<!DOCTYPE html>

<html>

<head>

<title>CleanCab App</title>

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" type="text/css" href="styleph.css">

<link rel="manifest" href="manifest.json">

</head>

<body onload="init()">

<!-- Simulate a smartphone / tablet -->

<div class="mobile-containerx">

<!-- Top Navigation Menu -->

<div class="topnav">

<a href="javascript:showDiv('connectingDiv')" class="active">CleanCab</a>

<div id="idleLinks">

<a href="javascript:connect()">Connect</a>

<a href="javascript:showDiv('contactDiv')">Contact</a>

</div>

<div id="connectedLinks">

<a href="javascript:disconnect()">Disconnect</a>

<a id="menuItemLogin" href="javascript:showDiv('loginDiv')">Login</a>

<a href="javascript:showDiv('statusDiv')">Status</a>

<a href="javascript:showDiv('commissionDiv')">Commission</a>

<a href="javascript:showDiv('alarmHistoryDiv')">Alarm History</a>

<a href="javascript:showDiv('filterDetailsDiv')">Filter Details</a>

<a href="javascript:showDiv('filterHistoryDiv')">Filter History</a>

<a href="javascript:showDiv('contactDiv')">Contact</a>

</div>

<a href="javascript:void(0);" class="icon" onclick="toggleMenu()">

<i style="display:inline-block;font:normal normal normal 14px/1 FontAwesome;font-size:inherit;text-rendering:auto;-webkit-font-smoothing:antialiased;-moz-osx-font-smoothing:grayscale">

&#9776;

</i>

</a>

</div>

<!--div id="debugDiv">

<div class="container">

<div class="row">

<p id="phdbg"></p>

</div>

<div class="row">

<input id="phdbgBtn" type="button" value="Clear" onclick="phdbgClear()" class="btn">

</div>

</div>

</div-->

<div id="connectingDiv" class="hiddenitemdiv">

<h3>Connection</h3>

<div class="container">

<div class="row">

<div style="width: 100%;height: 150px;background-image: url('cleancab.png'); background-repeat: no-repeat; background-size: contain; background-position: center;">

</div>

</div>

<div class="row">

<p id="connectingTicks"></p>

<p id="connectingState"></p>

</div>

</div>

</div>

<div id="statusDiv" class="hiddenitemdiv">

<h3>Status</h3>

<div class="container">

<form>

<div class="row">

<div class="col-25">

<label>Fan</label>

</div>

<div class="col-65">

<div class="row-progress">

<div id="sFanSpeedBar" class="container-progress" style="width:10%">10%</div>

</div>

</div>

</div>

<div class="row">

<div class="col-25">

<label>Pressure</label>

</div>

<div class="col-65">

<div class="row-progress">

<div id="sPressureBar" class="container-progress" style="width:10%">10pa</div>

</div>

</div>

</div>

<div class="row">

<div class="col-25">

<label>Door</label>

</div>

<div class="col-65">

<input type="text" id="sDoorState" readonly value="Disabledx">

</div>

</div>

<div class="row">

<div class="col-25">

<label>Alarm</label>

</div>

<div class="col-65">

<input type="text" id="sAlarmState" readonly value="Disabledx">

</div>

</div>

<div class="row">

<div class="col-25">

<label>CO2</label>

</div>

<div class="col-65">

<input type="text" id="sCO2State" readonly value="Disabledx">

</div>

</div>

</form>

</div>

<h3>Running Times</h3>

<div class="container">

<form>

<div class="row">

<div class="col-25">

<label>Motor</label>

</div>

<div class="col-65">

<input type="text" id="sMotorTime" readonly value="123">

</div>

<div class="col-5">

<label class="units">hrs</label>

</div>

</div>

<div class="row">

<div class="col-25">

<label>HEPA</label>

</div>

<div class="col-65">

<input type="text" id="sHepaFilterTime" readonly value="123">

</div>

<div class="col-5">

<label class="units">hrs</label>

</div>

</div>

<div class="row">

<div class="col-25">

<label>Main Filter</label>

</div>

<div class="col-65">

<input type="text" id="sMainFilterTime" readonly value="123">

</div>

<div class="col-5">

<label class="units">hrs</label>

</div>

</div>

</form>

</div>

</div>

<div id="filterDetailsDiv" class="hiddenitemdiv">

<h3>Replace Filters</h3>

<div class="container">

<form>

<div class="row" id="addHepaDiv">

<div class="col-35">

<label>Hepa Filter</label>

</div>

<div class="col-65">

<input id="addHepaBtn" type="button" value="Replace" onclick="replaceFilter(1)" class="btn">

</div>

</div>

<div class="row" id="addMainDiv">

<div class="col-35">

<label>Main Filter</label>

</div>

<div class="col-65">

<input id="addMainBtn" type="button" value="Replace" onclick="replaceFilter(2)" class="btn">

</div>

</div>

<div id="filterForm" style="display:none;">

<div class="row">

<div class="col-35">

<label>Part Number</label>

</div>

<div class="col-65">

<input type="text" id="filterPartNumber" value="">

</div>

</div>

<div class="row">

<div class="col-35">

<label>Serial Number</label>

</div>

<div class="col-65">

<input type="text" id="filterSerialNumber" value="">

</div>

</div>

<div class="row">

<div class="col-35">

<label>Installer Name</label>

</div>

<div class="col-65">

<input type="text" id="filterInstallerName" value="">

</div>

</div>

<div class="row">

<div class="col-35">

</div>

<div class="col-65">

<input id="filterSaveBtn" type="button" value="Save" onclick="filterSave()" class="btn">

</div>

</div>

</div>

</form>

</div>

<h3>HEPA Filter Details</h3>

<div class="container">

<div class="row">

<div class="col-35">

<label>Installed Date</label>

</div>

<div class="col-65">

<input type="date" id="hepaInstalledDate" readonly value="2024-02-18">

</div>

</div>

<div class="row">

<div class="col-35">

<label>Part Number</label>

</div>

<div class="col-65">

<input type="text" id="hepaPartNumber" readonly value="">

</div>

</div>

<div class="row">

<div class="col-35">

<label>Serial Number</label>

</div>

<div class="col-65">

<input type="text" id="hepaSerialNumber" readonly value="">

</div>

</div>

<div class="row">

<div class="col-35">

<label>Installer Name</label>

</div>

<div class="col-65">

<input type="text" id="hepaInstallerName" readonly value="">

</div>

</div>

</div>

<h3>Main Filter Details</h3>

<div class="container">

<div class="row">

<div class="col-35">

<label>Installed Date</label>

</div>

<div class="col-65">

<input type="date" id="mainInstalledDate" readonly value="2024-02-18">

</div>

</div>

<div class="row">

<div class="col-35">

<label>Part Number</label>

</div>

<div class="col-65">

<input type="text" id="mainPartNumber" readonly value="">

</div>

</div>

<div class="row">

<div class="col-35">

<label>Serial Number</label>

</div>

<div class="col-65">

<input type="text" id="mainSerialNumber" readonly value="">

</div>

</div>

<div class="row">

<div class="col-35">

<label>Installer Name</label>

</div>

<div class="col-65">

<input type="text" id="mainInstallerName" readonly value="">

</div>

</div>

</div>

</div>

<div id="filterHistoryDiv" class="hiddenitemdiv">

<h3>Filter History</h3>

<div class="container">

<p id="filterHistorydbg"></p>

<div class="row">

<div class="col-30">

<button id="filterHistoryPrior" onclick="gotoFilterRec(-1)">Newer</button>

</div>

<div class="col-40 pad-10-lr">

<input type="text" id="filterHistoryPage" readonly value="1/100">

</div>

<div class="col-30 units">

<button id="filterHistoryNext" onclick="gotoFilterRec(1)">Older</button>

</div>

</div>

<div class="container" id="filterHistoryDataDiv">

</div>

<div class="row">

<div class="col-70">

</div>

<div class="col-30 units">

<button id="filterHistorySave" onclick="saveFilterRecords()">Save to file</button>

