**COSC 6397: Cyber Physical Systems**

**Project Report**

**Sujeet Kuchibhotla**

**Nripun Sredar**

**Shaunak Vairagare**

**May 2011**

**Department of Computer Science**

**University of Houston**

**Overview:**

Arduino Fio is a microcontroller board intended for wireless applications. It has 14 digital I/O pins and 8 analog inputs. The main aim of this project is to attach sensors like temperature sensors, soil moisture sensors and other such analog or digital sensors to the Arduino Fio boards and be able to get values from these sensors to a control station wirelessly. The programming of the Arduino Fio boards is also done wirelessly. Many such Arduino Fio boards can thus be deployed to create a network of sensor stations which communicate wirelessly with the control station.

**Programming Arduino:**

Arduino can be programmed in two ways as shown below:

1. Preburned with bootloader
   * Two ways to upload sketches( s/w written using Arduino)
     1. FTDI – SUB cable
     2. Wirelessly over pair of XBee radios
2. Bypass the bootloader and program the ATmega328 with an external programmer

We have programmed the Arduino Fio wirelessly in this project. The configuration steps needed to start wirelessly programming the Arduino Fio using XBee’s are listed in the next section.

**Configuring XBee:**

The typical setup for this project is a control station with an XBee unit and a remote Arduino Fio with a XBee unit. The control station XBee unit is used to wirelessly program the remote Arduino board. These XBee’s are the only communication channel. Both the XBee units viz. the control station XBee and the remote XBee need to be configured differently so that they can communicate with each other. This configuration can be done using the FioXBeeConfigTool. This tool can be obtained at the following URL:

<http://funnel.googlecode.com/files/XBeeConfigTool.zip>

The configuration steps are listed below:

1. Connect the Programming Radio to the PC using the XBee adapter
2. Install the driver
3. Go to Device Manager. Right Click USB COM Port 9 and select properties. Select Port Settings Tab. Click Advanced. Check mark Set RTS On Close.
4. Open FioXBeeConfigTool
5. Select COM9
6. Select Programming radio
7. Select 57600 Baud rate
8. Set the PAN ID to 1234
9. MY ID – 0000
10. DL ID – FFFF

Click Configure/ **Message : Configured Successfully**

1. Connect the Arduino Fio Radio to the PC using the XBee adapter
2. Open FioXBeeConfigTool
3. Select COM9
4. Select Arduino Fio radio
5. Select 57600 Baud rate
6. Set the PAN ID to 1234
7. MY ID – 0001
8. DL ID – 0000
9. Click Configure/ **Message : Configured Successfully**

**Sensors:**

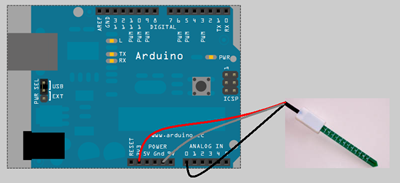
Arduino Fio supports both digital inputs and Analog inputs. We have used two analog sensors for our purpose. We have used an analog temperature sensor and a moisture sensor to connect to the Arduino Fio board.

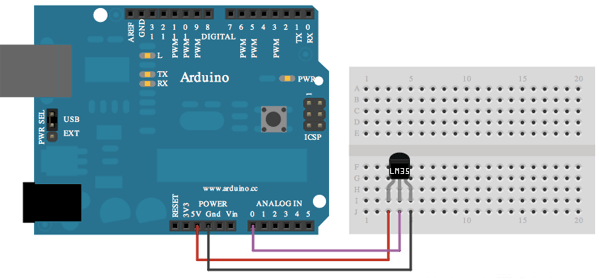
Temperature Sensor: LM35Z (<http://www.national.com/mpf/LM/LM35.html#Overview>)

Moisture Sensor: VG400 ([http://www.vegetronix.com](http://www.vegetronix.com/))

**Circuit Diagrams:**

The sensors are connected to the Arduino Fio board as shown in the figures below:





**Arduino Programming:**

Arduino can be programmed wirelessly by the following steps:

1. Open Arduino IDE
2. Select Tools/Board/Arduino Fio
3. Select Tools/Serial Port/ COM9 (Should be the same port used while configuring XBee)
4. Go to File/Examples/Basics->Blink (Or any other custom sketch)
5. Change the delay of the blink duration
6. Click Save. Compile. Upload. If upload is successful, the message “Done uploading” is displayed.

**Program:**

There are three software modules in this project.

