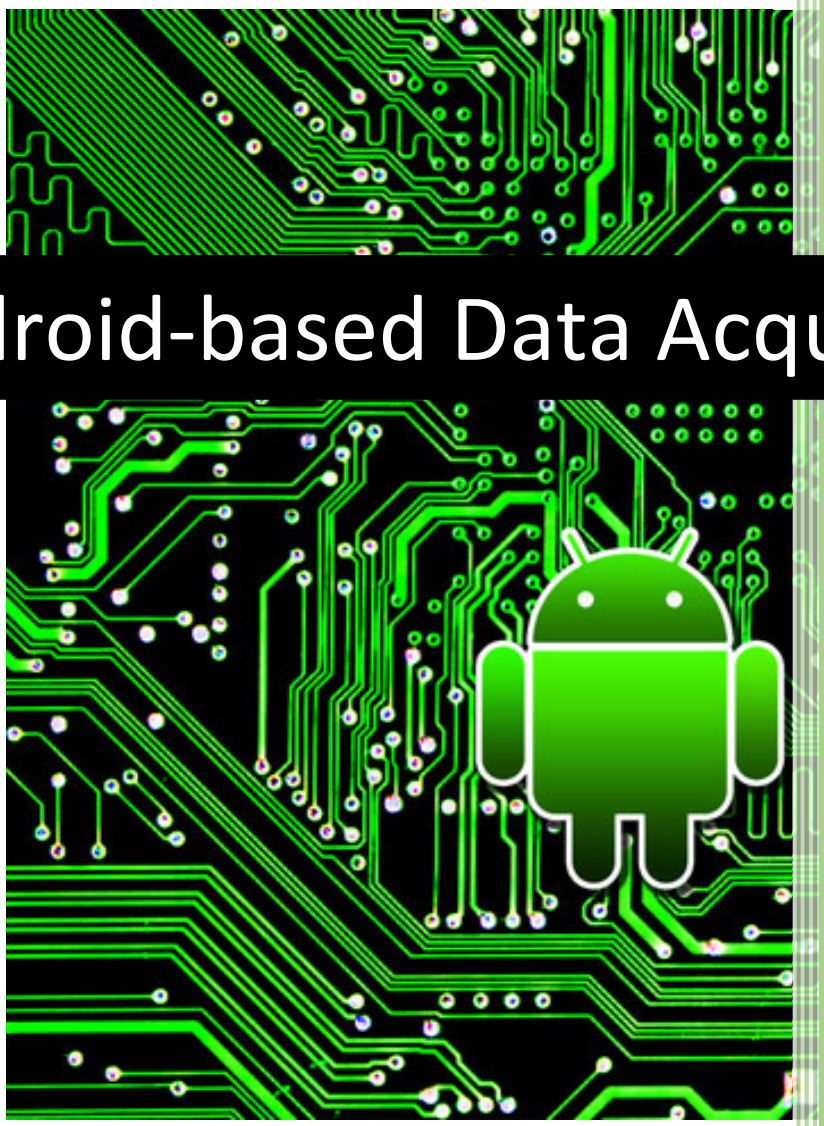
Fall 2014 – Spring 2015

Android-based Data Acquisition System



Authors:
Tochukwu Akujuo 108094561
Saket Ati 1080177378
Zeev Douek 108334308

Faculty Advisors:
Dr. Mónica Bugallo
Dr. Helio Takai

Abstract

Android OS based cellular phones and tablets systems are becoming ubiquitous. They are popular, powerful and equipped with a long list of sensors. Many will no longer serve their original purpose and will be recycled. For a number of applications they can still be used effectively, e.g. in experiments where the data collection speeds are relatively low, but connection to the internet and cloud services are required. We would like to explore the use of Android based tablets and cell phones as data collection nodes for physical science experiments. To complement the sensors in the device, we will use the Arduino micro controller. Thus this project entails two main goals. First to establish a communication between an Android based tablet, or cell phone, with an Arduino micro controller and, second to establish a connection between the android devices to a centrally localized computer via wireless network. If successful we will use this setup to acquire data from a cosmic ray detector.

Table of Contents

Section 1: Goals and Impacts

The goal of our project was twofold. Our first objective was to take an existing system that Dr. Takai's team at Brookhaven National Laboratory uses and to improve upon the design. We concentrated on minimizing the hardware of the system by integrating the functions of several existing boards into one development board. We retained parts of the existing design. In the process, we aimed to make the design expandable and easily reproducible. This will enable Dr. Takai and his team to be able to set up more of these systems in the future. This hardware collects data from the sensors and Yahoo Weather and uploads the data to a text file on DropBox.

Our second objective was to develop a front-end Android application that will enable users to access the data collected by the hardware. The app has the capability to allow users to pick data points that they want to observe and fetch that data from Dropbox. The app makes use of the Hello Charts API to display the data in a graphical format. This will allow the users to observe trends and patterns in a huge dataset.

Our project does not have immediate impact on society. It is academic in nature and intended to improve the existing system that Dr. Takai and his team uses. His team will be able easily access the data and observe trends and patterns in a pictorial format rather than a plain text format. This is a great advantage as you can imagine how difficult it is to observe hundreds of lines of numbers. Using our system, Dr. Takai's team will be able to conveniently reproduce and set up more sensors in the future.

Section 2: Background

The system currently being used has a few different development boards (seen in figure 1) doing specific tasks.

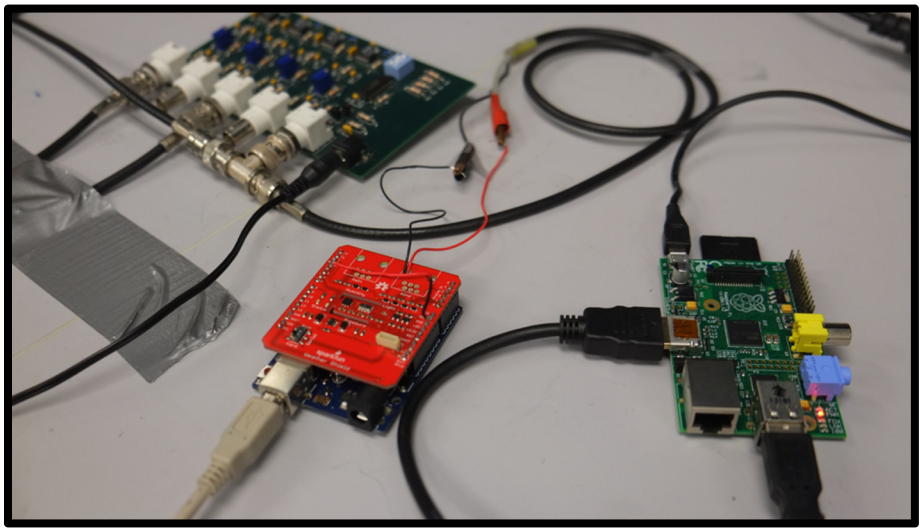


Figure 1 Current Data Acquisition System. Courtesy: Dr. Takai's Notes

Firstly, there are the cosmic-ray detectors that detect cosmic rays and send an analog signal to the coincidence board. The purpose of the coincidence board is as follows: the output detector of the board is active only when the two input signals receive a signal within a time window that is accepted as concurrent and in parallel at both inputs. The purpose of using coincidence detection is to reduce the chances of the output being triggered due to the inherent noise of both the detector itself and/or the environment. By using coincidence detection the probability that noise will active the output is squared, as result the chances of a false detection is severely reduced. A digital signal is sent to the Arduino Uno Board when both the signals have a detection, and the output is active. The Arduino Uno has a weather shield connected to it that records ambient weather data. It is to be noted that currently this system sits in-doors. Hence the data being recorded by the weather shield is room temperature. Every five minutes, the Arduino sends the count of the number of detections and the weather data over to the Raspberry Pi that then uploads this new information to a text file on Google Drive.

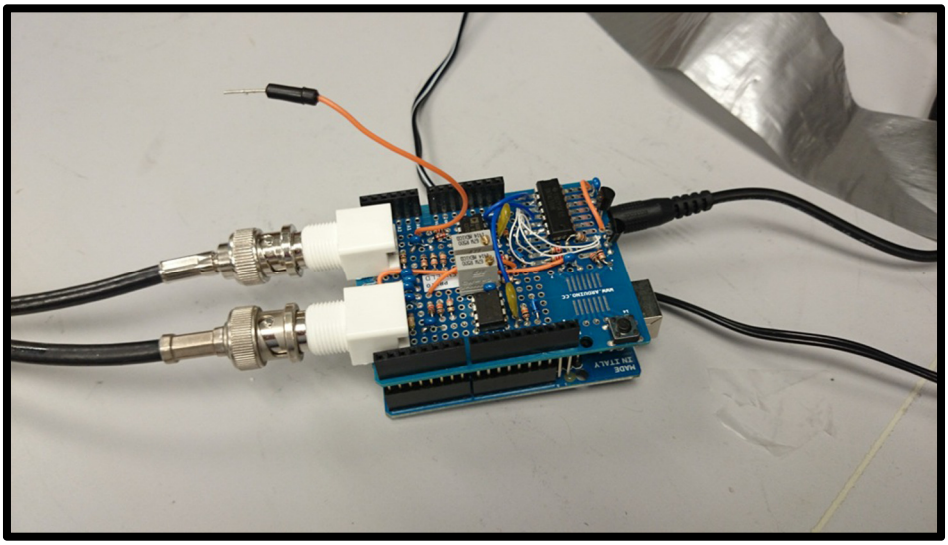


Figure 2 Coincidence Board. Couresty Dr. Takai's Notes

Previous developers have designed a coincidence shield that is much smaller and simply plugs into the Arduino board just like the weather shield (see in figure 2). The advantage of this is, it can be stacked on the Arduino along with the weather shield and others as necessary. The drawback for us is that we have no control over the design and pin assignments of this shield and have to treat it as a black box. We will work our new design around the existing design.