</div>

</div>

</div>

</div>

<div id="alarmHistoryDiv" class="hiddenitemdiv">

<h3>Alarm History</h3>

<div class="container">

<p id="alarmHistorydbg"></p>

<div class="row">

<div class="col-30">

<button id="alarmHistoryPrior" onclick="gotoAlarmRec(-1)">Newer</button>

</div>

<div class="col-40 pad-10-lr">

<input type="text" id="alarmHistoryPage" readonly value="1/100">

</div>

<div class="col-30 units">

<button id="alarmHistoryNext" onclick="gotoAlarmRec(1)">Older</button>

</div>

</div>

<div class="container" id="alarmHistoryDataDiv">

</div>

<div class="row">

<div class="col-70">

</div>

<div class="col-30 units">

<button id="alarmHistorySave" onclick="saveAlarmRecords()">Save to file</button>

</div>

</div>

</div>

</div>

<div id="commissionDiv" class="hiddenitemdiv">

<h3>Commission</h3>

<div class="container">

<form id="commissionForm">

<div class="row">

<div class="col-25">

<label>Installed Date</label>

</div>

<div class="col-75">

<input type="date" id="cInstalledDate" readonly value="2024-02-18">

</div>

</div>

<div class="row">

<div class="col-25">

<label>Company</label>

</div>

<div class="col-75">

<input type="text" id="cInstallerCompany" maxlength="99" readonly value="">

</div>

</div>

<div class="row">

<div class="col-25">

<label>Installed By</label>

</div>

<div class="col-75">

<input type="text" id="cInstallerName" maxlength="99" readonly value="">

</div>

</div>

<div class="row">

<div class="col-25">

<label>Contact</label>

</div>

<div class="col-75">

<input type="text" id="cInstallerPhone" maxlength="19" readonly value="">

</div>

</div>

<div class="row">

<div class="col-25">

<label>Machine Name</label>

</div>

<div class="col-75">

<input type="text" id="cMachineName" maxlength="99" readonly value="">

</div>

</div>

<div class="row">

<div class="col-25">

<label>Plant Number</label>

</div>

<div class="col-75">

<input type="text" id="cPlantNumber" maxlength="29" readonly value="">

</div>

</div>

<div class="row">

<div class="col-25">

<label>Plant Hours</label>

</div>

<div class="col-75">

<input type="text" id="cPlantHours" maxlength="9" readonly value="">

</div>

</div>

<div class="row" id="cSaveRow" style="display:none;">

<input type="submit" value="Save" class="btn">

</div>

</form>

<div class="row">

<div class="col-25">

<label>Check for leaks</label>

</div>

<div class="col-75">

<label class="switch">

<input id="testModeCheckbox" type="checkbox" onclick="testModeClicked()">

<span class="slider round"></span>

</label>

</div>

</div>

</div>

</div>

<div id="calibrateDiv" class="hiddenitemdiv">

<h3>Calibration</h3>

<div class="container">

<div class="row">

<p id="calDbg"></p>

</div>

<div class="row">

<div class="col-25">

<label>ADC Reading</label>

</div>

<div class="col-65">

<div class="row-progress">

<div id="calAdcBar" class="container-progress" style="width:10%">10%</div>

</div>

</div>

</div>

<div class="row">

<div class="col-25">

<label>Pressure</label>

</div>

<div class="col-65">

<div class="row-progress">

<div id="calPressureBar" class="container-progress" style="width:10%">10%</div>

</div>

</div>

</div>

</div>

<div class="container">

<div class="row">

<div class="col-25">

<label>Real Pressure</label>

</div>

<div class="col-65">

<input type="number" id="calRealPressure" min="0" max="300" value="0">

</div>

</div>

<div class="row">

<div class="col-25">

</div>

<div class="col-65">

<input type="button" value="Set Zero Point" onclick="calibratePoint(1)">

</div>

</div>

<div class="row">

<div class="col-25">

</div>

<div class="col-65">

<input type="button" value="Set Point 2" onclick="calibratePoint(2)">

</div>

</div>

<div class="row">

<div class="col-25">

</div>

<div class="col-65">

<input type="button" value="Clear Data Points" onclick="calibratePoint(0)">

</div>

</div>

</div>

<h3>Log Management</h3>

<div class="container">

<div class="row">

<div class="col-25">

</div>

<div class="col-65">

<input type="button" value="Erase Filter Log" onclick="eraseLog(0)">

</div>

</div>

<div class="row">

<div class="col-25">

</div>

<div class="col-65">

<input type="button" value="Erase Alarm Log" onclick="eraseLog(1)">

</div>

</div>

<div class="row">

<div class="col-25">

</div>

<div class="col-65">

<input type="button" value="Erase Running Counts" onclick="eraseLog(2)">

</div>

</div>

</div>

</div>

<div id="contactDiv" class="hiddenitemdiv">

<h3>Contact</h3>

<div class="container">

<div class="row">

<div style="width: 100%;height: 150px;background-image: url('cleancab.png'); background-repeat: no-repeat; background-size: contain; background-position: center;">

</div>

</div>

<div class="row">

<p>Address :<a href="">3 Rose Smith Lane, Muckadilla QLD 4461</a></p>

</div>

<div class="row">

<p>Phone :<a href="tel:+427477740">0427 477 740</a></p>

</div>

<div class="row">

<p>Email :<a href="mailto:info@cleancab.au">info@cleancab.au</a></p>

</div>

<div class="row">

<p>Facebook :<a href="https://www.facebook.com/profile.php?id=61550062888002">CLEANCAB filtration system</a></p>

</div>

</div>

</div>

<div id="settingDiv" class="hiddenitemdiv">

<h3>Settings</h3>

<div class="container">

<form id="settingForm">

<div class="row">

<div class="col-40">

<label>Operating Mode</label>

</div>

<div class="col-5">

</div>

<div class="col-50">

<input type="radio" name="fmode" value="0" checked onclick="handleModeClick(this);"><label>Fixed</label><br>

<input type="radio" name="fmode" value="1" onclick="handleModeClick(this);"><label>Adaptive</label>

</div>

<div class="col-5">

<label></label>

</div>

</div>

<div class="row">

<div class="col-40">

<label>Ramp Rate</label>

</div>

<div class="col-5">

<div class="tooltip">?<span class="tooltiptext">How fast the blower changes speed. Default is 20ms</span></div>

</div>

<div class="col-50">

<input type="number" id="frampRate" name="frampRate" min="1" max="200" placeholder="20">

</div>

<div class="col-5">

<label class="units">ms</label>

</div>

</div>

<div class="row" id="settingStartPause">

<div class="col-40">

<label>Startup Pause Time</label>

</div>

<div class="col-5">

<div class="tooltip">?<span class="tooltiptext">The time to hold the fan at max speed after power on</span></div>

</div>

<div class="col-50">

<input type="number" id="fstartPause" name="fstartPause" min="0" max="300" placeholder="30">

</div>

<div class="col-5">

<label class="units">sec</label>

</div>

</div>

<div class="row">

<div class="col-40">

<label>Door Opened Fan Speed</label>

</div>

<div class="col-5">

<div class="tooltip">?<span class="tooltiptext">The fan speed when the door is opened</span></div>

</div>

<div class="col-50">

<input type="number" id="fdoorSpeed" name="fdoorSpeed" min="0" max="100" placeholder="20">

</div>

<div class="col-5">

<label class="units">%</label>

</div>

</div>

<div class="row">

<div class="col-40">

<label>Fixed Fan Speed</label>

</div>

<div class="col-5">

<div class="tooltip">?<span class="tooltiptext">The required fan speed</span></div>

</div>

<div class="col-50">

<input type="number" id="ffixedSpeed" name="ffixedSpeed" min="0" max="100" placeholder="70">

</div>

<div class="col-5">

<label class="units">%</label>

</div>

</div>

<div class="row">

<div class="col-40">

<label>Target Pressure</label>

</div>

<div class="col-5">

<div class="tooltip">?<span class="tooltiptext">The target pressure for the cabin (pa)</span></div>

</div>

<div class="col-50">

<input type="number" id="ftargetPressure" name="ftargetPressure" min="10" max="200" placeholder="30" disabled>

</div>

<div class="col-5">

<label class="units">pa</label>

</div>

</div>

<div class="row">

<div class="col-40">

<label>Step Size</label>

</div>

<div class="col-5">

<div class="tooltip">?<span class="tooltiptext">The fan speed change amount to obtain the target pressure</span></div>

</div>

<div class="col-50">

<input type="number" id="fstepSize" name="fstepSize" min="1" max="100" placeholder="10" disabled>

</div>

<div class="col-5">

<label class="units">%</label>

</div>

</div>

<div class="row">

<div class="col-40">

<label>Hold Time</label>

</div>

<div class="col-5">

<div class="tooltip">?<span class="tooltiptext">The rate at which the fan speed is updated to obtain the target pressure</span></div>

</div>

<div class="col-50">

<input type="number" id="fholdTime" name="fholdTime" min="0" max="300" placeholder="30" disabled>

</div>

<div class="col-5">

<label class="units">sec</label>

</div>

</div>

<div class="row">

<div class="col-40">

<label>Min Fan Speed</label>

</div>

<div class="col-5">

<div class="tooltip">?<span class="tooltiptext">The minimum fan speed</span></div>

</div>

<div class="col-50">

<input type="number" id="fminSpeed" name="fminSpeed" min="0" max="100" placeholder="10" disabled>

</div>

<div class="col-5">

<label class="units">%</label>

</div>

</div>

<div class="row" id="settingDoorMode">

<div class="col-40">

<label>Door Detect Mode</label>

</div>

<div class="col-5">

<div class="tooltip">?<span class="tooltiptext">How the door switch detector works</span></div>

</div>

<div class="col-50">

<input type="radio" name="fdoorMode" value="0" checked><label>Disabled</label><br>

<input type="radio" name="fdoorMode" value="1"><label>N/O</label><br>

<input type="radio" name="fdoorMode" value="2"><label>N/C</label>

</div>

</div>

<div class="row" id="settingAlarmMode">

<div class="col-40">

<label>Alarm Detect Mode</label>

</div>

<div class="col-5">

</div>

<div class="col-50">

<input type="radio" name="falarmMode" value="0" checked><label>Disabled</label><br>

<input type="radio" name="falarmMode" value="1"><label>N/O</label><br>

<input type="radio" name="falarmMode" value="2"><label>N/C</label>

</div>

</div>

<div class="row" id="settingCO2Mode">

<div class="col-40">

<label>CO2 Detect Mode</label>

</div>

<div class="col-5">

</div>

<div class="col-50">

<input type="radio" name="fco2Mode" value="0" checked><label>Disabled</label><br>

<input type="radio" name="fco2Mode" value="1"><label>N/O</label><br>

<input type="radio" name="fco2Mode" value="2"><label>N/C</label>

</div>

</div>

<div class="row">

<input id="settingSaveBtn" type="submit" value="Save Settings" class="btn">

</div>

</form>

</div>

</div>

<div id="loginDiv" class="hiddenitemdiv">

<h3 id="loginDivHeader">Login</h3>

<div class="container">

<div class="row">

<div class="col-35">

</div>

<div class="col-65">

<label id="loginDbg">Enter login password</label>

</div>

</div>

<div class="row" id="loginPasswordRow">

<div class="col-35">

<label>Password</label>

</div>

<div class="col-65">

<input id="loginPassword" type="password" value="">

</div>

</div>

<div class="row">

<div class="col-35">

</div>

<div class="col-65">

<input id="loginBtn" type="button" value="Login" onclick="loginClicked()">

</div>

</div>

</div>

</div>

<!-- End smartphone / tablet look -->

</div>

<script>

//=====================================================

//

// Service Worker

//

//=====================================================

if ('serviceWorker' in navigator)

