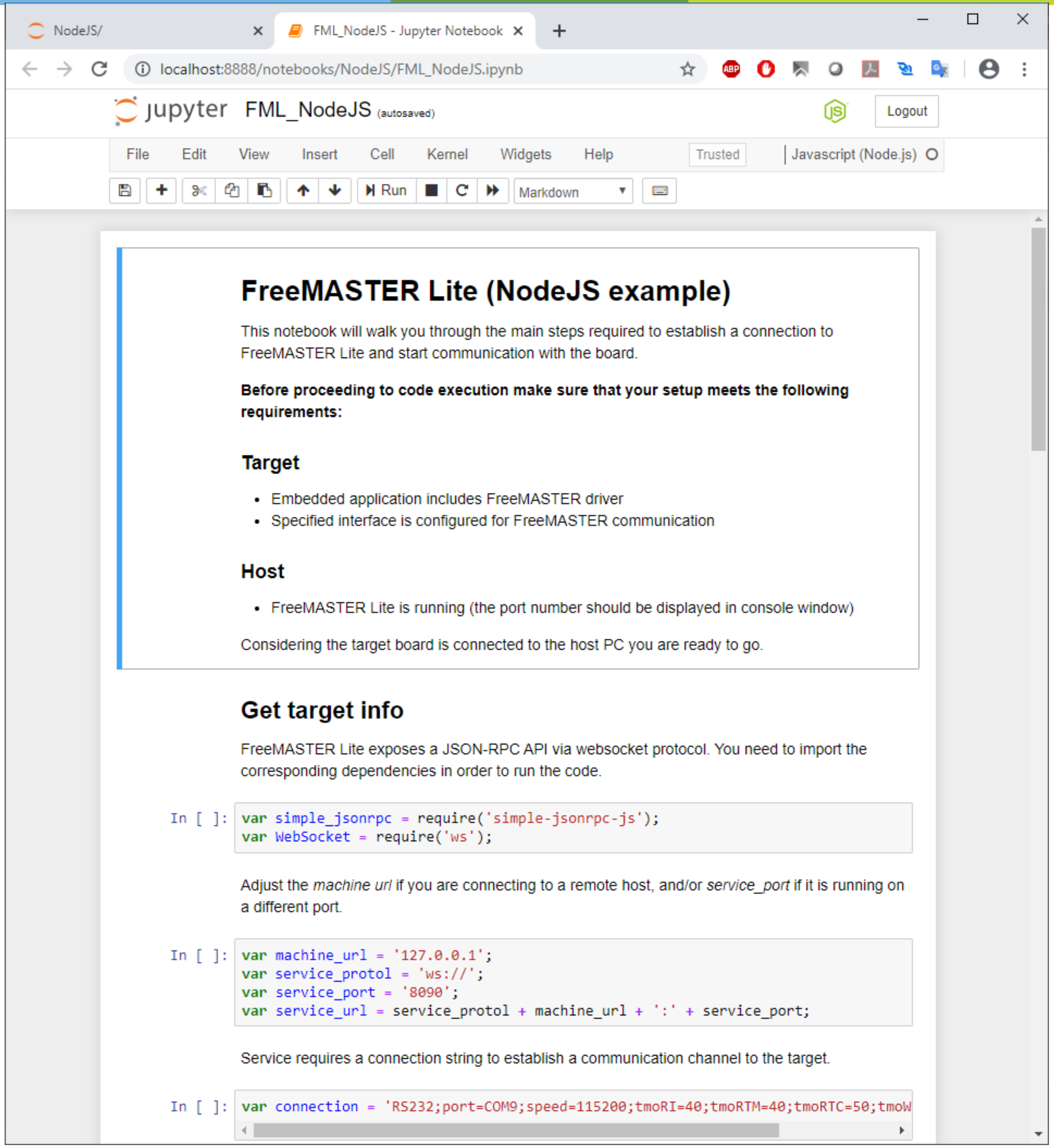
_>npm install

- Start Jupyter notebook

_>`jupyter notebook`

- Use browser based IDE

to interact with the board



NodeJS/
FML_NodeJS - Jupyter Notebook
localhost:8888/notebooks/NodeJS/FML_NodeJS.ipynb
jupyter FML_NodeJS (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Javascript (Node.js)
Run
Markdown

FreeMASTER Lite (NodeJS example)

This notebook will walk you through the main steps required to establish a connection to FreeMASTER Lite and start communication with the board.

