Dust Sensing Project

Table of Content

28 November 2010

Introduction

The dust sensor project aims to build an affordable air quality wireless sensor node prototype. It can work in two modes: online and offline. In online mode, the dust sensor periodically sends the dust level to receiver; in offline mode, the dust sensor stores the dust level periodically to its internal memory, and can be collected when needed.

This document explains not only how to setup server, but also how to re-produce the dust sensor. To find more information about this project and obtain the latest version of this document, please visit http://dust.sensorapp.net/.

Contacts

Project Leader	Dr. Shanika Karunasekera	http://ww2.cs.mu.oz.au/~shanika/
Research Fellow	Dr. Sutharshan Rajasegarar	http://people.eng.unimelb.edu.au/sraja/
Research Assistant	Paul Peng Deng	http://resume.sensorapp.net/

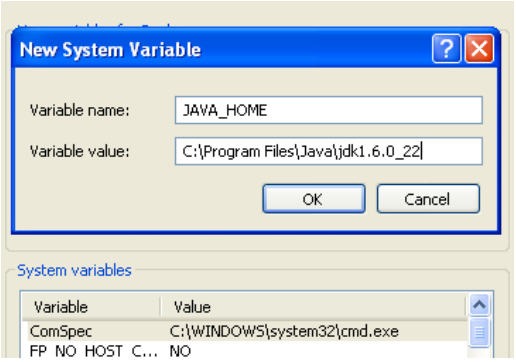
How to Use

Prerequisite Software

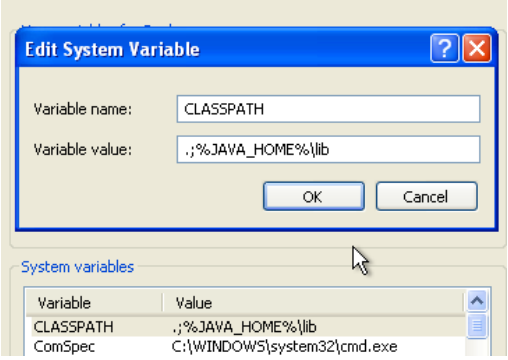
- Operating system: Windows (32bit version) with administrator privilege login
- Java Development Kit (JDK): http://www.oracle.com/technetwork/java/javase/downloads/index.html
- LiveGraph: http://www.live-graph.org/download.html
- Sun SPOT SDK Version: Yellow-101117-1
- Internet access

Install JDK

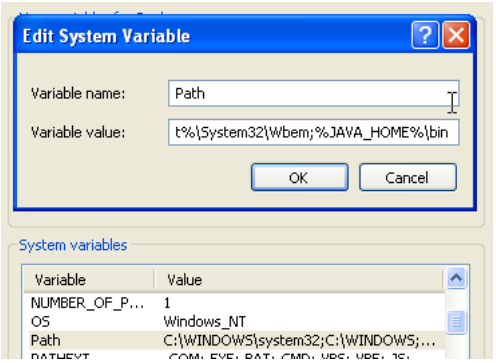
1. Install Java Development Kit with all default settings
2. Go to Control Panel → System → Advanced → Environment Variables
3. Click the New button below System Variables and fill in JDK path to create a new variable. For example, I have my JDK installed in C:\Program Files\Java\jdk1.6.0_22.



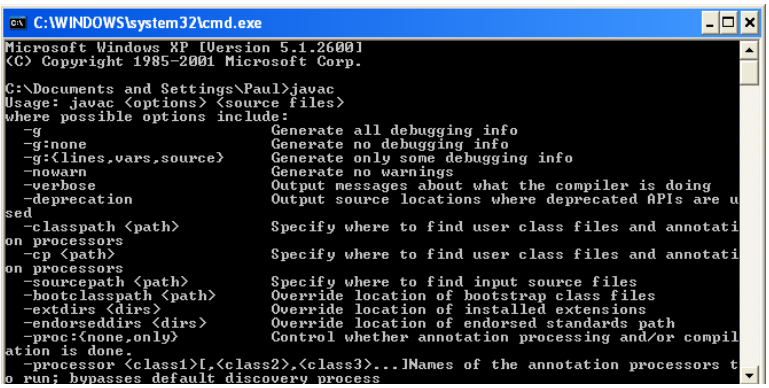
4. Click the New button below System Variables and fill in .;%JAVA_HOME%\lib



5. Find and double click Path variable. Append ;%JAVA_HOME%\bin



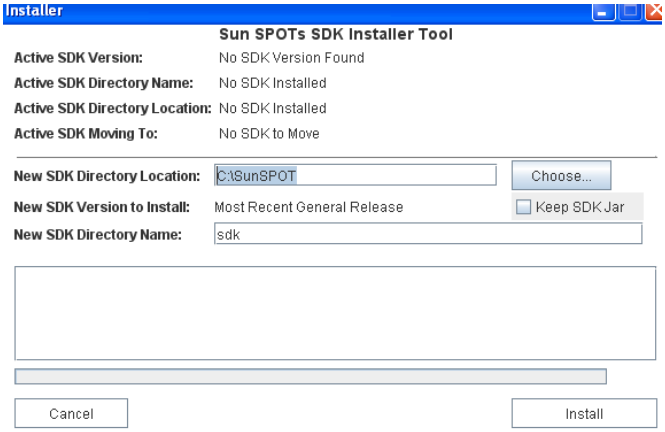
6. Click OK to save these changes and close this window
7. To verify if the configuration is correct, please open a console window and enter command javac, if you can see some print out like this. It means all good.