{

navigator.serviceWorker.register('serviceworker.js')

.then(function(registration)

{

console.log('ServiceWorker registration successful!');

})

.catch(function(err)

{

console.log('ServiceWorker registration failed: ', err);

});

}

//=====================================================

//

// Initialization

//

//=====================================================

//const phdbg = document.querySelector("#phdbg");

//var phdbgc = 0;

//

//function phdbgClear()

//{

// phdbg.innerHTML = "";

//}

const DBG\_INFO = 0;

const DBG\_CONNECT = 1;

const DBG\_DEBUG = 2;

const connectingState = document.querySelector("#connectingState");

const connectingTicks = document.querySelector("#connectingTicks");

var selectDisplay;

var activeLinks;

var filterHistory;

var alarmHistory;

var authorizeControl;

var calibrateControl;

function init()

{

selectDisplay = "";

activeLinks = "idleLinks";

filterHistory = new filterHistoryClass("filter", "Filter");

alarmHistory = new alarmHistoryClass("alarm", "Alarm");

authorizeControl = new authorizeClass();

calibrateControl = new calibrateClass();

onDisconnected();

setTimeout(secondTimer, 500);

}

//=====================================================

//

// Menu helper

//

//=====================================================

function menuHide()

{

var x = document.getElementById(activeLinks);

if (x)

{

x.style.display = "none";

}

}

function toggleMenu()

{

var x = document.getElementById(activeLinks);

if (x.style.display === "block")

{

x.style.display = "none";

}

else

{

x.style.display = "block";

}

}

function hideSelected()

{

if (selectDisplay != "")

{

var x = document.getElementById(selectDisplay);

if (x)

{

x.style.display = "none";

}

}

}

function showSelected()

{

if (selectDisplay != "")

{

var x = document.getElementById(selectDisplay);

if (x)

{

x.style.display = "block";

}

if (selectDisplay === "filterHistoryDiv")

{

if (filterHistory)

{

filterHistory.loadHistoryDiv(0);

}

}

else if (selectDisplay === "alarmHistoryDiv")

{

if (alarmHistory)

{

alarmHistory.loadHistoryDiv(0);

}

}

}

}

function showDiv(divName)

{

hideSelected();

selectDisplay = divName;

showSelected();

menuHide();

}

//===================================================

//

// Helper functions

//

//===================================================

function showDetailsDiv(frecId)

{

element = document.getElementById(frecId);

if (element)

{

if (element.style.display === "none")

{

element.style.display = "block";

}

else

{

element.style.display = "none";

}

}

}

function pad2(num)

{

var s = "0" + num;

return s.substr(s.length - 2);

}

function setRadioInput(fieldName, selectValue)

{

var radios = document.getElementsByName(fieldName);

for (var j = 0; j < radios.length; j++)

{

if (radios[j].value == selectValue)

{

radios[j].checked = true;

break;

}

}

}

function getRadioInput(fieldName)

{

var radios = document.getElementsByName(fieldName);

for (var j = 0; j < radios.length; j++)

{

if (radios[j].checked)

{

return radios[j].value;

}

}

return 0;

}

function getNowDateForInput()

{

const date = new Date();

thedate = date.getFullYear() + "-";

let realMonth = date.getMonth() + 1;

if (realMonth < 10)

{

thedate += '0';

}

thedate += realMonth + '-';

if (date.getDate() < 10)

{

thedate += '0';

}

thedate += date.getDate();

return thedate;

}

function toHours(mins)

{

var hours = mins / 60;

return hours.toFixed(1);

}

//==========================================================

//

// Log reader base class (aka History)

//

//==========================================================

// Log helper functions

function logDataValid(data)

{

return data.getUint8(1) > 0;

}

function logDataRecordNum(data)

{

if (protocolVersion == 0)

{

return 0;

}

return data.getUint16(2, true);

}

function logDataNumOfRecords(data)

{

if (protocolVersion == 0)

{

return 0;

}

return data.getUint16(4, true);

}

// Log data:

// Version 0: id,valid,day, month, year, hour, min,sec,...

// Version 1: id,valid,rec num lo,rec num hi,num recs lo,num recs hi,day,month,year

function logDataDate(data)

{

let offset = (protocolVersion == 0) ? 2 : 6;

let theDay = pad2(data.getUint8(offset + 0));

let theMonth = pad2(data.getUint8(offset + 1));

let theYear = "20" + pad2(data.getUint8(offset + 2));

let theDate = theYear + "-" + theMonth + "-" + theDay;

return theDate;

}

function logDataTime(data)

{

let offset = (protocolVersion == 0) ? 5 : 9;

let theHour = pad2(data.getUint8(offset + 0));

let theMinute = pad2(data.getUint8(offset + 1));

let theSecond = pad2(data.getUint8(offset + 2));

let theTime = theHour + ":" + theMinute + ":" + theSecond;

return theTime;

}

// Log base class

class historyClass

{

constructor(historyName, blename)

{

this.elementDbg = document.getElementById(historyName + "Historydbg");

this.elementDataDiv = document.getElementById(historyName + "HistoryDataDiv");

this.elementPrior = document.getElementById(historyName + "HistoryPrior");

this.elementNext = document.getElementById(historyName + "HistoryNext");

this.elementPage = document.getElementById(historyName + "HistoryPage");

this.elementSave = document.getElementById(historyName + "HistorySave");

this.fileData = "";

this.top = 0;

this.count = 0;

this.numRecords = 0;

this.blename = blename;

this.fileName = historyName + "-history.txt";

this.name = historyName;

this.id = 0;

this.exporting = false;

this.readState = 0;

}

setExportFileHeader()

{

this.fileData = "";

}

exportRecords()

{

bleiface.setMessage(DBG\_INFO, "History " + this.name + ": Start Export");

this.elementDbg.innerHTML = "Exporting data...";

this.elementNext.disabled = true;

this.elementPrior.disabled = true;

this.elementSave.disabled = true;

this.exporting = true;

this.readState = 1;

this.setExportFileHeader();

this.count = 2; // actually a retry count

}

loadHistoryDiv(thePage)

{

bleiface.setMessage(DBG\_INFO, "History " + this.name + ": Start Loading Page");

this.elementDbg.innerHTML = "Loading data...";

this.elementNext.disabled = true;

this.elementPrior.disabled = true;

this.elementSave.disabled = true;

while (this.elementDataDiv.firstChild)

{

this.elementDataDiv.removeChild(this.elementDataDiv.firstChild);

}

this.exporting = false;

this.readState = 1;

this.top = thePage;

this.count = 10;

this.numRecords = 0;

}

handleHistoryLoading()

{

let result = false;

if (this.readState > 0)

{

let item = bleiface.getItem(this.blename);

if (item)

{

bleiface.setMessage(DBG\_INFO, "History " + this.name + ": State: " + this.readState);

switch (this.readState)

{

case 1:

// perform a Start command

if (this.exporting)

{

item.setReadRecord(3);

}

else

{

item.setGotoRecord(0, this.top + 9);

}

this.id = item.getId();

result = true;

this.readState = 2;

break;

case 2:

// performing the first read after a GOTO record request or a get First record request

result = true;

this.readState = 1; // if we fail then go back to sending the GOTO request

bleiface.itemReadWithBusy(item);

break;

case 3:

// the first record after the GOTO had been read, so keep reading until the end

if (this.count > 0)

{

this.count--;

result = true;

bleiface.itemReadWithBusy(item);

}

else

{

this.ReadDone();

}

break;

}

}

else

{

this.readState = 0;

}

}

return result;

}

readDone()

{

bleiface.setMessage(DBG\_INFO, "History " + this.name + ": Finished");

if (this.exporting)

{

var tempLink = document.createElement("a");

var taBlob = new Blob([this.fileData], {type: 'text/plain'});

tempLink.setAttribute('href', URL.createObjectURL(taBlob));

tempLink.setAttribute('download', this.fileName);

tempLink.click();

URL.revokeObjectURL(tempLink.href);

this.fileData = "";

}

if (this.numRecords == 0)

{

this.elementDbg.innerHTML = "No records found";

}

else

{

this.elementDbg.innerHTML = "";

}

this.elementNext.disabled = (this.top + 10) >= this.numRecords;;

this.elementPrior.disabled = this.top == 0;

this.elementPage.value = this.top + "/" + this.numRecords;

this.elementSave.disabled = false;

this.exporting = false;

this.readState = 0;

}

addRowElement(element, theLabel, theValue)

{

var rowDiv = document.createElement("div");

var labelDiv = document.createElement("div");

var valueDiv = document.createElement("div");

rowDiv.setAttribute("class", "row");

labelDiv.setAttribute("class", "col-35");

valueDiv.setAttribute("class", "col-65");

labelDiv.innerHTML = "<label>" + theLabel + "</label>";

valueDiv.innerHTML = "<input type='text' readonly value='" + theValue + "'>";

rowDiv.appendChild(labelDiv);

rowDiv.appendChild(valueDiv);

element.appendChild(rowDiv);

}

dataValid(data)

{

return false;

}

dataRecordNumber(data)

{

return 0;

}

dataNumOfRecords(data)

{

return 0;

}

dataAddToFileData(data)

{

}

dataAddDetailElements(detailsDiv, data)

{

}

dataDate(data)

{

return "";

}

dataRecordType(data)

{

return "";

}

handleRecordRead(id, data)

{

if (this.id == id &&

this.readState > 0)

{

bleiface.setMessage(DBG\_INFO, "History " + this.name + ": Received. Len=" + data.length);

if (this.readState == 1)

