IoT Platform Tutorial 5:
Energy-Aware System

In Hwan “Chris” Baek
chris.inhwan.baek@gmail.com

Kuan Chen
mikeck1989@ucla.edu

William Kaiser
kaiser@ee.ucla.edu

Introduction

In 2014, Intel introduced its Edison board to enhance the development of IoT platform. Admittedly, the new x86 board is much more than powerful enough as an embedded system. Developers, customers and users can build countless applications with this board, but its power dissipation is also more severe than less powerful boards. In spite of being designed for low power consumption, Intel Atom “Tangier” on the Edison is still considerably more power-hungry than many low power microcontrollers such as ATmega328, the chip on Arduino Uno.

In several applications (like environmental monitoring) electronic devices can work periodically, instead of working all the time. Base on this situation, we created a power-saving platform, which can reduce the power consumption and longer the power life intensively. Furthermore, the platform will not affect the internal work of IoT device itself. Developers still can create millions different applications on this new platform.

In this tutorial, you will learn:
1. About the background principles of our energy-aware system,
2. To set up the energy-aware system,
3. About the code, and
4. To modify the system to fit your own need.

Things Needed

1. An Intel Edison with Arduino-compatible breakout,
2. a micro USB cable,
3. a A-to-B USB cable,
4. a Grove Starter kit,
5. an Arduino Uno,
6. a power adapter,
7. a power extender/relay, and
8. a PC or Mac

Background Principles

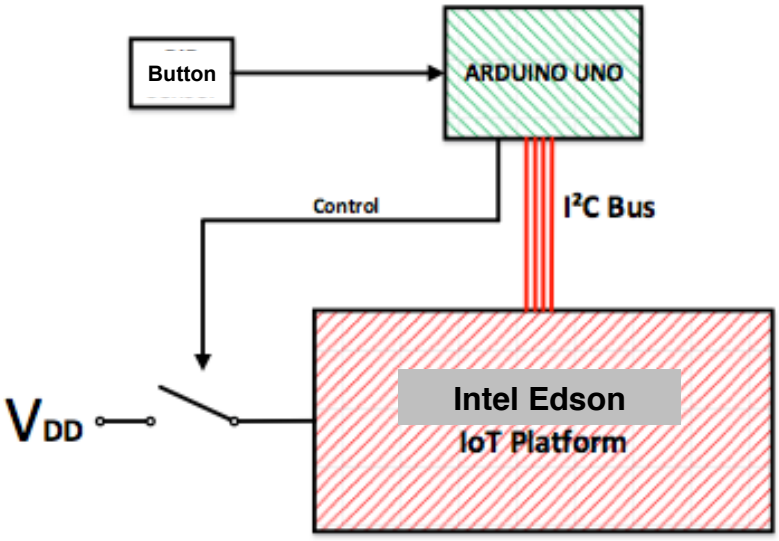
Although the name “Energy-aware system” may mislead, the system is designed to save power. In many applications, the IoT platform is idle most of time, wasting power for not performing any useful task. Even when the system is idle, it needs to maintain the current state. Furthermore, any IC suffers from leakage power, which is the power consumed by unintended leakage that does not contribute to the IC’s function [1]. In order to prevent such waste of power, system needs to shut down when it does not have any tasks to perform.

However, power consumption of a system is not completely prevented by merely shutdown. Standby power or vampire power is another concern. Standby power is the power consumed by devices when they are plugged into power but switched off. It is necessary to completely remove a device from its power supply to prevent standby power consumption.

Based on these principles, we have designed the “Energy-aware system”. The fundamental idea of the power-saving platform is this: A low-power Arduino Uno board is used to schedule the work of the main system, the Intel Edison board. The Arduino Uno wakes up the Edison, let it finish its work and then removes its power supply completely. There are two types of wake-ups. The first type, namely “regular wake-up”, is that the Arduino Uno wakes up the Edison based on working cycles. In other words, every time the Edison shuts down, the Arduino calculates what time it should wake up and starts a timer. It works like an alarm clock. Instead of sounding the alarm, it connect the power supply to the Edison to boot it up. The second type, namely “alert wake-up, is that the Arduino immediately wakes up the Edison when an event occurs. In this tutorial, a button is used to demonstrate the alert wake-up as the system considers motion detection as an event.

