Research Abstract

Ilisha Ramachandran

Bryn Mawr College

12th June 2016

Project Title: Developing a distributed embedded computing application development framework for educational and research purposes

Background Statement

More and more researchers and educators are utilizing relatively low cost Development Boards[[1]](#footnote-1) such as the Raspberry Pi[[2]](#footnote-2), and the Intel Edison[[3]](#footnote-3), to accomplish a variety of tasks, including running scientific experiments , environmental sensing, remote monitoring and teaching students how to program.

However ordinarily, a setup, such as a sensor network to track environmental parameters in a rainforest, or a light sensor experiment in a student teaching lab, is limited by the following:

* The setups generally tend to be “closed” environments”; allowing only a single researcher or student access to the board and usually only allowing one application to read and control sensors and actuators that are linked to the development boards.
* The setups generally support only specific languages or IDEs[[4]](#footnote-4) that researchers and students may not necessarily be familiar with.
* Each setup may require complicated software or code to control sensors and actuators connected to the Development Boards. Or require technicians with specific skills to calibrate the sensors, making duplicating the setup challenging.
* Each Development Board or setup used requires the student or researcher to know how to setup the board, configure it, and load the specific software before starting to use the experiment or setup. This can take hours if not days.

Project goal

The goal of this project is to:

* Develop a system that allows researchers and students remotely access and develop local applications to control a shared development board.
* Develop Web APIs that can be used to read sensors and control actuators on any Linux-based development board.
* Utilize high level languages such as Python and Javascript to read sensors and control actuators on a development board.

The proposed Architecture is described in the Figure 1 below:



Fig 1 – Proposed Architecture

Narrative of Proposed System

Figure 1 describes the proposed solution. The following narrative provides an idea of how all the elements come together. Researchers will utilize a cloud based IDE, such as Jupyter (used by Bryn Mawr College to teach students how to program in Python). Researchers would write their applications using Python, and utilize APIs to control two categories of devices on the Development Board, Sensors and Actuators. Sensors are generally devise that can sense/read environmental parameters such as temperature and humidity. Actuators are devices that create specific actions on the board, such as activating a Buzzer or a relay (to turn on a motor for example).

The Development Board for this Project will utilize an Intel Edison Board which runs Yocto Linux[[5]](#footnote-5). The software framework that will be used to implement the “Web APIs or Web Services” is Node.JS[[6]](#footnote-6).

Node.js is based on Google’s V8 Javascript engine. V8 is open source and is used in Chrome and some of Google web APIs. It’s fast and supports an event driven architecture that allows it to be used effectively in embedded systems.

As an example, say Researcher 1 wishes to read the temperature being recorded by the temperature sensor on the Edison Board. These are the steps that will be followed:

1. Using a Browser, such as Chrome, the Researcher types the following command on the address window of the Chrome Browser:
   * <http://192.168.1.2/sensor/readTemperature>
2. This command is sent to the Intel Edison (which for this exercise uses the IP Address 192.168.1.2)
3. The URI[[7]](#footnote-7) “/sensors/ReadTemperature” implements the Javascript code which reads the Temperature sensor value and returns it to Researcher 1. He will see the reading, ie :”24 degrees Celsius”, displayed on his browser.

Each sensor and each actuator would have its own URI. In this way we provide an “abstraction layer”, meaning that the Researcher doesn’t need to understand how to program the Intel Edison to obtain a temperature or any other reading.

In the same way, the same command can be used by Researcher 2 and Researcher 3.. in fact as many[[8]](#footnote-8) researchers as the Node.JS engine on the Intel Edison can handle.

