加速器学会2012

WebSocketを用いたEPICS Channel Access Clientの開発と実装

総研大　内山 暁仁

KEK　古川 和朗

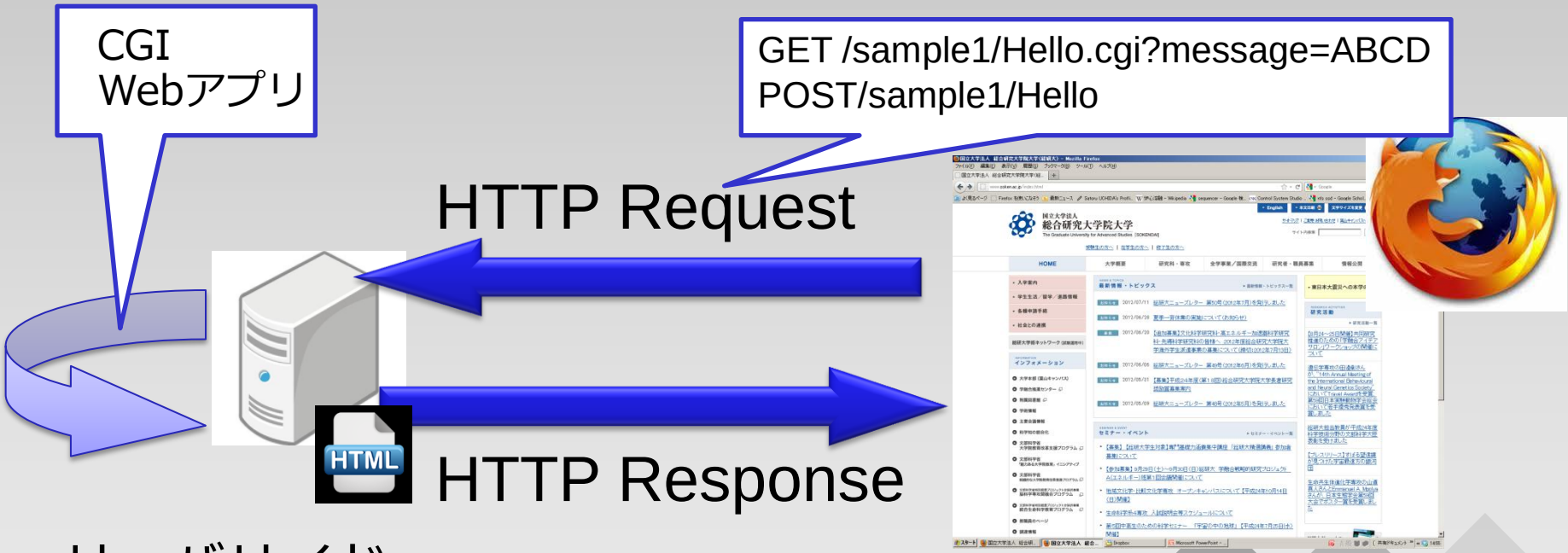
理研仁科　日暮祥英

Webを用いたシステムのメリット

- ✓ WANを利用したシステム提供
- ✓ オペレーションの場所の制約が無くなる
- ✓ 専用ソフトウェアが必要なくなる(ブラウザを利用)
- ✓ GUI開発の容易さ
- ✓ ブラウザが実装されている機器はスマートフォンやPDA色々ある

HTTP(HyperText Transfer Protocol)