Install Spot Manager

1. Start your browser, go to http://www.sunspotworld.com/spotmanager
2. Click the big icon to download and run SpotManager
3. SpotManager will start to check if you have everything needed
4. Do **NOT** install NetBeans
5. Install Ant
6. Do **NOT** install NetBeans Modules

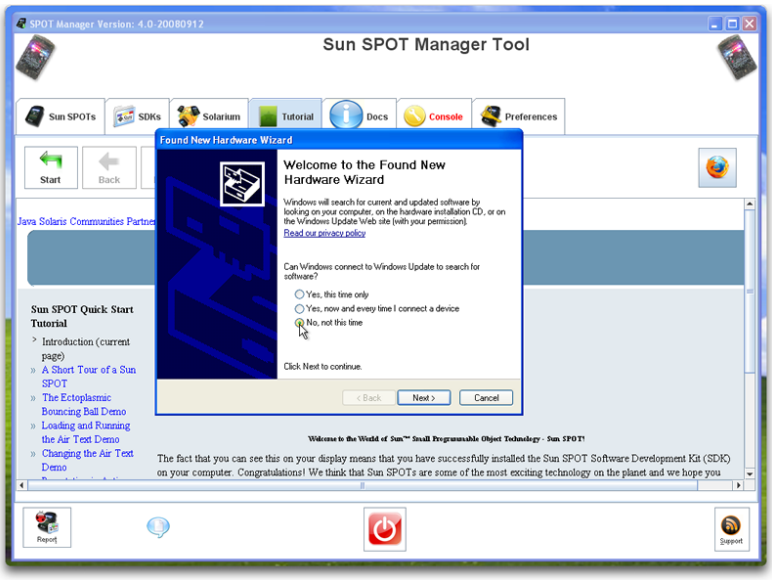
7. Install the Sun SPOTs SDK to C:\SunSPOT



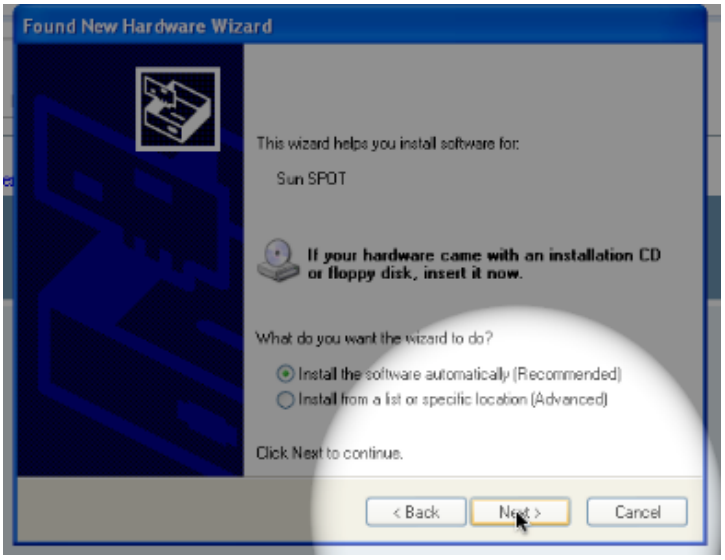
8. Once installation is done, reboot your pc
9. Run Spot Manager again, change to SDKs tab and check if you have the right version of the SDK (Yellow-101117-1)

Plug in Sun SPOT base station

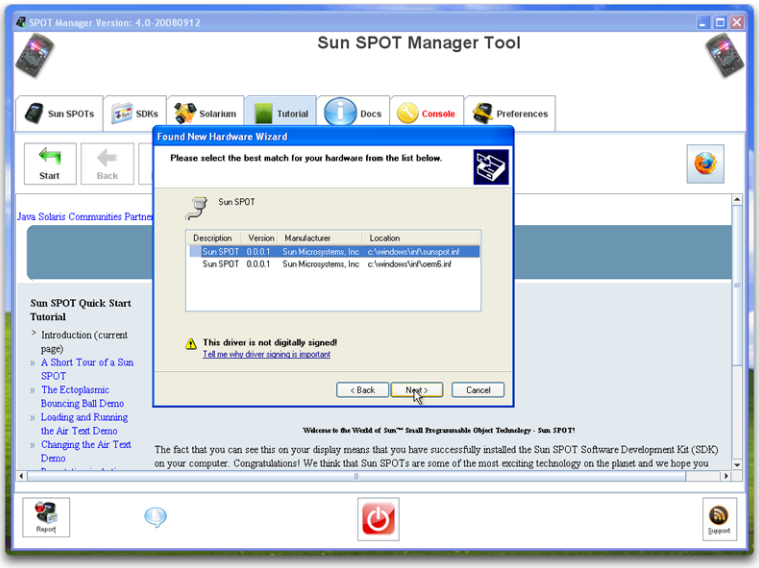
1. Windows asks for Sun SPOT driver, select No, not this time and Next



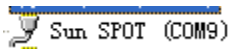
2. Select Install from a list on specific location (Advanced) and Next



3. Windows should find the proper driver for Sun SPOT

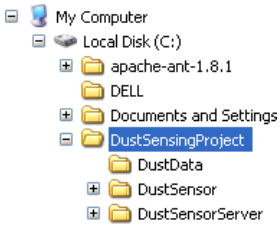


4. Go to Control Panel → System → Hardware → Device Manager, write down the Sun SPOT COM port number, we need to use it later. In my computer, system assigns **COM9** to Sun SPOT device.



