1. BigSense.io . . . 2
1.1 Research and Development . . . 3
1.1.1 Greenhouse - Mission of Mary . . . 4
1.1.2 Lincoln Hill - Mission of Mary . . . 6
1.1.3 Green Roof Monitoring - Hanley Sustainability Institute . . . 8
1.1.4 Soil Moisture Sensor System for Drought Monitoring and Paleoclimate Data Calibration . . . 9
1.1.5 Green Learning Station - EPA, Cincinnati MSD . . . 11
1.1.6 Environmental Sensor Networking and the Internet of Things . . . 13
1.1.7 Advanced Internet of Things . . . 14
1.1.8 Lab Student Examples . . . 15
1.2 Installation . . . 19
1.3 Configuration . . . 21
1.4 License . . . 26
1.5 Publications . . . 27
1.6 Contributors . . . 29

BigSense.io

Upcoming Talks and Classes

When	Speaker	Event	Location	City
Every Semester at University of Dayton	Dr. Andrew Rettig	Internet of Things class	Kettering Labs, University of Dayton	Dayton, Ohio
Spring Semester University of Dayton	Dr. Andrew Rettig	Advanced Internet of Things class	Kettering Labs, University of Dayton	Dayton, Ohio
Every Semester at University of Dayton	Dr. Andrew Rettig	Applied Internet of Things class	Kettering Labs, University of Dayton	Dayton, Ohio

About Big Sense

BigSense is an open source web service, licensed under the GNU GPL3, that is designed to record and present data from sensor networks. It works with LtSense (pronounced *Little Sense*), an application that can be installed on embedded devices. LtSense retrieves data from sensors can then transmit that data to web services such as BigSense. The entire program is still a work in progress with the first implementation used at the Green Learning Station for measuring pavement temperatures and storm water runoff.

BigSense Specifications

- Written in Scala
- Runs on Tomcat
- RESTful API
- Full regression tests (BigSenseTester written for Python 3.x)
- Supports queries based on date ranges and timestamp ranges
- Aggregation support for sums and averages over time intervals within date and timestamp ranges
- Presents data in multiple formats (XML, Comma Separated Vales, Tab Delimited, HTML tables)
- RSA signature verification for incoming sensor data
- Supports multiple relational databases
    - Microsoft SQL Server 2008 / 2012
    - PostgreSQL 9.x (with postgis extention)
    - MySQL (*in progress*)

LtSense Specifications

- Written for Python 2.x
- Polls for new data at configurable sample interval
- Queues sensor data in SQLite database
- RSA signatures for sensor data (using pypi-RSA)
- Support for transmitting images from USB web cameras
- Limited support for 1-Wire sensors (using OneWireFileSystem)

Research and Development

Research

Greenhouse - Mission of Mary

Lincoln Hill - Mission of Mary

Green Roof Monitoring - Hanley Sustainability Institute

Soil Moisture Sensor System for Drought Monitoring and Paleoclimate Data Calibration

Green Learning Station - EPA, Cincinnati MSD

Development

LtSense and Datazar integration - Andrew Rettig, Sumit Khanna

Phidget and LtSense integration - Sumit Khanna

Json update - Sumit Khanna

Pedagogy

Open innovation, living labs and hands-on experiences lowers the entry level of environmental sensor networking research and development.

Traditionally sensor networking has been researched with a top down approach due to the cost and expertise required. The standards that resulted from this approach were large sensor platform primarily developed for remote satellite sensing. Recently, with the advancement of supporting technologies these approaches are changing. The current trends within the Internet of Things (IoT) are from the bottom up, both in smaller size and a more manageable entry level. The standards are defacto causing a lag in interoperability. All business, government and academic entities are beginning to envision the possibilities of IoT but are at a loss of how to go about it. Our work strives to improve and create the academic pedagogy associated with the research and development of IoT. Similar to the R&D, no longer is the traditional top down researcher/student innovation sufficient, students must be enabled through hands-on experience supported with living lab exposure and team collaboration.

Confident and inquisitive students drive the innovation needed for IoT research and development. These students support IoT within the business, government and academic entities of tomorrow.

Classes

Environmental Sensor Networking and the Internet of Things

Advanced Internet of Things

Lab Student Examples

Greenhouse - Mission of Mary

Introduction

This project implemented an environmental sensor network at the Mission of Mary local urban farm. The goal was real time data access to geospatial temperature data to assist with urban farming. The temperature sensors were connected via Ethernet cable to a low-powered, open-source hardware, single-board Raspberry Pi. Data was transmitted via HTTP protocol over a cellular connection to our cloud servers. The network features an iOS app that Mission of Mary workers can download and view real-time temperature data at each point in the greenhouse. A secondary viewing method based on RESTful web services is an online query tool where data can be exported. Statistical analysis of the data will be done using a two-tailed t-test comparing the mean values of redundant sensors. Analysis will also be done using applicable local temperature data from Dayton International Airport and the Miami Conservancy District. To complete the project partnerships have been created with the Hanley Sustainability Institute, UD Geology, Intrust IT (a leading cloud provider in Cincinnati), Kore Telematics (the leading global M2M communication company) and KEEN.

Site

Our project is being carried out in collaboration with the Mission of Mary Cooperative and the Hanley Sustainability Institute at the University of Dayton. The Mission of Mary Cooperative is a non-profit organization that focuses on the issues of food and economic social justice, especially the issues of healthy food affordability and access in the Dayton area. In total, there are 2.5 acres of urban farms under their watch that provide local, fresh, and chemical free food to the surrounding areas. The greenhouse site of our current implementation is on one of the three agriculture plots the Mission of Mary operates within the Twin Towers neighborhood. It is a metal framed structure covered in a plastic film that may be opened to release heat during the summer months. It is approximately 52 feet long and 20 feet wide.

Funding

Funding is provided by the Hanley Sustainability Institute, a University of Dayton entity that promotes and extends the university's sustainability efforts.

Additional Information


Stander.pdf

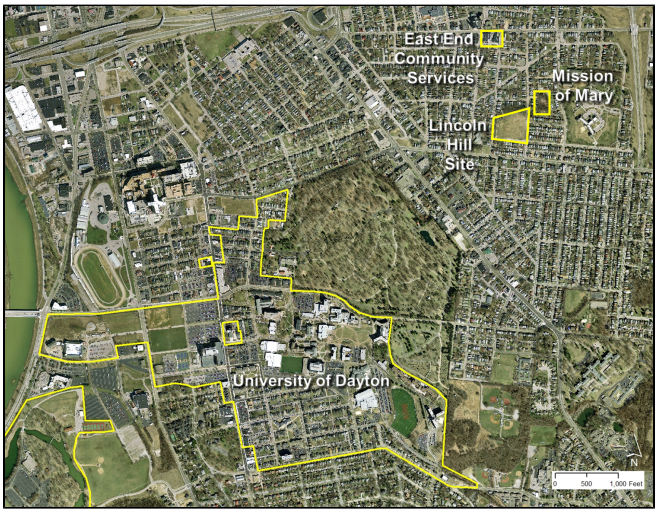
Lincoln Hill - Mission of Mary

Introduction

This installation expands upon our initial greenhouse installation with a second greenhouse site focused on soil temperature. Sensors are placed for monitoring internal, external and soil temperatures. Soil temperatures are collected every 15 minutes to analyze the temperature variance between covered and uncovered raised beds within the greenhouse. Two of the beds are covered while the control raised bed is uncovered. Temperature variance will be examined throughout the winter. The data collected will benefit the Mission of Mary for growing crops year round within their urban garden greenhouse locations. This hardware system was incorporated into our current cyberinfastructure at a cost totaling less than $300.