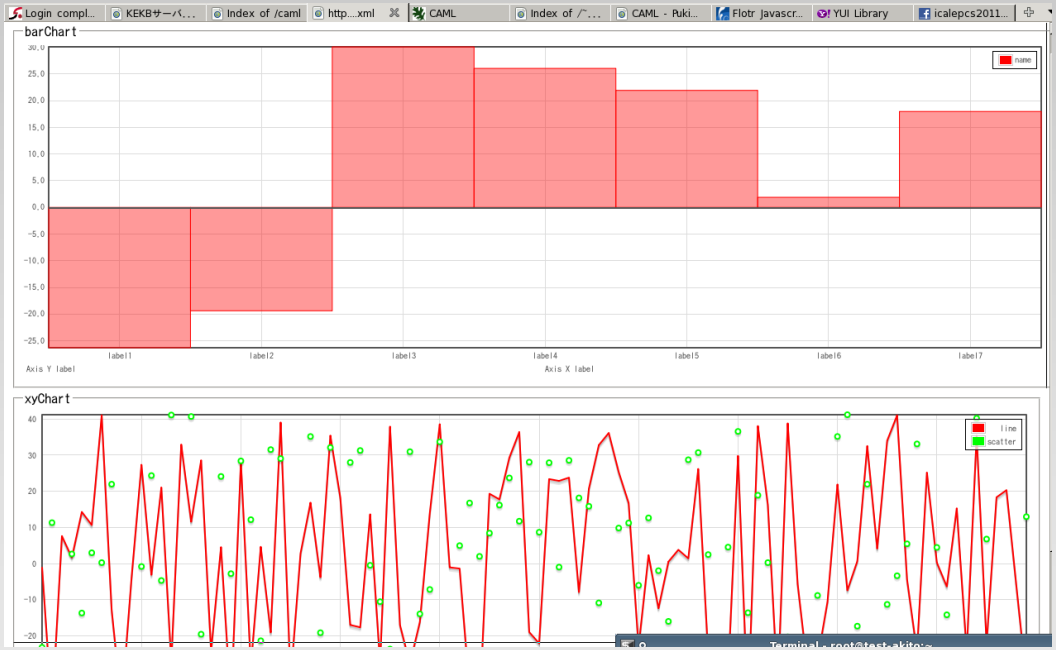
コネクションを張ってから切断(disconnect)するまでの一連の動作

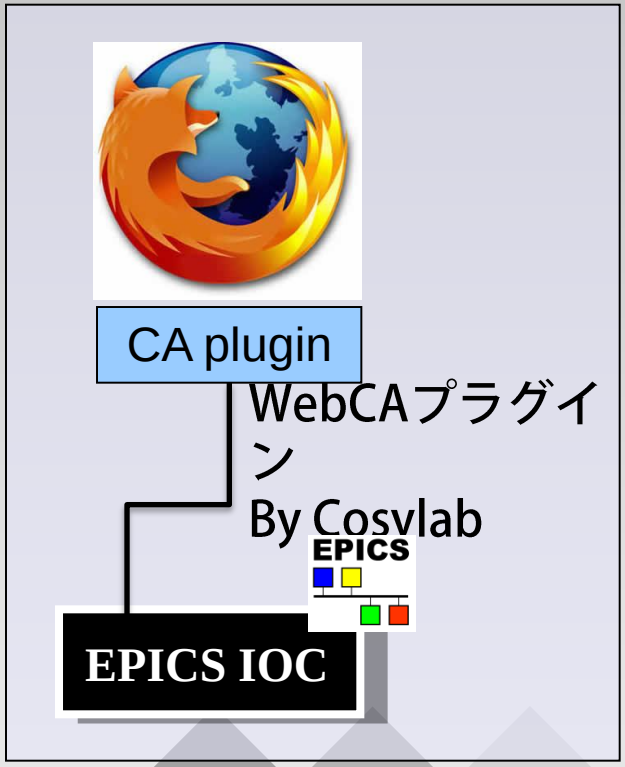


EPICSにおけるWebブラウザからの制御(1)

CAML

Channel **A**ccess **M**arkup **L**anguage　by SNS





- ブラウザに依存
- 直接Channel AccessするのでLANやPC以外での利用は困難
- サンプルでは 1 sec

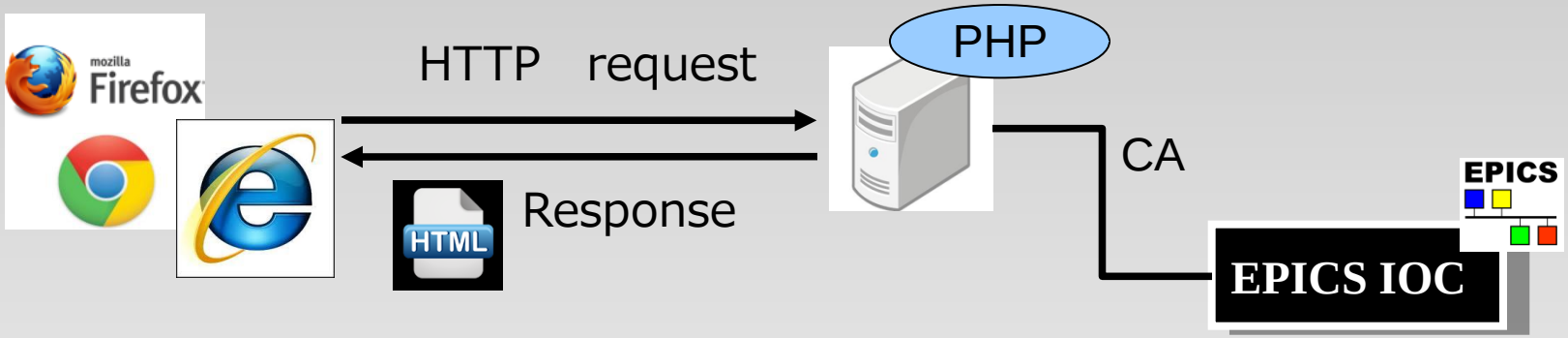
A. UCHIYAMA

EPICSにおけるWebブラウザからの制御(2)

PHP EPICS Module

by PSI

・PHPから**C**hannel **A**ccessを呼べるモジュール

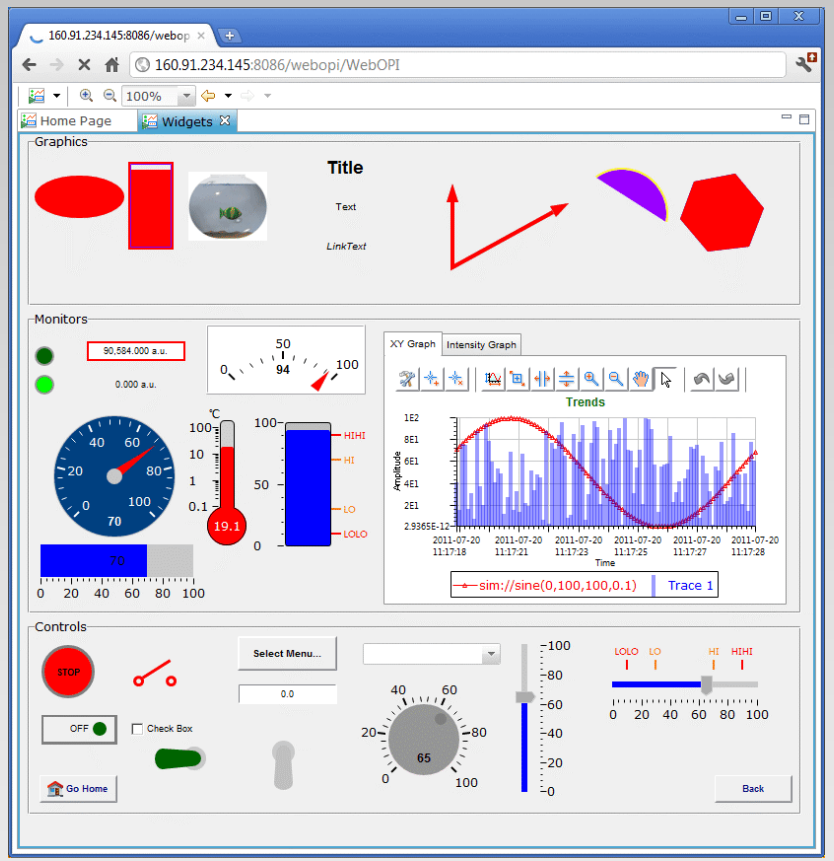


・サポートは止まっている(Only PHP4)
・CAのAPIの提供にとどまるのでウェブアプリは書く必要がある
・CAのモニタ機能（イベント・ドリブンな通知）の利用が困難
・Ajaxを使いある程度(イントラネットで約50 msec)の周期でアクセスを確認

