This document is licensed under a Creative Commons Attribution 4.0 License.

OICSensorBoard#

- Server application for Edison which demonstrates Iotivity server capabilities through the integration of an add-on breadboard that hosts temperature, ambient light and LED resources.
- Client application to test server functionality, discovering and communicating with these resources.

Connecting Sensors#

For this application, we use an Edison Arduino breakout board. In addition, it requires the following components from SunFounder Project Universal Starter Kit for Arduino UNO…, which can be purchased from Amazon

- temperature sensor
- ambient light sensor
- jumper wires
- breadboard
- 10K resistors

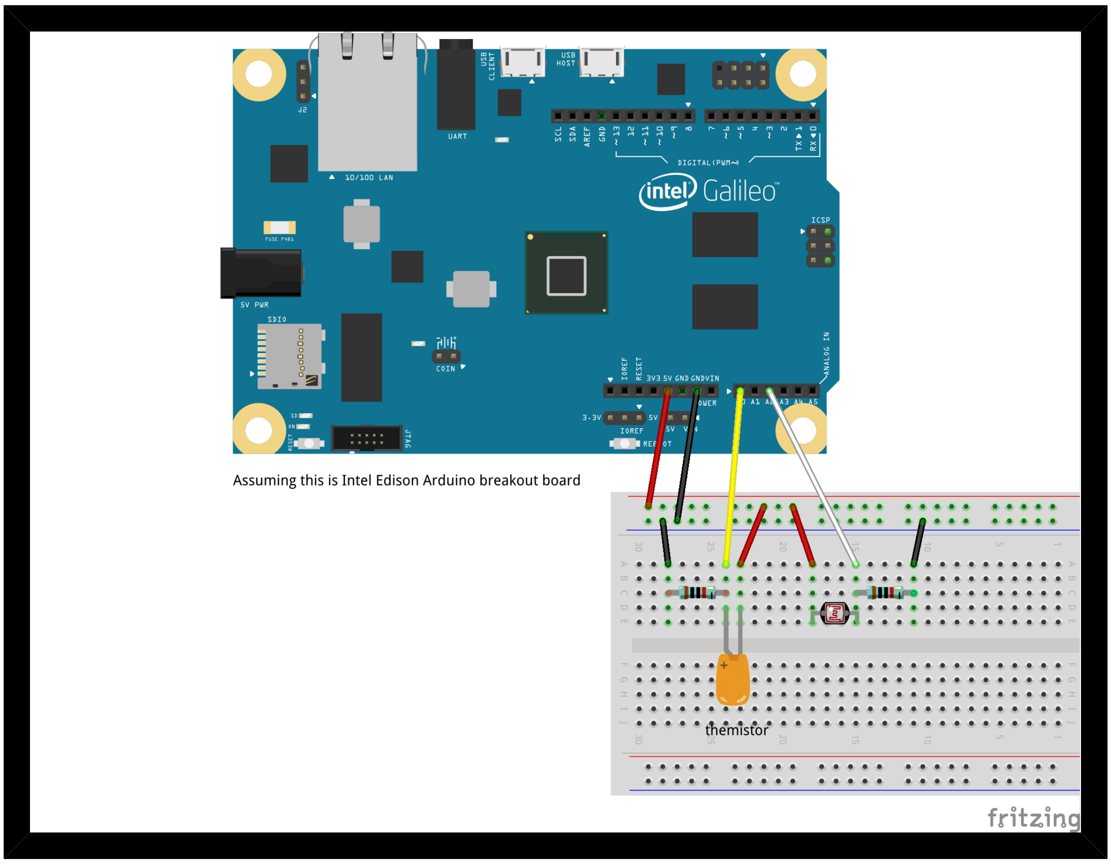
Edison Analog Inputs

Similar to Galileo, the Edison Arduino breakout board has six analog input pins, labeled A0 to A5. The analog inputs, via an analog to digital converter (ADC) provides 12 bits resolution (value range 0 to 4096). The ADC measures from 0 to +5 volts.