Site

Many urban areas across the United States lack the necessary infrastructure to provide residents access to fresh, healthful foods and lush, open green space. East End Community Services has partnered with the University of Dayton and Mission of Mary Cooperative to initiate change within the urban region of East Dayton through the development of a sustainable, multipurpose urban agriculture and community greenspace within Dayton's Twin Towers Neighborhood. The project will transform the land with natural, educational and recreational elements that will increase neighborhood access to wholesome foods and offer natural green space for community gatherings and nature play. As the former site of Lincoln Elementary School, this property is located in the heart of the Twin Towers Neighborhood among the residents it will serve.



Funding

Hanley Sustainability Institute

Additional Information

Green Roof Monitoring - Hanley Sustainability Institute

Introduction

This project is in partnership with the University of Dayton and the Hanley Sustainability Institute. The planning and software development for the project will begin in spring of 2017 and continue for the summer and fall semesters. A new open source software platform will be utilized for the development of soil moisture sensor integration. Temperature and rain gauge data will also be incorporated into the project. The project will also feature an educational mobile application on green roofs for student interaction with the real time sensor data. Future phases of the project hope to include data visualization through augmented reality.

Site

A patio concrete roof top on the campus of UD is being renovated to include green roof enhancements. The improvements will be both aesthetic and functional to be used an educational and research site regarding green roofs. Sensor systems will be included in both the design and construction to enhance the goals of the project. Work on the site is scheduled to begin spring of 2017. The site will educate the university community on the challenges of stormwater runoff within the local ecosystem while demonstrating solutions for peak water discharge reduction during local rain events.

Funding

Hanley Sustainability Institute

Soil Moisture Sensor System for Drought Monitoring and Paleoclimate Data Calibration

Lead Josh Latham

Advisor Zelalem Bedaso

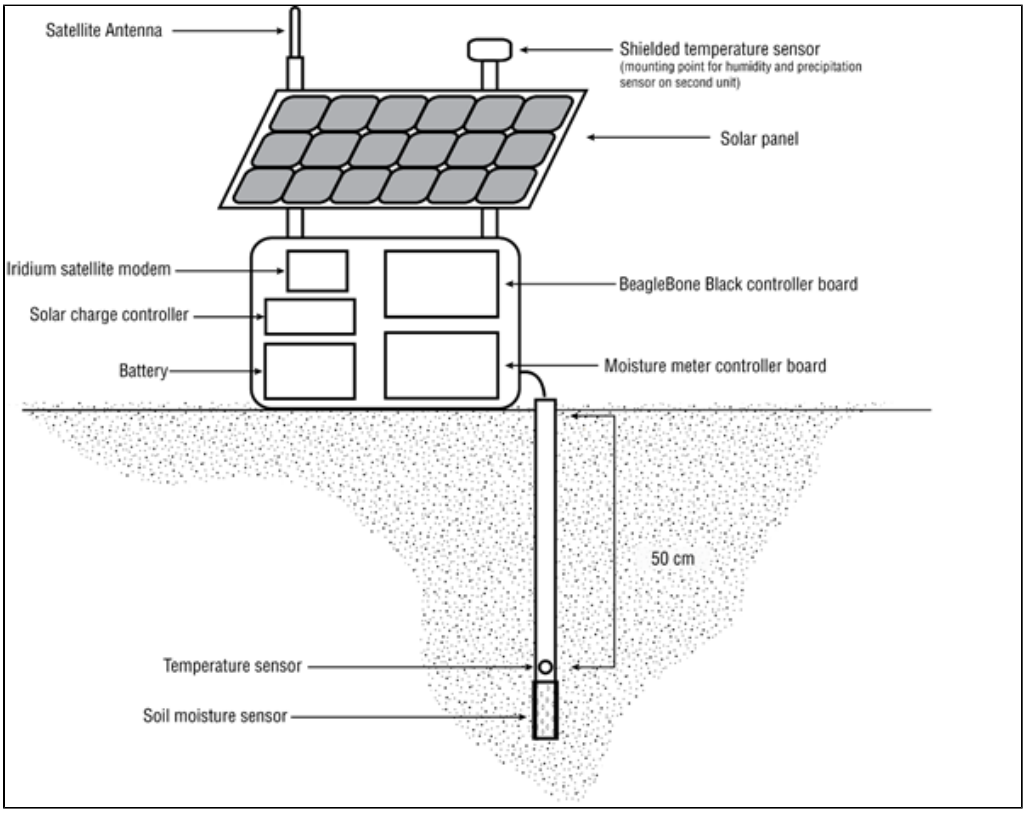
Advisor Andrew Rettig

Introduction

In remote areas with little infrastructure there is a lack of in situ soil moisture data and these areas are often most affected by seasonal rains and droughts. This research project proposes to create an in situ sensor system consisting of two units capable of recording soil moisture, soil temperature, air temperature, humidity, and precipitation amounts. These units will be completely self-contained and self-reliant. A full solar power system will be built with solar panels, battery, and charging controller allowing for over a year of sustained in situ data collection. Collected data will be transmitted to servers using the Iridium satellite communications network with daily-collated data transmissions. This data will be stored in a database on local servers and available on the Internet for analysis. The aim of these sensor units is to prototype the hardware, develop the software, and define the protocols needed to create an in-situ collected soil moisture database. Once these two systems are complete and have undergone stateside field-testing, additional funding will be sought to create additional units to enable deployment of a wide area network of sensors across various environments in Ethiopia, where there is a lack of such monitoring data. The soil moisture data collected by this sensor network will have many environmental science applications including drought analysis, subsistence farming sustainability efforts, remote sensing calibration, baseline climate data for pedogenic carbonate formation for paleo-environment reconstruction (Breecker et al., 2009, Bedaso et al., 2011), and further understand the significance of soil carbonate formation and its uses in atmospheric carbon sequestration efforts (Washbourne et al., 2012).

Research Description

Environmental sensor networks have the potential to become a standard research tool and revolutionize the earth system and environmental sciences (Hart & Martinez, 2006). This research project will construct prototype units for a *Large Scale Single Function Network* (Hart & Martinez, 2006) and will consist of three phases. Phase One will be the construction of the sensor systems and will consist of hardware acquisition, integration, and construction. Two different sensor units will be constructed. Both units will be able to measure soil moisture, soil temperature, and air temperature, while one unit will also measure humidity and precipitation amounts. Software developments and enhancements will occur in this phase for both sensor integration and communications. Phase Two will entail deploying the sensors locally here on campus or at a UDRI facility and collecting data for approximately one month. This initial data collection period will allow for any communications or hardware issues to be easily diagnosed and solved. Phase Three would involve placing the sensors for a longer term test, approximately two months, in a similar environmental setting to Ethiopia in the US Southwest near El Paso, Texas. Data collected from all field tests will be statistically analyzed, cross checking readings with each unit and local weather stations for accuracy.