Section 3: System Implementation and Testing

3.1. Implementation Problems

3.1.1. Arduino:

Due to the versatility of the Arduino Yún and all the help/tutorials available for it, we had no major implementation issues on this end. The one minor issue that we had to deal with was the Temboo Client service that we were using to make Yahoo Weather API calls had a limit on the number of calls that can be made per month on a free account. However, there was the option of getting more calls by inviting others to make a Temboo account. We made use of this option by creating dummy accounts that increased our limit on our main account. The other more complex alternative was to write our own Python script that would interact with Yahoo Weather website to get the weather information. This would have been more time consuming, error prone and would have required hours of testing to get it to work. Hence, we adhered to the Temboo option.

3.1.2. Android:

Due to the nature of the data that needs to be graph, one of the implementation problems that we had to face was that the android application needs to store the data fetched from the online cloud storage service in a data structure that is cohesive large data sets. The data structure that is to be selected must have the property of scalability in which performance does not suffer with a gross increase in entries within the selected data structure. As a result we used TreeMap to store the data. Another implementation problem was that the graphing APIs selected must be able to graph large data sets, and have a relatively easy to use for android. Also the overall aesthetics capabilities of the selected graphing API must be balanced with its performance capabilities in dealing with large datasets. We found the the HelloCharts API for Android properly addressed our concerns. Lastly, the final implementation problem we face was the the android application need to support multiple screen resolutions, since the application will run on both smartphones and tablets The application is intended to run on old android devices as a result the minimum Android API level supported must be keep as low as possible while maximizing the use of Graphic User Interfaces.

3.2. Final Implementation

3.2.1. The hardware system

The hardware system (consisting of the cosmic ray sensors, coincidence shield and Arduino Yun), has three main tasks as described below.

a. Count the number of cosmic ray detections per 5 minute intervals: The cosmic ray sensors and the coincidence shield are used exactly the way they were in the original system. The output from the coincidence shield

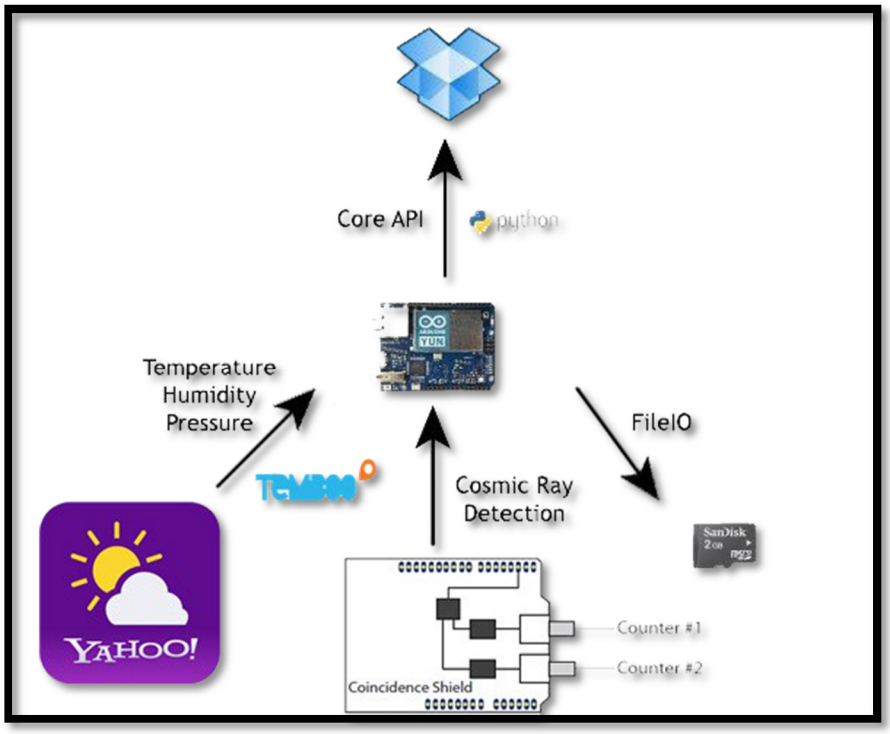
is connected to pin 3 of the Yún. The sketch uses an interrupt to count the detections. This interrupt driven approach has an advantage over polling approach because the system is not constantly polling pin 3 waiting for pulses. Using an interrupt allows the program to perform other background tasks while the count is incremented. The Interrupt Service Routine (ISR) simple does an increment on the count variable. So it is extremely fast and does not create timing issues with other parts of the program.

An interrupt is also used to count the number of seconds/minutes. The internal Timer/Counter 1 of the ATmega32 microcontroller is configured to call an interrupt every second. This is done by initializing the Compare Match Register to a pre-determined value (15,624). Every time the internal Timer reaches 15,624, the ISR corresponding to Timer 1 is called. This ISR checks if the seconds variable is equal to 60*5 (5 minutes) and sets a flag. When this flag is true, the main loop calls the appropriate functions to append a newline to file. This timing portion of the code was reused from the original system.

b. Obtain the temperature, pressure and barometric pressure from Yahoo Weather: Our system uses a third-party service called Temboo as an intermediary between our system and the Yahoo Weather API. Temboo is especially created for Internet connected devices and has special libraries for the Yún alone. One of the library, aka. Choreo, was the Yahoo Weather getWeatherByAddress choreo. The Temboo client is instantiated once using the appropriate authentication for our account. Our system provides two inputs to the choreo, units for the temperature (C) and the address of the location (zipcode). After the choreo is run, it returns a plethora of data in XML format. We added output filters to only obtain the temperature, humidity and pressure. The authentication information, inputs and output filters for the choreo are defined in the TembooSettings.txt file on the SD card. Refer to the updateWeather() function header in the sketch for a description of the settings file. To change the location, the line “-iAddress:11790” in the TembooSettings.txt file needs to be modified.

c. Upload data file to Dropbox: At the start of a new day, the data stored in a file is uploaded to Dropbox. The Yún calls a python scripts which makes use of Dropbox’s Python API to upload the file to Dropbox. In the event that there is an error connecting to Dropbox or the that internet connection lost, the python scripts saves the names of the files that

failed to upload to a file called 'send_later.txt', and notifies the Yún that a failed to upload the file has occurred. Yún will then call a reupload shells script which runs in the background attempting to reupload the files listed in the send_later text file.



Hardware System Implementation

The switch in Online Cloud Storage Service from Google Drive to Dropbox a necessary change in the project's implementation. Uploading files to Google Drive had transaction limits (limited number of times one can send files to Google Drive per month under a free account) and authentication with Google Drive was cumbersome. Whereas Dropbox provided APIs for both python and android which allowed the system to work seamlessly together.

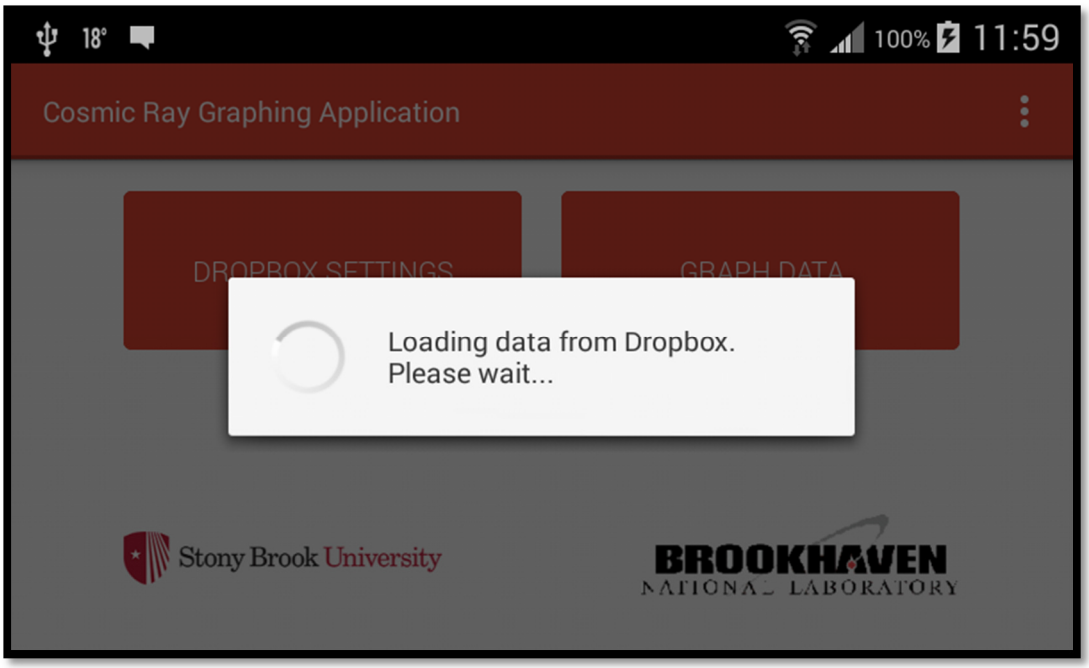
3.2.2. The software system

The software portion consists of an Android application, built specifically for the purpose of graphing sensor data. We will discuss the software implementation using a stepwise approach.