Install Server Application

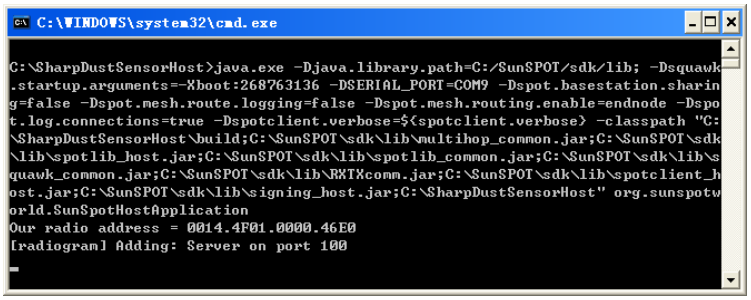
1. Download Server Application from http://dust.sensorapp.net/
2. Extract the ZIP archive to C:\, the file structure should be



3. Open Notepad to edit StartServer.bat in C:\DustSensingProject\DustSensorServer\DustSensorHost, replace the string -DSERIAL_PORT=**COM??** with -DSERIAL_PORT=**COM9** which is Sun SPOT COM port
4. Save and close Notepad

Run Dust Sensor Server

Double click the StartServer.bat, you should see a screen like this



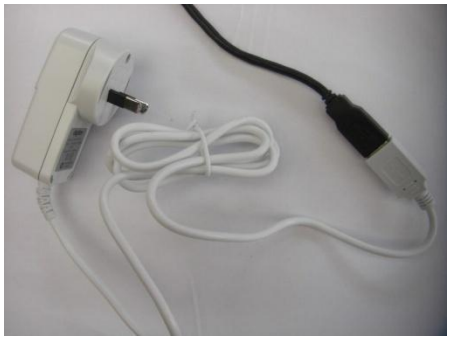
If you came across an error, please check:

a. the bases station is connected,
b. the COM port is properly set,
c. logged in as system administrator,
d. extracted Zip file to the right path C:\DustSensingProject,
e. installed Sun SPOT SDK version yellow-101117-1 to C:\SunSPOT

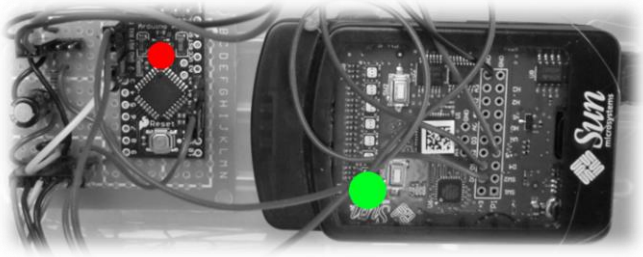
or contact me on pdeng@sensorapp.net

Run Dust Sensor

Connect the USB cable to power adapter, the sensor should up and running



After connected, wait about 10 seconds, the **RED** led should be always on and **Green** led flashes periodically when it is sensing.



If not, please reset dust sensor.

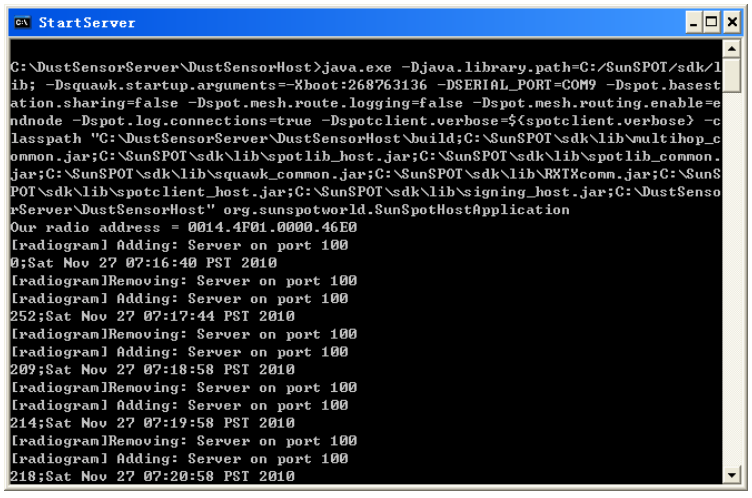
Collect Data

NOTE: the Dust Sensor Server is not designed to collect data from multiple dust sensors simultaneously. It means you need to make sure only the sensor you are working on is power on and active, others are disconnected from power and switched off.

