Monitoring Software using Sun Spots

Corey Andalora

February 19, 2008

Rochester Institute of Technology

Abstract

Sun has developed small devices named Spots designed to provide developers familiar with the Java programming language a platform for quickly creating embedded software applications with wireless communication abilities. The ad-hoc communication between the Sun Spots lends well for a sensor network that could be beneficial in a monitoring system for industrial applications. This document shall outline a software application developed for demonstrating a monitoring network for air conditioning units. The system involves embedded software to be deployed on a mesh of Sun Spots along with a base application that a user can use to view the current conditions of the air conditioners as well as historical data.

Hardware

The Sun Spots is built off of a 180MHz 32-bit ARM920T core processor with 512K of RAM and 4M of Flash memory encased in a plastic housing. The wireless communication is facilitated with a 2.4GHz radio and integrated antenna that is IEEE 802.15.4 compliant. The battery is a 3.7V rechargeable, 750 mAh lithium-ion that is charged through a USB adapter and lasts about seven hours under normal operating conditions. The sensor board includes a 3-axis accelerometer, a temperature sensor, and a light sensor. In addition, there are eight tri-color LEDs that work well for displaying information to a user as well as two switched that can be programmed to perform simple operations. Finally, it provides the ability to add external devices through six analog inputs readable by an ADC, five general-purpose I/O pins, and four high current output pins.

Software

The Sun Spots operate off of a Java ME compliant implementation called Squawk. The virtual machine operates directly out of memory and all device drivers are developed in Java. Any software to be flashed onto the Sun Spot is also written in Java.

Network

Sun Spots are programmed to use AODV (Ad-hoc On-demand Distance Vector) communication by default when sending and receiving data over their radios. The Spot radio stack implements LowPAN (Low power Wireless Personal Area Networks) on top of 802.15.4. It is possible to replace the default routing protocol with a different implementation by setting the routing manager for the LowPAN.

Server Application

The proposed application was designed to demonstrate a simple monitoring system utilizing the Sun Spots as sensors and relaying their findings to a central data store. Air conditioning units were the equipment selected for monitoring. Air conditioning units vibrate while operating so they were an obvious choice to use with the accelerometers provided by the Sun Spots. Excessive vibration would be considered faulty operation and

when detected should be reported to the user. In addition, air conditioners should be on if the temperature of a room is above a certain threshold. If the air conditioner is not on but the temperature given by the sensor exceeds the threshold, the air conditioner is considered inoperable because it isn't functioning while the room is hot. Finally, the Sun Spots are deemed faulty if a heartbeat is not received within a 60 second interval. All of these faults shall be displayed to the user so that he/she may take the appropriate actions.

The user interface provides a list of the Spots within the network as added automatically when operational. Each Spot is listed by address with a status indicator and when a Spot is selected, the features for the corresponding air conditioner are displayed. These features include the most recent acquisition time, temperature, and vibration features. If a specific feature is selected, then all of the data within one day's time is plotted on a graph for that feature. This plot will update in real time as new feature values arrive. The features are stored in a Microsoft Access database that can be queried by the software.

