6.033 Spring 2015 Design Project
Ambient Sensing for the MIT Campus

Last update: 4/20/2015, 2:00pm — See Errata for details
Also check out the list of Frequently Asked Questions (last updated 4/22/2015, 2:15pm)

0. Due Dates and Deliverables

There are three deliverables for this design project.

1. A proposal not exceeding 2000 words, due on March 20, 2015 at 5:00pm.
2. A 10-to-15-minute oral presentation given to your recitation instructor, to be scheduled with your recitation instructor, for some time between April 13, 2015 and April 23, 2015. The oral presentation will assess your progress and provide some feedback prior to your final report submission.
3. A design report not exceeding 5000 words, due on May 8, 2015 at 5:00pm

Each deliverable will have specific guidelines, which will be linked above.

The proposal and final report should be submitted via the online submission site. As with real-life system designs, 6.033 design projects are under-specified, and it is your job to complete the specification in a sensible way given the stated requirements of the project. As with designs in practice, the specifications often need some adjustment as the design is fleshed out. Moreover, requirements will likely be added or modified as time goes on. We recommend that you start early so that you can evolve your design over time. A good design is likely to take more than just a few days to develop. A good design will avoid unnecessary complexity and be as modular as possible, to enable it to evolve to changing requirements.

Late submission grading policy: If you submit any deliverable late, we will penalize you one letter grade per 48 hours, starting from 5 pm on the submission day. For example, if you submit the report anywhere from 1 minute to 48 hours late, and your report would have otherwise received a grade of "A", you will receive a "B"; if you submitted 49 hours late, you will receive a "C".

You must work in teams of three for this project. All three people on a team **must** have the same recitation instructor (you may team up with people from either of your instructor's sections). Note that although this is a team project, some of the deliverables have individual components. See the individual assignment links (above) for more information.

1. Introduction

Over the past few years, small hardware devices that combine computation, communication, and sensing have become inexpensive, thanks to advances in miniaturization, low-power design, and radio communication. Several thousand such battery-operated sensors can now be deployed in the buildings on the MIT campus at modest cost to help us understand the campus environment.

Many companies now sell such sensors: some examples are shown below.

Temperature (Nordic Semi)

Humidity (BeeWi)

Building sensors (Eve)

MIT facilities is in the final stages of procuring many such sensor nodes to monitor buildings on campus and build a campus-wide *ambient sensing* system. Some sensors record temperature, some record humidity, some detect water leaks, some record light intensity, some record vibrations, some detect smoke, some detect carbon monoxide, and so on. They will be attached to walls or placed in other suitable locations in rooms or corridors.

Sensor beacon on a wall (Estimote)

The sensor node includes a programmable general-purpose processor, some memory (64 kbytes RAM and 64 kbytes ROM), flash storage (8 Mbytes), a real-time clock, a low-power radio for communication, and the sensor itself. A sensor's reading contains a 32-bit timestamp and a 16-bit value. The communication technology uses a low-power radio with an energy-efficient link-layer protocol and a low communication range, called *Bluetooth Smart*

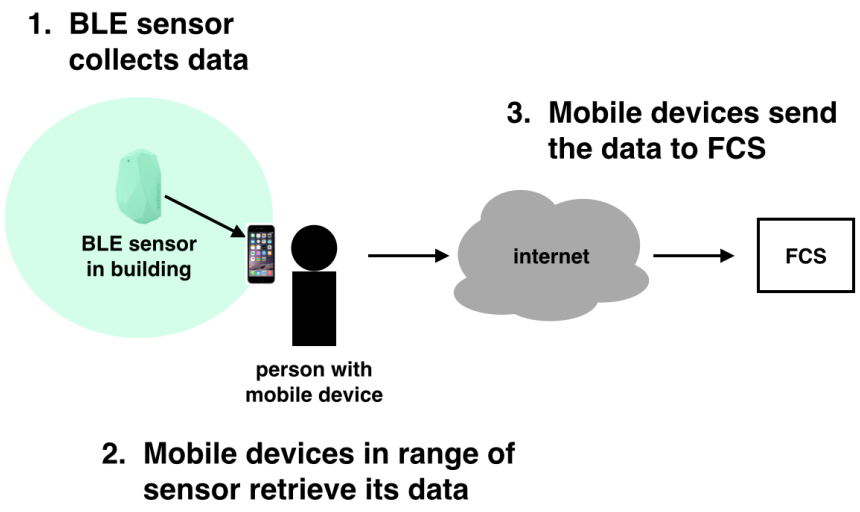
(aka *Bluetooth Low Energy*, or *BLE*). It is now widely available on many devices; the appendix has some details about how it works.