The data is stored in C:\DustSensingProject\DustData, any text editor can open and edit them.

From Online Sensor

When active dust sensor is running online mode, server will immediately print out and save to data file once message is received.



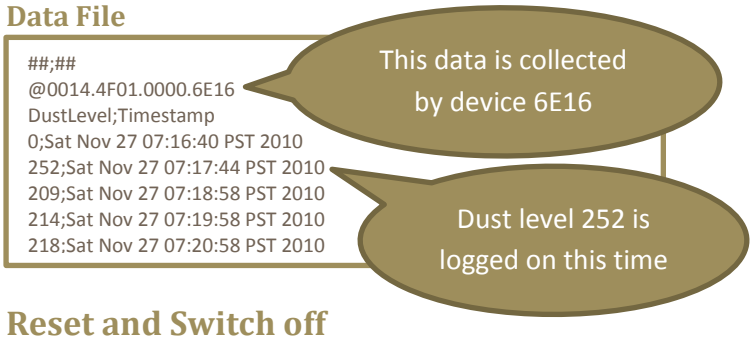
From Offline Sensor

If the active dust sensor is running offline mode. You need to do some manual operate:

- **Left button:** pause current sensing task and send off all stored data.
- **Right button:** DELETE stored data permanently

Reset Sun SPOT to resume.

Data File

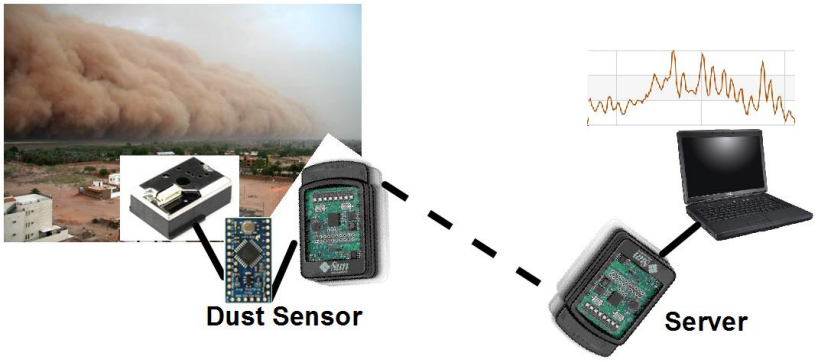


Reset and Switch off

To reset a dust sensor, please press the button here below

To switch off a dust sensor, please disconnect USB power, then press and hold the button for 5 seconds until the led near the button flashes red twice.

Architecture



- ***Dust Sensor:*** collect dust density level
- ***Server:*** receive dust density level sent from dust sensor

Dust Sensor Design

Sun SPOT is the main controller of this sensor node. It collects dust density in the air from the Sharp dust sensor via Arduino. The Sharp dust sensor requires a pulse signal to drive the internal LED, Sun SPOT can not generate such signal but Arduino can.

Components

Sun SPOT

Sun SPOT (Sun Small Programmable Object Technology) is a open source wireless sensor network (WSN) mote developed by Sun Microsystems. The device is built upon the IEEE 802.15.4 standard. Unlike other available mote systems, the Sun SPOT is built on the Squawk Java Virtual machine.

Data Sheet: http://www.sunspotworld.com/docs/Red/SunSPOT-TheoryOfOperation.pdf
Can be purchased from: http://www.sunspotworld.com/products/index.html

Processing

- 180 MHz 32 bit ARM920T core - 512K RAM - 4M Flash
- 2.4 GHz IEEE 802.15.4 radio with integrated antenna
- AT91 timer chip
- USB interface

Sensor Board

- 2G/6G three-axis accelerometer

- Temperature sensor
- Light sensor
- 8 tri-color LEDs
- 6 analog inputs
- 2 momentary switches
- 5 general purpose I/O pins and 4 high current output pins

Battery

- 3.7V rechargeable 750 mAh lithium-ion battery
- 30 µ A deep sleep mode
- Automatic battery management provided by the software

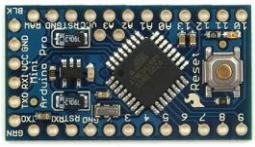
Open Source

Note! The current implementation uses hardware version v5. Oracle is upgrading hardware design to v8. Please select the right version when you download from source repository.

- Project homepage: https://spots.dev.java.net/
- Hardware Schematics: https://spots-hardware.dev.java.net/
- Firmware: https://squawk.dev.java.net/

Arduino Pro Mini

Arduino is an open-source electronics prototyping platform, designed to make the process of using electronics in multidisciplinary projects more accessible. The hardware consists of a simple open hardware design for the Arduino board with an Atmel AVR processor and on-board I/O support. The software consists of a standard programming language and the boot loader that runs on the board.