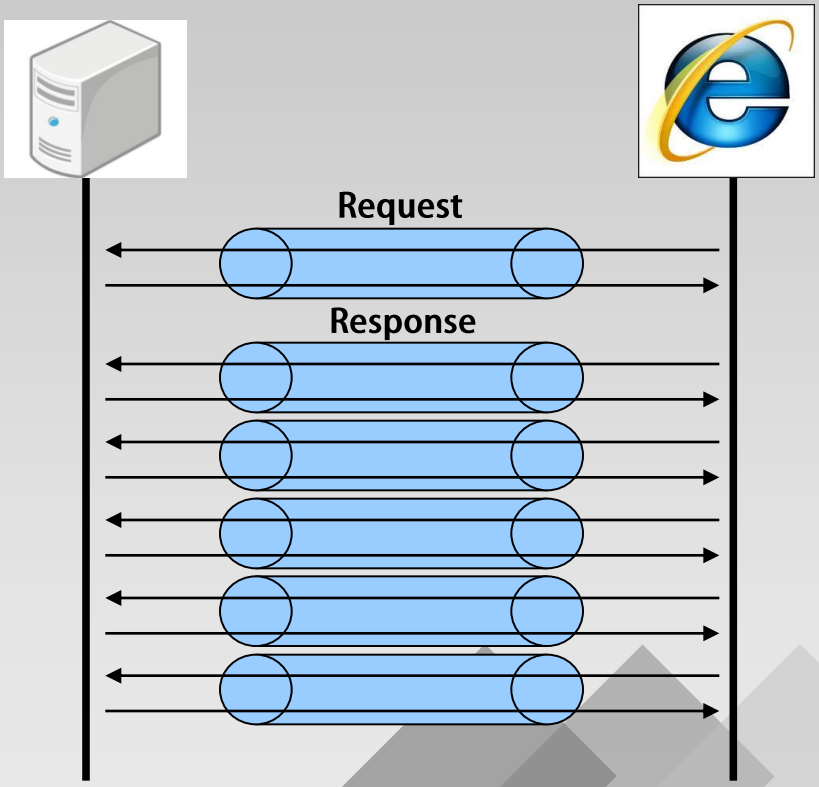
EPICSにおけるWebブラウザからの制御(3)

WebOPI
by SNS

・CSS/BOYで開発したOPIファイルをWebで提供可
- 要Tomcat 7



Ajax (Asynchronous JavaScript + XML)