System Workflow

The workflow of the energy-aware system is given by the diagrams below:



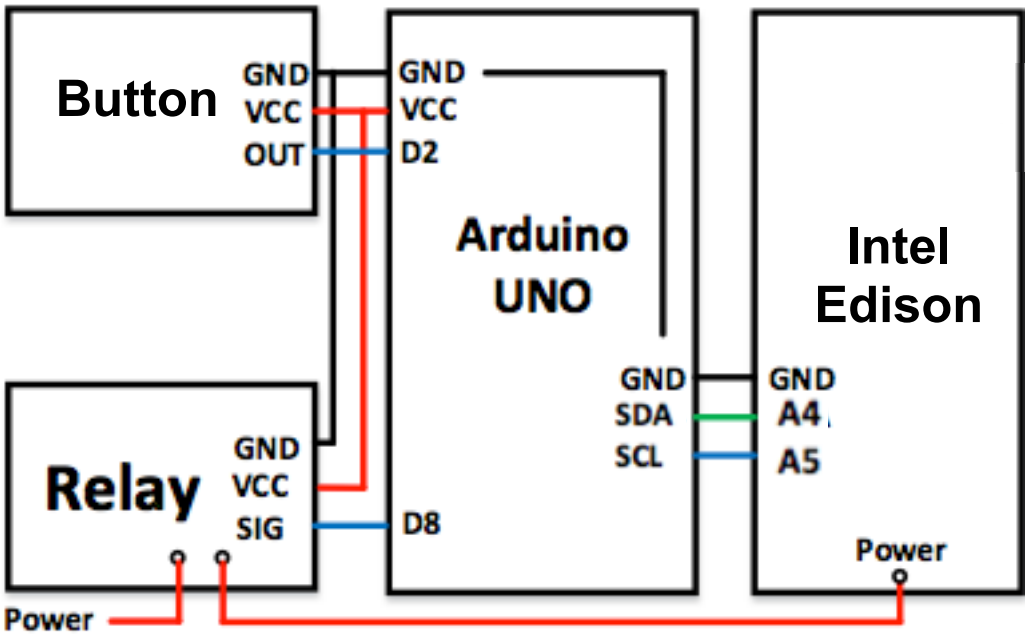
Edison communicates a future wake-up time to Arduino via I2C.
↓
Edison shuts down by executing a Linux command.
↓
Arduino removes V_{DD} power via relay from Edison.
↓
Arduino re-close V_{DD} relay at wake-up time or when PIR event occurs.
↓
Edison boots up when power is re-connected.
↓
Edison requests the wake-up type after completing the boot-up
↓
Edison does its work corresponding to the wake-up type (Alert or Regular)

The following YouTube video is a demo of the system. This demo is made with another IoT platform, Intel Galileo Gen2 and a PIR sensor to detect motion. The system workflow is identical to what is presented in this tutorial. The SMS message alert is an extra feature added for the demonstration and is not included.

- https://www.youtube.com/watch?v=k3CHlAJGxlg

System Setup

Now, we will assemble the hardware and observe how the energy-aware system works. The following is the overall wiring diagram.



To set up the energy-aware system, follow the steps below:

1. Check the list of components.
   - Intel Edison
   - Arduino Uno
   - Button from Grove Starter kit
   - Power extender/relay
   - Power supply for Intel Edison
   - USB cable for Arduino Uno
   - Breadboard
   - Cables and wires
2. Connect the power adapter to the Edison.
3. SSH into Edison.
4. **$ mkdir tutorial5_energyAware**
5. **$ cd tutorial5_energyAware**
6. Get the source files using git.
7. **$ git clone https://chrisIHbaek_reader:ucla_whi@bitbucket.org/chrisIHbaek/energyaware.git**
8. Copy the contents to the current directory. Enter "**cp ./energyaware/* .**"
9. There are 6 files. To see the general description of these files, enter "**cat README.txt**".
10. The description in README.txt is for Intel Galileo but this system is compatible with Intel Edison as well.
11. Transfer files to your computer. You can use sftp client software such as cyberduck. (Mac/Linux users may refer to **Appendix** of this tutorial to use sftp directly on a terminal window).
12. Once you have transferred all the files, compile energy_aware.c.