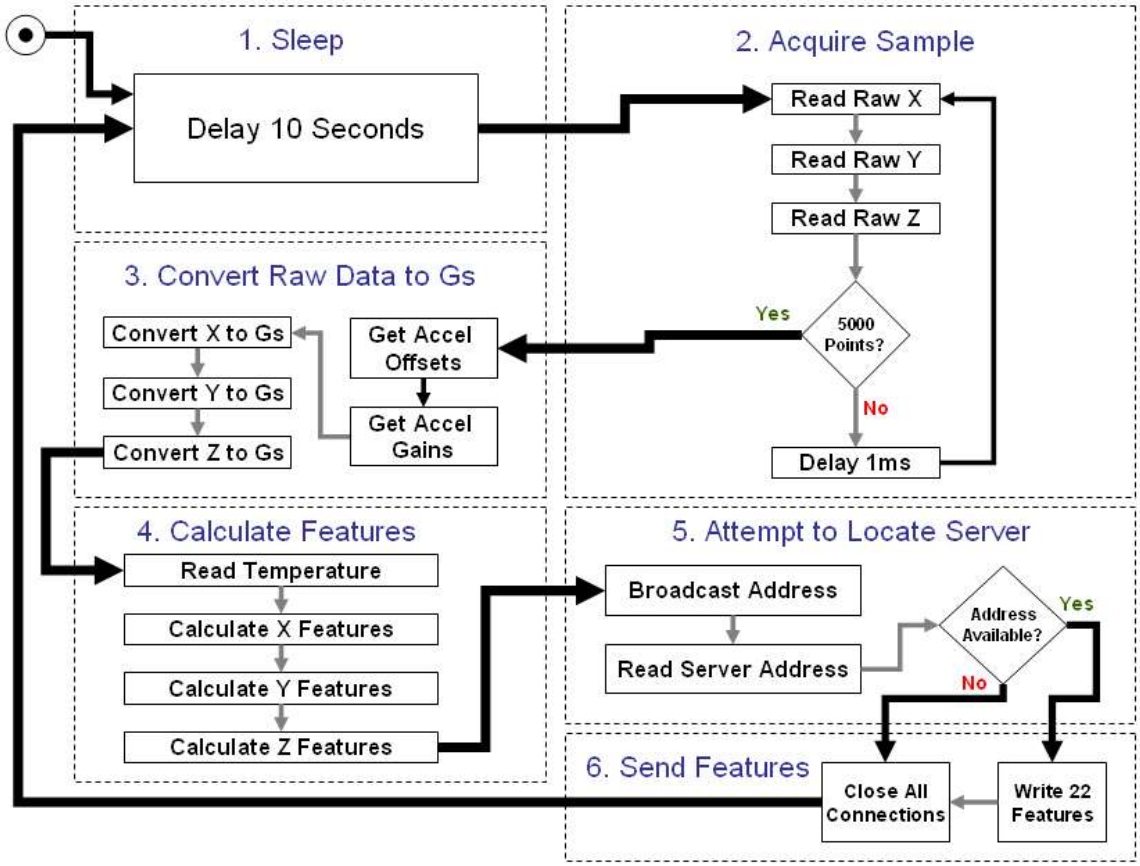
SPOT Software

Each SPOT contains the same software for acquiring data and calculating features. The entire application operates within one thread in order to ensure accurate timer execution and runs within an infinite loop. The following explains what each iteration of the loop performs:

1. Delay 10 seconds
2. Acquire raw data from X, Y, and Z axes of accelerometer for 5 seconds at 1 KHz
3. Convert raw accelerometer data to Gs
4. Calculate features
5. Attempt to locate server
6. If server was located, send features

The default behavior of the Sun Spot radio communication is to only have the radio on while wireless connections are held open. The software placed on the Spots was designed to open connections with the server only long enough to get the address and send the features and then immediately close the connection. This essentially keeps the radio usage to a minimum and thus conserves power. Because the Spots implement AODV, there is no need to perform routing and the data messages will find their destination automatically as long as a route exists.

The following figure depicts a representation of the Sun Spot software in flow chart form. Notice all six steps are broken down to show a bit more detail on their operations. After completing step 6, the process continues again at step 1. The features calculated and communication protocol will be discussed in more detail in the next sections.



Feature Calculation

The following features are calculated for the acquired data sample of each axis of the accelerometer:

- Minimum value
- Maximum value
- Average value
- Standard deviation
- Variance
- Root mean square
- Kurtosis

The minimum value doesn't provide anything too useful but may be necessary in case large negative accelerations occur. The maximum value is useful in late stage faults and for general anomaly detection. For the purposes of this software, it is used to indicate when the air conditioner has been switched on. The average, standard deviation, and variance can be useful for general fault identification. Root mean square measures the magnitude of variation and is the most common fault identifier in industry. Kurtosis is the fourth statistical moment and is a very common fault identifier. It shows how peaky the

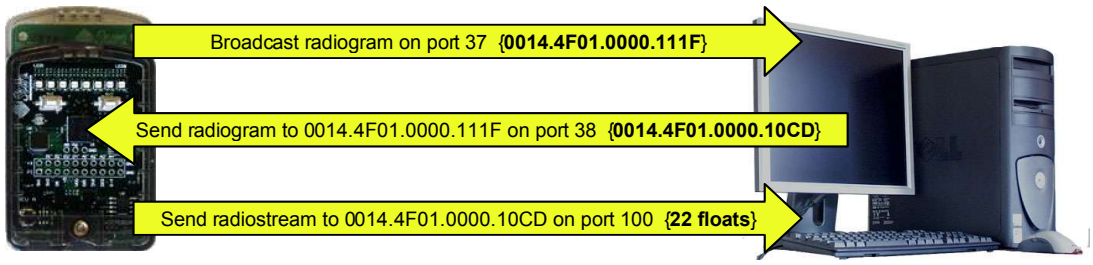
data is, which causes the distribution to appear lop sided and is one of the few features that have a definable healthy value regardless of system.