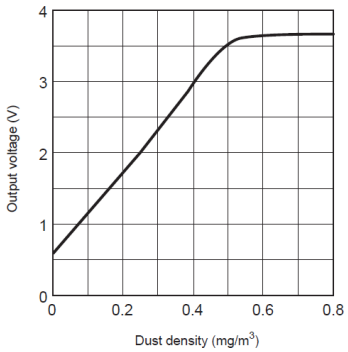
5V 16MHz model

Data Sheet: http://www.arduino.cc/en/Main/ArduinoBoardProMini
Can be purchased from: http://www.littlebirdelectronics.com/products/Arduino-Pro-Mini-328-%252d-5V%7B47%7D16MHz.html

Dust Sensor

Sharp's GP2Y1010AU0F is an optical air quality sensor, designed to sense dust particles. An infrared emitting diode and a phototransistor are diagonally arranged into this device, to allow it to detect the reflected light of dust in air. It is especially effective in detecting very fine particles like cigarette smoke, and is commonly used in air purifier systems.

The sensor has a very low current consumption (20mA max, 11mA typical), and can be powered with up to 7VDC. The output of the sensor is an analog voltage proportional to the measured dust density, with a sensitivity of 0.5V/0.1mg/m3.

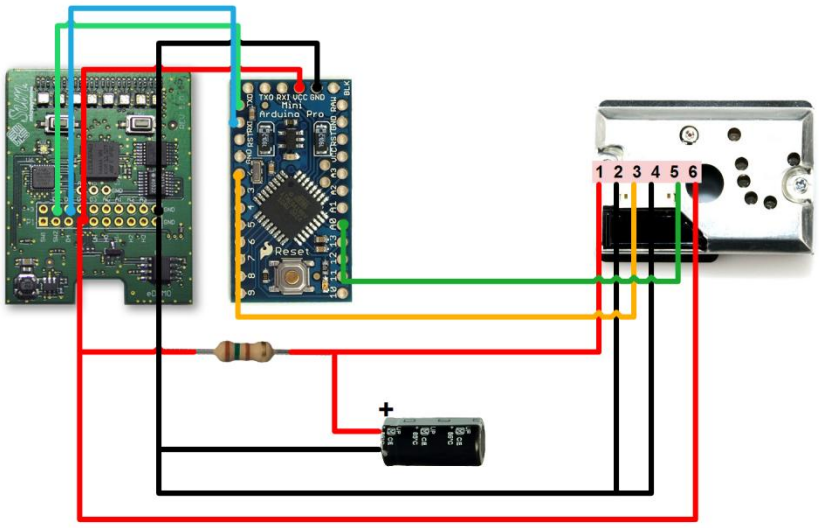


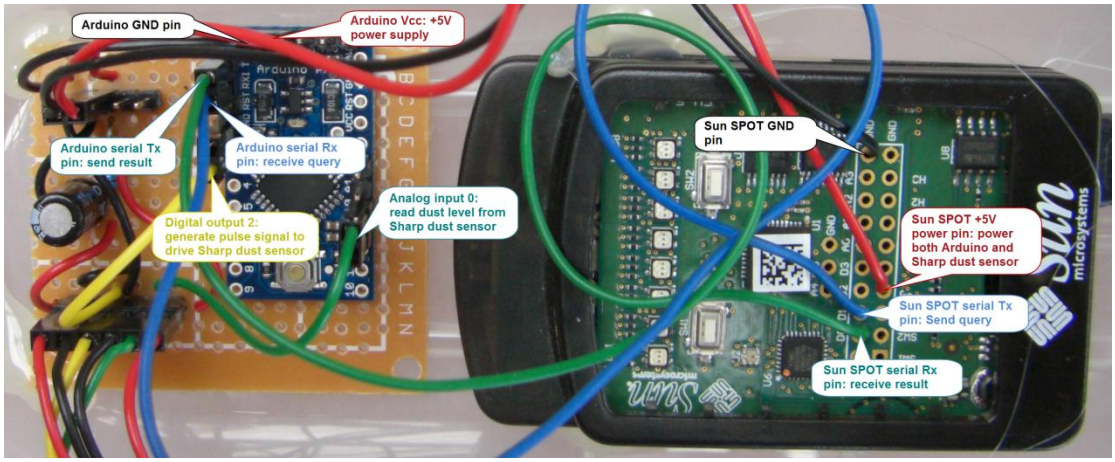
Data Sheet: http://www.sparkfun.com/datasheets/Sensors/gp2y1010au_e.pdf
Can be purchased from: http://www.sparkfun.com/products/9689
Mating connector: http://www.sparkfun.com/products/9690
Crimp pins: http://www.sparkfun.com/products/9728

Power

Data Sheet: http://bit.ly/idFnQI
Can be purchased from: Dick Smith Australia http://bit.ly/idFnQI