$ gcc -lmraa -o energy_aware energy_aware.c

13. "**energy_aware**" is the program to run on the Edison automatically on boot.
14. We can set it up with energyAwareSevice.sh, but we need to edit the code.
15. **$ vi energyAwareService.sh**
16. Edit "directory that contains energy_aware" in line 13.
    (e.g. /home/root/tutorial4_energyAware)

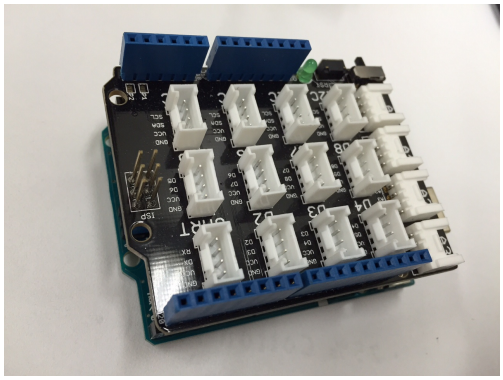
```
dir="directory that contains energy_aware"
user="root"
cmd="./energy_aware"

name="energyAwareService"
```

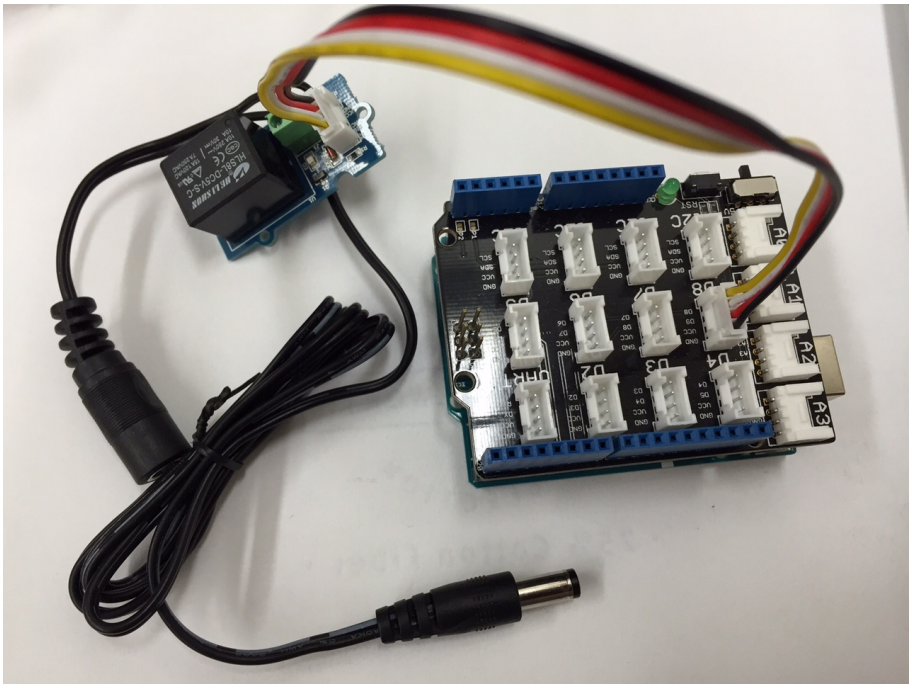
17. **$ cp energyAwareService.sh /etc/init.d/**
18. **$ cd /etc/init.d/**
19. **$ chmod +x energyAwareService.sh**
20. **$ update-rc.d energyAwareService.sh defaults**
21. Verify that references are created. Enter "**ls -l /etc/rc?.d/*energyAwareService.sh**".

```
root@edison:/etc/init.d# ls -l /etc/rc?.d/*energyAwareService.sh
lrwxrwxrwx    1 root     root            31 Jun  1 03:26 /etc/rc0.d/K20energyAwareService.sh -> ../init.d/energyAwareService.sh
lrwxrwxrwx    1 root     root            31 Jun  1 03:26 /etc/rc1.d/K20energyAwareService.sh -> ../init.d/energyAwareService.sh
lrwxrwxrwx    1 root     root            31 Jun  1 03:26 /etc/rc2.d/S20energyAwareService.sh -> ../init.d/energyAwareService.sh
lrwxrwxrwx    1 root     root            31 Jun  1 03:26 /etc/rc3.d/S20energyAwareService.sh -> ../init.d/energyAwareService.sh
lrwxrwxrwx    1 root     root            31 Jun  1 03:26 /etc/rc4.d/S20energyAwareService.sh -> ../init.d/energyAwareService.sh
lrwxrwxrwx    1 root     root            31 Jun  1 03:26 /etc/rc5.d/S20energyAwareService.sh -> ../init.d/energyAwareService.sh
lrwxrwxrwx    1 root     root            31 Jun  1 03:26 /etc/rc6.d/K20energyAwareService.sh -> ../init.d/energyAwareService.sh
```