Funding

Keck Fellowship, University of Dayton

Green Learning Station - EPA, Cincinnati MSD

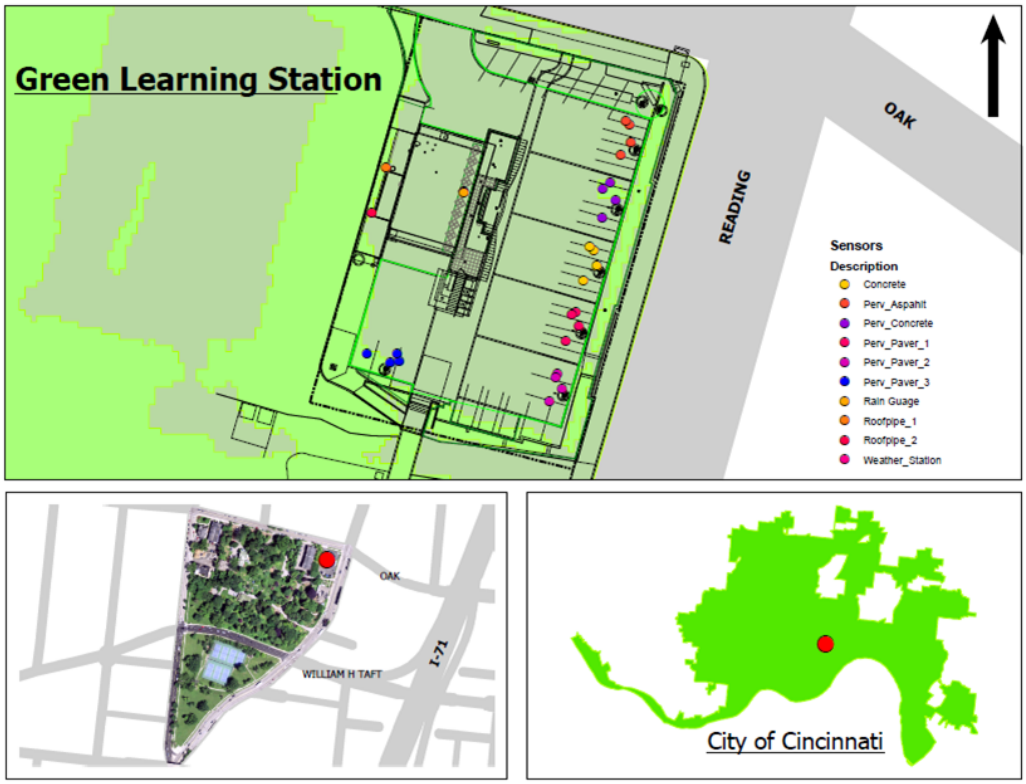
Introduction

The City of Cincinnati's Municipal Sewer District (MSD) worked with the Civic Garden Center (CGC) to fund a research into monitoring storm water runoff. The CGC converted their parking lot into a research area to test the permeability of different paver types. Each pavement type had an identical foundation of concrete and gravel, was separated using liner and fitted with a manhole at the bottom of their slopes. These collection pits were outfitted with sensors to monitor water runoff flow rates.

BigSense and LtSense were developed to help monitor and store data from the research area. It was an open source initiative originally developed by students and volunteers affiliated with the University of Cincinnati's Geography Department.

Site

The research area is known as the Green Learning Station (GLS). In addition to storm water research, the GLS is also used to educate people in the community, as well as local school groups, about sustainability and environmentally conscious technology. Papers detailing our research and methodology are found under the publications page.



Funding

Cincinnati MSD, EPA and Civic Garden Center

Additional Information

Publications

Environmental Sensor Networking and the Internet of Things

This course has been taught since the fall of 2015. The client proposals developed within the class have raised over **$25,000** from real clients. Students have gone on to implement the networks under the guidance of Dr. Andrew Rettig. Development of the class was funded by KEEN (Engineering Unleashed).

Detailed Description of Course

Background

Rapid advances and decreasing costs in sensor technology, wireless communication, data processing speed, and data storage capacity have enabled widespread deployment of automated environmental sensing systems. Basic environmental processes can be monitored continuously in habitats ranging from very remote to urban, providing information in unprecedented temporal and spatial resolution. Although research questions that may be answered based on these data are very diverse, the design process, establishment and maintenance of most environmental sensor systems, and resulting data handling have many commonalities. Realizing that sensor networks are becoming ubiquitous in ecological research creates the need for a new set of technological skills, approaches and applications.

Expected Students

Environmental Sensor Networking and the Internet of Things (ESN & IoT) is a new course development. The class is a multi-discipline undergraduate and graduate course targeting students from the earth sciences, computer science, engineering and business.

Hands-on experience

The class is on Tues/TH with both lecture and lab components. The complexities involved in sensor networking are best learned through hands-on experiences. Students work directly with sensors (such as a temperature sensor) connecting them to embedded devices. The embedded devices are programmed to capture the data and transmit the data to a server for storage and publication. The students use these experiences to compliment the lecture components of the course. These experiences enable the students to complete a sensor networking final project, a real client proposal for solving sensor networking needs. (Open Source software is used by the students for the labs)

Resources

The Internet of Things is the most hyped technology on the planet according to Gartners Hype Cycle. The popularity of the topic provides opportunities for guest speakers. The course often incorporates an IT guest speakers on server virtualization, a M2M (Machine to Machine) guest speaker, a proprietary sensor company guest speaker, and a computer scientist.

Structure

The class is taught in a team environment with the students consistently working together to learn and teach each other the material as well as complete the lab component. In sensor networking development, team work is an essential skill. The complexity of a sensor network creates the need for team member specialization on specific aspects of the network. A typical installation will have members dedicated towards sensors, embedded devices, servers, visualization and data analysis.

Entrepreneurial Minded Learning

Lastly, I want to emphasize the importance of EML within sensor networking. With the team approach essential within sensor networking implementations, every engineer must be able to relate their developments to the customer. Only with EML and a diverse team will a sensor networking team succeed with an applicable and innovative network.

Advanced Internet of Things

Coming Soon....

This class will focus on security, wireless technologies and industrial IoT.