・Ajax通信を利用してインタラクティブなWebを提供
・イントラネットな環境では 5 msec ～ 100 msec でアクセス

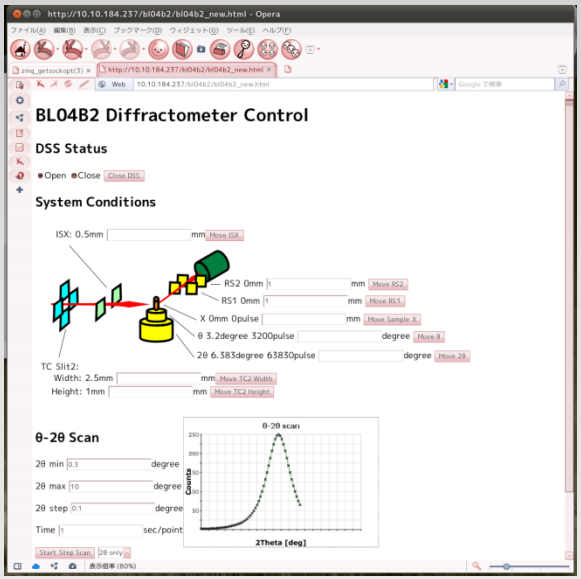
A. UCHIYAMA

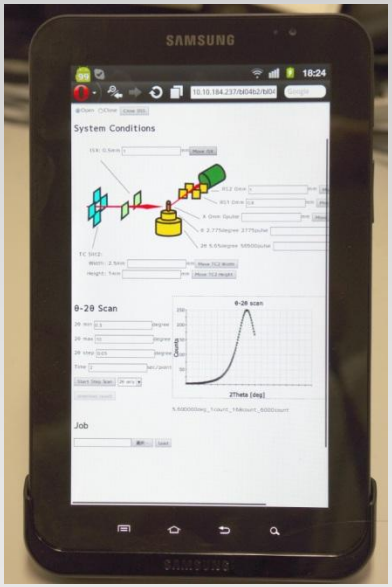
加速器制御におけるWebSocketの可能性

国際会議ICALEPCS2011@フランス グルノーブル 2011/10

–古川行人 氏@SPring-8がビームライン制御（MADOCA）におけるWebSocketの有用性を発表

•シンプルなCで書いたWebSocketサーバをプロトタイプで構築・動作確認





Reference: Proc. ICALEPCS2011, No.WEMAU10, Y. Furukawa

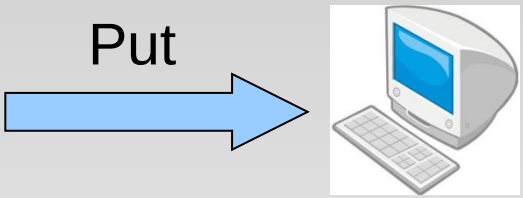
WebSocket（１）

◆リアルタイムに双方向通信を行う為のプロトコル

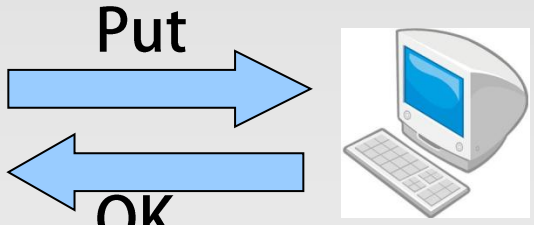
2011/12仕様決定

Ajaxの弱点

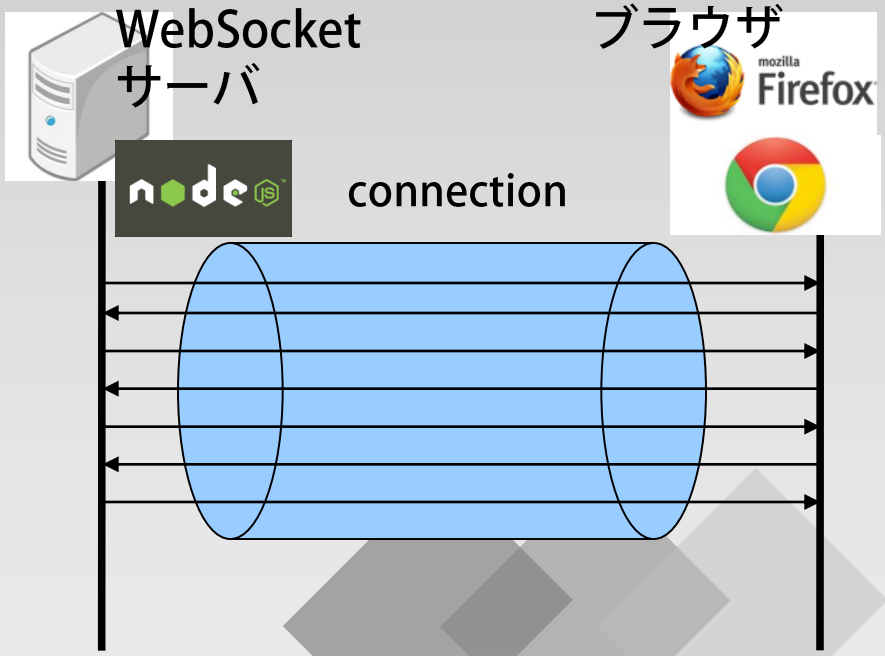
- 通信の都度コネクションを貼らなければならない
  - インタラクティブ化の実現には命令を送ってからの応答速度が重要
- 送った命令に対する確認の手段がない (Callbackがない)
- クライアントがサーバの状態を判断できない



Put ... 成功？失敗？



CallbackにOKが返ってきたから成功。



コネクション維持したまま双方向通信

WebSocket（２）

デメリット

現在全てのブラウザが対応しているわけではない

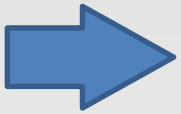

Mozilla Firefox ◎


Google Chrome ◎


Opera ◎


Internet Explorer９ ×

IE10からWebSocketに対応する事が決定


標準化

A. UCHIYAMA

Requirement

EPICSで実現するためには・・・

1. Channel Accessに対応するWebSocketサーバの開発
   - ✓Node.js（サーバサイドJavaScript言語）を利用
2. Node.jsからChannel Access Protocolを呼び出す機能を実装する
   - ✓C++アドオンで提供 (Channel Access for Node.js)
3. クライアントの実装
   - ✓可視化(グラフ・チャート)に向けてJavaScriptライブラリの選択
   - ✓テキストアップデートはDOM操作

Node.js

特徴

簡単にネットワーク・プログラムが書ける

サーバサイドJavaScript (通常JavaScriptはブラウザで動作する言語)

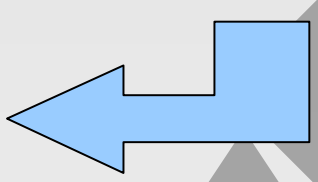
- 全部C言語などで実装する事は難易度が高い
- クライアントもサーバーサイドもJavaScriptで書く事が可能