{

this.readState = 3;

}

if (this.readState == 3)

{

if (this.exporting)

{

this.count = 2;

if (this.dataValid(data))

{

this.elementDbg.innerHTML = "Exporting data ... Record " + this.dataRecordNumber(data);

this.dataAddToFileData(data);

}

else

{

this.readDone();

}

}

else

{

// need to add the record

if (this.dataValid(data))

{

this.elementDbg.innerHTML = "Loading data ... Record " + this.dataRecordNumber(data);

this.numRecords = this.dataNumOfRecords(data);

// Add a new element.

var rowDiv = document.createElement("div");

var dateDiv = document.createElement("div");

var typeDiv = document.createElement("div");

var btnDiv = document.createElement("div");

var frecId = "frec" + this.name + "Id" + this.dataRecordNumber(data);

rowDiv.setAttribute("class", "row");

dateDiv.setAttribute("class", "col-30");

typeDiv.setAttribute("class", "col-40 units");

btnDiv.setAttribute("class", "col-30 units");

dateDiv.innerHTML = "<input type='text' readonly value='" + this.dataDate(data) + "' title='" + this.dataDate(data) + "'>";

typeDiv.innerHTML = "<input type='text' readonly value='" + this.dataRecordType(data) + "'>";

btnDiv.innerHTML = "<input type='button' value='Details...' onclick=" + '"' + "showDetailsDiv('" + frecId + "')" + '"' + ">";

rowDiv.appendChild(dateDiv);

rowDiv.appendChild(typeDiv);

rowDiv.appendChild(btnDiv);

if (this.elementDataDiv.firstChild)

{

this.elementDataDiv.insertBefore(rowDiv, this.elementDataDiv.firstChild);

}

else

{

this.elementDataDiv.appendChild(rowDiv);

}

var detailsDiv = document.createElement("div");

detailsDiv.setAttribute("class", "container hiddenitemdiv");

detailsDiv.setAttribute("id", frecId);

this.dataAddDetailElements(detailsDiv, data);

rowDiv.appendChild(detailsDiv);

detailsDiv.style.display = "none";

if (this.dataRecordNumber(data) == this.top)

{

this.readDone();

}

}

else

{

// not valid so finished reading

this.readDone();

}

}

}

}

}

gotoRecord(rec)

{

bleiface.setMessage(DBG\_INFO, "History " + this.name + ": Goto " + rec);

if (rec < 0)

{

if (this.top > 10)

{

this.top -= 10;

}

else

{

this.top = 0;

}

}

else

{

this.top += 10;

if (this.top >= this.numRecords)

{

this.top -= 10;

if (this.top < 0)

{

this.top = 0;

}

}

}

this.loadHistoryDiv(this.top)

}

}

//===================================================

//

// Bluetooth Characteristic base class

//

//===================================================

class blueToothIf

{

constructor(itemName, serviceIdx, characteristicUUID)

{

this.name = itemName;

this.serviceIdx = serviceIdx;

this.characteristicUUID = characteristicUUID.toLowerCase();

this.characteristic = null;

this.validIsValid = false;

this.connectState = 0; // 0 = idle, 1 = connecting, 2 = connected, 3 = failed

this.readValue = null;

this.initState = 0;

this.runState = 0;

}

getName()

{

return this.name;

}

disconnected()

{

this.characteristic = null;

this.readValue = null;

this.validIsValid = false;

this.connectState = 0;

this.initState = 0;

}

start()

{

this.connectState = 1;

}

done()

{

this.connectState = 2;

}

failed()

{

this.connectState = 3;

}

setCharactistic(characteristic)

{

this.characteristic = characteristic;

this.done();

}

connect(service)

{

this.characteristic = null;

this.readValue = null;

this.validIsValid = false;

this.connectState = 1;

return service[this.serviceIdx].getCharacteristic(this.characteristicUUID);

}

isConnected()

{

return this.connectState == 2;

}

isConnecting()

{

return this.connectState == 1;

}

isConnectFailed()

{

return this.connectState == 3;

}

isNotConnect()

{

return this.connectState == 0 || this.connectState == 3;

}

isValueValid()

{

return this.isConnected() && this.validIsValid;

}

getValue()

{

return this.readValue;

}

setRead(readValue)

{

this.validIsValid = true;

this.readValue = readValue;

}

read()

{

this.validIsValid = false;

this.readValue = null;

return this.characteristic.readValue();

}

write(data)

{

bleiface.setBusy(this.getName);

this.characteristic.writeValueWithResponse(data)

.then (cdata =>

{

bleiface.clearBusy();

return true;

})

.catch (error =>

{

bleiface.setError("Error:Write: " + error);

});

}

init()

{

this.initState = 9999;

}

isInit()

{

return this.initState == 9999;

}

run()

{

}

};

//==========================================================

//

// Test Switch

//

//==========================================================

// Variables

var testSwitch = document.querySelector("#testModeCheckbox");

// Callback events

function testModeClicked()

{

let item = bleiface.getItem("Test");

if (item)

{

item.setTestState(testSwitch.checked);

}

}

// Bluetooth interface

class blueToothTest extends blueToothIf

{

constructor ()

{

super ("Test", 0, "546a24ea-ec5b-461b-9cce-b9bce932a72c");

this.send = false;

this.testState = false;

}

disconnected()

{

super.disconnected();

this.send = false;

this.testState = false;

}

writeTestState(on)

{

let data = new Uint8Array(1);

data[0] = on ? 1 : 0;

this.write(data);

}

setTestState(on)

{

this.send = true;

this.testState = on;

}

init()

{

if (this.initState == 0)

{

super.init();

bleiface.setMessage(DBG\_CONNECT, "Turning off Test Mode");

this.writeTestState(false);

}

}

run()

{

if (this.send)

{

this.send = false;

this.writeTestState(this.testState);

}

}

}

//==========================================================

//

// Filter setting/change

//

//==========================================================

// Variables

var addingFilter = 0;

var addHepaBtn = document.querySelector("#addHepaBtn");

var addMainBtn = document.querySelector("#addMainBtn");

var filterForm = document.querySelector("#filterForm");

var addHepaDiv = document.querySelector("#addHepaDiv");

var addMainDiv = document.querySelector("#addMainDiv");

var filterPartNumber = document.querySelector("#filterPartNumber");

var filterInstallerName = document.querySelector("#filterInstallerName");

var filterSerialNumber = document.querySelector("#filterSerialNumber");

const hepaInstalledDate = document.querySelector("#hepaInstalledDate");

const hepaPartNumber = document.querySelector("#hepaPartNumber");

const hepaInstallerName = document.querySelector("#hepaInstallerName");

const hepaSerialNumber = document.querySelector("#hepaSerialNumber");

const mainInstalledDate = document.querySelector("#mainInstalledDate");

const mainPartNumber = document.querySelector("#mainPartNumber");

const mainInstallerName = document.querySelector("#mainInstallerName");

const mainSerialNumber = document.querySelector("#mainSerialNumber");

//Filter event handlers

function gotoFilterRec(recDir)

{

if (filterHistory)

{

filterHistory.gotoRecord(recDir);

}

}

function saveFilterRecords()

{

if (filterHistory)

{

filterHistory.exportRecords();

}

}

function replaceFilter(filterType)

{

if (addingFilter == filterType)

{

// hide the form

addHepaBtn.value = "Replace";

addMainBtn.value = "Replace";

addingFilter = 0;

filterForm.style.display="none";

}

else if (addingFilter == 0)

{

addingFilter = filterType;

if (filterType == 1)

addHepaBtn.value = "Cancel Replace";

if (filterType == 2)

addMainBtn.value = "Cancel Replace";

filterForm.style.display="block";

}

addMainBtn.disabled = addingFilter == 1;

addHepaBtn.disabled = addingFilter == 2;

addHepaDiv.style.display = (addingFilter != 2) ? "block" : "none"

addMainDiv.style.display = (addingFilter != 1) ? "block" : "none"

}

function filterSave()

{

if (addingFilter != 0)

{

item = bleiface.getItem("Filter");

if (item)

{

item.addFilter(addingFilter - 1,

filterPartNumber.value + "|" + filterInstallerName.value + "|" + filterSerialNumber.value);

}

if (addingFilter == 1)

{

hepaInstalledDate.value = getNowDateForInput();

hepaPartNumber.value = filterPartNumber.value;

hepaInstallerName.value = filterInstallerName.value;

hepaSerialNumber.value = filterSerialNumber.value;

}

else

{

mainInstalledDate.value = getNowDateForInput();

mainPartNumber.value = filterPartNumber.value;

mainInstallerName.value = filterInstallerName.value;

mainSerialNumber.value = filterSerialNumber.value;

}

replaceFilter(addingFilter);

filterPartNumber.value = "";

filterInstallerName.value = "";

filterSerialNumber.value = "";

}

}

// filter helper functions

function filterDataFilterType(data)

{

let offset = (protocolVersion == 0) ? 8 : 9;

if (data.getUint8(offset) == 0)

{

return "Hepa Filter";

}

return "Main Filter";

}

function filterDataHoursRun(data)

{

let offset = (protocolVersion == 0) ? 9 : 10;

mins = data.getUint16(offset, true) / 60;

return mins.toFixed(1);

}

function filterDataStrings(data)

{

let offset = (protocolVersion == 0) ? 11 : 12;

thelen = data.getUint8(offset);

const view2 = new DataView(data.buffer, offset + 1, thelen);

const decodedValue = new TextDecoder().decode(view2);

return decodedValue.split("|");

}

// Filter history class

class filterHistoryClass extends historyClass

{

setExportFileHeader()

{

this.fileData = "Date,Filter Type,Hours Run,Part Number,Serial Number,Installer Name\r\n";

}

dataValid(data)

{

return logDataValid(data);

}

dataRecordNumber(data)