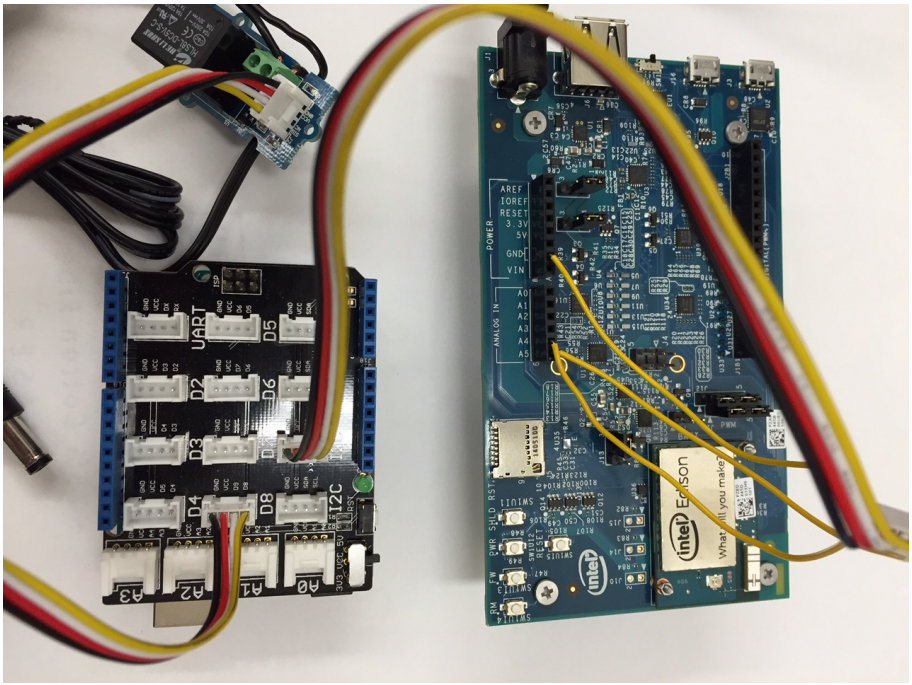
22. The setup on the Edison is complete! You can learn more about how to start a program automatically on boot in Linux at http://www.tldp.org/HOWTO/HighQuality-Apps-HOWTO/boot.html.
23. Turn off the Edison and remove all cable/adapter.
24. Upload "**arduino_example.ino**" to the Arduino Uno using Arduino IDE.
25. Once the upload is complete, unplug the USB cable to turn off the Arduino Uno.
26. Now, we need to assemble the hardware components.
27. Attach the Grove Base shield to the Arduino Uno.



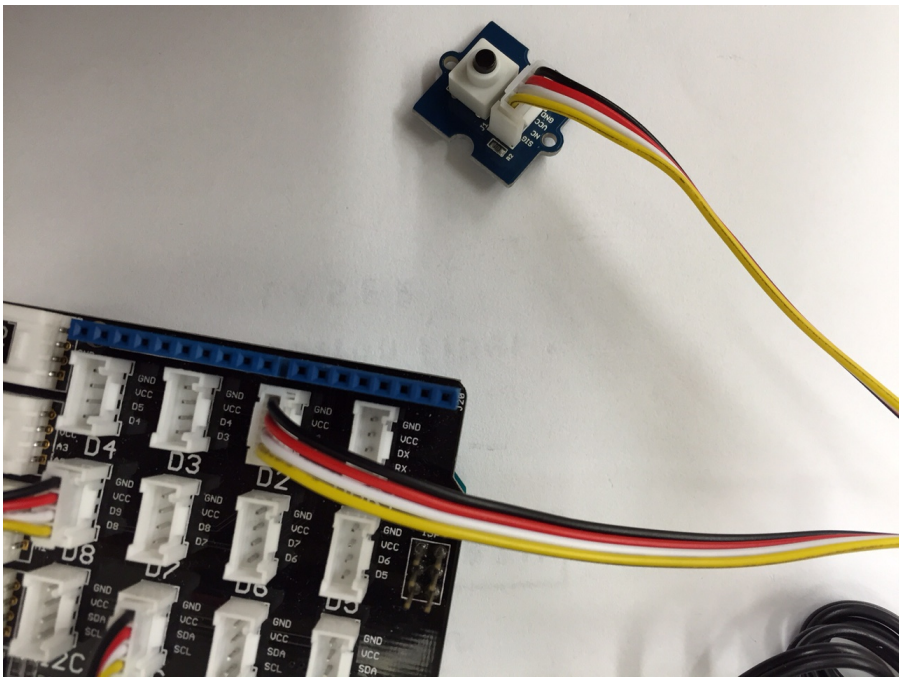
28. Using a cable, connect the power extender/relay to D8 of the Base shield as shown in the picture below.