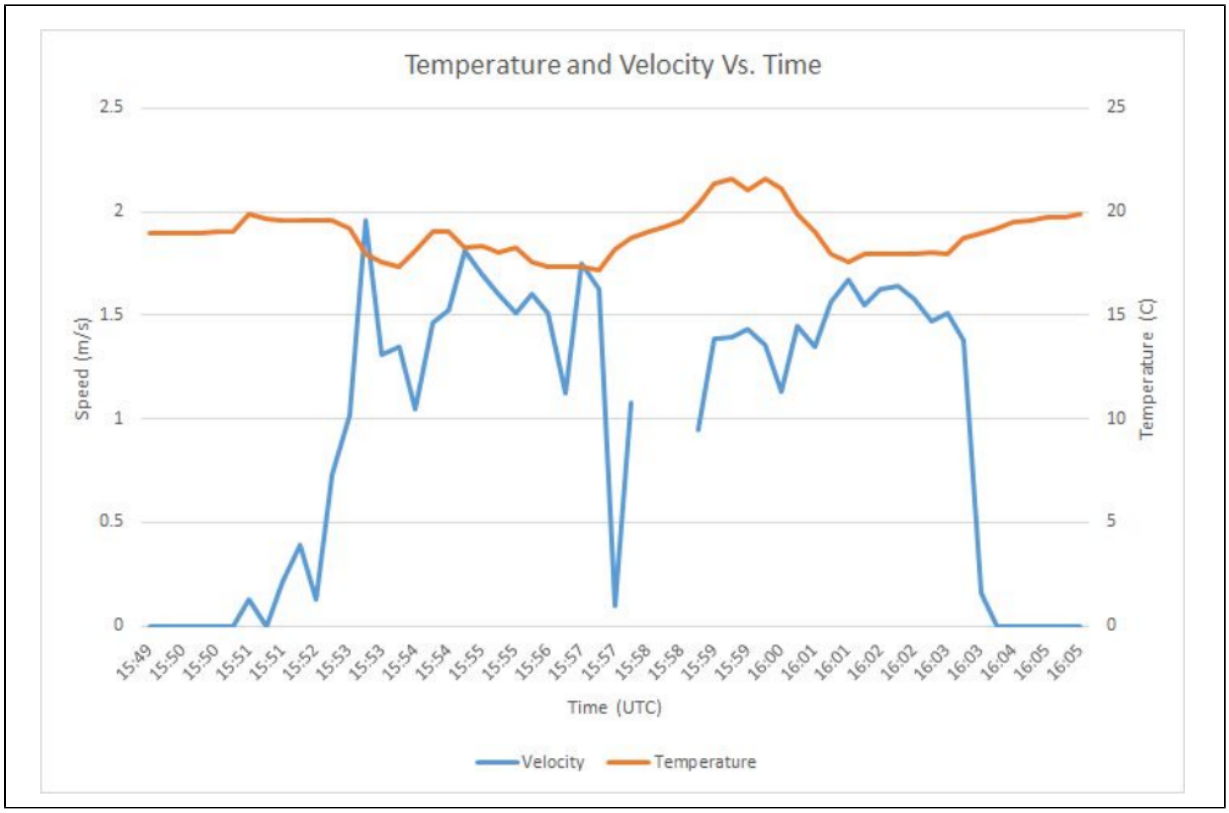
Lab Student Examples

The students created these examples using the lab hardware and installing the software for a complete custom end to end sensor network for real-time streaming.

Lab 2:

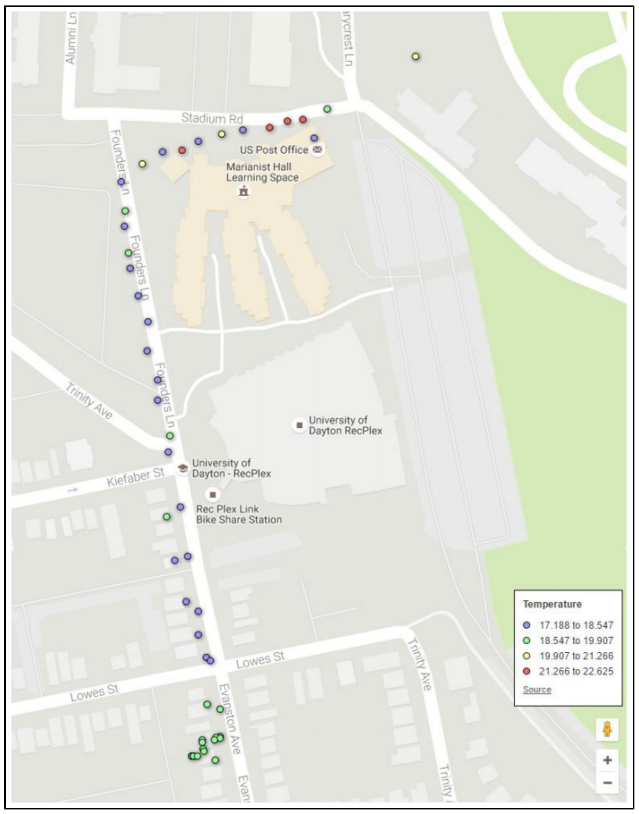
Step 5: Chart Creation from their streaming IoT device

The team downloaded a .csv that had a date range containing a walking trip one team member took to the post office while recording temperature and gps data. The following graph shows both temperature and speed graphed versus time of day. From this graph, it can be seen that for the first two minutes after the beaglebone was powered up, the board was stationary. This was when the team member was packing up the board into his backpack to take it to the post office. The next 6 minutes are during the walk, the temperature can be seen to be lower outside than inside the team member's house. From 15:57 to 15:59, the velocity data drops out. This is because the team member entered the post office, and the GPS module lost its lock. The higher temperature in the post office can also be seen during this time. The next 5 minutes or so show the walk back to the team member's house, and the last two minutes, where the velocity drops back to 0 is when the device was being unpacked and prepared for shutdown.



Step 6: Google Fusion Tables Map Creation

The following map was created in google fusion tables using the data from the trip to the post office and back. The legend shows that the color of each dot depends on the temperature at that point, separated into 4 ranges, red being the hottest and blue-purple being the coldest. While the temperature does fluctuate quite a bit, it can definitely be seen that it was cooler outside than in either the team member's house or the post office. There is also an outlier data point north west of the post office where the team member never went. This was a glitch in the gps lock, and slightly erroneous data was reported.



Example 2 Step 6:

For part 6 of the lab, the team placed the Beagle Bone temperature measurement rig into a motor vehicle. The GPS was mounted on the roof of the car in order to get a clear signal as the car moved. The temperature probe was placed out the window in order to check for varying temperature as the car moved. The car was then driven for a few miles around Dayton. The raw results of this activity are shown in Figure 8. Plots of the temperature and speed as a function of time are shown in Figures 9 and 10, respectively. Finally, maps were created using Google Fusion of the temperature and speed data over varying positions, shown in Figures 11 and 12, respectively.

Looking at the plots and maps, a general trend can be observed. As the speed of the vehicle increases, the temperature measured at the probe decreases. At the starting and ending points of the experiment, the temperature probe measured its max values, since the vehicle was at rest. The other speed variations can be attributed to the start and stop movements of the vehicle in traffic, resulting in fluctuations in the measured temperature.