Starting the Application

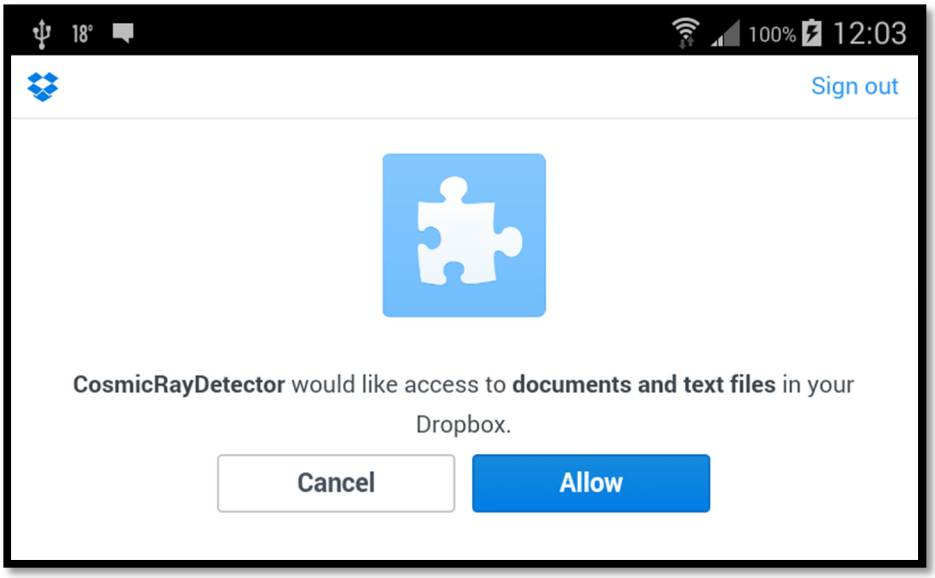
The name of our Cosmic Ray Graphing Application is shortened to “Cosmic Ray”. The application can be found in the app drawer of the device on which it is installed.

Loading Data from the Cloud

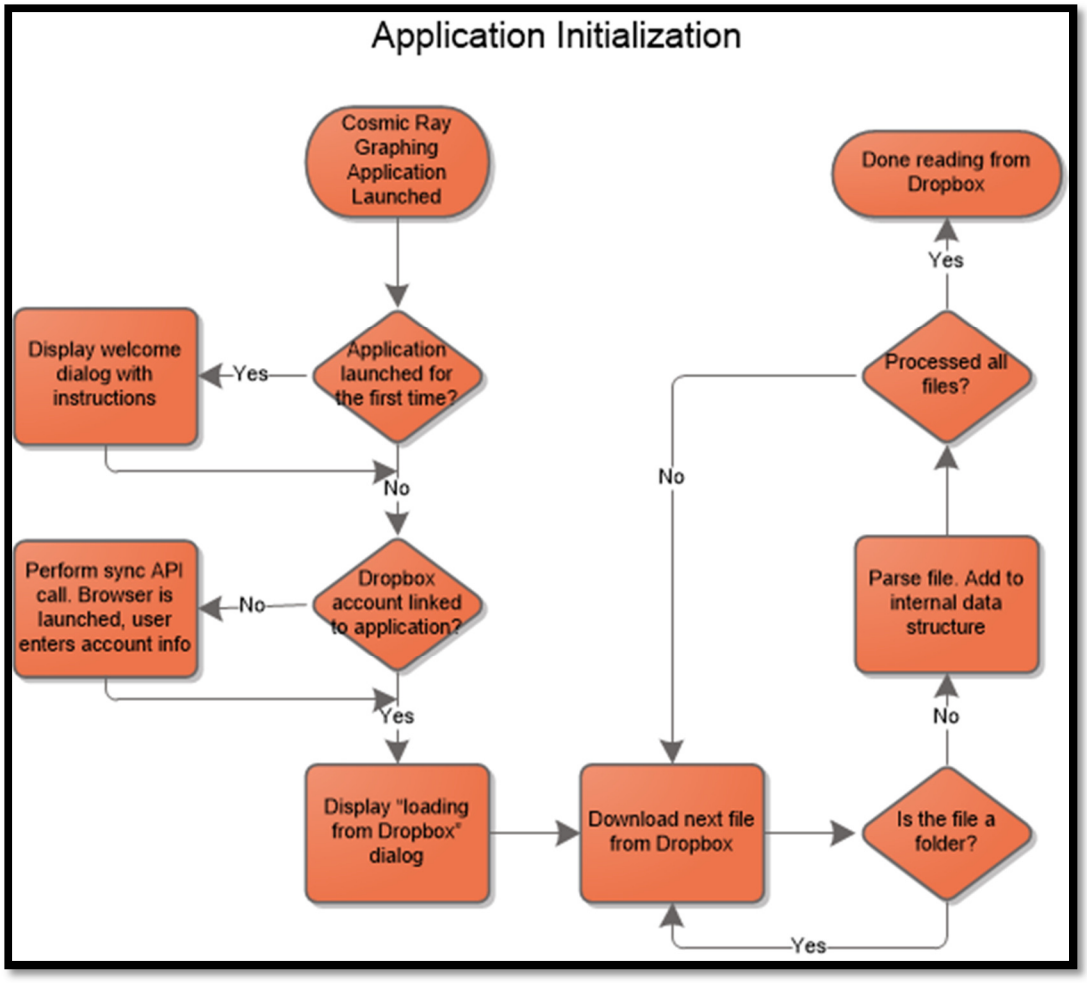


Once launched, the application will attempt to connect to a Dropbox account containing sensor data files. It will then automatically load all of the data from the cloud onto the device. If the application was not previously run on the device, or if it does not have a Dropbox account associated with it, the user will be asked to provide permission to access their Dropbox account.

Account Authentication via Browser



The application initialization process is outlined below:



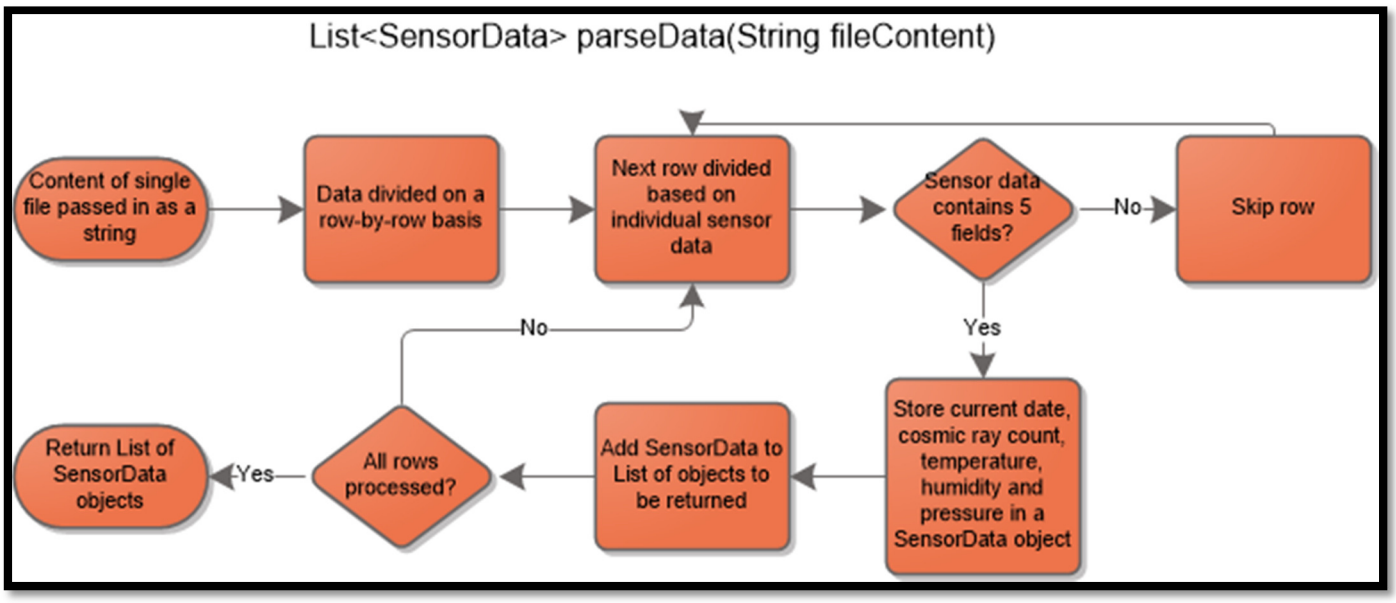
Our application is designed to specifically parse files containing data in the following format:

MM_DD_YY HH:MM:SS, RAY_COUNT, TEMP_C, HUMIDITY_PERCENT, PRESSURE_HPA

Sample Line:

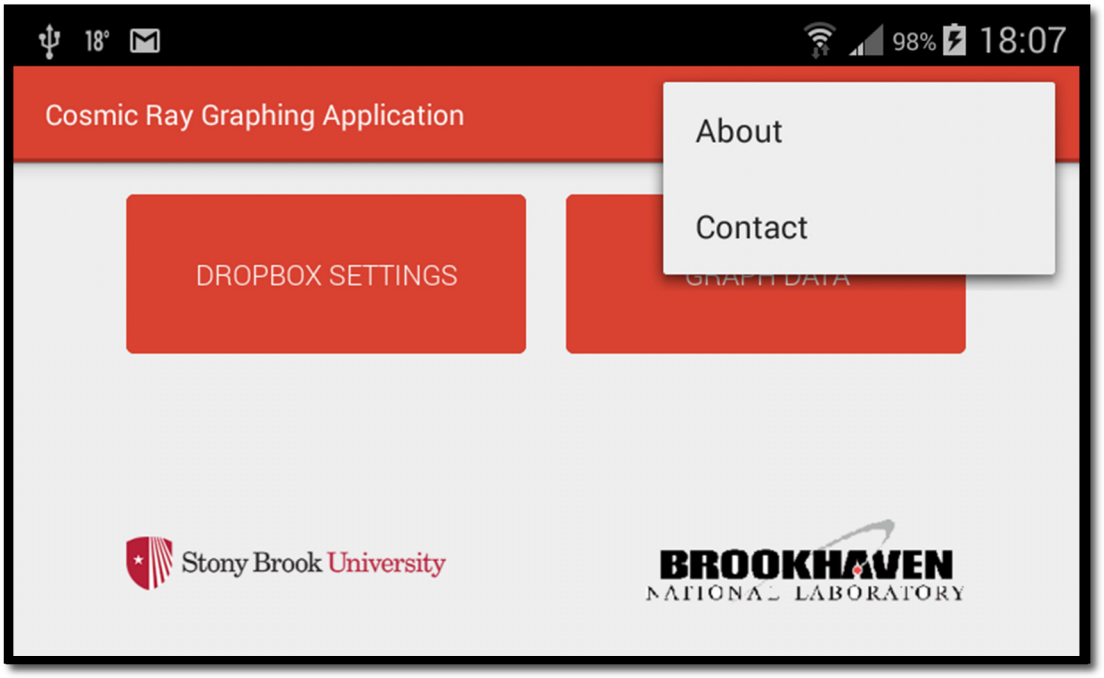
05_10_15 00:50:33, 7805, 14, 93, 1024

The parsing behavior is as follows:



Main Activity

Once data has been loaded from the cloud, the user is presented with the main activity screen. This screen is serves as a branching point through which all of the application's other activities may be launched.



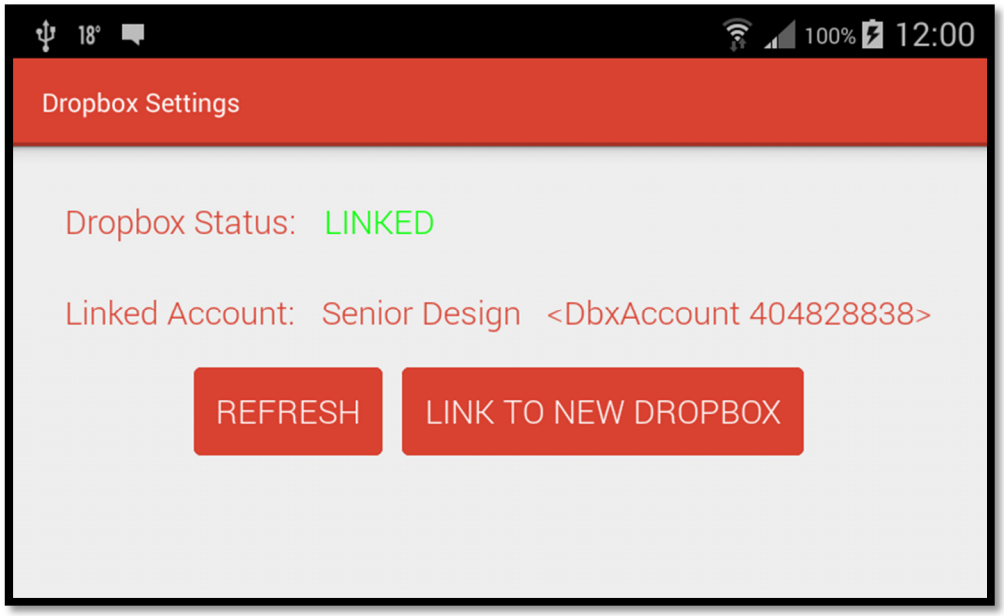
The “Dropbox Settings” button is used to verify synchronization between the app and Dropbox. It also allows the user the re-synchronize the app to a new Dropbox account if need be.

The “Graph Data” button sends the user to the graph activity where they may chose the data and timeframe to be graphed.

The Action Bar contains additional hidden buttons at the top right corner:

- Clicking on the “About” button will send the user to an activity displaying a short description of the application.
- Clicking on the “Contact” button will display an activity containing the names and email addresses of the faculty and students involved in creating the application.

Dropbox Activity

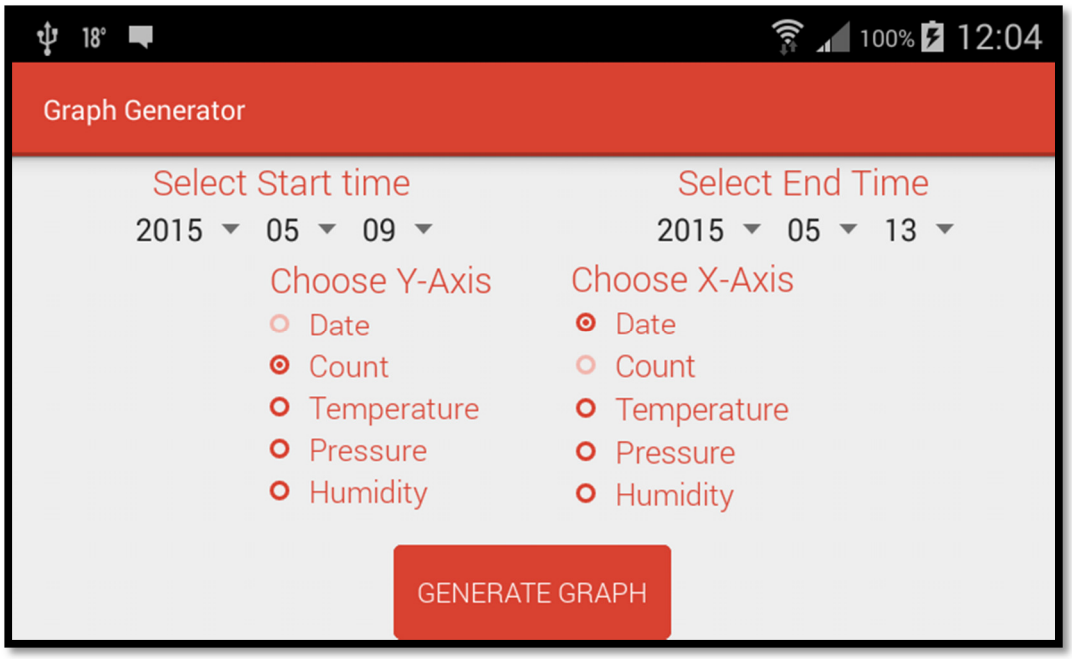


The “Dropbox Status” indicates whether a Dropbox account is currently linked to the application. This status may be updated using the “Refresh” button.