{

return logDataRecordNum(data);

}

dataNumOfRecords(data)

{

return logDataNumOfRecords(data);

}

dataAddToFileData(data)

{

this.fileData += this.dataDate(data) + "," +

this.dataRecordType(data) + "," +

filterDataHoursRun(data) + ",";

const filterStrings = filterDataStrings(data);

if (filterStrings.length >= 3)

{

this.fileData += filterStrings[0] + "," +

filterStrings[2] + "," +

filterStrings[1] + "\r\n";

}

else

{

this.fileData += ",,\r\n";

}

}

dataAddDetailElements(detailsDiv, data)

{

this.addRowElement(detailsDiv, "Hours Run", filterDataHoursRun(data));

const filterStrings = filterDataStrings(data);

if (filterStrings.length >= 3)

{

this.addRowElement(detailsDiv, "Part Number", filterStrings[0]);

this.addRowElement(detailsDiv, "Serial Number", filterStrings[2]);

this.addRowElement(detailsDiv, "Installer Name", filterStrings[1]);

}

}

dataDate(data)

{

return logDataDate(data);

}

dataRecordType(data)

{

return filterDataFilterType(data);

}

}

// filter ble class

class blueToothFilter extends blueToothIf

{

constructor ()

{

super ("Filter", 1, "f8885b19-317b-451c-bb17-ad16498fa694");

this.id = 0;

this.retries = 0;

this.send = false;

this.filterId = 0;

this.filterData = null;

}

disconnected()

{

super.disconnected();

this.send = false;

}

addFilter(filterId, filterData)

{

this.send = true;

this.filterId = filterId;

this.filterData = filterData;

}

advanceId()

{

this.id++;

if (this.id > 200)

{

this.id = 0;

}

}

getId()

{

return this.id;

}

setReadRecord(rec)

{

bleiface.setMessage(DBG\_INFO, this.getName() + " set record to read " + rec);

this.advanceId();

this.retries = 0;

let data = new Uint8Array(2);

data[0] = this.id;

data[1] = rec;

this.write(data);

}

setGotoRecord(cmd, rec)

{

bleiface.setMessage(DBG\_INFO, this.getName() + " set record to goto " + rec);

this.advanceId();

this.retries = 0;

let data = new Uint8Array(4);

data[0] = this.id;

data[1] = cmd;

data[2] = rec & 0x00ff;

data[3] = (rec >> 8) & 0x00ff;

this.write(data);

}

writeFilter(filterId, filterData)

{

const encodedValue = new TextEncoder().encode(filterData);

const sdata = new Uint8Array(3 + encodedValue.length);

bleiface.setMessage(DBG\_INFO, this.getName() + " write filter data " + filterId);

sdata[0] = 3;

sdata[1] = 5;

sdata[2] = filterId;

sdata.set(encodedValue, 3);

this.write(sdata);

}

run()

{

if (this.send)

{

this.send = false;

this.writeFilter(this.filterId, this.filterData);

}

else

{

filterHistory.handleHistoryLoading();

}

}

init()

{

switch (this.initState)

{

case 0:

bleiface.setMessage(DBG\_CONNECT, "Reading current Hepa Filter");

this.initState++;

this.setReadRecord(1); // read the current Hepa Filter

break;

case 1:

this.initState++;

bleiface.itemReadWithBusy(this);

break;

case 2:

if (!this.isValueValid() ||

this.getValue().getUint8(0) != this.id)

{

if (this.isValueValid())

{

bleiface.setError("Value is valid");

bleiface.setError("Data = " + this.getValue() + " id=" + this.id);

}

else

{

bleiface.setError("Value is invalid");

}

this.retries++;

if (this.retries > 3)

{

bleiface.setError("Failed to read current HEPA Filter");

break;

}

bleiface.itemReadWithBusy(this);

break;

}

else

{

this.initState++;

if (logDataValid(this.getValue()))

{

hepaInstalledDate.value = logDataDate(this.getValue());

const filterStrings = filterDataStrings(this.getValue());

if (filterStrings.length >= 3)

{

hepaPartNumber.value = filterStrings[0];

hepaInstallerName.value = filterStrings[1];

hepaSerialNumber.value = filterStrings[2];

}

}

}

bleiface.setMessage(DBG\_CONNECT, "Reading current Main Filter");

this.setReadRecord(2); // read the current Main Filter

break;

case 3:

this.initState++;

bleiface.itemReadWithBusy(this);

break;

case 4:

if (!this.isValueValid() ||

this.getValue().getUint8(0) != this.id)

{

this.retries++;

if (this.retries > 3)

{

bleiface.setError("Failed to read current Main Filter");

break;

}

bleiface.itemReadWithBusy(this);

break;

}

else

{

super.init()

if (logDataValid(this.getValue()))

{

mainInstalledDate.value = logDataDate(this.getValue());

const filterStrings = filterDataStrings(this.getValue());

if (filterStrings.length >= 3)

{

mainPartNumber.value = filterStrings[0];

mainInstallerName.value = filterStrings[1];

mainSerialNumber.value = filterStrings[2];

}

}

}

break;

}

}

setRead(readValue)

{

super.setRead(readValue);

if (this.isInit())

{

var id = readValue.getUint8(0);

filterHistory.handleRecordRead(id, readValue);

}

}

}

//==========================================================

//

// Real Time Clock

//

//==========================================================

// Bluetooth interface

class blueToothRtc extends blueToothIf

{

constructor ()

{

super ("RTC", 0, "469cbb22-dc08-4f1a-9e52-2626391ae161");

}

init()

{

if (this.initState == 0)

{

super.init();

const date = new Date();

bleiface.setMessage(DBG\_CONNECT, "Setting RTC");

let data = new Uint8Array(7);

data[0] = date.getDate();

data[1] = date.getMonth() + 1;

data[2] = date.getFullYear() & 0x00ff;

data[3] = (date.getFullYear() >> 8) & 0x00ff;

data[4] = date.getHours();

data[5] = date.getMinutes();

data[6] = date.getSeconds();

this.write(data);

}

}

}

//==========================================================

//

// Commission

//

//==========================================================

// variables

var commissionDataValid = false;

const commissionForm = document.querySelector("#commissionForm");

const cInstalledDate = document.querySelector("#cInstalledDate");

const cInstallerCompany = document.querySelector("#cInstallerCompany");

const cInstallerName = document.querySelector("#cInstallerName");

const cInstallerPhone = document.querySelector("#cInstallerPhone");

const cMachineName = document.querySelector("#cMachineName");

const cPlantNumber = document.querySelector("#cPlantNumber");

const cPlantHours = document.querySelector("#cPlantHours");

const cSaveRow = document.querySelector("#cSaveRow");

// Callback events

commissionForm.addEventListener("submit", (event) =>

{

event.preventDefault();

var cdata = cInstalledDate.value + "|" +

cInstallerName.value + "|" +

cInstallerPhone.value + "|" +

cInstallerCompany.value + "|" +

cMachineName.value + "|" +

cPlantNumber.value + '|' +

cPlantHours.value;

var item = bleiface.getItem("Commission");

if (item)

{

item.setCommission(cdata);

}

});

// helper functions

function authorizeCommission()

{

let allowed = true;

if (commissionDataValid)

{

allowed = false;

if (authorizeControl &&

authorizeControl.getAuthorized())

{

allowed = true;

}

}

cSaveRow.style.display = (!allowed) ? "none" : "block";

cInstallerName.readOnly = !allowed;

cInstallerPhone.readOnly = !allowed;

cInstallerCompany.readOnly = !allowed;

cMachineName.readOnly = !allowed;

cPlantNumber.readOnly = !allowed;

cPlantHours.readOnly = !allowed;

}

// bluetooth interface class

class blueToothCommission extends blueToothIf

{

constructor ()

{

super ("Commission", 0, "f31e779c-7729-4073-8319-a0a2f33524ed");

this.send = false;

this.sendValue = "";

}

disconnected()

{

this.send = false;

}

setCommission(commissionData)

{

this.sendValue = commissionData;

this.send = true;

}

setRead(readValue)

{

super.setRead(readValue);

commissionDataValid = false;

const decodedValue = new TextDecoder().decode(readValue);

const myArray = decodedValue.split("|");

cSaveRow.style.display = "none";

if (myArray.length >= 7)

{

cInstalledDate.value = myArray[1];

cInstallerName.value = myArray[2];

cInstallerPhone.value = myArray[3];

cInstallerCompany.value = myArray[4];

cMachineName.value = myArray[5];

cPlantNumber.value = myArray[6];

if (myArray.length >= 8)

{

cPlantHours.value = myArray[7];

}

else

{

cPlantHours.value = "";

}

commissionDataValid = (myArray[0] === "V");

}

if (!commissionDataValid)

{

cInstalledDate.value = getNowDateForInput();

}

authorizeCommission();

}

run()

{

if (this.send)

{

this.send = false;

const encodedValue = new TextEncoder().encode(this.sendValue);

this.write(encodedValue);

}

}

}

//==========================================================

//

// Settings

//

//==========================================================

// variables

const settingForm = document.querySelector("#settingForm");

// Settings fields

const fieldRampRate = document.querySelector("#frampRate");

const fieldStartPause = document.querySelector("#fstartPause");

const fieldDoorSpeed = document.querySelector("#fdoorSpeed");

const fieldFixedSpeed = document.querySelector("#ffixedSpeed");

const fieldTargetPressure = document.querySelector("#ftargetPressure");

const fieldMinSpeed = document.querySelector("#fminSpeed");

const fieldStepSize = document.querySelector("#fstepSize");

const fieldHoldTime = document.querySelector("#fholdTime");