Node自身はGoogle V8 JavaScript Engineを元に開発

- Google V8 EngineはChromeでJavaScriptを実装する為に開発
- 新しい言語であるためAPIの仕様が変わる可能性がある

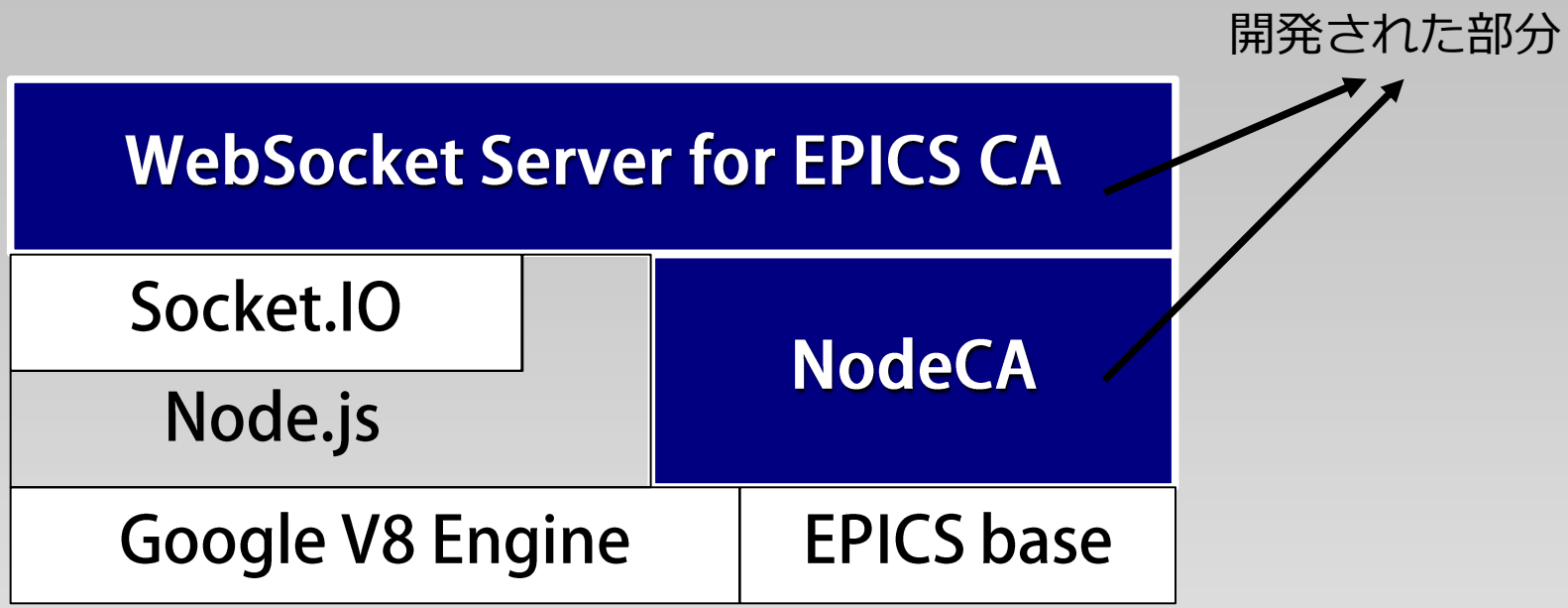
WebSocketのAPIを提供するライブラリがある

Socket.IO



A. UCHIYAMA

NodeCA・WebSocketサーバ概要



caget Example(サーバサイド)

```
var pv = require("../build/Release/nodeca");
val = pv.caget("akito12:example");
console.log(val);
```