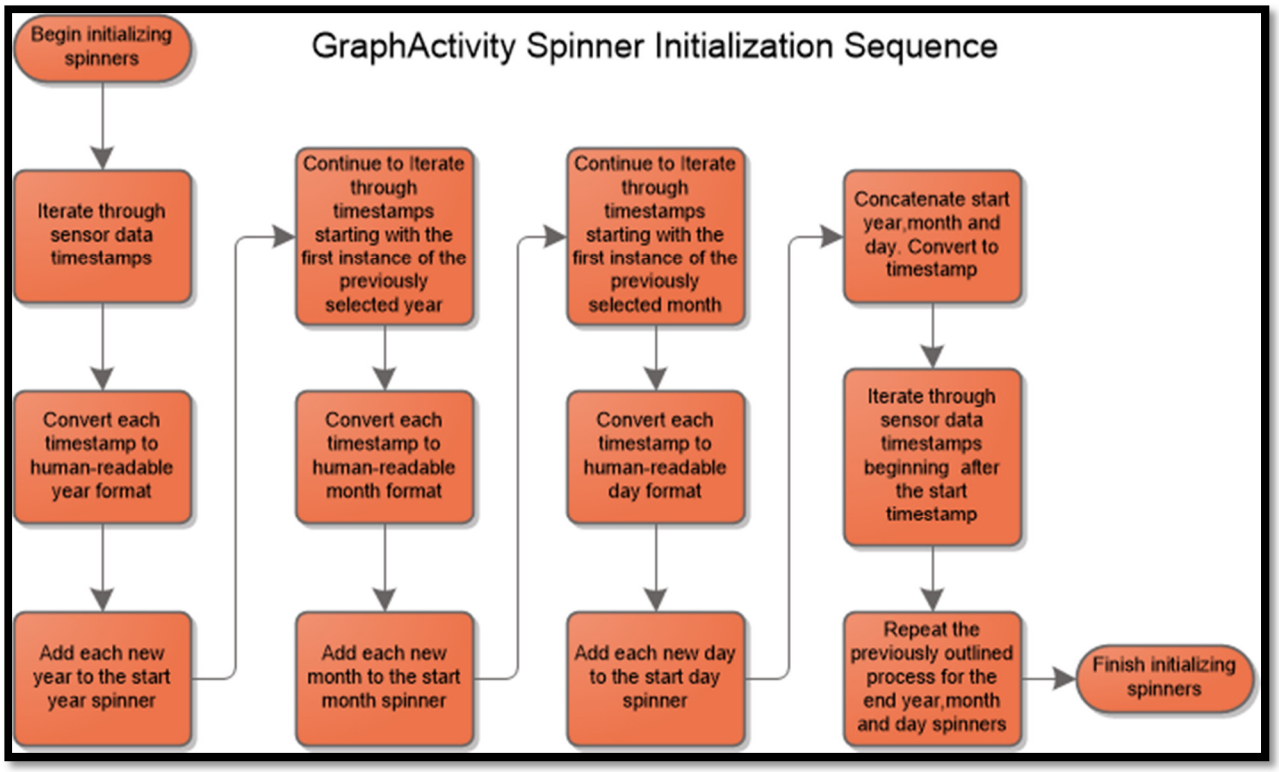
The “Linked Account” text field lists the full name of the account owner and their account number. If no account is linked, the field will display “None”.

“Link To New Dropbox” may be used to link the application to a new Dropbox account.

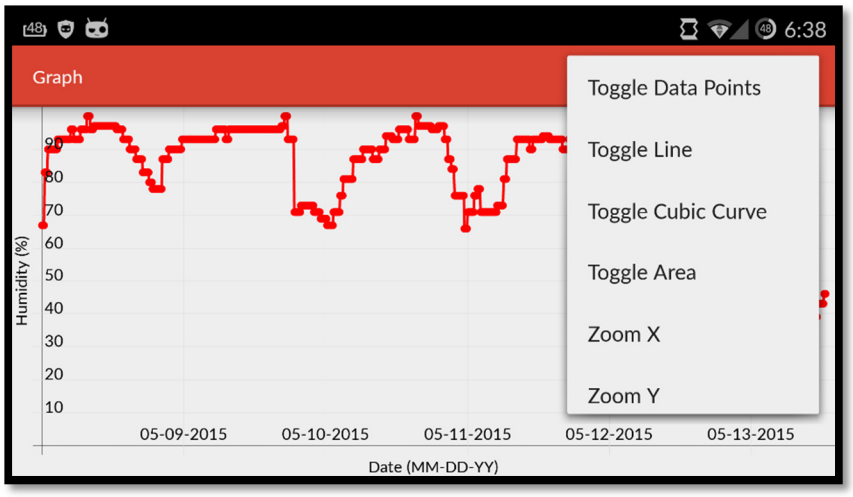
Graph Generator Activity



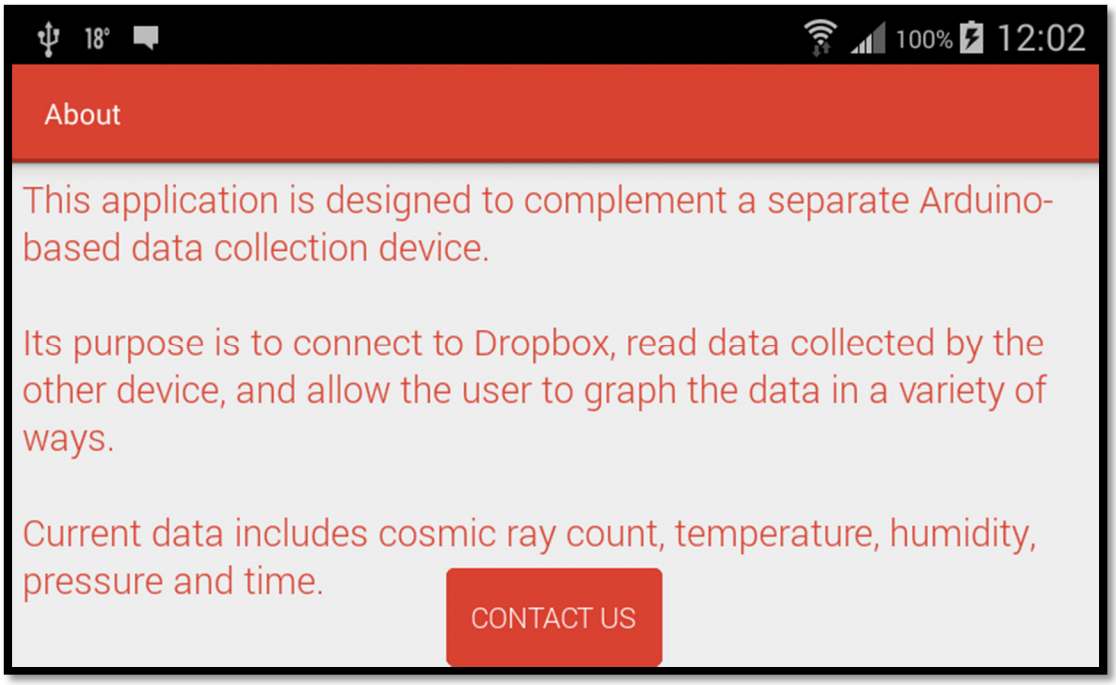
The graph generator activity contains the primary functionality of the application. The user begins by selecting a range of dates to be graphed using the six spinners at the top of the screen. The spinners are designed to only display data which exists in the user's Dropbox. Likewise, the "End Time" field updates based on the "Start Time" field, such that the end time is always after or equal to the start time. The initialization sequence for the spinners is outlined below:



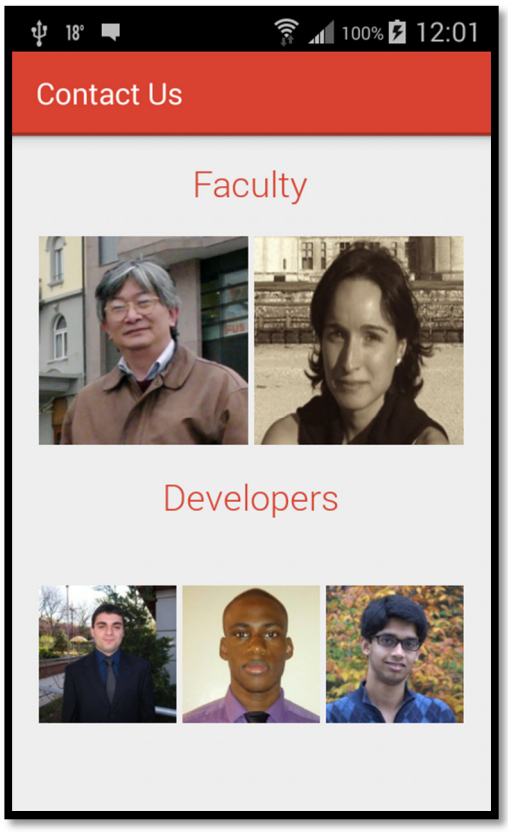
After selecting a range of dates, the user may dynamically select the data to be graphed on the X and Y axes. The radio buttons are designed to be mutually exclusive, such that the user may never graph a field against itself. The graphing screen provides variety of self-explanatory options to change the look of the graph:



About Activity

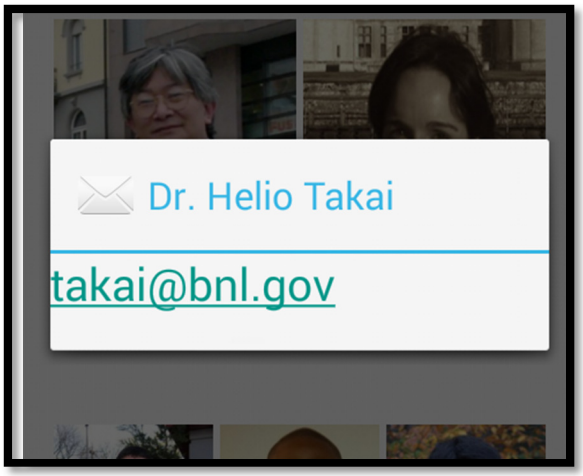


The about activity contains information about the application. Clicking “Contact Us” takes the user to the contact activity:



Contact Us Activity

The contact us activity displays clickable pictures of the faculty and students. Clicking on any images displays the person’s name and email address:



3.3. Testing

In order to make testing easier, the hardware system was developed in incremental steps. First the sensors were mounted and the required functions to test them were written. This was tested by simply printing lines to the screen to see if the sensor data was being collected appropriately. Next, we developed the file system to be able to write data to a txt file on the SD card. Following that, we implemented the Temboo system to get weather data. This data was printed to the screen and written to the SD card to verify that it was accurate. Finally, the script to take the data from the SD card and upload it to Dropbox was written. Once all these major blocks were written and tested, we put them together, carefully testing them all along. The hardest part was to design redundancies in case of network failures. Two main tools that were helpful for testing were the Serial monitor on the Arduino IDE and SSH client. We were able to monitor, in real-time, what the system was doing using print statements to the serial monitor. This was crucial in testing the flow of the complete system. The SSH capability allowed us to remotely access the data on the SD card of the Yún and monitor that data that was being written to it correctly.

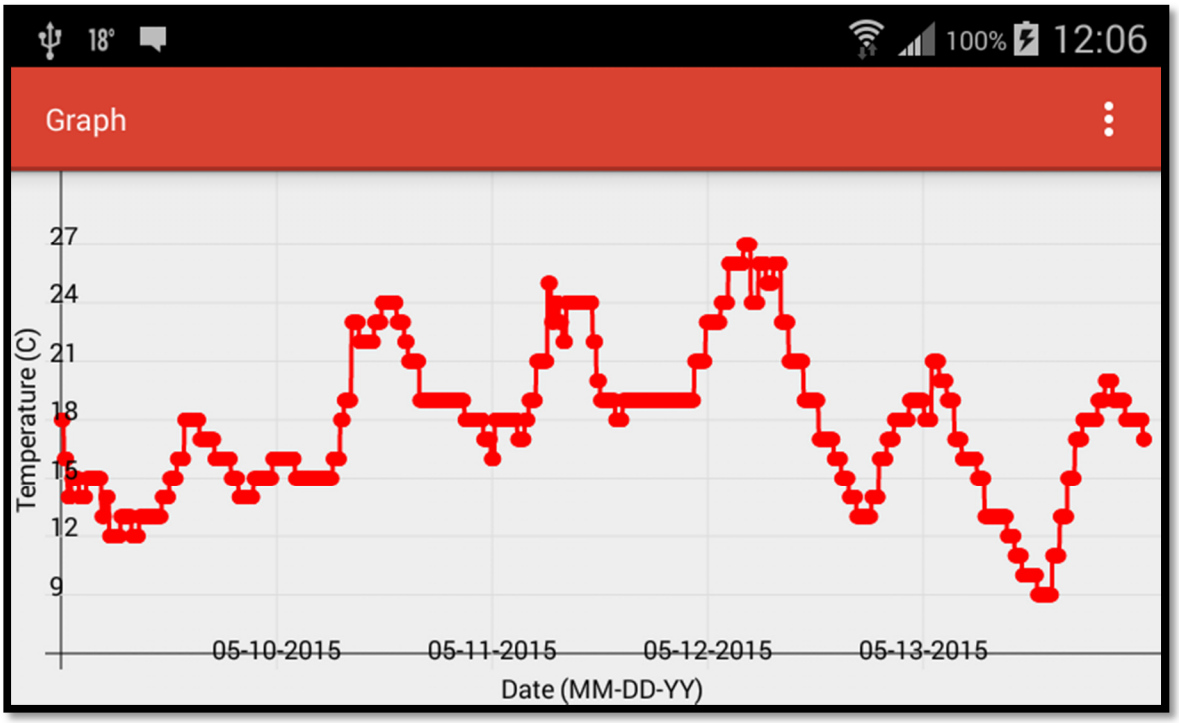
In testing the android app, log statements were used to keep track of each the step the application was doing to console. Each feature of the application was developed incremental and tested extensively before adding a new feature. The logs file allowed us to quickly identify what actions were causing the application to crash or produce undesired results. Testing the application on various mobile devices and running it on the emulator allowed us to address UI resolution issues early and promptly.

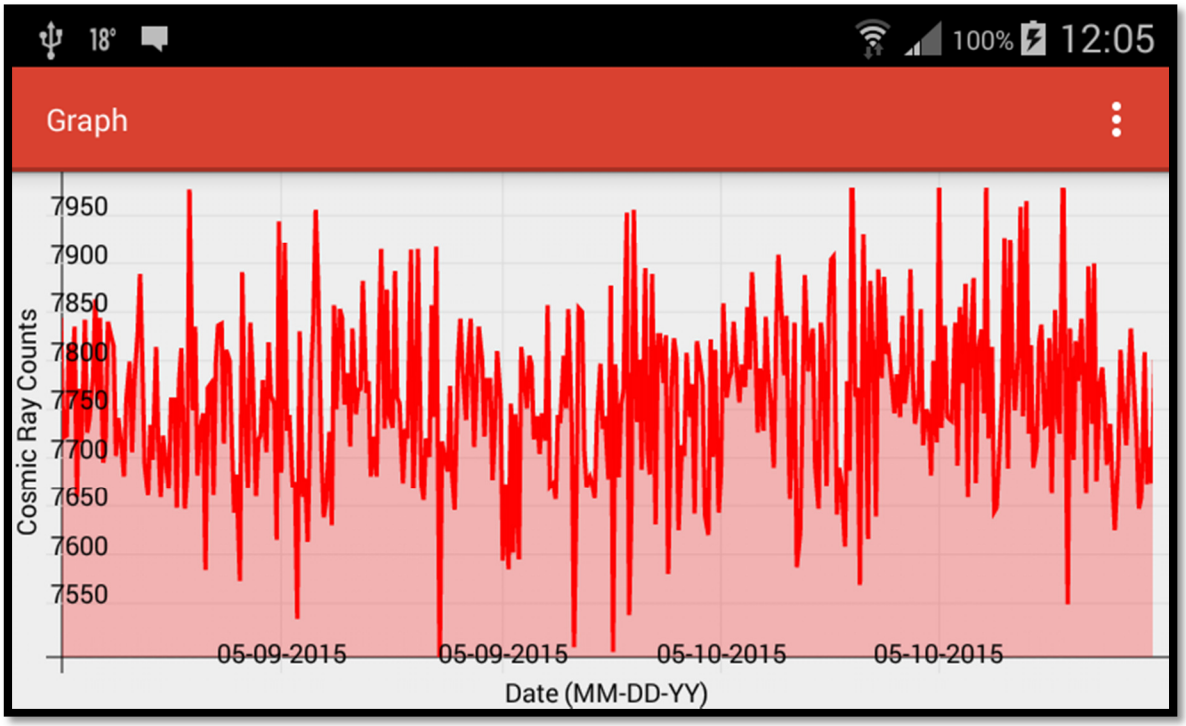
Section 4: Discussion

4.1. Result Analysis

Our project allows the user to generate 20 different types of graphs, spanning any time range. The graph data may be used for further research into the behavior patterns of cosmic rays as they pertain to weather data. Shown below are two sample graphs. The first is of temperature vs time, containing data points and a connecting line. The second is of cosmic rays vs time, containing a shaded area and a line.

Sample Graphs





4.2. Multi-disciplinary Experience/Issues

Working on this project requires a skillset which combines high-level Java programming with embedded system design. All of us come from the field of computer engineering, albeit with different focuses:

Zeev is majoring in Computer Engineering with a focus on software. He has taken several courses which are relevant to the project: CSE 114/214/219 (Java 1,2,3) and CSE 230 (Intermediate C++) for programming, and ESE 380 (Embedded System Design I) for hardware. He has worked as a Java developer for a major U.S bank and will be working for a software company after graduation.

Saket is majoring in Computer Engineering while focusing his course work in embedded systems design. He has taken ESE 380/381 (Embedded Systems Design I & II), ESE 224 (Problem Solving using C++) and CSE 114 (Java 1). He has also worked as a teaching assistant for ESE 380. He has experience designing embedded systems using microcontrollers as well as Arduino and Raspberry Pi development boards. He runs Arduino tutorials for Freshman students at the Institute Of Electrical and Electronics Engineers (IEEE) Student Branch at Stony Brook.

Tochukwu is majoring in Computer Engineering with a minor in Computer Science. He has focused his course work in embedded systems, and system programming. Tochukwu has taken CSE 114/214/219 (Java 1,2,3) which have sharpened his Java language skills, ESE 124/214 and CSE 376 (Intro C, Problem Solving using C++, and Advance C/Unix) which has helped Tochukwu understand the concepts of probability, program robustness, and modular design, and ESE 380/381 (Embedded System Design), and CSE 308 (Software Engineering).Tochukwu has worked as a teaching assistant for CSE 114, and developed a video game presentation application for FSA..

