Flexible IoT Solutions using Zephyr and JavaScript

Brian Jones
Software Engineer
Intel Open Source Technology Center
Do more with Less!
Obviously there are limits…

RAM and ROM are limited, so the device’s purpose has to be focused.

Devices need to be reflashed to change the code on them.

Different drivers for different boards makes code reuse difficult.
What if we want to be more flexible?

Multiple programs or modes?

Modules?

Update your code on the fly?

Reusable code?
Enter JavaScript Runtime for Zephyr OS

https://www.zephyrproject.org
http://jerryscript.net
https://github.com/intel/zephyr.js
How does ZJS work?

Zephyr supplies the native API

JerryScript is our JavaScript engine

ZJS exposes JavaScript APIs and the ability to run JS files at will
// Copyright (c) 2016-2017, Intel Corporation.
// Test code to use the gyroscope (subclass of Generic Sensor) API.
// to communicate with the IMU108 inertia sensor on the Arduino 108.
// and monitor the rate of rotation around the x, y, and z axis.

var sensor = new Gyroscope({
  frequency: updateFrequency
});

Sensor.onchange = function() {
  console.log('Rotation (rad/s): ' +
    'x: ' + sensor.x +
    'y: ' + sensor.y +
    'z: ' + sensor.z);
};

sensor.onactivate = function() {
  console.log('Activated!');
};

sensor.onerror = function(event) {
  console.log('Error: ' + event.error.name +
    ' - ' + event.error.message);
};

sensor.start();

Ready.
JavaScript benefits

- Easy and quick to write.
- Code can be modularized.
- Reuse parts for other projects.
- Can be run at any time without restarting the board.
- Code can be edited on the fly.
- Use it on any ZJS supported board.
… and hurdles

• JavaScript parser takes up space.
• Speed is not as predictable.
• Security can be tricky.
So where do we put all our JS code?
Example code

//app1.js
var aio = require("aio");
var gpio = require("gpio");
var spi = require("spi");

console.log("Starting first program..");
… // Do some things
… // Event occurs

// Stops the current JS and runs
// the code in app2.js
runJS("app2.js");

// app2.js
var i2c = require("i2c");

… // Do some more stuff
… // Finished, return to app1

runJS("app1.js");
//main.js
var aio = require("aio");
var gpio = require("gpio");
var spi = require("spi");
var app1 = require("app1.js");
var app2 = require("app2.js");
var app3 = require("app3.js");
var app4 = require("app4.js");
console.log("Starting main program..");
app1.init();
app2.init();
app3.init();
app4.init();
console.log(app1.checkTemp());
....

// app1.js
var app1API = {};

app1API.init = function () {
    // init the module
}

app1API.checkTemp = function () {
    // return the temp
}

module.exports = app1API;
Some possibilities...
Single JS files

- Production
- Debug
- Beta
- Minimal

Main Menu → Data record → Chart → Map
Modulated code that only uses what you need.

Full suite

Power save
A combination of both
What could you make?

HUD with various display modes.

A guitar pedal that changes its modulation mode on tap.

Home brewing buddy with all your recipes ready to go.

Garden helper tool that learns from data and gets updated remotely to work better.

Load program based on the hardware it’s plugged into using sensor.
Thanks for joining me!