// event callbacks

settingForm.addEventListener("submit", (event) =>

{

event.preventDefault();

const buffer = new ArrayBuffer(24);

const view = new DataView(buffer);

view.setUint16( 0, getRadioInput("fmode"), true);

view.setUint16( 2, fieldRampRate.value, true);

view.setUint16( 4, fieldFixedSpeed.value, true);

view.setUint16( 6, fieldDoorSpeed.value, true);

view.setUint16( 8, fieldMinSpeed.value, true);

view.setUint16(10, fieldTargetPressure.value, true);

view.setUint16(12, fieldStepSize.value, true);

view.setUint16(14, fieldHoldTime.value, true);

view.setUint16(16, fieldStartPause.value, true);

view.setUint16(18, getRadioInput("fdoorMode"), true);

view.setUint16(20, getRadioInput("falarmMode"), true);

view.setUint16(22, getRadioInput("fco2Mode"), true);

var item = bleiface.getItem("Settings");

if (item)

{

item.sendSettings(buffer);

}

});

function handleModeClick(myRadio)

{

if (myRadio.value == 0)

{

fieldFixedSpeed.disabled = false;

fieldTargetPressure.disabled = true;

fieldMinSpeed.disabled = true;

fieldStepSize.disabled = true;

fieldHoldTime.disabled = true;

}

else

{

fieldFixedSpeed.disabled = true;

fieldTargetPressure.disabled = false;

fieldMinSpeed.disabled = false;

fieldStepSize.disabled = false;

fieldHoldTime.disabled = false;

}

}

// bluetooth interface

class blueToothSettings extends blueToothIf

{

constructor ()

{

super ("Settings", 0, "3bfa30cb-f9f9-4122-9367-979de74970b6");

this.send = false;

this.sendData = null;

}

disconnected()

{

super.disconnected();

this.send = false;

}

setRead(readValue)

{

super.setRead(readValue);

var mode = readValue.getUint16(0, true);

setRadioInput("fmode", readValue.getUint16(0, true));

fieldRampRate.value = readValue.getUint16(2, true);

fieldFixedSpeed.value = readValue.getUint16(4, true);

fieldDoorSpeed.value = readValue.getUint16(6, true);

fieldMinSpeed.value = readValue.getUint16(8, true);

fieldTargetPressure.value = readValue.getUint16(10, true);

fieldStepSize.value = readValue.getUint16(12, true);

fieldHoldTime.value = readValue.getUint16(14, true);

fieldStartPause.value = readValue.getUint16(16, true);

setRadioInput("fdoorMode", readValue.getUint16(18, true));

setRadioInput("falarmMode", readValue.getUint16(20, true));

setRadioInput("fco2Mode", readValue.getUint16(22, true));

fieldFixedSpeed.disabled = mode == 1;

fieldTargetPressure.disabled = mode == 0;

fieldMinSpeed.disabled = mode == 0;

fieldStepSize.disabled = mode == 0;

fieldHoldTime.disabled = mode == 0;

}

sendSettings(data)

{

this.send = true;

this.sendData = data;

}

run()

{

if (this.send)

{

this.send = false;

this.write(this.sendData);

}

}

}

//==========================================================

//

// Status

//

//==========================================================

// variables

const sFanSpeedBar = document.querySelector("#sFanSpeedBar");

const sPressureBar = document.querySelector("#sPressureBar");

const sDoorState = document.querySelector("#sDoorState");

const sAlarmState = document.querySelector("#sAlarmState");

const sCO2State = document.querySelector("#sCO2State");

const sMotorTime = document.querySelector("#sMotorTime");

const sHepaFilterTime = document.querySelector("#sHepaFilterTime");

const sMainFilterTime = document.querySelector("#sMainFilterTime");

// bluetooth interface

class blueToothStatus extends blueToothIf

{

constructor ()

{

super ("Status", 0, "d65a4ca2-5c9a-43ff-aac3-3091ba7e165b");

}

valueToText(v, v0, v3, vx)

{

if (v == 3) return v3;

if (v == 0) return v0;

return vx;

}

decodeFanSpeed(fan)

{

let tempFan = (((fan \* 100) + 0.5) / 255 >> 0);

if (tempFan > 100)

{

tempFan = 100;

}

return tempFan;

}

setRead(readValue)

{

super.setRead(readValue);

let fan = this.decodeFanSpeed(readValue.getUint8(0));

sFanSpeedBar.style.width = fan + '%';

sFanSpeedBar.innerHTML = fan + '%';

sDoorState.value = this.valueToText(readValue.getUint8(1), "Closed", "Disabled", "Opened");

sAlarmState.value = this.valueToText(readValue.getUint8(2), "Inactive", "Disabled", "Alarmed");

sCO2State.value = this.valueToText(readValue.getUint8(3), "Inactive", "Disabled", "Alarmed");

//NOTE: With the pressure. need to scale to 100% (ie choose a range and if greater the set to 100)

let pressure = (readValue.getUint16(4, true) / 2 >> 0);

if (pressure > 100)

{

pressure = 100;

}

sPressureBar.style.width = pressure + '%';

sPressureBar.innerHTML = readValue.getUint16(4, true) + 'pa';

sHepaFilterTime.value = toHours(readValue.getUint32(6, true));

sMainFilterTime.value = toHours(readValue.getUint32(10, true));

sMotorTime.value = toHours(readValue.getUint32(14, true));

}

run()

{

if (selectDisplay === "statusDiv")

{

bleiface.itemReadWithBusy(this);

}

}

}

//==========================================================

//

// General

//

//==========================================================

// variables

var protocolVersion = 0;

// bluetooth interface

class blueToothGeneral extends blueToothIf

{

constructor ()

{

super ("General", 1, "59bb5014-1e25-4cdb-aab9-c675d7d5b608");

this.version = "";

}

getVersion()

{

return this.version;

}

setRead(readValue)

{

super.setRead(readValue);

if (readValue.getUint8(0) == 0)

{

// this is the new version of the general message

protocolVersion = readValue.getUint8(1);

this.version = readValue.getUint8(6) + "." + pad2(readValue.getUint8(7));

}

else

{

protocolVersion = 0;

const decodedValue = new TextDecoder().decode(readValue);

this.version = decodedValue;

}

bleiface.setMessage(DBG\_CONNECT, "Version: " + this.version);

}

}

//==========================================================

//

// Calibrate

//

//==========================================================

// callback functions

function calibratePoint(thePoint)

{

if (calibrateControl)

{

calibrateControl.executeSetCalibrate(thePoint);

}

}

function eraseLog(theLog)

{

if (calibrateControl)

{

calibrateControl.executeEraseLog(theLog);

}

}

// Calibrate class

class calibrateClass

{

constructor()

{

this.elementPressureBar = document.getElementById("calPressureBar");

this.elementAdcBar = document.getElementById("calAdcBar");

this.elementRealPressure = document.getElementById("calRealPressure");

this.elementDbg = document.getElementById("calDbg");

this.pressure = 0;

this.adc = 0;

this.one\_pa = 0;

this.two\_pa = 0;

this.send\_req = false;

this.send\_cmd = 0;

this.send\_idx = 0;

this.send\_value = 0;

this.timer = 0;

this.deleteAlarm = false;

this.deleteFilter = false;

}

tick()

{

if (this.timer > 0)

{

this.timer--;

if (this.timer == 0)

{

this.elementDbg.innerHTML = "";

}

}

}

restart()

{

this.pressure = 0;

this.adc = 0;

this.one\_pa = 0;

this.two\_pa = 0;

this.elementRealPressure.value = "0";

this.elementPressureBar.style.width = '0%';

this.elementPressureBar.innerHTML = "Not Calibrated";

this.elementAdcBar.style.width = '0%';

this.elementAdcBar.innerHTML = '0';

this.send\_req = false;

this.deleteAlarm = false;

this.deleteFilter = false;

}

processData(data)

{

if (data.getUint8(0) == 0)

{

this.one\_pa = data.getUint16(1, true);

this.two\_pa = data.getUint16(3, true);

this.adc = data.getUint16(5, true);

this.pressure = data.getUint16(7, true);

let temp = (((this.adc \* 100) + 0.5) / 4096 >> 0);

if (temp > 100)

{

temp = 100;

}

this.elementAdcBar.style.width = temp + '%';

this.elementAdcBar.innerHTML = this.adc;

if (this.pressure == 0xffff)

{

this.elementPressureBar.style.width = '0%';

this.elementPressureBar.innerHTML = "Not Calibrated";

}

else

{

let temp = (((this.pressure \* 100) + 0.5) / 250 >> 0);

if (temp > 100)

{

temp = 100;

}

this.elementPressureBar.style.width = temp + '%';

this.elementPressureBar.innerHTML = this.pressure;

}

if (this.deleteAlarm && (data.getUint8(9) == 0))

{

this.deleteAlarm = false;

this.elementDbg.innerHTML = "Alarm Log Deleted";

this.timer = 5;

}

if (this.deleteFilter && (data.getUint8(10) == 0))

{

this.deleteFilter = false;

this.elementDbg.innerHTML = "Filter Log Deleted";

this.timer = 5;

}

}

}

executeSetCalibrate(thePoint)

{

this.send\_req = false;

this.send\_idx = 0;

if (thePoint == 0)

{

this.send\_cmd = 1; // delete

}

else

{

this.send\_cmd = 0; // set

this.send\_value = 0;

this.send\_idx = thePoint - 1;

}

this.send\_value = this.elementRealPressure.value;

this.send\_req = true;

bleiface.setMessage(DBG\_INFO, "Sending calibrate: Cmd=" + this.send\_cmd + ", Value=" + this.send\_value + "Idx=" + this.send\_idx);

}

executeEraseLog(theLog)

{

bleiface.setMessage(DBG\_INFO, "Erase Log: Log=" + theLog);

this.send\_req = false;

this.send\_cmd = theLog + 2;

this.send\_idx = 0;

this.send\_value = 0;

this.send\_req = true;