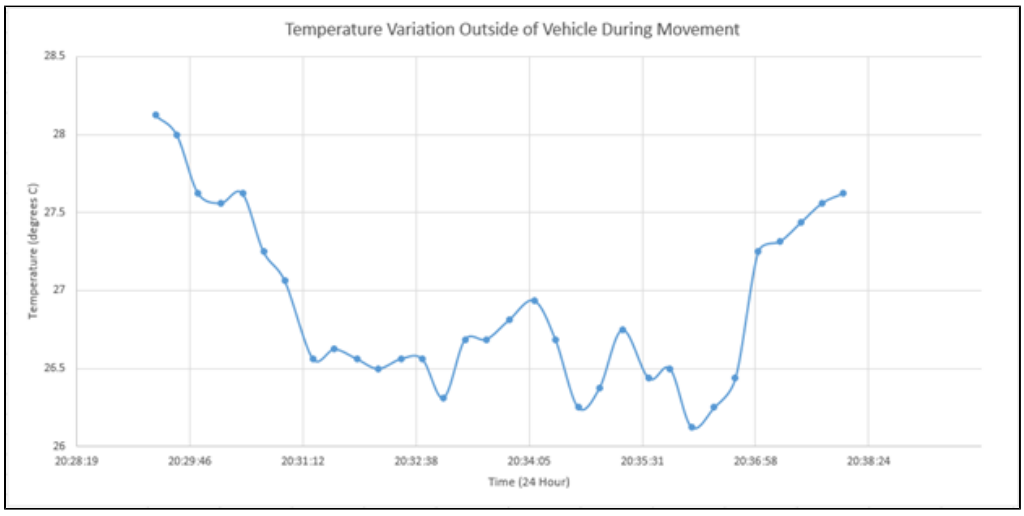


Figure 9: Plot of the Temperature Variation Measured vs. Time when Placing the Temperature Sensor Outside of a Moving Vehicle