Before proceeding to code execution make sure that your setup meets the following requirements:

Target

- Embedded application includes FreeMASTER driver
- Specified interface is configured for FreeMASTER communication

Host

- FreeMASTER Lite is running (the port number should be displayed in console window)

Considering the target board is connected to the host PC you are ready to go.

Get target info

FreeMASTER Lite exposes a JSON-RPC API via websocket protocol. You need to import the corresponding dependencies in order to run the code.

```
In [ ]: var simple_jsonrpc = require('simple-jsonrpc-js');
        var WebSocket = require('ws');
```

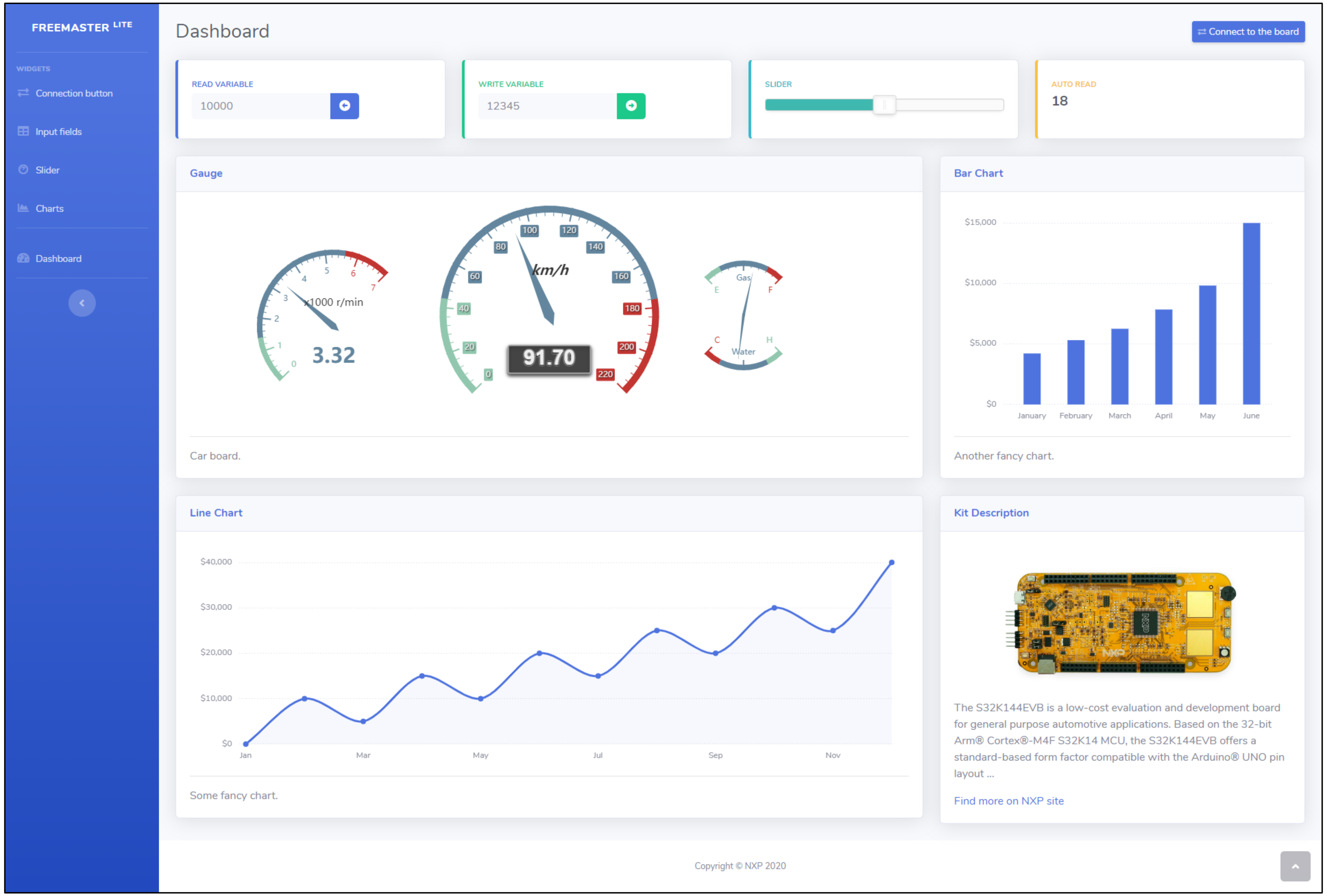
Adjust the *machine url* if you are connecting to a remote host, and/or *service_port* if it is running on a different port.

```
In [ ]: var machine_url = '127.0.0.1';
        var service_protol = 'ws://';
        var service_port = '8090';
        var service_url = service_protol + machine_url + ':' + service_port;
```