}

handleActions(item)

{

if (this.send\_req)

{

this.send\_req = false;

item.sendRequest(this.send\_cmd, this.send\_idx, this.send\_value);

if (this.send\_cmd == 2)

{

this.deleteFilter = true;

this.elementDbg.innerHTML = "Deleting Filter Log...";

}

if (this.send\_cmd == 3)

{

this.deleteAlarm = true;

this.elementDbg.innerHTML = "Deleting Alarm Log...";

}

return true;

}

if (selectDisplay === "calibrateDiv")

{

bleiface.itemReadWithBusy(item);

return true;

}

return false;

}

}

// bluetooth interface

class blueToothCalibrate extends blueToothIf

{

constructor ()

{

super ("Calibrate", 1, "aba886d1-094b-4808-87a8-6d1d73afd8fe");

}

setRead(readValue)

{

super.setRead(readValue);

calibrateControl.processData(readValue);

}

sendRequest(sendCmd, sendIdx, sendValue)

{

let data = new Uint8Array(4);

const buffer = new ArrayBuffer(4);

const view = new DataView(buffer);

data[0] = sendCmd;

data[1] = sendIdx;

data[2] = sendValue & 0x00ff;

data[3] = (sendValue >> 8) & 0x00ff;

this.write(data);

}

run()

{

calibrateControl.handleActions(this);

}

}

//==========================================================

//

// Authorize

//

//==========================================================

// callback events

function loginClicked()

{

authorizeControl.handleClicked();

}

// authorize class

class authorizeClass

{

constructor()

{

this.authorized = false;

this.sendLogin = false;

this.sendPassword = "";

this.queryLogin = false;

this.count = 0;

this.timer = 0;

this.elementBtn = document.getElementById("loginBtn");

this.elementPassword = document.getElementById("loginPassword");

this.elementDbg = document.getElementById("loginDbg");

this.elementMenuItem = document.getElementById("menuItemLogin");

this.elementHeader = document.getElementById("loginDivHeader");

this.elementPasswordDiv = document.getElementById("loginPasswordRow");

this.calibrateMenuItem = 0;

this.settingMenuItem = 0;

}

handleClicked()

{

if (!this.authorized)

{

this.sendPassword = this.elementPassword.value;

this.sendLogin = true;

}

else

{

this.sendPassword = "";

this.sendLogin = true;

}

}

tick()

{

if (this.timer > 0)

{

this.timer--;

if (this.timer == 0)

{

if (!this.authorized)

{

this.elementDbg.innerHTML = "Enter password to login";

}

else

{

this.elementDbg.innerHTML = "Click button to logout";

}

}

}

}

restart()

{

this.setAuthorized(false);

this.sendLogin = true;

this.sendPassword = "";

this.queryLogin = false;

this.elementPassword.value = "";

}

getAuthorized()

{

return this.authorized;

}

setAuthorized(authorized)

{

if (authorized != this.authorized)

{

this.authorized = authorized;

if (authorized)

{

this.elementBtn.value = "Logout";

this.elementMenuItem.innerHTML = "Logout";

this.elementHeader.innerHTML = "Logout";

this.elementPasswordDiv.style.display = "none";

if (!this.calibrateMenuItem)

{

this.calibrateMenuItem = document.createElement("a");

this.calibrateMenuItem.setAttribute("href", "javascript:showDiv('calibrateDiv')");

this.calibrateMenuItem.innerHTML = "Calibrate";

this.elementMenuItem.parentNode.insertBefore(this.calibrateMenuItem, this.elementMenuItem.nextSibling);

}

if (!this.settingMenuItem)

{

this.settingMenuItem = document.createElement("a");

this.settingMenuItem.setAttribute("href", "javascript:showDiv('settingDiv')");

this.settingMenuItem.innerHTML = "Settings";

this.elementMenuItem.parentNode.insertBefore(this.settingMenuItem, this.elementMenuItem.nextSibling);

}

}

else

{

this.elementBtn.value = "Login";

this.elementMenuItem.innerHTML = "Login";

this.elementHeader.innerHTML = "Login";

this.elementPasswordDiv.style.display = "block";

if (this.calibrateMenuItem)

{

this.calibrateMenuItem.remove();

this.calibrateMenuItem = 0;

}

if (this.settingMenuItem)

{

this.settingMenuItem.remove();

this.settingMenuItem = 0;

}

}

this.elementPassword.readOnly = authorized;

}

authorizeCommission();

}

processData(data)

{

const decodedValue = new TextDecoder().decode(data);

let wasAuthorized = this.authorized;

this.queryLogin = false;

this.setAuthorized(decodedValue == "Y");

if (this.sendPassword == "")

{

if (!this.authorized && wasAuthorized)

{

this.timer = 5;

this.elementDbg.innerHTML = "Successful Logout";

}

}

else

{

if (!this.authorized)

{

this.timer = 5;

this.elementDbg.innerHTML = "Login failed";

}

else

{

this.timer = 5;

this.elementDbg.innerHTML = "Successful Login";

this.elementPassword.value = "";

}

}

}

handleActions(item)

{

if (this.sendLogin)

{

this.timer = 5;

this.elementDbg.innerHTML = "Trying...";

this.sendLogin = false;

this.queryLogin = true;

this.count = 2;

item.sendLogin(this.sendPassword);

return true;

}

if (this.queryLogin)

{

if (this.count == 0)

{

this.timer = 5;

this.elementDbg.innerHTML = "Login/Logout failed. No response";

this.queryLogin = false;

this.setAuthorized(false);

}

else

{

this.count--;

bleiface.itemReadWithBusy(item);

}

return true;

}

return false;

}

}

// bluetooth interface

class blueToothAuthorize extends blueToothIf

{

constructor ()

{

super ("Authorize", 1, "af297289-ad63-457c-bc81-de7db0450df7");

}

setRead(readValue)

{

super.setRead(readValue);

authorizeControl.processData(readValue);

}

sendLogin(password)

{

const encodedValue = new TextEncoder().encode(password);

this.write(encodedValue);

}

run()

{

authorizeControl.handleActions(this);

}

}

//==========================================================

//

// Alarm log blue tooth

//

//==========================================================

// Event callback

function gotoAlarmRec(recDir)

{

if (alarmHistory)

{

alarmHistory.gotoRecord(recDir);

}

}

function saveAlarmRecords()

{

if (alarmHistory)

{

alarmHistory.exportRecords();

}

}

// Alarm history class

class alarmHistoryClass extends historyClass

{

setExportFileHeader()

{

this.fileData = "Date,Pressure Alarm,CO2 Alarm,Door,Fan Speed,Pressure\r\n";

}

dataValid(data)

{

return logDataValid(data);

}

dataRecordNumber(data)

{

return logDataRecordNum(data);

}

dataNumOfRecords(data)

{

return logDataNumOfRecords(data);

}

dataAddToFileData(data)

{

this.fileData += this.dataDate(data) + "," +

this.dataRecordType(data) + "," +

this.dataDoorState(data) + "," +

this.dataFanSpeed(data) + "," +

this.dataPressure(data) + "\r\n";

}

dataAddDetailElements(detailsDiv, data)

{

this.addRowElement(detailsDiv, "Door State", this.dataDoorState(data));

this.addRowElement(detailsDiv, "Fan Speed", this.dataFanSpeed(data));

this.addRowElement(detailsDiv, "Pressure", this.dataPressure(data));

}

dataDate(data)

{

return logDataDate(data) + " " + logDataTime(data);

}

dataRecordType(data)

{

if ((data.getUint8(13) & 0x80) == 0x80)

{

if ((data.getUint8(13) & 0x01) == 0x01)

{

return "CO2 Active";

}

return "CO2 Idle";

}

if ((data.getUint8(12) & 0x01) == 0x01)

{

return "Alarm Active";

}

return "Alarm Idle";

}

dataDoorState(data)

{

if (data.getUint8(14) == 0)

{

return "Door Closed";

}

return "Door Opened";

}

dataFanSpeed(data)

{

return data.getUint8(15);

}

dataPressure(data)

{

return data.getUint16(16, true);

}

}

// Alarm Bluetooth interface class

class blueToothAlarm extends blueToothIf

{

constructor ()

{

super ("Alarm", 1, "3df84905-2b48-4223-8588-7f52f8cbb71d");

this.id = 0;

this.retries = 0;

}

advanceId()

{

this.id++;

if (this.id > 200)

{

this.id = 0;

}

}

getId()

{

return this.id;

}

setReadRecord(rec)

{

bleiface.setMessage(DBG\_INFO, this.getName() + " set record to read " + rec);

this.advanceId();

this.retries = 0;

let data = new Uint8Array(2);

data[0] = this.id;

data[1] = rec;

this.write(data);

}

setGotoRecord(cmd, rec)

{

bleiface.setMessage(DBG\_INFO, this.getName() + " set record to goto " + rec);

this.advanceId();

this.retries = 0;

let data = new Uint8Array(4);

data[0] = this.id;

data[1] = cmd;

data[2] = rec & 0x00ff;

data[3] = (rec >> 8) & 0x00ff;

this.write(data);

}

setRead(readValue)

{

super.setRead(readValue);

if (this.isInit())

{

var id = readValue.getUint8(0);

alarmHistory.handleRecordRead(id, readValue);

}

}

run()

{

alarmHistory.handleHistoryLoading();

}

}

//==========================================================

//

// Blue tooth device

//

//==========================================================

class blueToothDevice

{

constructor()