Primary Project Success Criteria

We will measure the success of the project based on the following criteria:

1. Demonstrate that Node.js can be used to implement the basic information flow outlined in Figure 1.
2. Implement the following URI/Web APIs on the Intel Edison, using Node.JS and Express[[9]](#footnote-9):

* [readTemperature](http://www.seeedstudio.com/wiki/Grove_-_Tempture%26Humidity_Sensor_(High-Accuracy_%26Mini)_v1.0)
* readHumidity
* [readLightLevel](http://www.seeedstudio.com/wiki/Grove_-_Light_Sensor)
* readUVLevel
* readPIRMotion
* readEncoder
* readButtonLevel
* setLCDMessage
* setRelayOn, setRelayOff
* setBuzzerOn, setBuzzerOff

1. Use Chrome to test the URI/Web APIs on the Intel Edison. Each api defined above will be tested.
2. Use multiple instances of Chrome on a PC, to simulate multiple researchers using the same resource (ie, using the readTemperature API from multiple instances of Chrome, on the same PC).
3. Use Chrome on a Mobile Phone, to test if the APIs work. This will prove that the solution is ‘platform agnostic”, meaning any device that supports a browser can be used to access the resources on the Intel Edison.
4. Write Python “modules” that implement the APIs, to make it easy for developers (using the Jupyter system) to use the APIs to access the sensors and actuators on the Intel Edison.
5. Write sample code on Jupyter to test each of the APIs.

Secondary Project Success Criteria

The following goals are provided as secondary goals, but will only be implemented if this researcher has sufficient time. If not, this will serve as the basis for future research projects.

1. Develop a framework to test/simulate how many simultaneous Researchers can access the Intel Edison/Node.js framework without losing data, and within a reasonable latency[[10]](#footnote-10) period.
2. Learn how to use Load Stress test tools, such as Apache Jmeter[[11]](#footnote-11).
3. Come up with recommendations as to how many researchers can access an Intel Edison, running Node.JS and the Express framework.

Primary Project Deliverables

1. Installation and setup instructions for the Intel Edison, Node.js, Express and the Grove Indoor Environment Sensing kit.
2. A Test Jig to simulate an environmental sensing scenario, including a water flow sensor, temperature and humidity sensors.
3. JavaScript source code implementing the Web APIs outlined in the previous section (Intel Edison), and the JavaScript code to control the various sensors.
4. Python Script/modules implementing the Web APIs.
5. Sample Python code (implemented as a Jupyter Project) demonstrating how to use each of the APIs.
6. Project Report outlining all relevant aspects of the project.
7. Upload project to Instructables.com
8. Provide all source code via a github account

1. A Development Board, in the context of this Abstract, refers to a board that contains a CPU, Sensor and Actuators [↑](#footnote-ref-1)
2. Raspberry PI is a Linux-based Development board developed by Cambridge University. See <https://www.raspberrypi.org/> [↑](#footnote-ref-2)
3. The Intel Edison is a System on a Module (SOM) that integrates a full “Pentium grade” processor with Bluetooth and WiFi, running on Yocto Linux [↑](#footnote-ref-3)
4. IDE’s, or Integrated Development Environments, refer to software tools that allow developers to write, and debug software and also control hardware development boards. [↑](#footnote-ref-4)
5. Yocto Linux or the Yocto Project is a framework that allows Embedded Linux flavors to be rapidly configured and deployed on specific development boards. The Intel Edison uses the Yocto to manage the Linux implementation on the Edison. See <https://en.wikipedia.org/wiki/Yocto_Project>. [↑](#footnote-ref-5)
6. Node.js is a runtime environment that allows for the development of Server side Web Applications. Node.JS is based on Google’s V8 Javascript engine which is used in by Chrome and also Google’s Web APIs/services [↑](#footnote-ref-6)
7. See the following for details on what a URI is <https://en.wikipedia.org/wiki/Uniform_Resource_Identifier> [↑](#footnote-ref-7)
8. One of the objective of this project will be to try and simulate multiple researchers/users of the same resources. How many users can access the system simultaneously? [↑](#footnote-ref-8)
9. <http://expressjs.com/> - Express is a minimalist webframework for node.js that allows developers to build a simple framework for developing web applications. We will use this framework to build our WebAPIs on Intel Edison. [↑](#footnote-ref-9)
10. Latency refers to the time from when an API is called by a Researcher, to the time that Intel Edison returns a result. [↑](#footnote-ref-10)
11. Apache J Meter is a tool that is used to “stress test” Web Applications and APIs. We will use this tool to simulate multiple users/access to the Intel Edison and see when it breaks! See <https://en.wikipedia.org/wiki/Apache_JMeter> [↑](#footnote-ref-11)