One challenge we face is bridging the gap between astrophysics and engineering. Given our lack of knowledge about cosmic rays, we must treat the cosmic ray detector and its associated coincidence shield as black boxes. This approach, although practical, may limit our ability to further customize the system (e.g. provide an arduino-compatible alternative to the coincidence shield). We will further explore the importance of cosmic rays, time permitting.

4.3. Professional/Ethical considerations

Because our project is largely software based, we must take software licensing issues into consideration. Most of the open-source APIs we will be using are released under the Apache 2.0 license, which allows us to freely modify, distribute and even sell software making use of them. We must, however, make sure to include the appropriate copyright and licensing documentation so that our project complies with the licensing agreement. We must also state any changes made to the original codebase.

4.4. Impact of project on Society/Environment

Our project does not have any immediate foreseeable effects on society. It is academic in nature, and is intended to provide Dr. Takai and Brookhaven National Lab with a practical, easy to use, reproducible system which will assist them in their larger research goals. In the long run, accurate detection of cosmic rays and ambient data will hopefully assist BNL in predicting astronomical phenomena, leading to more practical applications of the collected data.

Section 5: Conclusions

The system we've developed successfully meets the desired specifications outlined in the project description. The Yún successfully records the data it receives from the Yahoo Weather and the Coincidence Board to a file that is written to the micro SD card. At the end of the day the file is uploaded to Dropbox via python script.

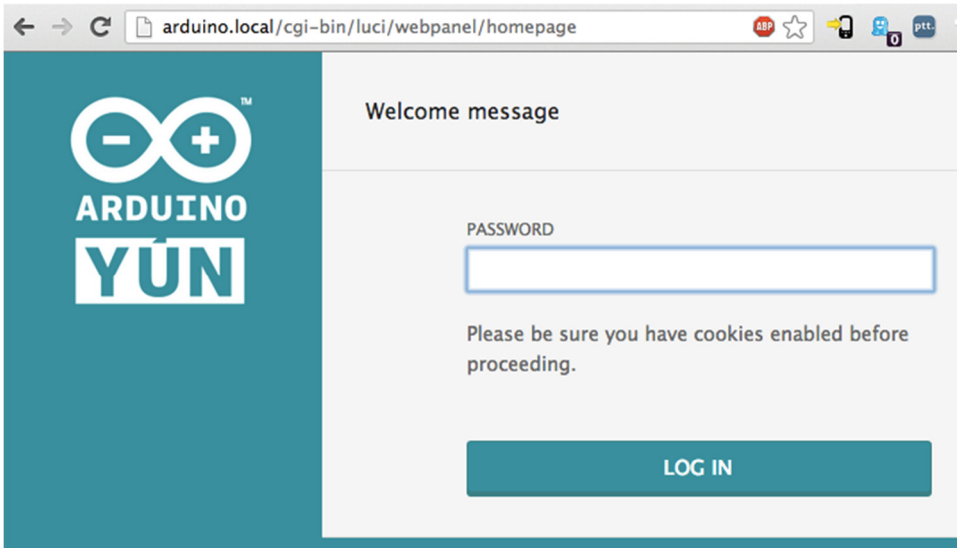
The Android application works as outlined in the projection specifications. The application successfully links to Dropbox, fetches and caches the data on the client device, and graphs it in a variety of forms. The app is designed to filter out invalid or un-related Dropbox files, to provide a robust graphing solution which complements the Arduino-based system.

Appendix – Guide to using the Yún

Contents

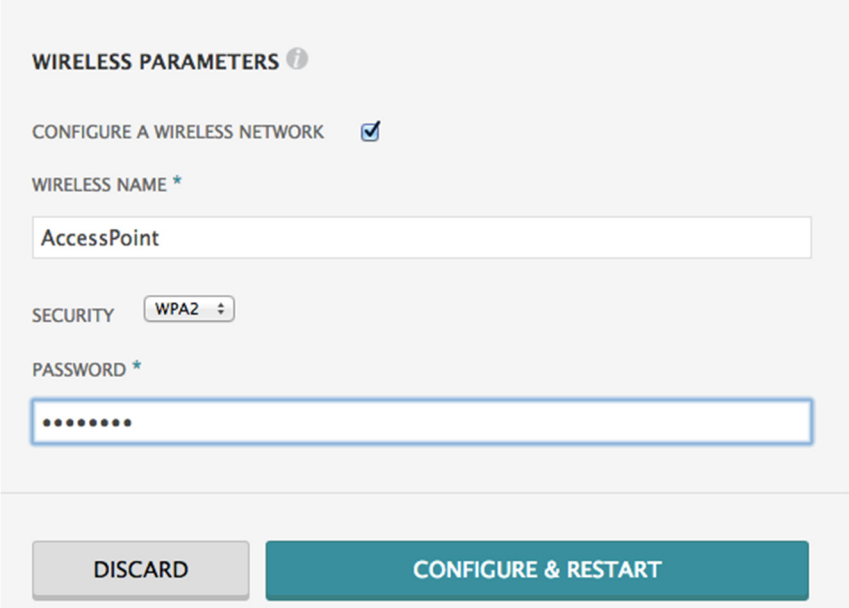
1. Configuring the onboard WiFi

- When you first power up the Yun, it creates its own WiFi network called *ArduinoYun-XXXXXXXXXXXX.* Connect your computer to this network.
- Once you have obtained an IP Address, open a web browser and enter, *http://arduino.local* in the address bar. After a few moments, a web page will appear asking you for the password. Enter the password and click *Log In* button. (The default password is "*arduino*").



Arduino Yun web page Welcome screen

- On the next page you will see some information about the Yun's network connections. Click on the "*Configure*" button.
- The next page you see will allow you to change the name of your Arduino Yun, the password, the time zone and network settings.

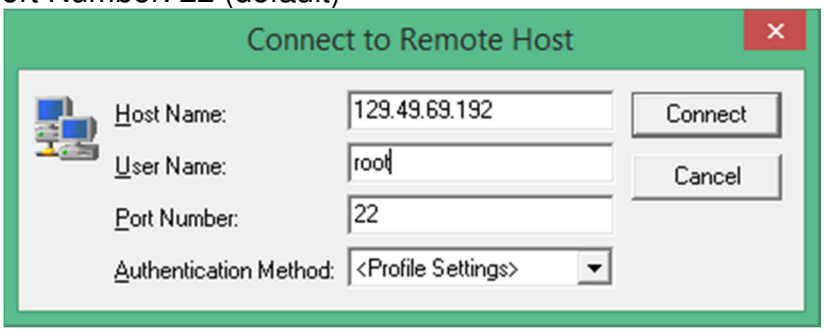


Arduino Yun Wireless Parameters

- Under the wireless parameters section, select the wifi network you wish to connect to. Select the security type and enter the password.
- When you click "*Configure & Restart*", the Arduino will restart itself and connect to the wireless network you specified. The *ArduinoYun-XXXXXXXXXXXX* will shut down after a few moments and you can connect to the wireless network that the Yun is on.

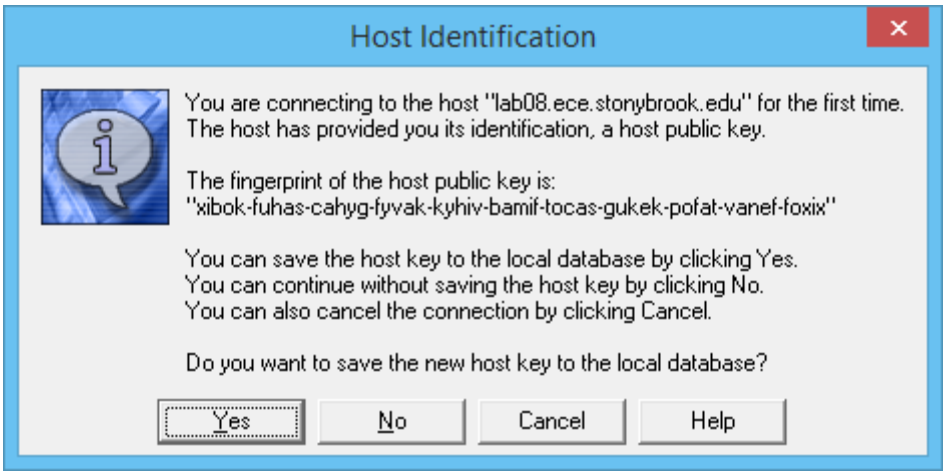
2. Connect via SSH from Remote Network

a) Download a SSH client like PuTTY or Secure Shell. These instructions use Secure Shell.
b) Open Secure Shell and click the "*Quick Connect*" button.
c) Enter the following information and click "*Connect*":
   - Host name: 129.49.69.192
   - User Name: root
   - Port Number: 22 (default)



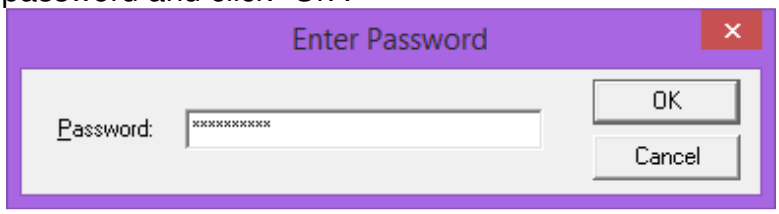
Remote Connection via SSH

d) If the "Host Identification" window pops up, click "*Yes*".