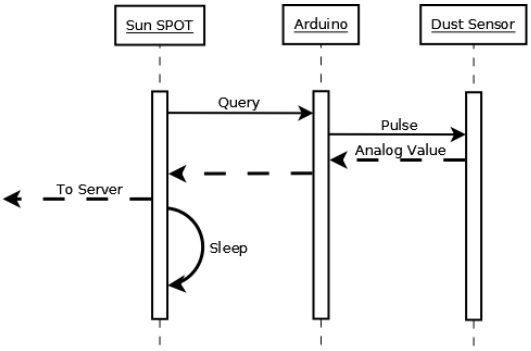
Connections



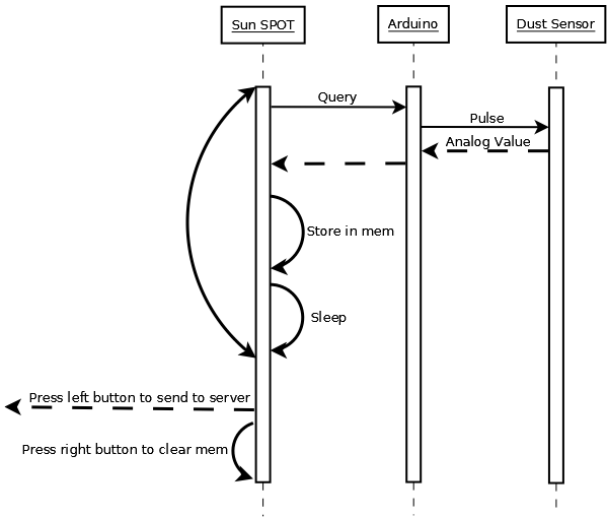


Embedded Application

Online Sensing

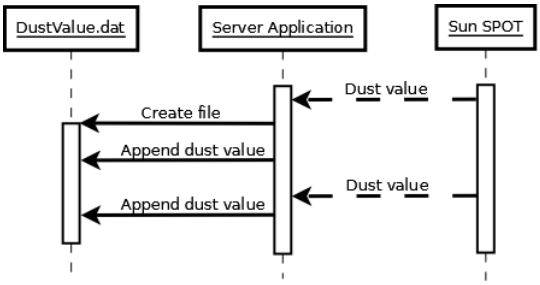


Offline Sensing

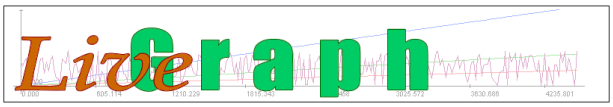


Server Design

Receiver Application

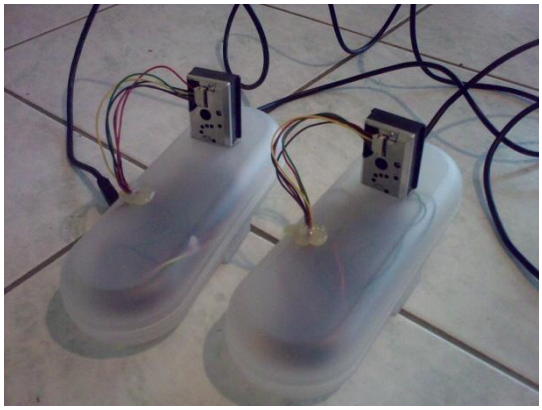


Visualization Tool



Plot graphs in real-time while the CSV data is still being generated by your application. Very easy GUI to manage 1000s of data series efficiently.

Show Off



Source Code

Visit http://dust.sensorapp.net/ to download source code and compiled binary.

Arduino Code

```
char incomingByte;
int inputPin=0;
int inputVal=0;

int ledPower=2;
int delayTime=280;
int delayTime2=40;
float offTime=9680;

void setup() {
  Serial.begin(9600);        // opens serial port, sets data rate to 9600 bps
  pinMode(ledPower,OUTPUT);
  pinMode(4, OUTPUT);
}

void loop(){

  if(Serial.available() > 0){
    incomingByte = Serial.read();
    if(incomingByte==81){ // if "Q" is received from serial
      Serial.println(readDustSensor()); // return 0-1024@5V
    }
  }
}

int readDustSensor(){
  digitalWrite(ledPower,LOW); // turn the LED on
  delayMicroseconds(delayTime);
  inputVal=analogRead(inputPin);
  delayMicroseconds(delayTime2);
  digitalWrite(ledPower,HIGH); // turn the LED off
  return inputVal;
}
```