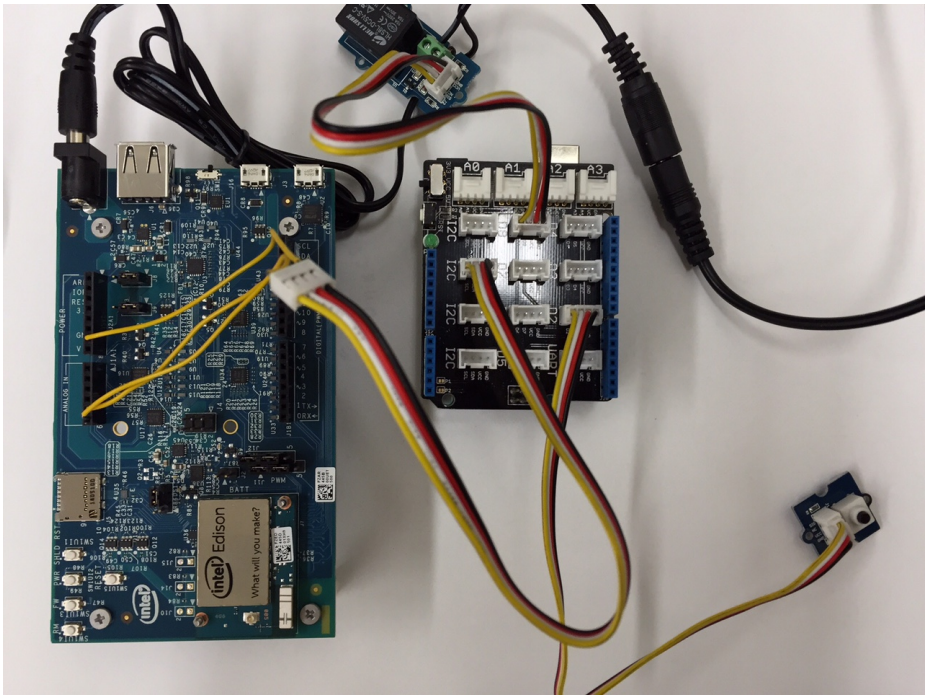
29. Connect wires between the Edison and the Arduino Uno to establish I2C.
    - Arduino Uno's GDN - Edison's GND
    - Arduino Uno's SDA (white) - Edison's A4
    - Arduino Uno's SCL (yellow) - Edison's A5



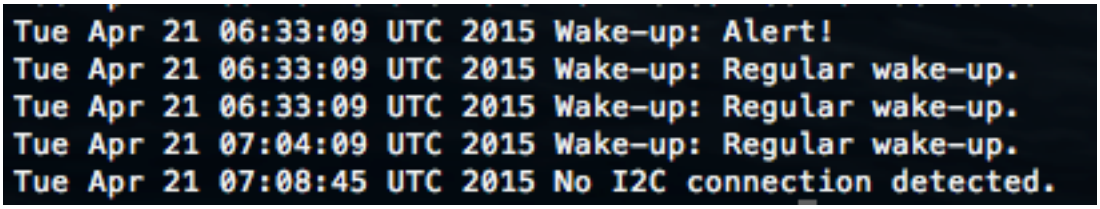
30. Connect a button to D2 of the Base shield as shown in the picture below.



31. Plug the power extender into the Edison and the power supply (plugged into a power outlet) into the extender.
32. You should have a system that looks like the following.



33. Now connect the USB cable to the Arduino Uno to power it on.
34. The energy-aware system should start working.
35. Press the button to simulate an alert wake-up.
36. After the Edison turns off, wait for enough time until it turns on again to simulate a regular wake-up.
37. Repeat steps 18 and 19 few more times.
38. When the Edison is off, unplug the USB cable from the Arduino Uno to turn off the energy-aware system.
39. Detach the power extender/relay, then plug the power supply into the Edison.
40. SSH into the Edison to inspect the simulation results.
41. Navigate to the directory that contains “energy_aware”. (e.g. ~/tutorial4_energyAware)
42. Now, you should have “log.txt”.
43. Enter “**cat log.txt**” to see the log messages.



44. As you can see, the messages show the timestamp and either wake-up type or an error message. In the screenshot above, “No I2C connection detected.” message implies that you have turned on the Edison without other Energy-aware system components connected. This message will be recorded whenever the Edison is turned on as a standalone platform.