{

this.msg = "";

this.connectState = 0;

this.readingItem = null;

this.device = null;

this.server = null;

this.serviceUUID = new Array("0666a01a-035e-4a01-9bd8-77b2b56a6bd1",

"0666a01b-035e-4a01-9bd8-77b2b56a6bd1");

this.service = new Array(null, null);

this.bleItem = new Array();

this.bleItem.push(new blueToothCommission());

this.bleItem.push(new blueToothSettings());

this.bleItem.push(new blueToothStatus());

this.bleItem.push(new blueToothRtc());

this.bleItem.push(new blueToothGeneral());

this.bleItem.push(new blueToothFilter());

this.bleItem.push(new blueToothAlarm());

this.bleItem.push(new blueToothTest());

this.bleItem.push(new blueToothAuthorize());

this.bleItem.push(new blueToothCalibrate());

this.busy = false;

this.infoStr = "";

this.initDone = false;

this.tickCount = 0;

}

disconnect()

{

if (this.isConnected())

{

this.connectState = 0;

this.device.gatt.disconnect();

connectingTicks.innerHTML = "";

}

}

disconnected()

{

this.device = null;

this.service[0] = null;

this.service[1] = null;

this.server = null;

this.connectState = 0;

this.busy = false;

this.initDone = false;

this.tickCount = 0;

this.bleItem.forEach((item) => item.disconnected());

connectingTicks.innerHTML = "";

}

start()

{

connectingState.innerHTML = "";

this.tickCount = 0;

connectingTicks.innerHTML = "0";

this.setMessage(DBG\_CONNECT, "Connecting to device");

this.connectState = 1;

this.bleItem.forEach((item) => item.start());

}

done()

{

this.connectState = 2;

this.bleItem.forEach((item) => item.done());

setTimeout(bluetoothPoll, 10);

}

setError(error)

{

this.setMessage(DBG\_ERROR, "Last: " + this.msg + " Fail:Reason:" + error);

this.connectState = 3;

connectingTicks.innerHTML = "";

this.bleItem.forEach((item) => item.failed());

this.device = null;

this.service[0] = null;

this.service[1] = null;

this.server = null;

}

init()

{

let result = false;

if (!this.isBusy() && !this.initDone)

{

for (var item of this.bleItem)

{

if (!item.isInit())

{

result = true;

item.init();

}

if (this.isBusy())

{

break;

}

}

if (!result)

{

this.initDone = true;

connectingTicks.innerHTML = "";

connectingState.innerHTML = "";

this.setMessage(DBG\_CONNECT, "Connected to device");

this.setMessage(DBG\_CONNECT, "");

this.setMessage(DBG\_CONNECT, "Version : " + this.getItem("General").getVersion());

activeLinks = "connectedLinks";

if (selectDisplay === "connectingDiv")

{

showDiv("statusDiv");

}

}

}

return this.isBusy() || result;

}

run()

{

if (!this.isBusy())

{

for (var item of this.bleItem)

{

item.run();

if (this.isBusy())

{

break;

}

}

}

return this.isBusy();

}

setBusy(infoStr)

{

this.busy = true;

this.infoStr = infoStr;

}

clearBusy()

{

this.busy = false;

}

isBusy()

{

return this.busy || (!this.isConnected);

}

isConnected()

{

return this.connectState == 2;

}

setDevice(device)

{

this.device = device;

}

setServer(server)

{

this.server = server;

}

setService(idx, service)

{

this.service[idx] = service;

}

setItem(name, characteristic)

{

this.getItem(name).setCharactistic(characteristic);

}

setMessage(msgType, msg)

{

this.msg = msg;

switch (msgType)

{

case DBG\_CONNECT:

connectingState.innerHTML += "<br>" + msg;

break;

case DBG\_INFO:

//phdbg.innerHTML += "<br>" + msg;

break;

case DBG\_ERROR:

connectingError.innerHTML += "<br>" + msg;

break;

}

}

getServiceUUID(idx)

{

return this.serviceUUID[idx].toLowerCase();

}

serverConnect()

{

this.setMessage(DBG\_CONNECT, "Discovering services");

return this.device.gatt.connect();

}

serviceConnect(idx)

{

return this.server.getPrimaryService(this.serviceUUID[idx]);

}

itemConnect(name)

{

this.setMessage(DBG\_INFO, "Finding " + name);

return this.getItem(name).connect(this.service);

}

itemRead(name)

{

this.setMessage(DBG\_INFO, "Reading " + name);

return this.getItem(name).read();

}

setRead(name, valueRead)

{

this.getItem(name).setRead(valueRead);

}

itemReadWithBusy(item)

{

bleiface.setBusy(item.getName());

this.readingItem = item;

item.read()

.then (cdata =>

{

bleiface.readingItem.setRead(cdata);

bleiface.clearBusy();

return true;

})

.catch (error =>

{

bleiface.setError(DBG\_ERROR, "Reading " + bleiface.readingItem.getName() + " error: " + error);

});

}

getItem(name)

{

for (var item of this.bleItem)

{

if (item.getName() === name)

{

return item;

}

}

setError(DBG\_ERROR, "Unable to find '" + name + "' as a ble interface");

return null;

}

tick()

{

if (this.connectState == 1 ||

(this.connectState == 2 && !this.initDone))

{

this.tickCount++;

connectingTicks.innerHTML = this.tickCount;

}

}

}

//==========================================================

//

// Connect to blue tooth process

//

//==========================================================

var bleiface = new blueToothDevice();

function isWebBluetoothEnabled()

{

if (!navigator.bluetooth)

{

menuHide();

showDiv("connectingDiv");

connectingState.innerHTML = "Web Bluetooth API is not available in this browser/device!";

return false

}

return true

}

function connectToBluetooth()

{

bleiface.disconnected();

if (isWebBluetoothEnabled())

{

navigator.bluetooth.requestDevice({

filters: [{name: 'CLEANCAB'}],

optionalServices: [bleiface.getServiceUUID(0), bleiface.getServiceUUID(1)]

})

.then(device =>

{

bleiface.start();

device.addEventListener('gattserverdisconnected', onDisconnected);

bleiface.setDevice(device);

return bleiface.serverConnect();

})

.then(server =>

{

bleiface.setServer(server);

return bleiface.serviceConnect(0);

})

.then(service0 =>

{

bleiface.setService(0, service0);

return bleiface.serviceConnect(1);

})

.then(service1 =>

{

bleiface.setService(1, service1);

return bleiface.itemConnect("General");

})

.then(item0 =>

{

bleiface.setItem("General", item0);

return bleiface.itemConnect("Test");

})

.then(item1 =>

{

bleiface.setItem("Test", item1);

return bleiface.itemConnect("Filter");

})

.then(item2 =>

{

bleiface.setItem("Filter", item2);

return bleiface.itemConnect("RTC");

})

.then(item3 =>

{

bleiface.setItem("RTC", item3);

return bleiface.itemConnect("Commission");

})

.then(item4 =>

{

bleiface.setItem("Commission", item4);

return bleiface.itemConnect("Settings");

})

.then(item5 =>

{

bleiface.setItem("Settings", item5);

return bleiface.itemConnect("Status");

})

.then(item6 =>

{

bleiface.setItem("Status", item6);

return bleiface.itemConnect("Calibrate");

})

.then(item7 =>

{

bleiface.setItem("Calibrate", item7);

return bleiface.itemConnect("Authorize");

})

.then(item8 =>

{

bleiface.setItem("Authorize", item8);

return bleiface.itemConnect("Alarm");

})

.then(item9 =>

{

bleiface.setItem("Alarm", item9);

bleiface.setMessage(DBG\_CONNECT, "Discover done. Reading...");

return bleiface.itemRead("General");

})

.then(cdata =>

{

bleiface.setRead("General", cdata);

return bleiface.itemRead("Commission");

})

.then(cdata =>

{

bleiface.setRead("Commission", cdata);

return bleiface.itemRead("Settings");

})

.then(cdata =>

{

bleiface.setRead("Settings", cdata);

return bleiface.itemRead("Status");

})

.then(cdata =>

{

bleiface.setRead("Status", cdata);

bleiface.setMessage(DBG\_CONNECT, "Reading done. Initializing...");

return bleiface.done();

})

.catch(error =>

{

bleiface.setError(error);

});

}

}

function onDisconnected(event)

{

menuHide();

activeLinks = "idleLinks";

showDiv("connectingDiv");

connectingState.innerHTML = "Device disconnected";

//phdbg.innerHTML = "";

}

function connect()

{

if (isWebBluetoothEnabled())

{

authorizeControl.restart();

calibrateControl.restart();

testSwitch.checked = false;

menuHide();

showDiv("connectingDiv");

//connectingState.innerHTML = "Connecting...";

connectToBluetooth();

}

/\*

if (isWebBluetoothEnabled())

{

device.addEventListener('gattserverdisconnected', onDisconnected);

setTimeout(bluetoothPoll, 1000);

activeLinks = "connectedLinks";

connectingState.innerHTML = "Connected";

})

.catch(error =>

{

connectingState.innerHTML = "Connection Failed: " + error;

});

}

\*/

}

function disconnect()

{

menuHide();

activeLinks = "idleLinks";

showDiv("connectingDiv");

connectingState.innerHTML = "Device disconnected";

bleiface.disconnect();

//phdbg.innerHTML = "";

}

function secondTimer()

{

if (authorizeControl)

{

authorizeControl.tick();

}

if (calibrateControl)

{

calibrateControl.tick();

}

if (bleiface)

{

bleiface.tick();

}

setTimeout(secondTimer, 500);

}

function bluetoothPoll()

{

if (bleiface.isConnected())

{

bleiface.init();

if (!bleiface.isBusy())

{

bleiface.run();

}

if (bleiface.isBusy())

{

setTimeout(bluetoothPoll, 20);

}

else

{

setTimeout(bluetoothPoll, 200);

}

}

}

window.onerror = function myErrorHandler(errorMsg, url, lineNumber) {

alert("Error occured: " + errorMsg);//or any message

return false;

}

</script>

</body>

</html>