* + - 1. Sketches flashed onto the Arduino Fio board (Arduino program)
      2. Script which communicates with Arduino Fio board (Ruby script)
      3. Web-app that displays associated data content (Ruby on Rails app)

**Arduino Program:**

This program is flashed on to each Arduino Fio Board. The program reads input on the serial port and upon receiving appropriate query input, it sends out requested data over the serial port. The program expects input in a pre-defined format as shown below:

Command Format used to query Arduino:

Command : $#1#temp#moist@ --> Arduino with id #1 will return temperature and moisture in format $#1#23#513@

Command : $#1#temp@ --> Arduino with id #1 will return temperature in format $#1#23@

Command : $#2#moist@ --> Arduino with id #2 will return moisture in format $#1#513@

Temperature is returned in Celsius and moisture is returned in units ranging from 0 to 1024 depending on degree of saturation.

**Ruby Script:**

The Ruby script is used to send queries to the Fio boards to obtain sensor data from that board. The command format for the queries is as shown in the previous section.

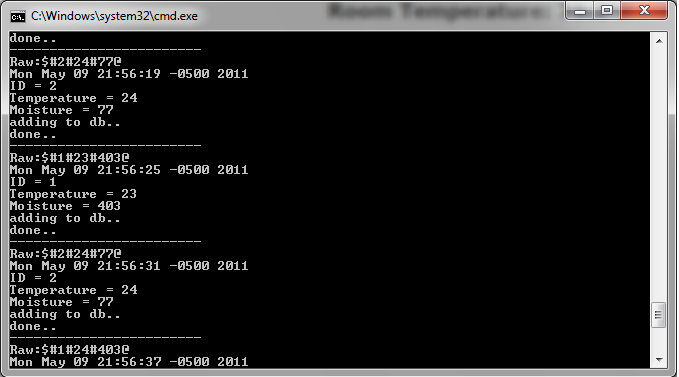
**Ruby on Rails App:**

The Web-app enables one to add a sensor station (Fio Board). It also provides a dashboard for each sensor station which shows current sensor readings along with the graphs for each sensor.

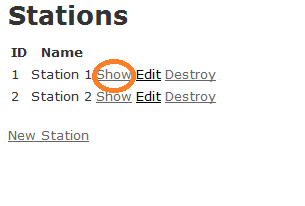
**Program Output Snapshots:**

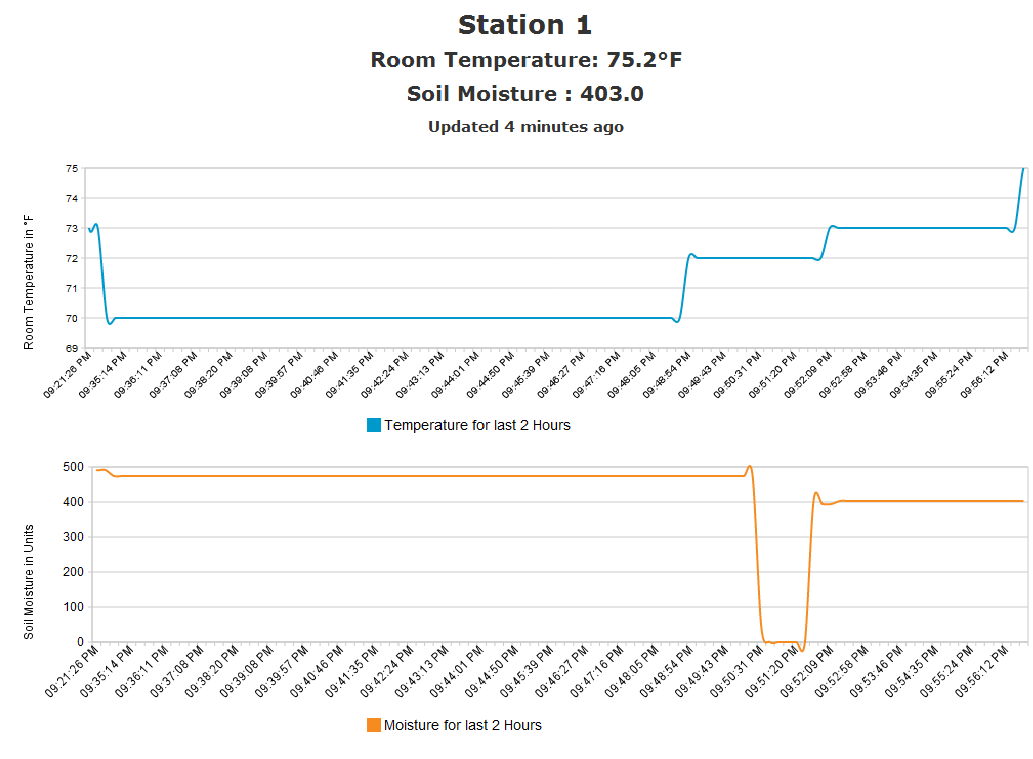
The Arduino program that is uploaded on each Fio board enables one to query individual Fio boards from a controller station using the ruby script. The web-app displays data obtained from the sensors associated with each Arduino board within a browser. The web-app displays historical data which it retrieves from a database and has graphical output too along with sensor data display.