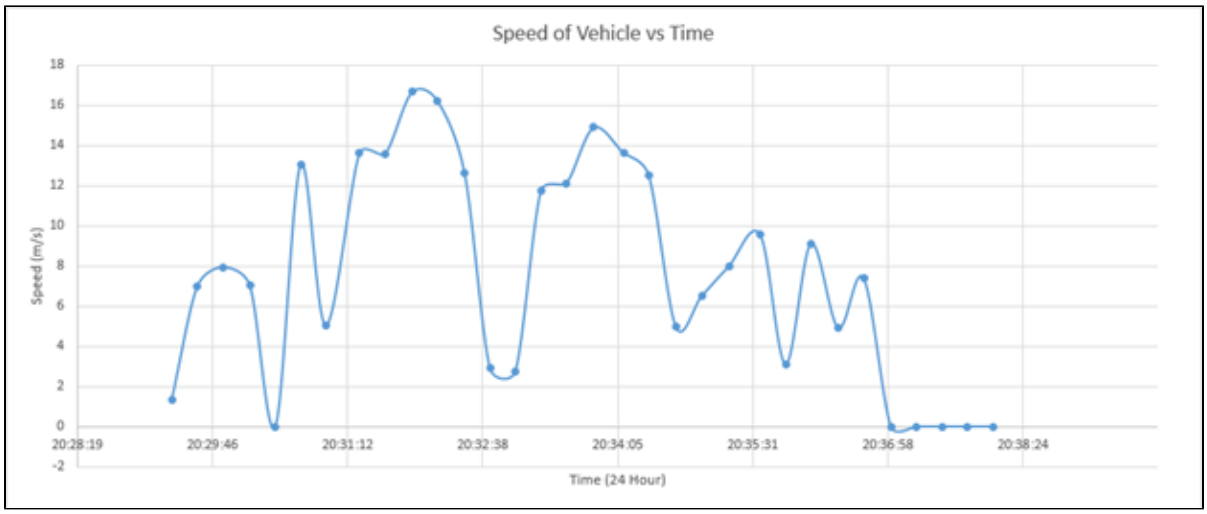


Figure 10: Plot of the Speed of the Vehicle vs. Time

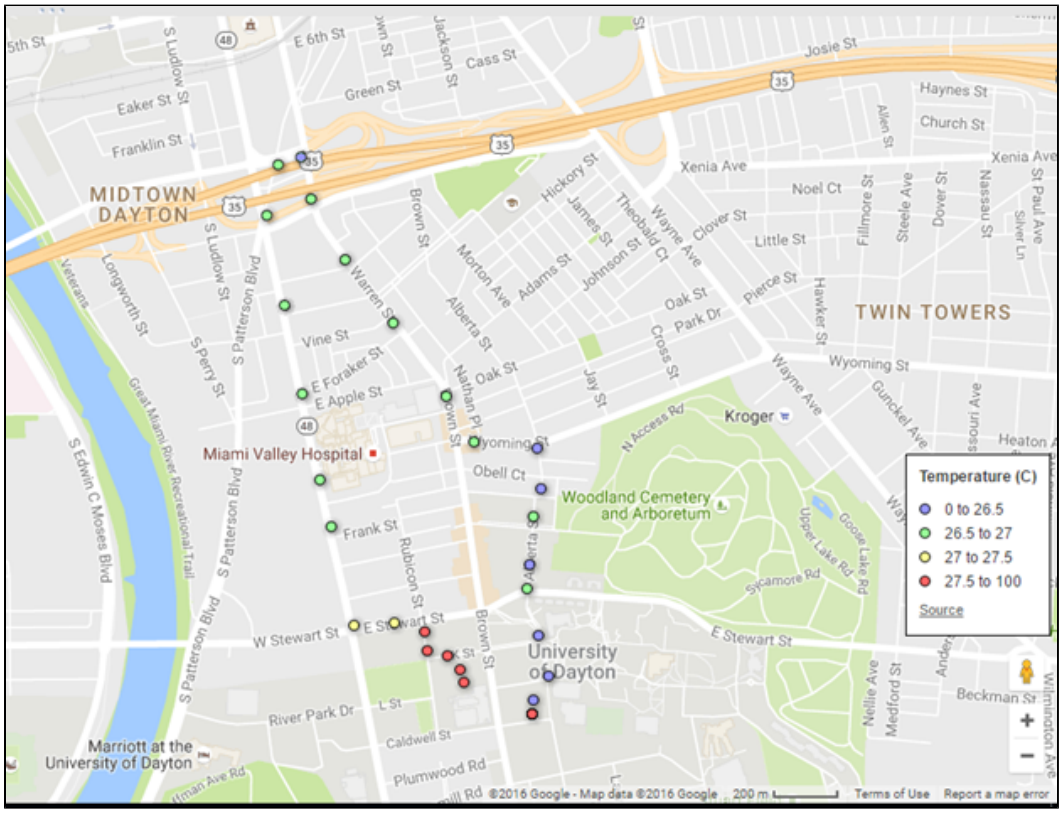


Figure 11: Map of Temperature Data Collected While Moving

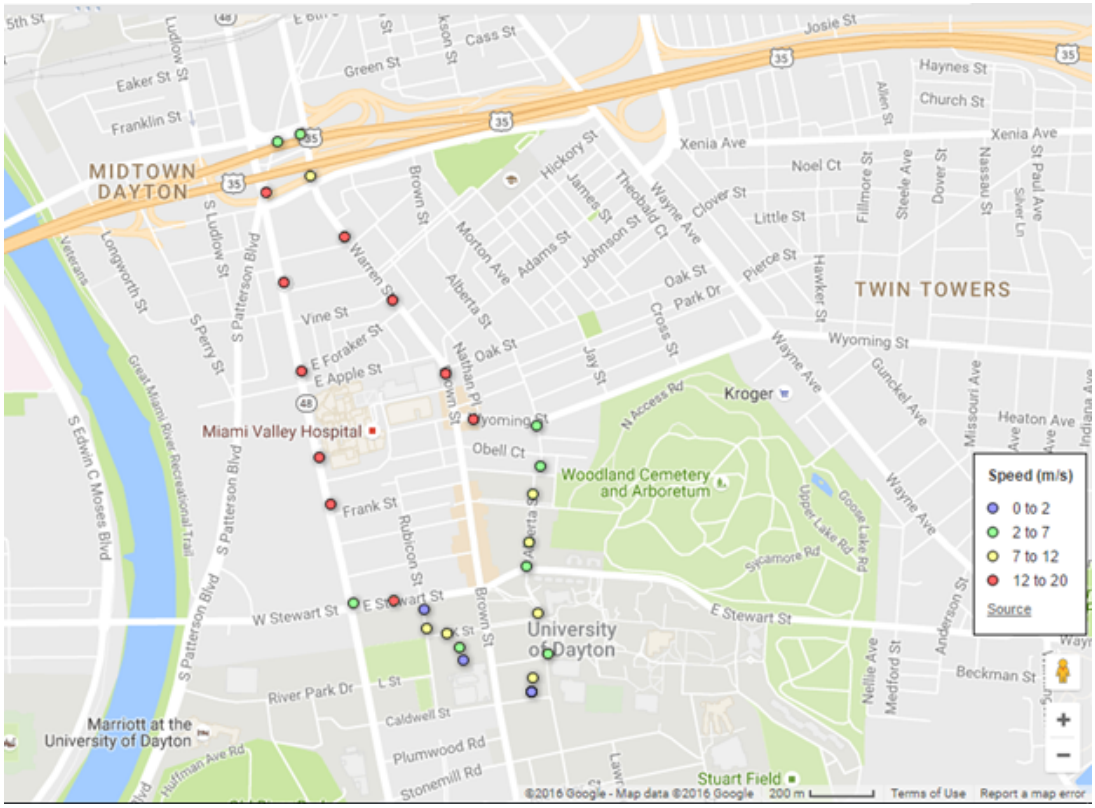


Figure 12: Map of Vehicle Speed While Moving

Installation

BigSense and LtSense currently have official package repositories for several Linux distributions.

- Ubuntu/Debian
- CentOS/openSUSE

Ubuntu/Debian

Repositories are provided for distributions that use **apt** for package management such as Debian, Ubuntu and their derivatives. To get started using these repositories, first install the BigSense repository key.

```
curl https://repo.bigsense.io/bigsense.io.key | sudo apt-key add -
```

There are three repositories for the three major init system types: **Upstart, SystemV** and **SystemD**. There are also three components for each repository **n ightly, testing** and **stable.**

- If you're using **Ubuntu 14.04 Trusty**, **Ubuntu 14.10 Utopic** or any other distribution that uses **Upstart**, you'll want to use the **Upstart** repository.
- If you're using **Debian 7 Wheezy, Devuan** or any distribution that uses **SystemV**, you'll want to use the **SystemV** repository.
- If you're using **Debian 7.8 Wheezy for Beagle Boards**, their custom Debian image uses **SystemD**.
- If you're using **Debian 8 Jessie, Ubuntu 15+** or any distribution that uses **SystemD**, you'll want to use the **SystemD** repository.

Upstart:

```
sudo add-apt-repository 'deb https://repo.bigsense.io/debs upstart stable'
```

SystemV:

```
sudo add-apt-repository 'deb https://repo.bigsense.io/debs systemv stable'
```

SystemD:

```
sudo add-apt-repository 'deb https://repo.bigsense.io/debs systemd stable'
```

You can now install the BigSense server or the LtSense client. If you're installing BigSense, you will also need a Java Runtime Environment. The following example installs an OpenJDK Java 8 Runtime.

BigSense:

```
sudo apt-get update
sudo apt-get install openjdk-8-jre-headless bigsense
```

BigSense is now installed, however you'll need to setup a database and restart the BigSense service for it to be functional. See Configuration for more information.

LtSense:

```
sudo apt-get update
sudo apt-get install ltsense
```

LtSense is now installed, however you'll need to adjust its settings to point to a BigSense instance, configure its sensors and restart it for it to start streaming data. See Configuration for more information.

CentOS/openSUSE

There is a single yum repository for **SystemD** based rpm distributions. **CentOS 7** and **openSUSE 13** have been tested, but any RPM based distribution that supports **SystemD** services should work (e.g **Fedora 15**, **Redhat Enterprise Linux 7)**. To start, add the BigSense RPM PGP key:

```
sudo curl https://repo.bigsense.io/bigsense.io.key --create-dirs -o /etc/pki/rpm-gpg/RPM-GPG-KEY-BigSense
sudo rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-BigSense
```

Next, add the repository by creating ***BigSense.repo*** and setting it up like so

- CentOS 7 location: ***/etc/yum.repos.d/***
- OpenSuse 13 location: ***/etc/zypp/repos.d/***

```
[BigSense]
name=BigSense Repository for RPM
baseurl=https://repo.bigsense.io/rpms/stable
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-BigSense
```

You can now install the BigSense server or the LtSense client. If you're installing BigSense, you will also need a Java Runtime Environment. The following example installs a Java 7 Runtime.

BigSense (CentOS):

```
sudo yum install java-1.8.0-openjdk-headless bigsense
```

BigSense (openSUSE):

```
sudo zypper install java-1_8_0-openjdk-headless bigsense
```

BigSense is now installed, however you'll need to setup a database and restart the BigSense service for it to be functional. See Configuration for more information.

LtSense (CentOS):

```
sudo yum install ltsense
```

LtSense (openSUSE):

```
sudo zypper install ltsense
```

LtSense is now installed, however you'll need to adjust its settings to point to a BigSense instance, configure its sensors and restart it for it to start streaming data. See Configuration for more information.

Configuration

[ ] [BigSense] [LtSense]

BigSense

BigSense needs to connect to a database for its storage. Currently the following databases are supported:

- MySQL 5+
- Postgres 9.2+ (postgis extension required)
- Microsoft SQL 2012+

Example configuration files can be found in */etc/bigsense/examples*. Copy the appropriate file for your database type into ***/etc/bigsense/bigsense.conf*.** Here's an example for a Postgres configuration file:

```
dbms=pgsql
dbHostname=localhost
dbPort=3456
dbDatabase=bigsense
dbUser=db_bigsense
dbPass=bigsense
dboUser=postgres
dboPass=
securityManager=Disabled
httpPort=8181
server=tomcat
```

- **dbms** can be either **pgsql**, **mysql** or **mssql** for Postgres, MySQL and Microsoft SQL respectively
- **dbHostname, dbPort** and **dbDatabase** are database connection parameters
- **dbUser/dbPass** and **dboUser/dboPass** specify the two database users
    - The DBO user can update the schema and is only used upon startup of BigSense
    - The non-DBO user is used for all further connections when reading from/writing to the database
- **securityManager** can be used to verify all incoming sensor data. Currently can be set to **Disabled** or **Signature**. (TODO: link to security section)
- **httpPort** the port the web service will run on
- **server** can be either **jetty** or **tomcat**

Database

Bigsense will create the schema with appropriate access and user setup within the databases.