Codes

Now, we will study the codes. As you have seen in the previous section, the code written for the Arduino Uno is Arduino sketch code. The code written for the Edison is in C. It is not necessary to understand or modify the C code to use our Energy-aware system. Instead, you may modify other code to change the behavior of the system. Follow the steps below to learn how you can modify some code to fit the system to your specific needs.

1. Open “arduino_example.ino”.
    1) There are five main work this code does: reading from the button, turning on the on-board LED to indicate the motion detection, relay control, I2C communication with the master device (Edison), setting up the next regular wake-up time.
    2) The code was initially written for a system that uses Intel Galileo (instead of Edison) and a PIR sensor (instead of a button).
    3) Read through the code. The code must be easy to follow.
    4) Modification may be done in the last function “void receiveEven(int howMany){ … }”. This function read a numerical value from the Edison and use the value to calculate the next wake-up time. In fact, the numerical value is fixed as 1. In the line “wake_up_time = sleep_start_time/1000 + (sleep_period*60);”, you may change 60 to any value desired to change the next wake-up time.
2. If there is an alert wake-up, “alert.sh” script runs. If there is a regular wake-up, “regular.sh” script runs. There scripts are located along with other files. Explanations of how they work and what can be modified are given as comments on the top of the each script.
    1) For example, in regular.sh, the suggested modification is the path to the directory and the command to run.
    2) To implement the SMS message demo shown in the YouTube video from “System Workflow” section, you can modify the SMS message example in Python from Tutorial 3.
    3) Modify the Python code in a way that it will send “Alert! Motion detected!” to your phone. Then, save the code in /home/root/sms.
    4) Modify alert.sh so that dir is set to “/home/root/sms" and cmd is set to “python sms.py”.
3. Set up the energy-aware system again and see if it works as intended.

As mentioned earlier, understanding the C code is not necessary to configure the system to fit your need. Nevertheless, the system cannot be fully understood without understanding the C code. Open “energy_aware.c”. The code is simple and must be easy to understand. The following is how it works.

1) First, it checks whether there is an I2C connection. If there is no I2C connection, it calls system(), which makes a system call to run a Linux command. For instance, system("echo $(date) No I2C connection detected. >> log.txt”) creates log.txt if there is no log.txt and writes a line of message that include the timestamp given by $(date) and the error message to log.txt. If there is log.txt already, it appends the message to log.txt If you are not familiar with “echo” command, please refer to http://www.tecmint.com/echo-command-in-linux/.
2) Then, it reads a byte via the I2C bus. If a proper byte message is sent from the Arduino Uno, the byte must be either ‘A’ or ‘R’. ‘A’ means that an alert event is detected. ‘R’ means that the Arduino Uno turned on the Edison based on the regular wakeup algorithm.
3) It checks whether the byte is -1. The return value of mraa_i2c_read_byte() is -1 when there is no byte data received. If it is -1, an error message will be written to log.txt. If it is not -1, it checks whether it is ‘A’, ‘R’, or neither. If neither, another type of error message will be written to log.txt.

4) In case of ‘A’, it writes a wake-up message to log.txt and then, runs a shell script named “alert.sh”.
5) In case of ‘R’, it writes a wake-up message to log.txt and then, runs a shell script named “regular.sh”
6) As soon as the script is done running, it send a value 1 to the Arduino Uno via I2C.
7) Then, the I2C is closed and runs “shutdown -h now” command to shut down the Edison.

Appendix

Downloading energy-aware system files via SFTP.

You can download the energy-aware system files to your computer via SFTP. The Yocto Embedded Linux image has SFTP installed. Follow the steps below to download the files via SFTP.

1. Open terminal.
2. Login process is similar to SSH. Instead of ssh, use sftp. (e.g. **stfp root@164.67.185.189**)
3. Any Linux command applies to the Edison and commands followed by “l” apply to your local machine.
   - ls - list files and directories in the current directory on the Edison.
   - lls - list files and directories in the current directory on your computer.
   - cd - change directory on the Edison.
   - lcd - change directory on your computer.
4. Navigate to the directory where you downloaded the system files (e.g. ~/tutorial5_energyAware).
5. Navigate to the directory you want to download the files on your computer.
6. Enter “**get -r ***” to download all files (You may not see log.txt).

References

1. http://www.eetimes.com/document.asp?doc_id=1264175
2. http://standby.lbl.gov/faq.html
3. http://www.tldp.org/HOWTO/HighQuality-Apps-HOWTO/boot.html
4. http://www.tecmint.com/echo-command-in-linux/