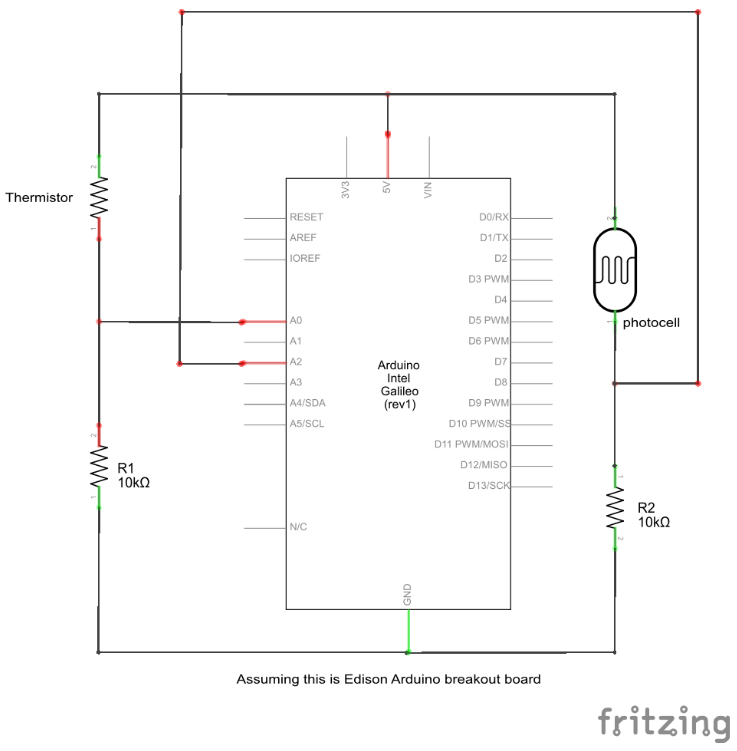
OICSensorBoard uses an open-source library called libmraa, which is a low-level communication library for Intel® platforms. The libmraa library provides C/C++ wrappers for configuring the dev kit board pins. It is part of Edison BSP.

Connecting the Components

We connect the temperature sensor to A0 and the light sensor to A2 as shown in following diagram.



Schematic



Build System#

OICSensorBoard uses scons as its build framework. You must therefore fetch and install scons on your build machine before proceeding.

Building the server for Edison#

1. Follow the IoTivity guide on Yocto Support to build the IoTivity stack with the Edison BSP. As a result, IoTivity runtimes would be installed into the Edison OS image.
2. Build the Yocto toolchain for Edison with the following configuration:
   -- Update conf/local.conf with IMAGE_INSTALL_append = " iotivity-dev mraa".
   -- Build toolchain (bitbake -c populate_sdk edison-image), and install toolchain on your build machine.
3. Initialize the Yocto toolchain's environment setup script by soucring it into your shell.
4. Run "make server". This builds the server executable for Edison.
5. Transfer "server" executable to the Edison using scp or a micro-SD card.

Building the client application#

1. Modify the CLIENTARCH and BUILDTYPE variables if required to match the set up on your client target.
2. Run "make client". This builds the client executable.

Running the server on Edison#

1. Open up a shell on the Edison via either USB-UART port or ssh.
2. Ensure that the Edison has a network connection using either Wifi or Ethernet.
3. Navigate to the directory containing the **server** executable and run it.

```
# ./server
Press Ctrl-C to quit....
Running IoTServer constructor
Successfully created Room.Temperature resource
Successfully created Ambient.Light resource
Successfully created Platform.Led resource
```

Running the client application#

Run **client** on your client machine.

```
Running IoTClient constructor
Performing Discovery...

Found Resource
Resource Types:
   Room.Temperature
Resource Interfaces:
   core.edison.resources
Resource uri: /temperature
host: coap://192.168.1.6:59120

Found Resource
Resource Types:
   Ambient.Light
Resource Interfaces:
   core.edison.resources
Resource uri: /ambientlight
host: coap://192.168.1.6:59120

Found Resource
Resource Types:
   Platform.Led
Resource Interfaces:
   core.edison.resources
Resource uri: /led
host: coap://192.168.1.6:59120

Enter:
0) Display this menu
1) Get temperature Reading
2) Start Temperature Observer
3) Stop Temperature Observer
4) Get ambient light reading
5) Start Ambient Light Observer
6) Stop Ambient Light Observer
7) Turn LED ON
8) Turn LED OFF
9) Quit
```

We can now control on-board LED and query or observe the temperature and light sensor readings. The temperature value is in Celsius, and light sensor returns raw ADC value (between 0 and 4096.)

Supported Resources and Methods#

Interface: “core.edison.resources”

Resources

Resource Uri: /led

Resource type: “Platform.Led”

Method: PUT, Key:”switch”, value type: integer in [0,1] (0 = OFF, 1 = ON)

Method: GET, Key:”switch”, value type : integer in [0,1]

Resource Uri: /temperature

Resource type: “Room.Temperature”

METHOD:GET, Key:”temperature”, value type: double giving value in celsius.

METHOD:OBSERVE notifications every 1.5s.

Resource Uri: /ambientlight

Resource type: “Ambient.Light”

METHOD:GET, Key:”ambientlight”, value type: integer giving “light level”

METHOD:OBSERVE notifications every 1.5s.