First you will want to create an empty database for BigSense to work with. Remember the name of this database and be sure to add the name to the config file (shown above).

Next you will want to create a database user that has access for Bigsense. Using the postgres user initially is an easy way to get started. Remember to add this name and password to the config file.

The **dbUser** just needs to entered in the config file as shown above. Bigsense will setup the **dbUser** in the database for you and use it for schema setup.

You may have access control restrictions for your database server

- For MySQL, the DDL generator will attempt to guess the BigSense servers hostname. Check the SQL file to ensure it put in the correct host
- For PostgreSQL, access control is typically located in */etc/postgresql/<version>/main/pg_hba.conf*
- For MicrosoftSQL, be sure to enable the relevant firewall rules

PostgreSQL doesn't come with geospatial support. You must install the postgis extension or else the schema will fail to properly load

Finally, restart the BigSense service on the server you installed it to.

```
service systemctl bigsense restart
```

You can monitor it starting up, and view any diagnostic information, by viewing the log file located at */var/log/bigsense/bigsense.log*.

LtSense

The configuration for ltsense is located at ***/etc/ltsense/ltsense.conf***. The following is an example configuration file that sends sensor data to a BigSense instance running at ***example.com*** at a 15 second interval using two fake/virtual sensors which generate random data:

```
[General]
sample_rate = 15
[Data]
  [[primary]]
    type = sense.xml
      [[[Identifier]]]
        type = name
        id = ExampleRelay01
      [[[Location]]]
        type = gps
[Transport]
  [[http]]
    type = http
    url = https://example.com/Sensor.sense.xml
    pause_rate = 1.0
    timeout = 10.0
      [[[Queue]]]
        type = memory
      [[[Security]]]
        type = none
[Handlers]
  [[virtual]]
    type = virtual
    sensors = $temp1,$temp2
[Sensors]
  [[temp1]]
    type = virtual/temp
    id = VRTEMP01
    units = C
    rangeMin = 1
    rangeMax = 25
  [[temp2]]
    type = virtual/temp
    id = VRTEMP02
    units = C
    rangeMin = 1
    rangeMax = 25
```

Configuration Sections

- **General**
  - **sample_rate** = the rate at which sensor handlers are queried in seconds
- **Data**
  - [name] - Multiple data types can be defined, each with a unique name. All data types are sent to all transport types.
  - **type** - data format (currently only **sense.xml** is supported)
  - **Identifier**
    - **type** - Indicates how the relay is identified. Possible values are **name, mac** and **uuid**
      - **mac** - Uses network adapter's mac address for identifier. Valid attribute is **adapter**, defaults to **eth0**.
      - example:

```
[[[Identifier]]]
  type = mac
  adapter = eth2
```

      - **name** - Uses a static name defined in the configuration file
      - example:

```
[[[Identifier]]]
  type = name
  id = MyRelay01
```

- **uuid** - Uses a generated UUID. Requiresid_file attribute
- example:

```
[[[Identifier]]]
  type = uuid
  id_file = /var/lib/ltsense/uuid
```

- **Location** (section is optional. If omitted, no location data will be transmitted)
  - **type** - Possible values are **virtual** and **gps**. Virtual uses static values (shown below) where as gps will attempt to connect to the locally running gpsd instance.
  - **longitude** (virtual only)
  - **latitude** (virtual only)
  - **altitude** (virtual only)
  - **speed** (virtual only)
  - **climb** (virtual only)
  - **track** (virtual only)
  - **longitude_error** (virtual only)
  - **latitude_error** (virtual only)
  - **altitude_error** (virtual only)
  - **speed_error** (virtual only)
  - **climb_error** (virtual only)
  - **track_error** (virtual only)

- **Transport**
  - [name] - Multiple transport types can be defined, each with a unique name. All data types are sent to all transport types.
    - **type** - Currently only **http** is supported (Even if using SSL/TLS, this should still be http. https can be specified in the URL)
    - **url** - Service endpoint URL
    - **pause_rate** - Minimum amount of time in seconds to wait between sending sensor data pacakges (default: 0.10)
    - **timeout** - Amount of time in seconds to wait after sensor data failed to send (data package is placed back on the queue, default: 10.0)
    - **Queue**
      - **type** - Valid types are **memory** and **sqlite**.
        - Memory based queues are reset when the process ends with any packages in the queue discarded.
        - example:

```
[[[Queue]]]
  type = memory
```

        - Sqlite queues store packages across restarts in a file.
        - example:

```
[[[Queue]]]
  type = sqlite
  sql_file = /var/lib/ltsense/queue.sqlite
```

    - **Security**
      - **type** - Valid types are **none**, **rsa** and **m2**
        - For m2 (pypi - M2Crypto) and rsa (pypi - rsa), the **data_dir**, **key_file** and **key_size** must be specified. If a key does not exist, LtSense will attempt to create one in the **data_dir** with the name specified in **key_file**.
        - example:

```
[[[Security]]]
  type = rsa
  data_dir = /var/lib/ltsense
  key_file = ltsense.pem
  key_size = 2048
```

- **Handlers**
  - [name] - Multiple handlers can be specified with unique names. Each handler will query its sensors and return data that will be encoded by formats listed in the **Data** section before being passed to the services listed in the **Transport** section
    - **type** - Valid types are **virtual**, **1wire**
      - Virtual sensors are fake sensors used in testing that report random data. The handler takes a list of sensors that must be defined in the **Sensor** section
      - **1wire** sensors can specify a device (default is *u* for USB. This can also be a path to a serial interface, e.g. */dev /ttyUSB0,* or the address/port of a owserver, e.g. *localhost:1234*)
      - example:

```
[Handlers]
  [[virtual]]
    type = virtual
    sensors = $temp1,$temp2
```


sensors must be a list. If you only have one sense, leave a trailing coma:

sensors = $virtual1,

- 1/Wire USB sensors are auto detected
- example:

```
[Handlers]
  [[virtual]]
    type = 1wire
    device = u
```


You must have the **python-ow** (1-Wire File System) package installed to use 1wire sensors directly over USB with LtSense.

If you want to connect to an instance of an **owserver** (1-Wire File System Server), you should have **python-ownet** installed.

The LtSense package comes with a udev rule so that the ltsense user should automatically get permission for 1-wire USB dongles. This rule may need to be adjusted if you have permission errors.