For this application, a total of twelve features were combined for calculating the health of the air conditioner. Standard deviation, variance, RMS, and kurtosis were taken for all three axes and compared against average expected healthy values. The percent error is calculated, summed, and averaged with an allowance of 20%. This error is subtracted from one and rounded between 0 and 1 to give a health of the air conditioner where 1 is completely healthy and 0 is completely faulty.

Communication Protocol

There are two types of communication possible with the Sun Spot. First is the radiogram protocol which acts like UDP in that it sends data out without guarantee of delivery. This message is limited to a size of 256 bytes and is useful in delivering small broadcast messages. The other type of communication is called radiostream and allows for much larger data transfers as well as guaranteed delivery. This is similar to the TCP protocol.

The following figure demonstrates the communication that occurs after every new set of features is calculated. First the Sun Spot broadcasts its address on port 37 using a radiogram and then waits for a message on port 38. The server is constantly watching for broadcast messages on port 37 and when one is received, it reads the address contained within the message and opens a radiogram connection on port 38 to that address. The server then sends its address through a datagram and waits for a radiostream connection on 100. Finally, the Sun Spot opens a radiostream connection to the address read from port 38 on port 100 and sends the 22 calculated features as floats. The connection is closed which turns the radio off and then the Sun Spot starts the next iteration of the loop.

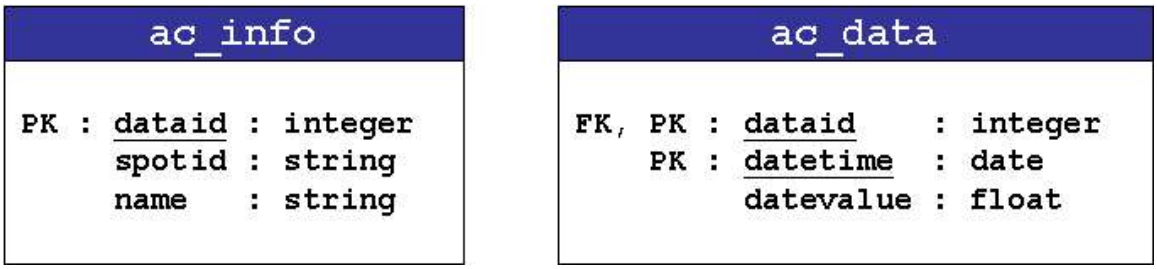


Database

The database resides on the same machine as the server software application. It is a simple Microsoft Access database but there is no reason why it couldn't be replaced with a more robust system because communication to it is provided through generic SQL commands.

The database consists of two simple tables: ac_info and ac_data depicted below. The table ac_info is used to describe all of the Sun Spots in the system and the ID numbers for each of the features. The primary key is the feature ID number and associated with it is the address of the Sun Spot and a name for the feature. The ac_data table is where all of the

feature values are stored. It contains a foreign key to the feature ID numbers in ac_info as well as the timestamp when the value was acquired and then the value itself. The primary key is constructed by combining the dataid and datetime fields.



In keeping with the theme of purely ad-hoc software, the server software will automatically set up the ac_info table as new Spots are added to the system. It finds the currently largest dataid given to a temperature feature and increments it by 100.

Deployment

In order to build the Sun Spot software, you must first install Java 5 and the Sun Spot SDK included with the devices. As part of this process you must have Apache Ant installed as well with all necessary environment variables set. See the user's manual that accompanied the Sun Spots for details. In addition, be sure to have your Sun Spots upgraded to version 3.0 of the SDK (Purple).

Note that the server software has only been tested on Windows and there may be problems with the Microsoft Access database on other operating systems because the software currently takes advantage of the Access ODBC driver.

This document will assume your project is located in C:\sunspot\demo. Once you meet the prerequisites open a command shell and navigate to the project folder for the server software (C:\sunspot\demo\MultiSpotDemonstration).

To compile host:

```
C:\sunspot\demo\MultiSpotDemonstration> ant host-compile
```

To run host:

First make sure you have the configured base-station connected to you machine.

```
C:\sunspot\demo\MultiSpotDemonstration> ant host-run
```

In order to deploy the software to the Sun Spots navigate to the spot project folder (C:\sunspot\demo\DemoSpot).