Script output:



Web-app output:



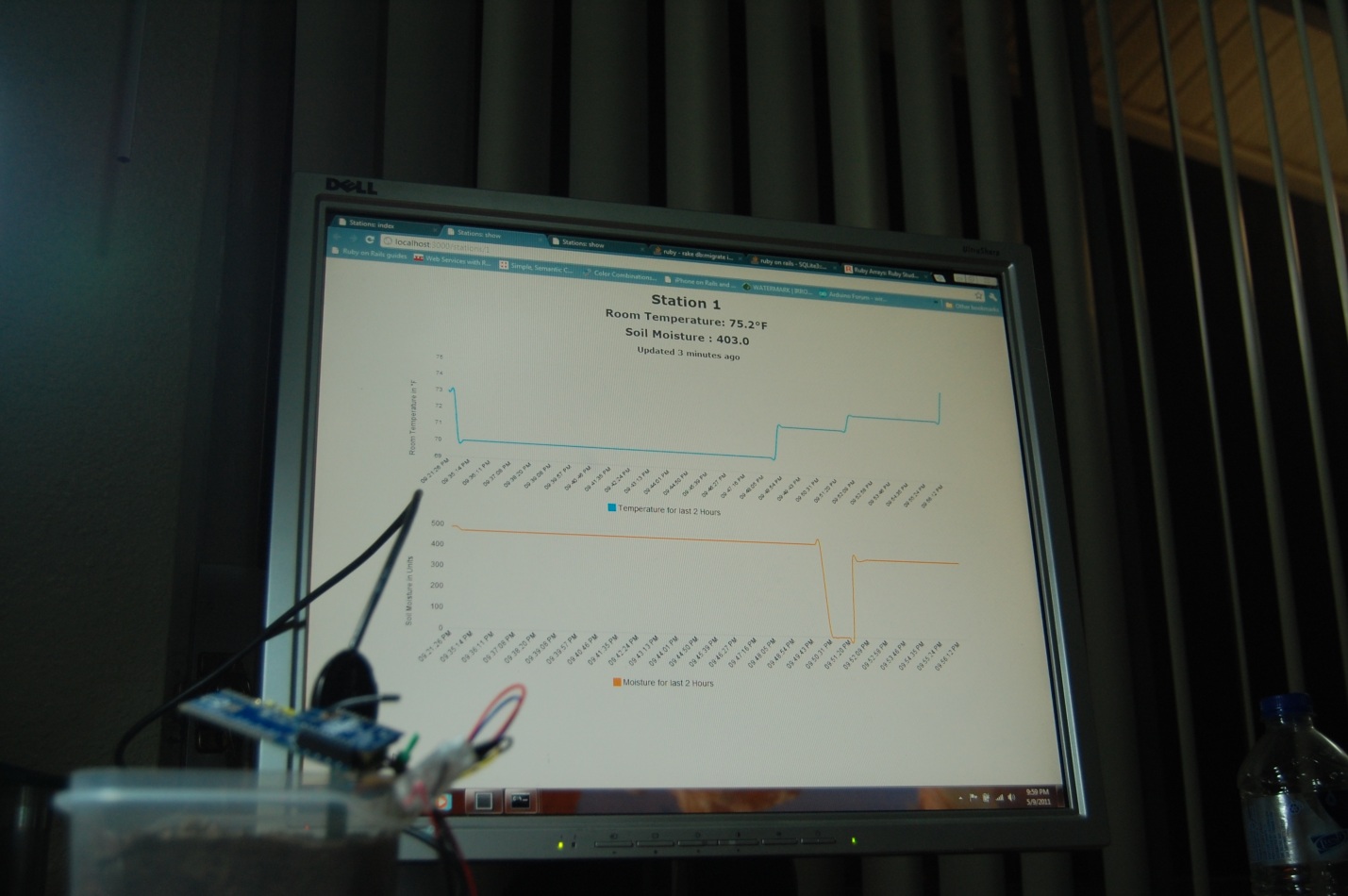
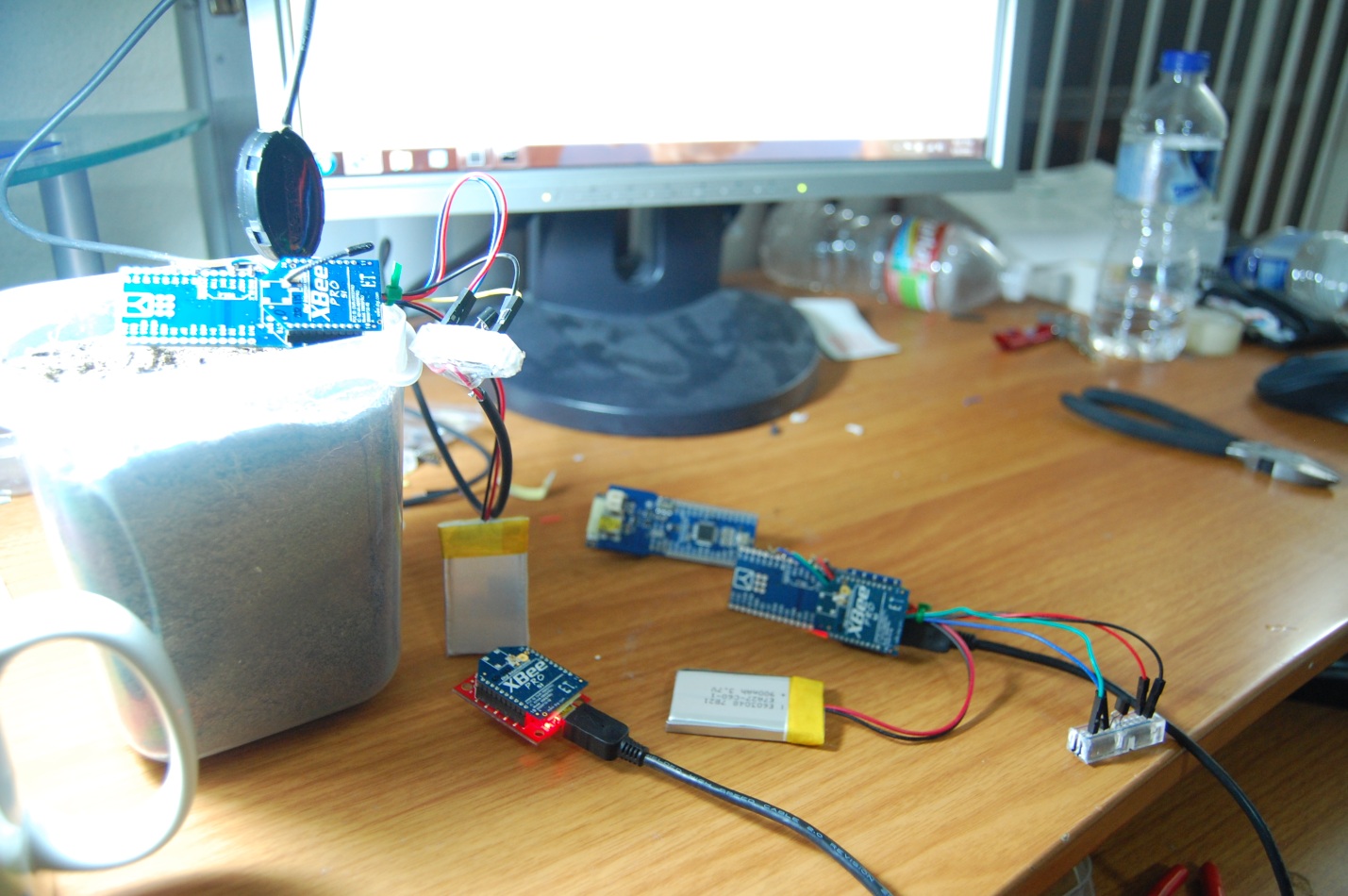
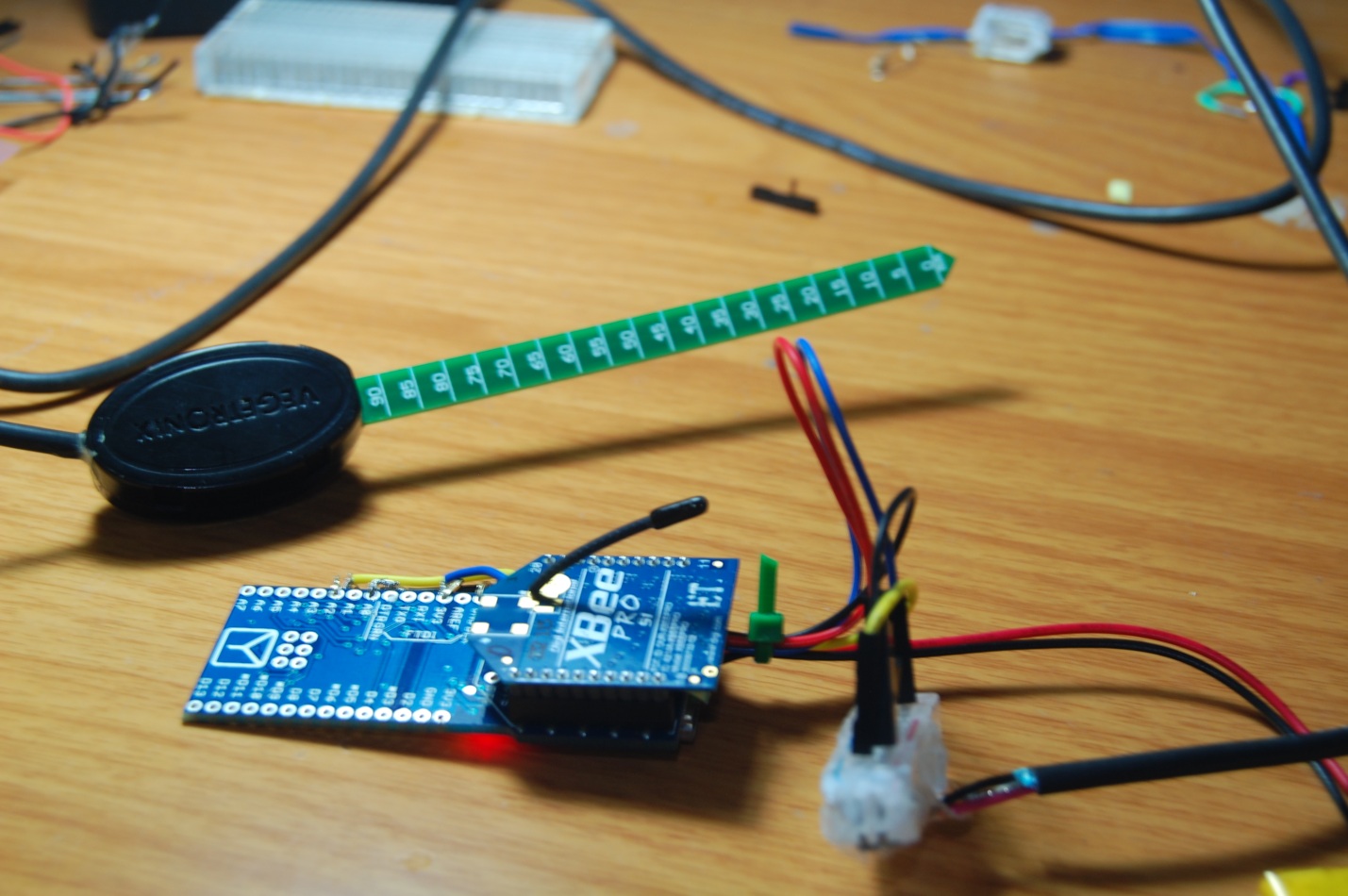
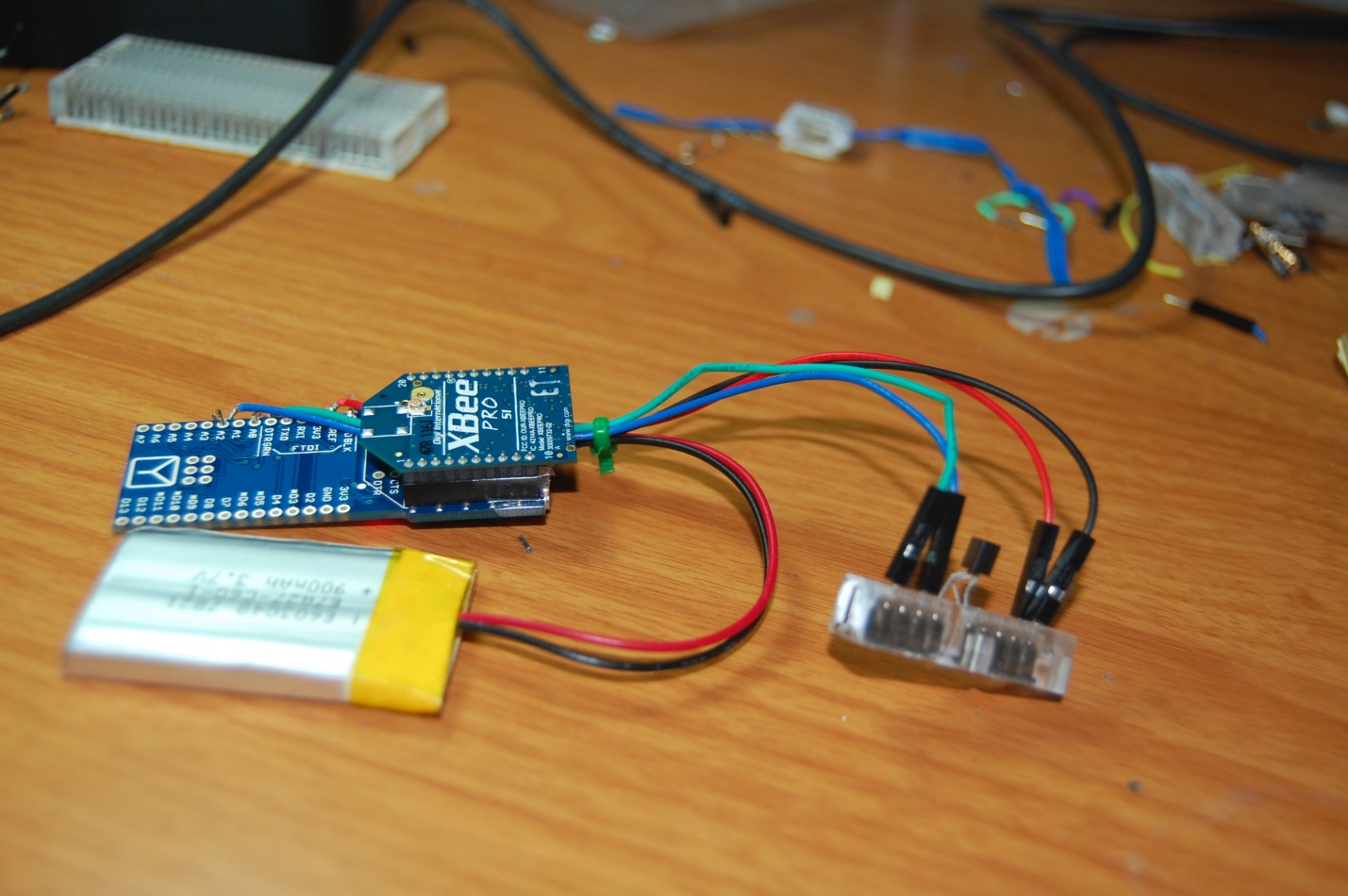


**Software Installations required on Windows:**

|  |  |
| --- | --- |
| |  | | --- | | 1. Download Arduino IDE (<http://www.arduino.cc/en/Main/Software>)  2. Download and Install Ruby 1.8.7 (<http://rubyinstaller.org/downloads/>)  3. Install Rails (on command prompt type: 'gem install rails -v 2.3.11' )  4. On Command Prompt type : 'gem install seer'  5. On Command Prompt type : 'gem install serialport'  **Instructions for the Program:**  1. Burn the sketch(in Sketches folder) into Arduino Fio using the Arduino IDE  2. Extract ruby script into 'arduino' folder.  3. To run the data acquisation script : run    "arduino/app/jobs> ruby collect\_data.rb  4. To run the Web GUI Application : run   "arduino> ruby script/server "  5. goto <http://localhost:3000/stations> | |

**Extensions:**

This project uses multiple XBee’s communicating in transparent mode. This can be further extended by making use of the API mode for Xbee. This mode enables packet based delivery and acknowledgment protocols ensuring robust data communication. Analog inputs can be directly wired to the Xbee station thus leaving the pins on the Arduino free for more inputs.

**Random Project Snaps:**