Sun SPOT

Online

```
package org.sunspotworld;

import com.sun.spot.peripheral.Spot;
import com.sun.spot.peripheral.radio.RadioFactory;
import com.sun.spot.resources.Resources;
import com.sun.spot.resources.transducers.ITriColorLEDArray;
import com.sun.spot.resources.transducers.LEDColor;
import com.sun.spot.sensorboard.EDemoBoard;
import com.sun.spot.service.BootloaderListenerService;
import com.sun.spot.util.IEEEAddress;
import com.sun.spot.util.Utils;
import java.io.IOException;
import javax.microedition.io.Connector;
import javax.microedition.io.Datagram;
import javax.microedition.io.DatagramConnection;

import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

public class SunSpotApplication extends MIDlet {

    private int dustLevel;
    private EDemoBoard demoBoard = EDemoBoard.getInstance();
    private ITriColorLEDArray leds = (ITriColorLEDArray) Resources.lookup(ITriColorLEDArray.class);
    private int SLEEP_TIME = 5;  // sleep time in seconds

    protected void startApp() throws MIDletStateChangeException {
        Spot.getInstance().getSleepManager().disableDeepSleep();

        BootloaderListenerService.getInstance().start();   // monitor the USB (if connected) and recognize commands from host

        long ourAddr = RadioFactory.getRadioPolicyManager().getIEEEAddress();
        System.out.println("Our radio address = " + IEEEAddress.toDottedHex(ourAddr));

        demoBoard.initUART(9600, false);  // open serial port
```

```
        Utils.sleep(5000);  // give 10 seconds to dust sensor to warm up

        String returnString;
        while (true) {
            leds.getLED(0).setColor(LEDColor.GREEN);
            leds.getLED(0).setOn();
            demoBoard.writeUART("Q");  // send query command to Arduino via serial port

            Utils.sleep(20);  // give Arduino and dust sensor some time to get back

            returnString = "";

            byte[] buffer = new byte[64];
            try {
                demoBoard.readUART(buffer, 0, buffer.length);  // read Arduino serial port
                returnString = returnString + new String(buffer, "US-ASCII").trim();  // format byte value
                dustLevel = Integer.parseInt(returnString);


                /**
                 * Please apply calibration to dust level here
                 */

                // if Sun SPOT is connected to PC, you should be able to see the following printout
                System.out.print(dustLevel);  // the dust level out of 1024
                System.out.println(", " + (double) dustLevel / 205 + " volts");  // the volts output from sensor

                sendOut(String.valueOf(dustLevel));
                leds.getLED(0).setOff();

            } catch (IOException ex) {
                ex.printStackTrace();
            }

            Utils.sleep(SLEEP_TIME * 1000 - 20);
        }

    }

    protected void pauseApp() {
        // This is not currently called by the Squawk VM
    }

    protected void destroyApp(boolean unconditional) throws MIDletStateChangeException {
    }

    private void sendOut(String msg) {
        try {
            DatagramConnection sendConn = (DatagramConnection) Connector.open("radiogram://broadcast:100");
            Datagram dg = sendConn.newDatagram(sendConn.getMaximumLength());
            dg.writeUTF(msg);
            sendConn.send(dg);
        } catch (IOException ex) {
        }
    }
}
```

Offline

```
package org.sunspotworld;

import com.sun.spot.peripheral.Spot;
import com.sun.spot.peripheral.radio.RadioFactory;
import com.sun.spot.resources.Resources;
import com.sun.spot.resources.transducers.ISwitch;
import com.sun.spot.resources.transducers.ISwitchListener;
import com.sun.spot.resources.transducers.ITriColorLEDArray;
import com.sun.spot.resources.transducers.LEDColor;
import com.sun.spot.resources.transducers.SwitchEvent;
import com.sun.spot.sensorboard.EDemoBoard;
import com.sun.spot.service.BootloaderListenerService;
import com.sun.spot.util.IEEEAddress;
import com.sun.spot.util.Utils;
import java.io.IOException;
import java.util.Date;
import javax.microedition.io.Connector;
import javax.microedition.io.Datagram;
import javax.microedition.io.DatagramConnection;

import javax.microedition.midlet.MIDlet;
```

```
import javax.microedition.midlet.MIDletStateChangeException;
import javax.microedition.rms.RecordStore;
import javax.microedition.rms.RecordStoreException;

public class SunSpotApplication extends MIDlet implements ISwitchListener {

    private ISwitch sw1, sw2;     // Variables to hold the two switches.
    private ITriColorLEDArray leds = (ITriColorLEDArray) Resources.lookup(ITriColorLEDArray.class);
    private boolean loopFlag = true;
    private String dustData;
    private RecordStore rms;
    private Date time;
    private EDemoBoard demoBoard = EDemoBoard.getInstance();
    private int SLEEP_TIME = 1; //sleep time in minutes

    protected void startApp() throws MIDletStateChangeException {
        Spot.getInstance().getSleepManager().disableDeepSleep();

        BootloaderListenerService.getInstance().start();   // monitor the USB (if connected) and recognize commands from host

        long ourAddr = RadioFactory.getRadioPolicyManager().getIEEEAddress();
        System.out.println("Our radio address = " + IEEEAddress.toDottedHex(ourAddr));

        try {
            monitorSwitches();
        } catch (IOException ex) {
        }

        demoBoard.initUART(9600, false);  // open serial port
        Utils.sleep(5000);  // give 10 seconds to dust sensor to warm up

        while (loopFlag) {
            leds.getLED(0).setColor(LEDColor.GREEN);
            leds.getLED(0).setOn();
            System.out.print("Sensing and ");
            dustData = getDustLevel();
            System.out.println("storing: " + dustData);
            store(dustData);
            leds.getLED(0).setOff();

            Utils.sleep(SLEEP_TIME * 60000);
        }
    }

    protected void pauseApp() {
        // This is not currently called by the Squawk VM
    }

    protected void destroyApp(boolean unconditional) throws MIDletStateChangeException {
    }

    private void monitorSwitches() throws IOException {

        sw1 = (ISwitch) Resources.lookup(ISwitch.class, "SW1");
        sw2 = (ISwitch) Resources.lookup(ISwitch.class, "SW2");

        sw1.addISwitchListener(this);      // enable automatic notification of switches
        sw2.addISwitchListener(this);
    }

    public void switchPressed(SwitchEvent evt) {
        int switchNum = (evt.getSwitch() == sw1) ? 1 : 2;
        System.out.println("Switch " + switchNum + " pressed.");
    }

    public void switchReleased(SwitchEvent evt) {
        int switchNum = (evt.getSwitch() == sw1) ? 1 : 2;
        System.out.println("Switch " + switchNum + " released.");
        if (switchNum == 1) {
            loopFlag = false;
            System.out.println("Sending...");
            sendRS();
        } else {
            System.out.println("Clear...");
            leds.getLED(7).setColor(LEDColor.RED);
            leds.getLED(7).setOn();
            clearRS();
            leds.getLED(7).setOff();
        }
    }
```

```
    private void clearRS() {
        try {
            rms.deleteRecordStore("Dust");
        } catch (RecordStoreException ex) {
        }
    }

    private void sendRS() {
        try {
            rms = RecordStore.openRecordStore("Dust", true);
            for (int j = 1; j < rms.getNumRecords(); j++) {
                System.out.println(new String(rms.getRecord(j)));
                int counter = (j - 1) % 8;
                leds.getLED(counter).setColor(LEDColor.GREEN);
                sendOut(new String(rms.getRecord(j)));
                leds.getLED(counter).setOn();
                Utils.sleep(100);
                leds.getLED(counter).setOff();
            }
            rms.closeRecordStore();
        } catch (Exception ex) {
        }
    }

    private void store(String data) {
        try {
            rms = RecordStore.openRecordStore("Dust", true);
            int num = rms.addRecord(data.getBytes(), 0, data.getBytes().length);
            System.out.println("Record number: " + num);
            rms.closeRecordStore();
        } catch (RecordStoreException ex) {
        }
    }

    private String getDustLevel() {
        int dustLevel = 0;
        demoBoard.writeUART("Q");  // send query command to Arduino via serial port

        //NOTE: this timestamp is California time, you need to add 18 hours to convert it to Melbourne local time
        time = new Date();  // create timestamp

        Utils.sleep(20);  // give Arduino and dust sensor some time to get back

        String returnString = "";

        byte[] buffer = new byte[64];
        try {
            demoBoard.readUART(buffer, 0, buffer.length);  // read Arduino serial port
            returnString = returnString + new String(buffer, "US-ASCII").trim();  // format byte value
            dustLevel = Integer.parseInt(returnString);

            /**
             * Please apply calibration to dust level here
             */
        } catch (IOException ex) {
        }


        return String.valueOf(dustLevel) + ";" + time.toString();
    }

    private void sendOut(String msg) {
        try {
            DatagramConnection sendConn = (DatagramConnection) Connector.open("radiogram://broadcast:100");
            Datagram dg = sendConn.newDatagram(sendConn.getMaximumLength());
            dg.writeUTF(msg);
            sendConn.send(dg);
        } catch (IOException ex) {
        }
    }
}
```