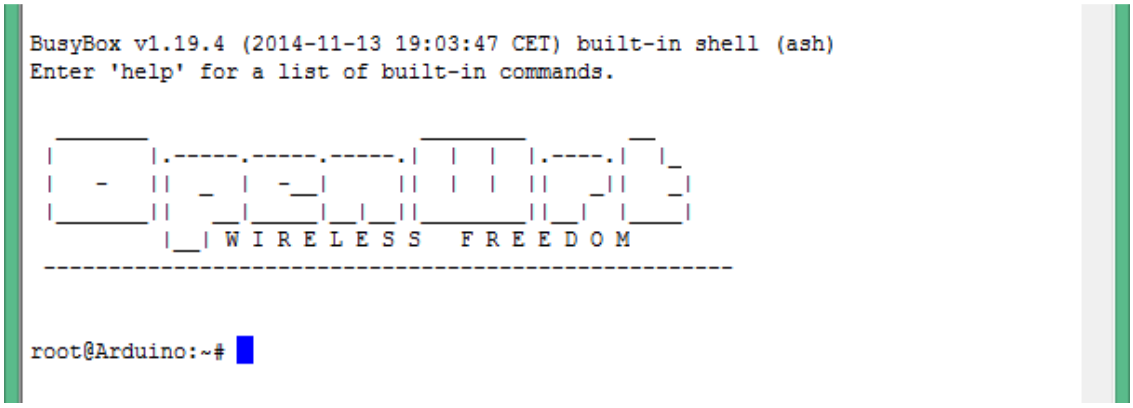
Host Identification Check

e) Enter the password and click "*Ok*".



Password

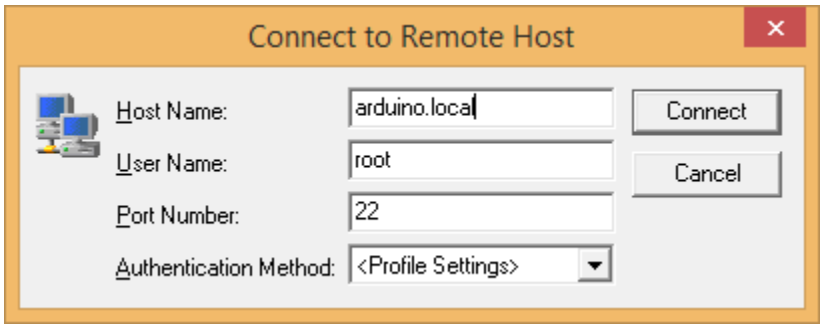
f) When you see the following screen, you are connected to the Arduino Yun.



SSH Window upon initial connection

3. Connect via SSH from Local Network

- Follow the steps as in the previous section. In step c, for the Host Name, enter "*arduino.local*".



Connect via Local Network

4. Change the Address for weather information

a) Once you are connected to the Arduino Yun via SSH. Type ***cd ../mnt/sda1/*** to navigate to the appropriate directory and press Enter. (Do not forget the space between cd and '..'):

b) Type ***ls*** to view the files in the current directory. You will be editing the *TembooSettings.txt* file to change the address.

c) Type ***vi TembooSettings.txt*** and press Enter. This will display the contents of the file in a text editor.

```
root@Arduino:~# cd ../mnt/sda1/
root@Arduino:/mnt/sda1# ls
System Volume Information
TembooSettings.txt
YunLogToDropbox
data
date.txt
openwrt-ar71xx-generic-yun-16M-squashfs-sysupgrade.bin
testData
root@Arduino:/mnt/sda1# vi TembooSettings.txt
```

Edit the TembooSettings.txt file

d) Press ***i*** to out the text editor to insert mode.

e) Using the arrow keys, navigate to the line that says "*-iAddress:11790*".

f) Backspace to erase the existing zip code and enter the new zipcode (or address in the format of *Street, City, State*).

g) Press ***esc*** to get out of insert mode.

h) Type ***:wq*** to save the changes and quit the text editor.

```
-acosmicraybnl
-useniorDesign
-pa685ed5ee7394953a6cb40d0f26110d9
-c/Library/Yahoo/Weather/GetWeatherByAddress
-iUnits:c
-iAddress:11790
-otemperature:/rss/channel/item/yweather:condition/@temp:Response
-opressure:/rss/channel/yweather:atmosphere/@pressure:Response
-ohumidity:/rss/channel/yweather:atmosphere/@humidity:Response
```

Changing the address parameter

5. **Manually display data from a text file**
   a) When you are in the */mnt/sda1/* directory, type ***cd data*** to navigate to the directory where all the data files are saved.
   b) Type ***ls*** to view the files in this directory.
   c) To print the data from a particular file to the screen, type ***cat filename.txt***.

```
root@Arduino:/mnt/sda1# cd data
root@Arduino:/mnt/sda1/data# ls
05_09_15.txt  05_11_15.txt  05_13_15.txt  05_15_15.txt
05_10_15.txt  05_12_15.txt  05_14_15.txt
root@Arduino:/mnt/sda1/data# cat 05_14_15.txt
05_14_15 00:01:03, 7940,     16,      46,     1019.3
05_14_15 00:06:03, 7980,     16,      46,     1019.3
05_14_15 00:11:03, 7932,     16,      46,     1019.3
05_14_15 00:16:03, 7990,     16,      46,     1019.3
05_14_15 00:21:03, 7770,     16,      46,     1019.3
05_14_15 00:26:03, 7905,     16,      46,     1019.3
05_14_15 00:31:03, 7724,     16,      46,     1019.3
05_14_15 00:36:04, 7969,     16,      46,     1019.3
05_14_15 00:41:04, 7844,     16,      46,     1020.5
```

Data collected on 05/14/15

6. **Setting Up the Python Script for different Dropbox Application Accounts**
   a) Once you are connected to the Arduino Yun via SSH. Type ***cd ../mnt/sda1/YunToDropbox*** to navigate to the appropriate directory and press Enter. (Do not forget the space between cd and '..')
   b) Type **python logscript.py** and press enter this will run the python script which will create an emtpy drop_auth.cfg file. This is the configuration file for python script uses to connect with dropbox.

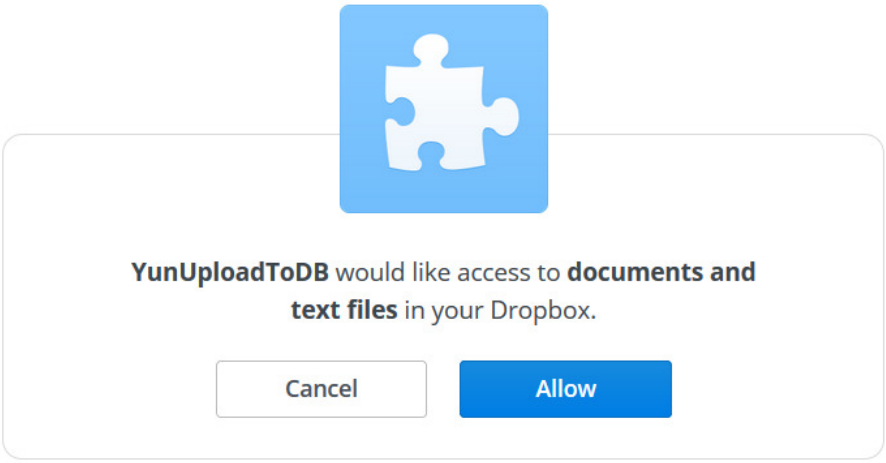
```
root@Arduino:/mnt/sda1/YunLogToDropbox/test# python logscript.py
```

   c) Type **vi drop_auth.cfg** and press enter. This will display the contents of the drop_auth.cfg file.

```
[dropbox]
app_secret =
app_key =
access_token =
```

   d) Press **i** on keyboard to set the editor to insert mode.
   e) Nagivate to the lines "app_secret = " and enter the app secret for the dropbox application. (Cosmic Ray BNL app_secret = 2v6dfjpufus3snz).
   f) Nagivate to the lines "app_key = " and enter the app key for the dropbox application. (Cosmic Ray BNL app_secret = zu45xlfjdx94lt).
   g) Press **esc** to get out of insert mode.
   h) Type **:wq** to save the changes and quit the text editor.ls

i) Type **python logscript.py** and press enter, This time the python script will authenticate the Yun with Dropbox.
j) Copy the URL to your web browser or CRTL + left click on the URL and authenticate the app.



k) Copy the key back into terminal and press enter. Once done the setup of the Python script is complete.