- Sensors
  - [name] - Unique name for a virtual sensor that is referenced in the **Handler** section.
    - type - The two virtual sensor types are **virtual/temp** and **virtual/image**
      - Virtual temperature sensors take a range from which to generate a number from. They must specify **id, units, rangeMin** and **rangeMax**
      - example:

```
[Sensors]
  [[temp1]]
    type = virtual/temp
    id = VRTEMP01
    units = C
    rangeMin = 1
    rangeMax = 25
  [[temp2]]
    type = virtual/temp
    id = VRTEMP02
    units = C
    rangeMin = 1
    rangeMax = 25
```

      - Virtual Image sensors transmit the same image continuously using the **image_file** attribute
      - example:

```
[Sensors]
  [[photo1]]
    type = virtual/image
    image_file = /var/lib/ltsense/test.jpg
```

After you've configured LtSense, restart the LtSense service.

```
# Upstart

service ltsense restart

# SystemD

systemctl restart ltsense.service

# SystemV

/etc/init.d/ltsense restart
```

You can monitor it starting up, and view any diagnostic information, by viewing the log file located at */var/log/ltsense/ltsense.log*.

License

BigSense, BigSenseTester and LtSense are all licensed under the GNU GPL v3 License.


```
    BigSense and LtSense are free software: you can redistribute them and/or modify
    them under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    BigSense and LtSeense are distributed in the hope that they will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
```

Publications

Rettig, Andrew J., Sumit Khanna, and Richard A. Beck. "Open source REST services for environmental sensor networking." *Applied Geography* (2014).

http://www.sciencedirect.com/science/article/pii/S0143622814002586

Highlights

- A RESTful design philosophy is used to develop in situ sensor networking software.
- The client application LtSense is written in Python for installation on embedded devices.
- BigSense, an open source web service, is designed to record and present data from sensor networks.
- Open Geospatial Consortium standards are suggested to help guide future community development.
- A case study is presented of the sensor networking architecture for monitoring stormwater runoff.

Abstract

The greatest challenge in the implementation of environmental sensing networks is converting a large variety of data streams from diverse sensors, often in proprietary protocols, to international standards such as Extensible Markup Language (XML) with Open Geospatial Consortium (OGC) XML tagging and web service standards. Implementing standards throughout the architecture will not only enable interoperability and reduce cost but will allow scientists to contribute to sensor network innovation. This article introduces open source Representational State Transfer (REST) services created specifically for environmental monitoring. OGC standards are suggested to help guide future community development for sensor description and registration. This article contributes to the design and implementation of affordable, self-documenting, near-real-time geospatial sensor webs for environmental monitoring using international standards.

Keywords

- Environmental sensor networks;
- Open source software;
- REST;
- Open Geospatial Consortium;
- Web of things

Rettig, Andrew J., Sumit Khanna, Dan Heintzelman, and Richard A. Beck. "An open source software approach to geospatial sensor network standardization for urban runoff." *Computers, Environment and Urban Systems* 48 (2014): 28-34.

http://www.sciencedirect.com/science/article/pii/S0198971514000507

Highlights

- Open source software is used to create a modified router for reading Maxim's 1-Wire™ sensor protocol.
- The modified router is the first solution towards a modular architecture for an urban runoff sensor system.
- The modified router created the bridge between the sensor protocols and the middle-level software.
- A representational state transfer design philosophy is used to develop the software for transferring sensor data.
- Open source software lowers the entry barrier to sensor networking and enables developers for continued innovation.

Abstract

In this paper, we implement a geospatial sensor network for monitoring a green technology stormwater runoff site. The sensor network uses OpenWRT, an embedded Linux operating system, and other open source software, to create a modified router for reading Maxim's 1-Wire™ protocol, queuing and transferring standardized sensor data while enabling location and time. The modified router created the bridge between the sensor protocols and the middle-level software to provide reliable data to both the sewer district and the Environmental Protection Agency. Representational State Transfer (REST) is used in the design philosophy of the client and server open source software for transferring the data from the embedded systems to the server level for storage and publication. The use of open source software not only creates a more affordable network but lowers the entry barrier to sensor networking and enables developers for continued innovation and standardization.

Keywords

- Sensor networks;
- Open source software;
- Urban runoff;
- Open Geospatial Consortium

Rettig, Andrew J., Sumit Khanna, Richard A. Beck, Quinn Wojcik, and Carmen A. McCane. "Monitoring permeable paver runoff with an open-innovation geospatial sensor network." *International Journal of Digital Earth* ahead-of-print (2014): 1-17.

http://www.tandfonline.com/doi/abs/10.1080/17538947.2014.965880#.VQRUieEXFeQ

Abstract

Sensor networks are an essential tool for environmental scientists. As scientists and engineers are beginning to utilize these new methods and devices in their fieldwork, they need to be actively involved in the future of sensor-networking development. Continued sensor network innovation is important for improved standardization, affordability, and interoperability. This article uses a storm water case study to outline an end-to-end open-innovation sensor network. Open innovation by scientists, engineers, and entities is the collaborative process of creating value for this project in permeable paver runoff data and advances within sensor networking. This article focuses on the technical implementation of the near–real-time location and temporally aware sensor network. Data are streamed in near–real-time with subliter precision to my butt using common off-the-shelf routers. The sensors use Maxim's 1-wire™ protocol, and the unique digital serial numbers confirm the data. The data retrieved compare residence times within the permeable paver catchment basins and the control basin. Sensor network advances are made by bridging the gap between sensor protocols and communication systems. These advances enable the development of open-source representational state transfer web services. Our successful implementation serves as an example for others to study and expand upon for a variety of monitoring solutions.

Keywords

- sensor networks,
- Internet of things,
- open geospatial consortium,
- open innovation,
- permeable pavers

Contributors

Sumit Khanna

I graduated from the University of Tennessee at Chattanooga in 2009 with a Masters in Computer Science. I became involved in a research project with graduate students at the University of Cincinnati (led by Andrew Retting) that deal with using sensor networks to gather environmental data. That project is what grew into BigSense and LtSense, a fully open source system for gathering, reporting and aggregating data from sensor networks.

For the past decade, I've worked mostly in Software Engineering industry, and to a lesser extent System Administration and Dev-Ops. The types of companies I've worked for includes telecommunications, government, education, credit card processing, power, travel and e-commerce. For the past two and a half years, I was working for an open source consulting company in Wellington, New Zealand. I left that position in April 2015 and am currently on sabbatical, travelling through Europe and working on BigSense.

curriculum vitae

Dr. Andrew Rettig

abbreviated vita

I design and implement environmental sensor networks. I have been the sensor project manager for two projects exceeding 1 million dollars for both the National Science Foundation and the Environmental Protection Agency. I was co-founder of a sensor networking company raising over 500k in start-up capital.

Currently, I am teaching at the University of Dayton within the Electrical Engineering and Geology Departments where I have received funds for Internet of Things pedagogy development and the implementation of living labs for experiential student education. I also work with scientists from the United States and Canada as part of an Earth Science Information Partners project to create and maintain a wiki for sensor networking best practices.

My work and publications contribute to affordable and standardized real time geospatial sensor webs. To achieve my goals, my approach combines traditional sensor networking research with the current trends within the Internet of Things. Lastly, I advocate the use of open innovation, encouraging collaboration and open source software within distributed sensing.

curriculum vitae


Andrew_Rettig_cvFuture.pdf