To compile client:

```
C:\sunspot\demo\DemoSpot> ant compile
```

To deploy client:

First make sure you have a wireless Sun Spot connected to your machine.

```
C:\sunspot\demo\DemoSpot> ant deploy
```

Reset the Sun Spot if prompted by pressing the power button.

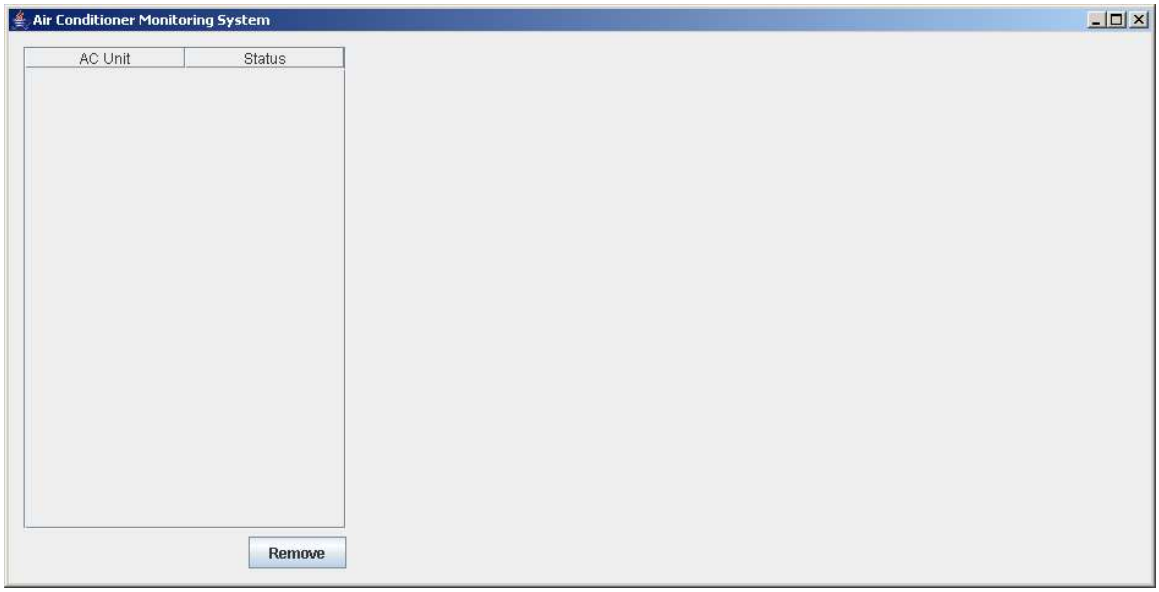
To run client:

The Sun Spot software will run automatically when the Sun Spot is rebooted. This can be done by resetting the Sun Spot using the power button.

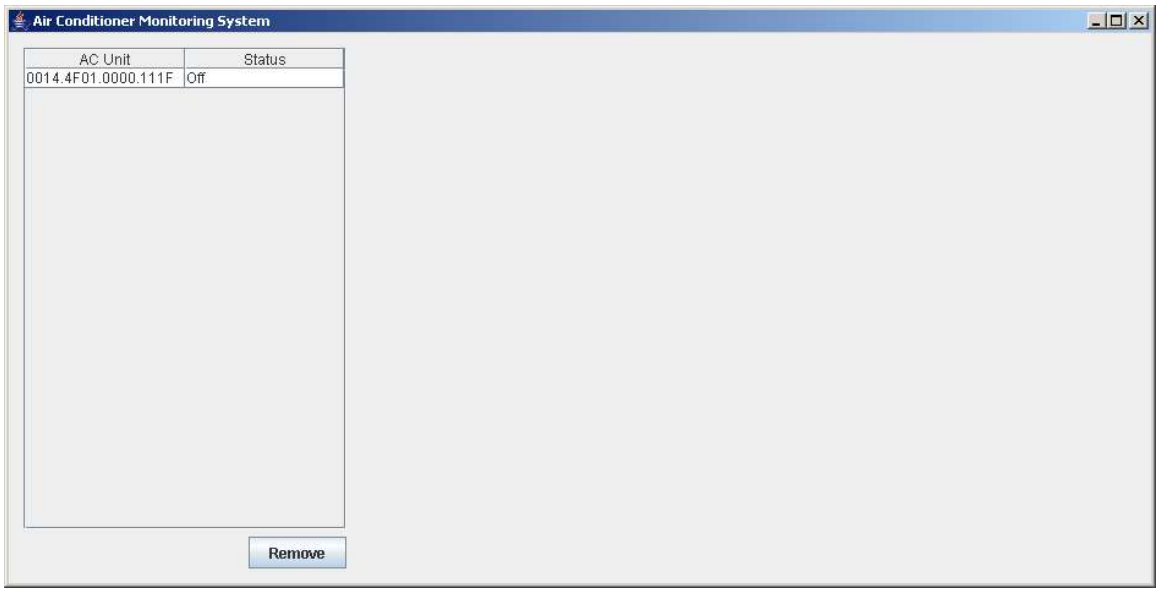
These folders are also set up to be used within Sun's NetBeans IDE software. You can add them as projects by selecting the folders from within NetBeans.