Server

```
package org.sunspotworld;

import com.sun.spot.peripheral.radio.RadioFactory;
import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.peripheral.NoAckException;
import com.sun.spot.util.IEEEAddress;
```

```
import java.io.*;
import java.util.Date;
import javax.microedition.io.*;

public class SunSpotHostApplication {

    private boolean flag = true;
    private Date time;
    private String dataFile;

    public void run() {

        long ourAddr = RadioFactory.getRadioPolicyManager().getIEEEAddress();
        System.out.println("Our radio address = " + IEEEAddress.toDottedHex(ourAddr));

        time = new Date();
        dataFile = "C:\\DustSensingProject\\DustData\\DustValue_" + time.toString().replace(":", "_") + ".dat";


        FileWriter fc = null;
        PrintWriter pc = null;
        try {
            fc = new java.io.FileWriter(dataFile, true);
            System.out.println("Create " + dataFile);
        } catch (IOException ex) {
            ex.printStackTrace();
        }
        pc = new java.io.PrintWriter(fc);

        while (true) {
            try {
                RadiogramConnection conn = (RadiogramConnection) Connector.open("radiogram://:100");  // receive any message that hits port 100
                Datagram dg = conn.newDatagram(conn.getMaximumLength());
                try {
                    conn.receive(dg);
                    String rawData = dg.readUTF();
                    time = new Date();

                    if (flag) {
                        pc.println("##;##");
                        pc.println("@" + dg.getAddress());
                        pc.println("DustLevel;Timestamp");
                        flag = false;
                    }

                    if (rawData.contains(";")) {
                        System.out.println(rawData);

                        // need to convert California time to Melbourne time

                        pc.println(rawData);
                        pc.flush();
                    } else {
                        System.out.println(rawData + "," + time.toString());
                        pc.println(rawData + ";" + time.toString());
                        pc.flush();
                    }
                } catch (NoAckException e) {
                } finally {
                    conn.close();
                }
            } catch (IOException e) {
            }
        }
    }

    /**
     * Start up the host application.
     *
     * @param args any command line arguments
     */
    public static void main(String[] args) {
        SunSpotHostApplication app = new SunSpotHostApplication();
        app.run();
    }
}
```

Get Help

Paul Peng Deng

Sensor Network Developer

Phone: +61 402 837 152
Skype: dengpengcd
Web: SenSorApp.net
E-mail: pdeng@sensorapp.net