Service requires a connection string to establish a communication channel to the target.

```
In [ ]: var connection = 'RS232;port=COM9;speed=115200;tmoRI=40;tmoRTM=40;tmoRTC=50;tmoW
```

EXAMPLE 3 WEB DASHBOARDS



Dashboard

READ VARIABLE: 10000

WRITE VARIABLE: 12345

AUTO READ: 18

Car board.

Another fancy chart.

Some fancy chart.

The S32K144EVB is a low-cost evaluation and development board for general purpose automotive applications. Based on the 32-bit Arm® Cortex®-M4F S32K14 MCU, the S32K144EVB offers a standard-based form factor compatible with the Arduino® UNO pin layout ...

Find more on NXP site

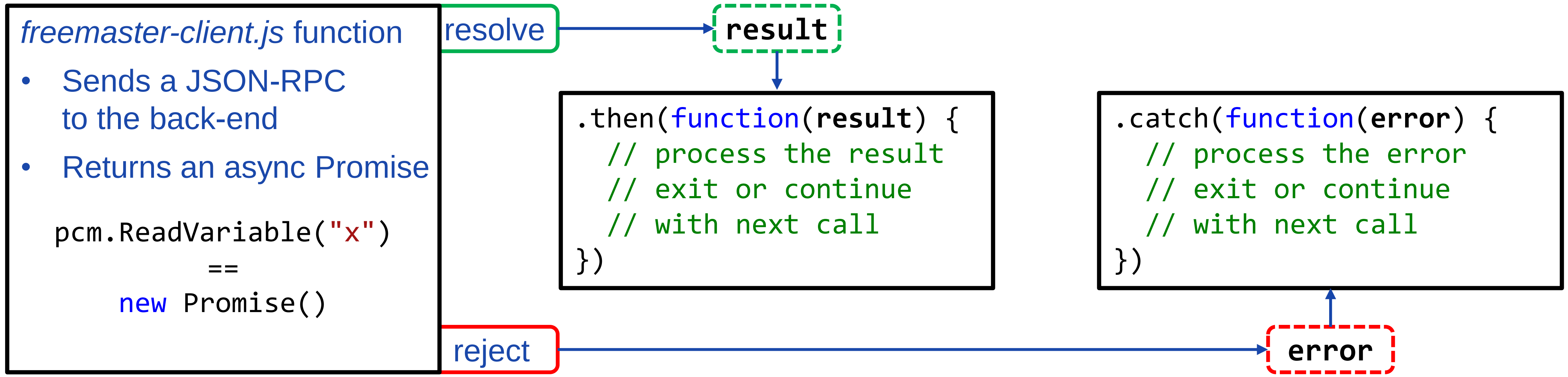
Copyright © NXP 2020

CODING SUMMARY
FREEMASTER-CLIENT.JS

- Include *simple-jsonrpc-js.js* and *freemaster-client.js* into your web page

```
<!-- FREEMASTER JS -->
<script src="simple-jsonrpc-js.js"></script>
<script src="freemaster-client.js"></script>
```