アドオン読み込み

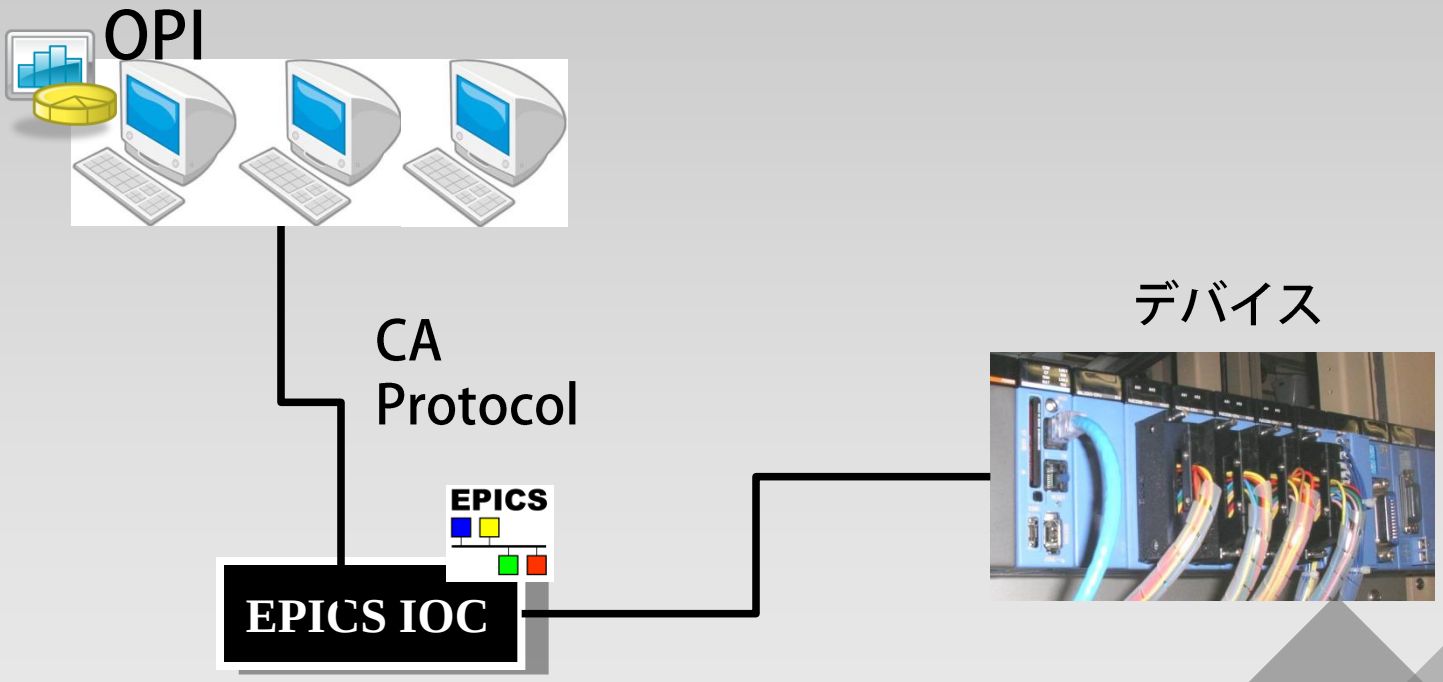
EPICSレコード指定

出力

コンセプトはできるだけシンプルに
プログラムが書ける事

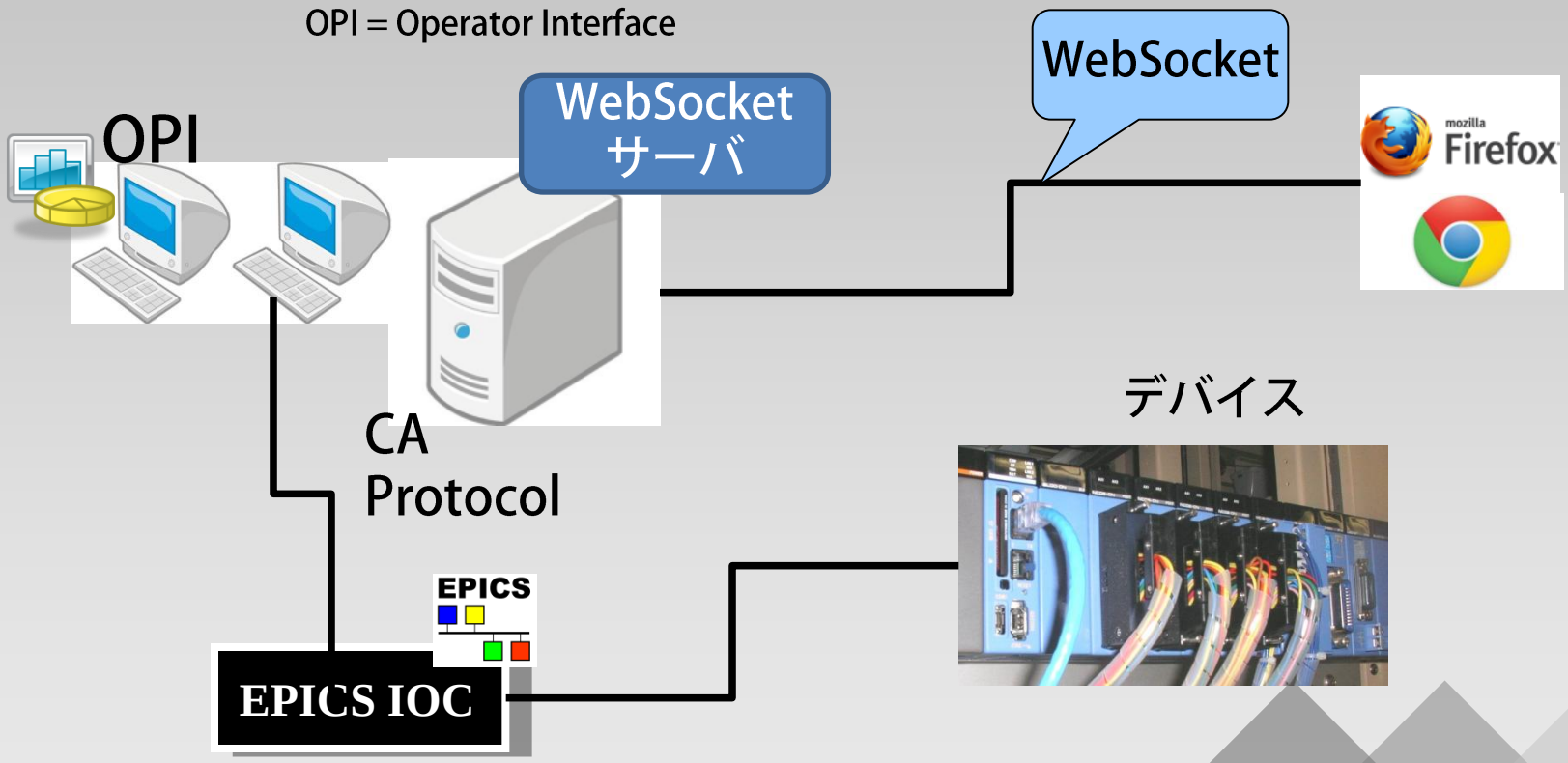
システム概要

通常のEPICSシステムの例



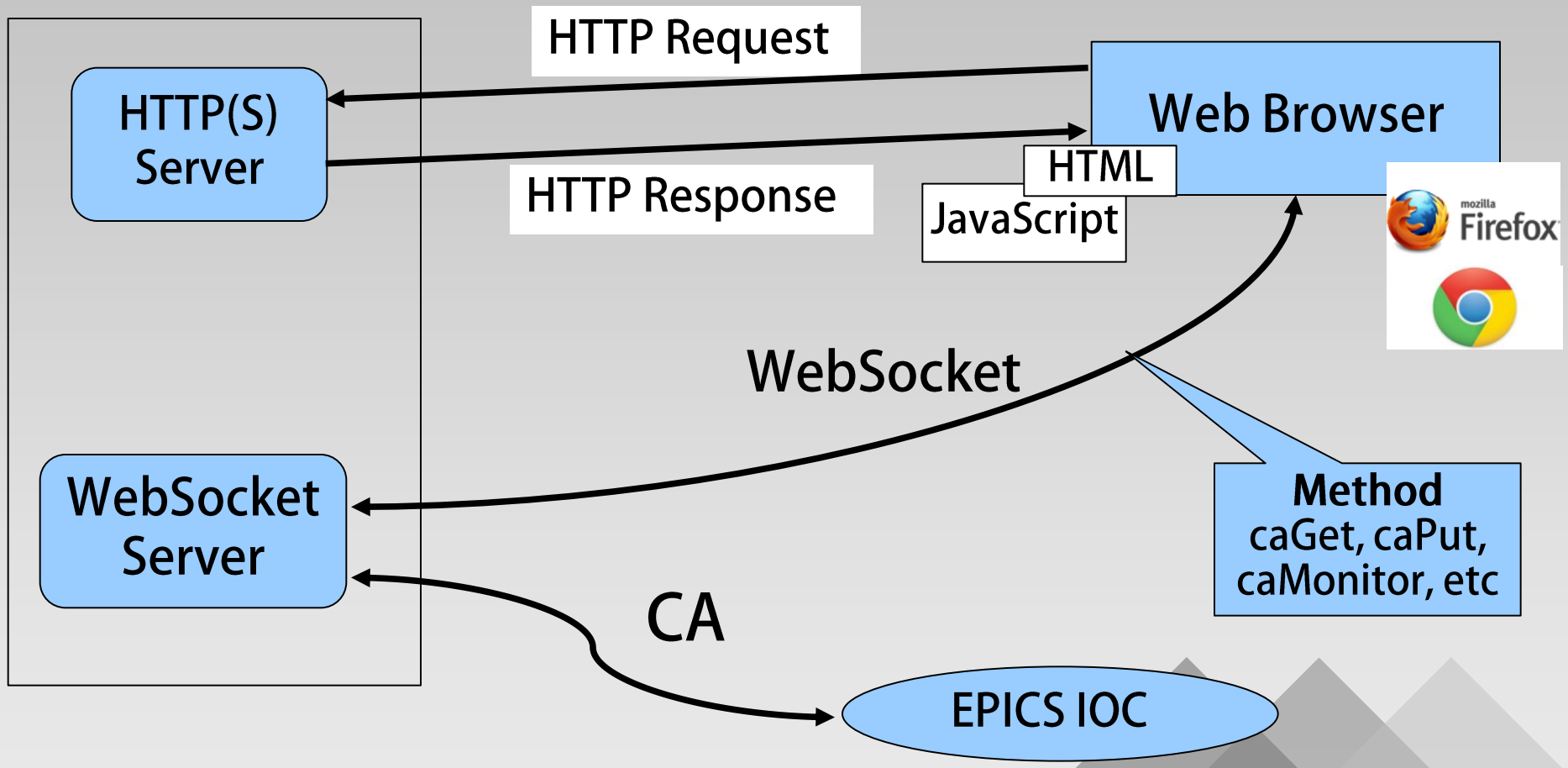
システム概要

OPIとWebSocketサーバが同列



WebSocket概要

・本開発ではWebSocketサーバとWebサーバを一緒に提供（同じポート）



A. UCHIYAMA

クライアントサイド実装(JavaScript)

```
/* 送信 */