User Manual

Start the server software as described above and you will get the following screen:

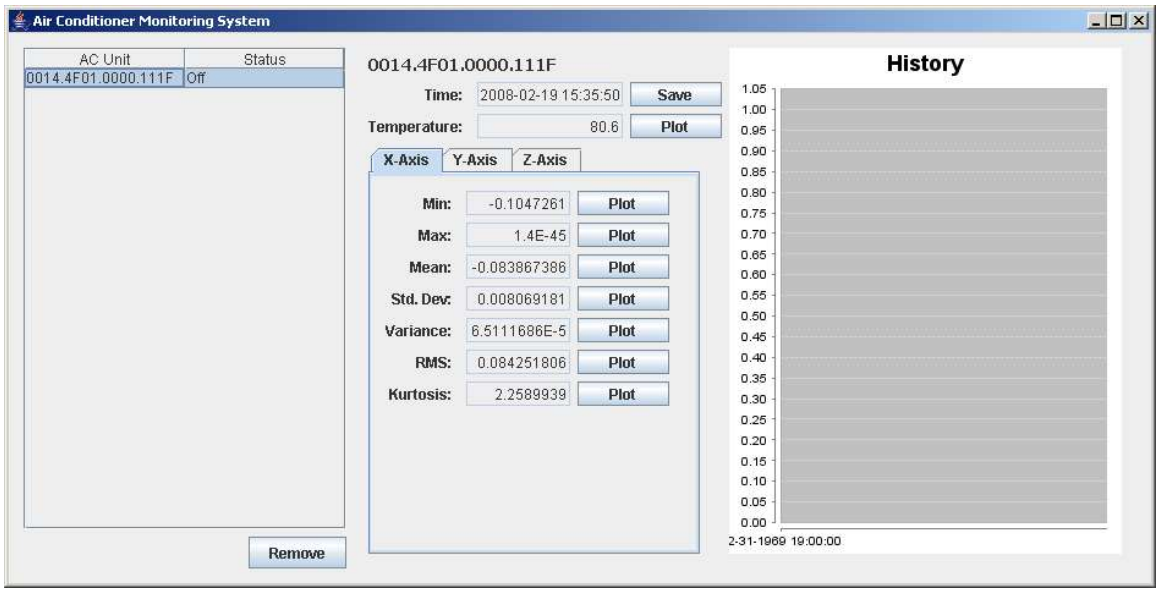


At this point you should start one or more Sun Spots set up with the client software. The following screen shows a Sun Spot added to the table:



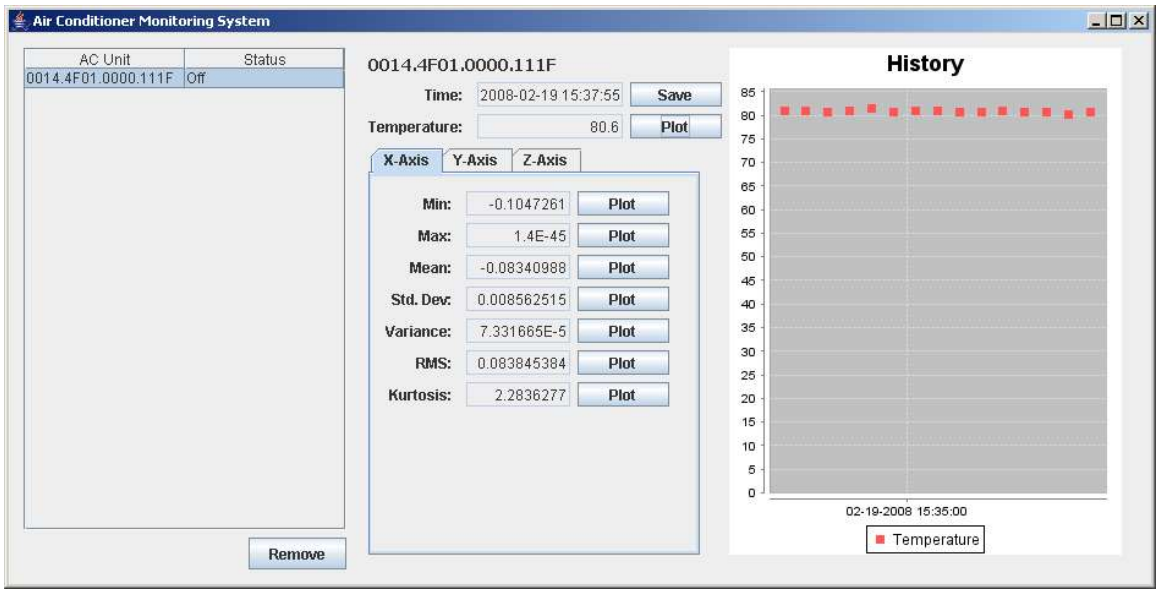
In the screenshot above, notice that the AC Unit is determined to be off. If this is the first time the server received data from this Spot it will insert appropriate rows into the database at this point. At any time a Sun Spot can be removed by selecting its address in the table and then clicking the "Remove" button. If the Sun Spot continues operation, it will re-add itself after it transmits its next set of features.

When a Sun Spot is selected, its features are displayed in the area to the right of the dialog:



This screen displays the time the last set of features were acquired along with the 22 features themselves. The "Save" button will save the current features to a user specified file. Clicking on the individual tabs for the accelerometer will display the seven features for the given axis. Any one of the features can be plotted in the history graph to the right by clicking the "Plot" button next to the desired feature. This will fetch the values for that

feature within the last 24 hour period. The following shows a short span of temperature values:

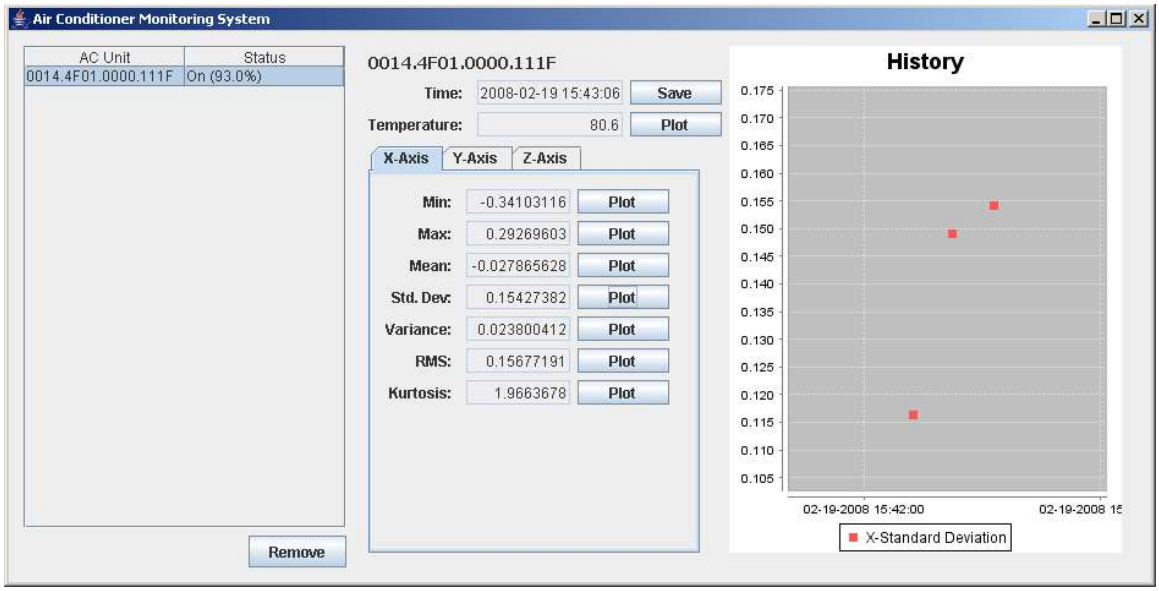


The graph is provided by JFreeChart (http://sourceforge.net/projects/jfreechart) and supports zooming, printing, and saving.