- *freemaster-client.js* functions return a JavaScript ***Promise*** that is executed ***asynchronously***



freemaster-client.js function

- Sends a JSON-RPC to the back-end
- Returns an async Promise

```
pcm.ReadVariable("x")
        ==
    new Promise()
```

```
.then(function(result) {
  // process the result
  // exit or continue
  // with next call
})
```

```
.catch(function(error) {
  // process the error
  // exit or continue
  // with next call
})
```

CODING SUMMARY
CONNECTING TO THE BOARD

- Initialize the *PCM* object once the page is loaded

```
$(document).ready(function() {
  pcm = new PCM(“localhost:8090", open_handler, close_handler, error_handler);
});
```

- Connect to the board once the client is connected to the server (inside *open_handler*)

```
function open_handler() {
  pcm.StartComm("connection name as set in configuration file")
    .then((response) => {
      // proceed with main application logic
    })
    .catch((error) => {
      // handle the error (board may be not connected or the connection string is wrong)
    });
};
```

CODING SUMMARY
READ & WRITE VARIABLE

- Read *TSA* or *ELF* file to map project variables to their corresponding addresses

```
pcm.ReadVariable("variable_name")
  .then((response) => {
    // process response.data property
  })
  .catch((error) => {
    // handle the error
  });
```

```
pcm.WriteVariable("variable_name", value)
  .then((response) => {
    // response does not carry any data
  })
  .catch((error) => {
    // handle the error
  });
```

- Read multiple calls and wait all to return before proceeding further

```
Promise.all([
    pcm.ReadVariable("variable_name_1")
    pcm.ReadVariable("variable_name_2")
  ]).then((responses) => {
      // process the array of responses in the same order as the functions were invoked in
    })
    .catch((error) => {
      // handle the error
    });
```

SHARE YOUR FREEMASTER DASHBOARD DESIGNS WITH THE NXP COMMUNITY
GET A FREE BOARD!

Why?

To build a robust community of support for FreeMASTER with idea share.

How to participate?

1. **Submit your idea** through June 19, 2020 to the NXP Community, request your board of choice (one of the following: i.MX RT1020 EVK, LPC55S28 development board and S32K144EVB), available on first come, first served basis until quantities are depleted.
2. Once you've created your code example, **post a brief description and a screenshot of your dashboard along with a ZIPped code** to ***the*** original blog comment thread.

Click here for complete details!

HOW TO CONTROL AND VISUALIZE DATA FROM YOUR EMBEDDED APPLICATION WITH FREEMASTER | A FOUR-PART WEBINAR SERIES

- **Part 1: Now Available On-Demand |** [Watch Now >]
  Get to Know the Easy-to-Use FreeMASTER Runtime Debugging Tool – Now Part of MCUXpresso SDK

- **Part 2: Now Available On-Demand |** [Watch Now >]
  Tips for Enhancing Embedded Applications with FreeMASTER UI from Various Development Environments like S32DS and Matlab/Simulink

- **Part 3: Now Available On-Demand |** [Watch Now >]
  Introduction to FreeMASTER Dashboard Coding Using HTML, JavaScript, ActiveX and JSON-RPC

- **Part 4: Today**
  Getting Started with FreeMASTER Lite and JSON-RPC Protocol: From Scripting to Visual Dashboards with Python and JavaScript

Q
&
A

SECURE CONNECTIONS
FOR A SMARTER WORLD