Working Without Wires
Simulating hardware components for IoT app development

Jared Kaplan  jkaplan@marvell.com
Sean Allen  sean@kinoma.com

@eclipsecon  March 9, 2016
Overview

1. What challenges did we face building products for IoT?
2. Why was simulation important for us?
3. How did we implement hardware simulation and other device specific functionality in Kinoma
What were the challenges?
We started with mobile…

- Many devices with different hardware interfaces and features
- Needed to be able to test various screen dimensions
- Simulator allowed us to test device specific functionality on the desktop
Created Kinoma platform

• Kinoma Porting Layer
• KinomaJS framework
• JavaScript apps
• Device simulators
We began prototyping new consumer electronic devices...

- Acquired by Marvell in 2011
- Starting building Linux based devices and products of our own
- Needed to interface with hardware at a lower level - GPIO, WiFi, Bluetooth, I2C, etc.
Realized an IoT prototyping platform would be useful

- Built a device for ourselves - allowed us to easily connect sensors and other hardware, and rapidly prototype new ideas
- Added support in KinomaJS to work with standard hardware interfaces
- Others became interested
So we developed...

- Kinoma prototyping devices using JavaScript for anyone
- Easy to connect and configure sensors and other hardware

Working Without Wires
A whole world of breakout boards and devices to hook up!
Why is Simulation Important?
Don’t try this at home!
Simulation helps when...

- You don’t want to have to debug hardware and wiring issues in the beginning while you are prototyping the software
- Hardware decisions haven’t been made, or you want to evaluate software libraries before committing to buy specific hardware
- You want to be able to work on the software when the hardware isn’t available
Also import because...

- Enables faster development cycles - time to edit, run, debug is better
- Work can be distributed more easily among team members
- Less costly
- Safer!
Temperature, barometric, and GPS sensors for real time data collection
What is our approach?
Using hardware with Kinoma

• Application running on the device interfaces with temperature sensor using a JavaScript module

Application

getTemp()
const pins = {
  i2c: { type: "I2C", address: 0x48 }
};

function configure( configuration ) {
  this.i2c.init();
}

function close() {
  this.i2c.close();
}

function getTemp() {
  var data = this.i2c.readWordDataSMB( 0 );
  var value = ((data & 0xFF) << 4) |
               ((data >> 8) >> 4);
  if (value & 0x800) {
    value -= 1;
    value = ~value & 0xFFF;
    value = -value;
  }
  return value * 0.0625;
}

export default { pins, configure, close, getTemp };
Adding a temperature simulator

- Simulator must export the same interface as device module
- Module path determined by runtime target
- No changes to application code!
const pins = {
  i2c: {
    type: "I2C", address: 0x48
  }
};

function configure( configuration ) {}  
function close() {}  
function getTemp() {
  return Math.random() * 120;
}

export default { pins, configure, close, getTemp };
Now with inspectors!

- Modules can add inspectors to the simulator UI
- Inspectors can simulate input data using controls or generated waveforms
- Inspectors visualize output using built-in level meters or custom interface
function configure( configuration ) {
    this.simulator = shell.delegate("addSimulatorPart", {
        header: {
            label: "Temperature Sensor",
            name: "TMP102",
            iconVariant: PinsSimulators.SENSOR_MODULE
        },
        axes:[new PinsSimulators.AnalogInputAxisDescription({
            valueLabel: "Celsius",
            valueID: "temperatureValue",
            defaultControl: PinsSimulators.SLIDER,
            minValue: -30, maxValue: 100, value: 0
        }) ]
    });
}

function close() {
    shell.delegate("removeSimulatorPart", this.simulator);
}

function getTemp() {
    return this.simulator.delegate("getValue")
    .temperatureValue;
}
Data driven inspectors

Digital Output
Pin 44
Value 1

Digital Input
Pin 62
Value 1

Analog Input
Pin 63
Value 0.9755

PWM
Pin 28
Value 0.4166

Serial
Tx 31, Rx 33

Transmit

Received
Custom inspectors

Tri Color LED
PWM LED

S7S
7-Segment Display
23:46

Icd
LCD
01234567
HELLO

Accelerometer
Simulated Shake

Shake To Clear

Working Without Wires
@kinoma @eclipsecon
Try it yourself!

- Download Kinoma Studio
- Install examples from IDE
- Or directly from GitHub: github.com/Kinoma
Thank You!

Jared Kaplan
jkaplan@marvell.com

Sean Allen
sean@kinoma.com

www.kinoma.com
github.com/Kinoma