The following screens show some of the different states that are detected by the monitoring software. They were created by attaching the Sun Spots to Honeywell model HT-800 fans.

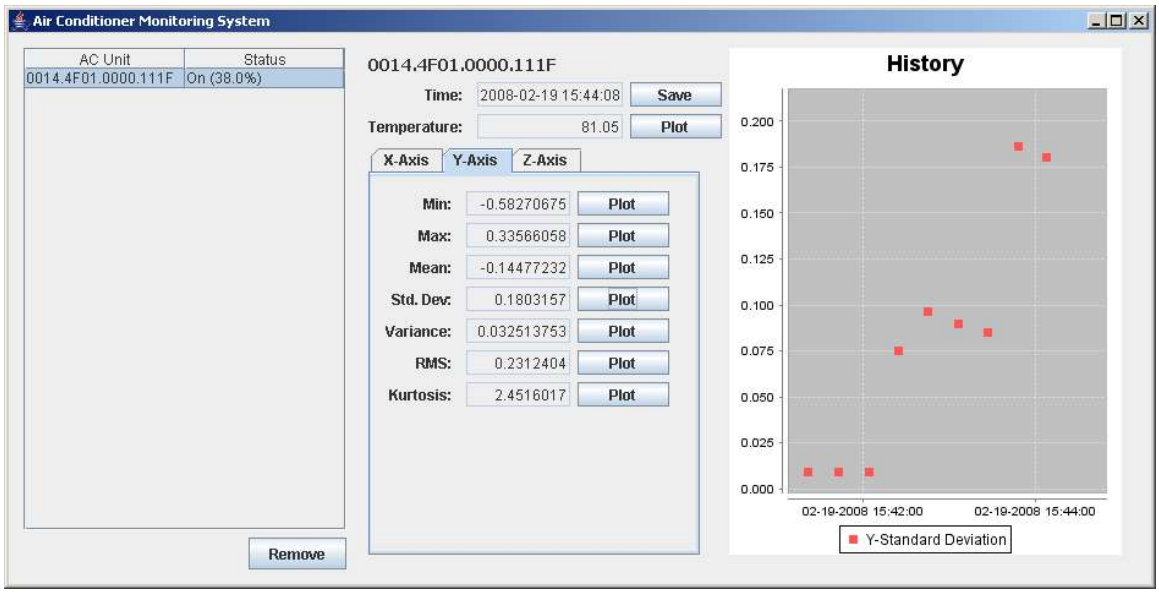
Healthy Air Conditioner:

- Fan on low setting



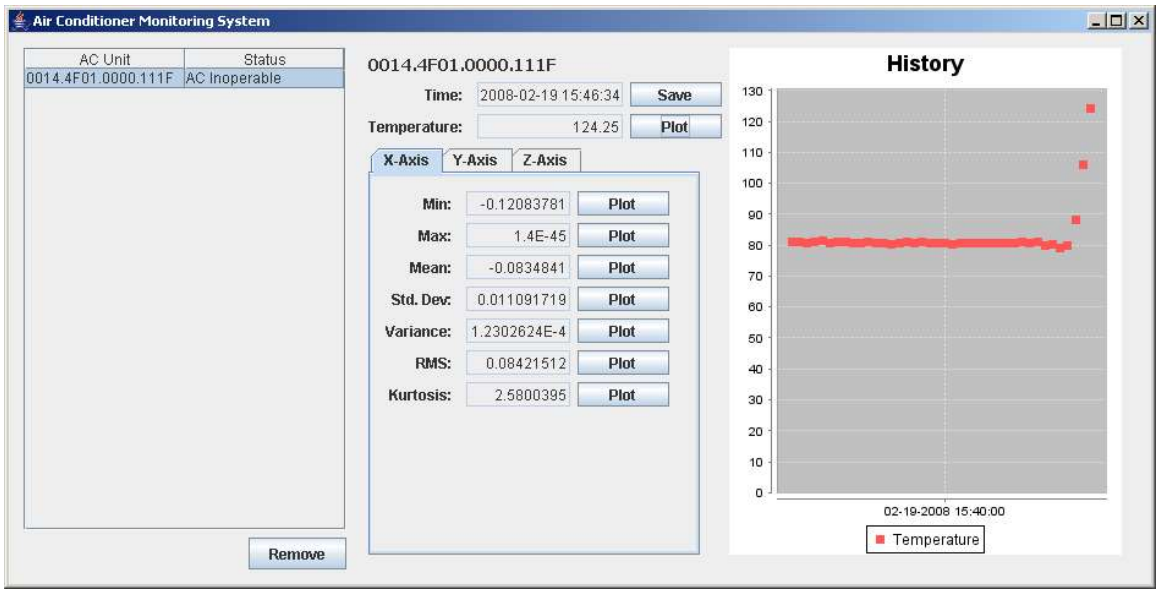
Faulty Air Conditioner:

- Fan on high setting



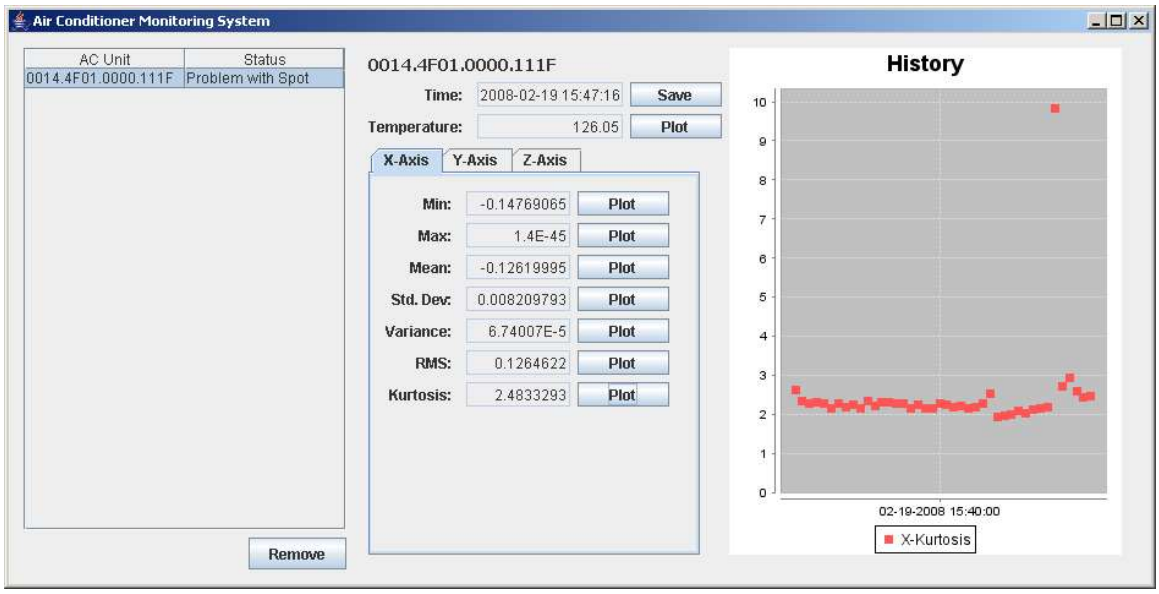
Inoperable Air Conditioner:

- Temperature high but fan is off



Problem with Sun Spot:

- No features received in over a minute



Conclusion

The Sun Spots are very easy to develop on due to the familiar Java programming language. There were few problems flashing the software onto the devices and establish communication among them. The Spot community is growing and can be very useful as a resource in assisting with development (https://www.sunspotworld.com/forums).

The accelerometer made available on the Sun Spots is not useful for high speed data acquisition requirements but as shown in this application can still work for slower acquisition rates. The temperature sensor is not terribly accurate because it is place so close to the circuitry and will typically be much warmer than the environment it is within. The battery life could become a problem but with well thought out conservation schemes, an application could operate for days without recharge.

The eight tri-color LEDs are extremely useful for indicating modes of operation and displaying fault codes to a user. With a large range of colors possible, a developer can take advantage of several combinations. Likewise, the two switches make for great mode switches and event triggers.

The ad-hoc routing for these devices is more than adequate for simple transfers within a reasonable range. The Spots network scales very nicely as developing for a large system would require little adaptation from a smaller system.

As a whole, it is the author's opinion that the Sun Spots are a great investment for hobbyists, education institutes, and companies interested in rapid prototyping. They are

simple to develop for with not too large of a learning curve as they require little time for someone to get an application running.