var socket =io.connect('/ca/gauge');
socket.emit('caMonitor',{
        pv: 'akito12Host:ai1',
        data_name: 'ai'
})

/* 受信 */
socket.on('ca', function(data){
     var value = data.ai;
});
```

EPICSレコード

データ名を指定

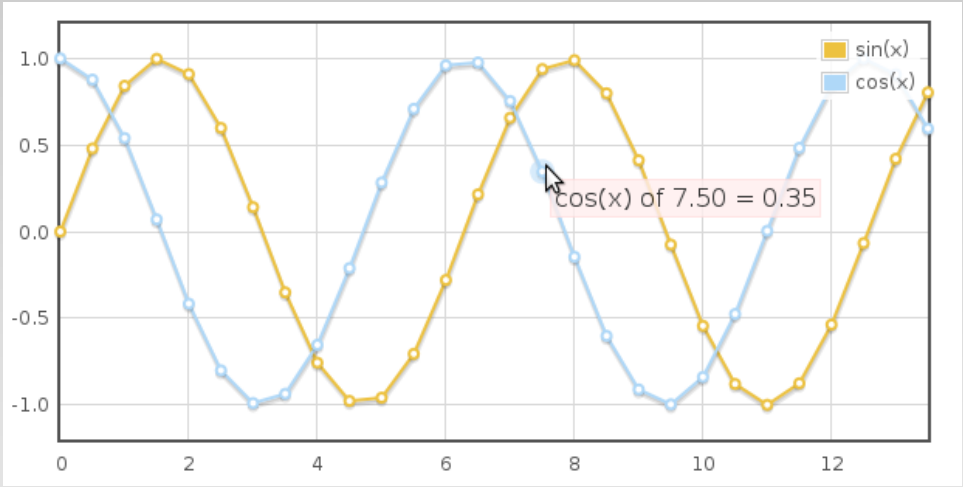
データ受信

クライアントサイド実装(可視化)

リアルタイムなデータを可視化するためJavaScriptライブラリを利用

- jsgauge
- flot

(jQueryベース, MITライセンス)



見栄えのするチャートをゼロから作るコストは大きい。
他の有料なチャートライブラリの使用も。

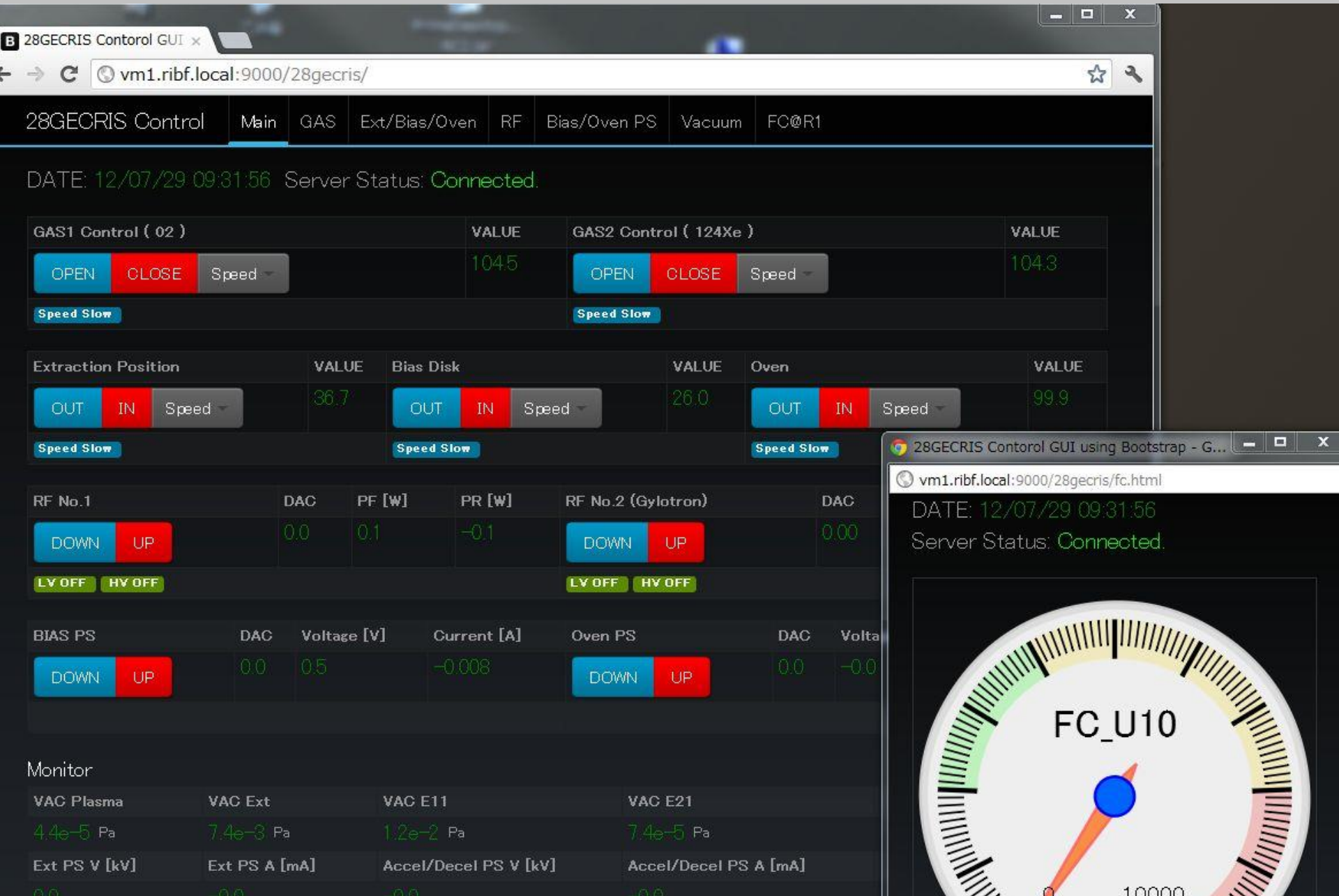
OPI用GUI開発 ≒ Web(HTML)開発

難易度は大幅に下がる

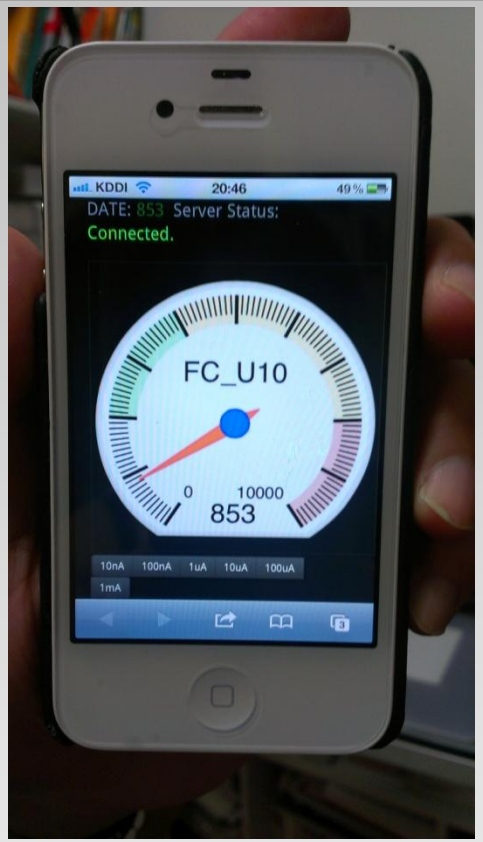
A. UCHIYAMA

クライアントサイド実装デモ

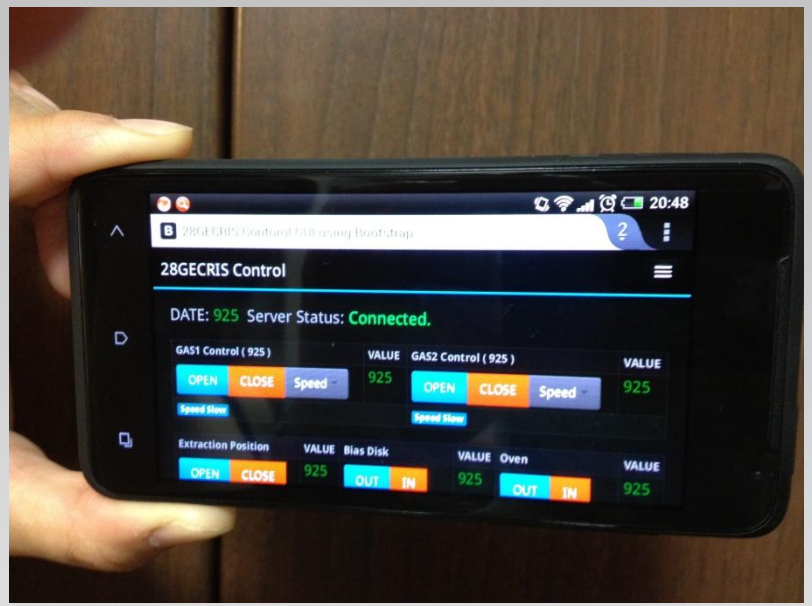
RIBF 28GHz SC-ECRイオン源制御の例



クライアントサイド実装(モバイル)



Safari on iPhone4S



Google Chrome on Android4.0

・SafariにはiPhone/iPod Touch向けにタッチイベントを取得する関数がある

マルチタッチアプリケーションの開発も可能

Objective-C＋cocoa Touchより難易度は大幅に下がる

まとめ

- ✓ 新しいHTMLの規格であるWebSocketからEPICS Channel Accessを行う事が可能になった
- ✓ 実現する為に
  - Node.jsからCAを利用するためにNodeCAを開発した
  - クライアントサイドからのCA要求に対応するWebSocketサーバを開発した
- ✓ ブラウザから制御を行い従来のアプリケーションと同程度のインタラクティブ性を実現した
- ✓ PCでなくiPhone4SやAndroidのブラウザからもWebSocketで制御できる事を確認した

ご清聴ありがとうございました