Facilities is interested in the following uses of a campus-wide sensor-node deployment:

1. **Archiving:** For each deployed sensor, the *Facilities Central Server (FCS)* should obtain a timestamped sequence of sensor readings to maintain a historical record for various kinds of analyses over the data. The system should be able to identify the location (MIT room number for rooms or a suitable location for places like the Infinite Corridor, such as "Infinite Corridor, Building 3, 1st Floor"), so that applications can bind any given sensor's data to a human-understandable location on campus.
2. **Anomaly detection:** Sensor nodes implement simple rules, specified and configured from the FCS by a building administrator, to proactively identify anomalous conditions in their sensor data. The goal is to communicate these conditions to the FCS as promptly as possible (given the communication constraints specified below; the opportunity for real-time synchronous communication is rare).
3. **User reports of issues:** Allow users to report problems via a mobile app running on their phone (or tablet or laptop), using a mobile to capture data from sensors to provide detailed evidence. For example, allowing users to report "my room is too cold".
4. **User retrieval of archived information:** Allow a user with authorization (established via MIT certificates) to retrieve archived information about public spaces as well as locations for which they are authorized (e.g., a student's dorm room, a staff member's office, etc.).

One approach to supporting these uses is by installing BLE access points all over campus to allow the sensor nodes to communicate with the FCS. Given the size of the MIT campus and the communication range of the BLE radios (about 10-20 meters in this system), Facilities has concluded that their budget does not permit setting up a dedicated infrastructure of BLE-equipped access points: it's just too expensive to purchase and maintain.

Fortunately, popular smartphone and tablet devices running iOS and Android now support BLE. For example, all iPhones 4S and later running iOS 7.0+, as well as many Android devices running Android 4.3 or later support BLE. Facilities estimates that almost all smartphones and tablets used by MIT students and staff will soon have BLE (if they don't already). They plan to use the mobile devices belonging to participating MIT users as *mobile gateways*, illustrated below. When a mobile device comes within BLE communication range of a sensor node, a background application running on the mobile device attempts to retrieve data from the sensor node to deliver to the FCS. As people walk around campus, they help collect data and also help the FCS issue directives to the sensor nodes. For less-populated parts of the campus, mobile devices carried by Facilities personnel and the janitorial staff provide coverage.



The Institute Committee on *Identifying Hot Things in Freezing Places* (aka IHTFP) has decided to form the IHTFP Task Force to design a system for campus-wide ambient sensing with the following elements:

1. **Mobile devices as data gateways**: The "gateways" that allow the sensor nodes to communicate with the FCS are the mobile devices carried by users. A user chooses to participate in the system by installing a mobile app on their device. The app is capable of operating in the background, listening for advertisement messages periodically broadcast by the sensor nodes. Such advertisement messages, or beacons, are a standard part of the BLE protocol; they include a 128-bit *region identifier*, a 16-bit *major id*, and a 16-bit *minor id*. You can set these three fields to anything you want for each beacon. The mobile app (described below) will be woken up by the mobile device's operating system when it comes within range of a specified *region identifier*; you may assume here that when the app is in the background, this wake-up will occur between 1 and 60 seconds after the device comes within range, but in best-effort fashion. Some useful details about BLE are in the appendix. The phone can search for at most 64 different region identifiers at a time.
2. **Mobile app:** When a user is close to a sensor node, the mobile device's operating system may wake the app up to run in the background, allowing the app to connect to the sensor node via a reliable BLE connection. To simplify the software on the sensor node, the sensor node maintains at most one active connection with a mobile device at any time. When connected, your design should decide whether you want a sensor node to broadcast advertisements or suppress them.

   When connected, the app and sensor node communicate over a protocol (which you must design) to retrieve data from the sensor node. In addition, the app may issue directives (originating from the FCS) to the sensor node over a protocol (which you must design). Examples of such commands include changing the configuration of the

node to set the threshold value above or below which the sensor node considers its readings anomalous, and to set the periodicity at which the node's sensor takes readings.

3. **Network protocol**: The system's network protocol should permit the FCS to communicate with the sensor nodes via the mobile app. It should include the methods used to communicate between the mobile app and the sensor nodes, as well as the FCS and the mobile app.
4. **User queries**: When run in the foreground, the app's user interface allows a user to retrieve sensor data for a specified time range for (1) any public space and (2) any space for which the user is authorized. The authorization uses MIT certificates; you may assume that the set of private spaces authorized for the user is specified in the certificate.
5. **Sensor processing and storage management:** Each sensor node wakes up periodically (some every second, some every few seconds) and records a reading along with the current time (from the real-time clock). Each sensor node has a certain amount of Flash storage, about 8 Megabytes. It should be used to store sensor data in a well-defined way, and the system must incorporate methods to manage this storage while supporting queries from the app to retrieve data and send it to the server.
6. **FCS data model**: To support queries from Facilities personnel and MIT users, and to allow Facilities administrators to maintain the configuration of the different sensor nodes, the system must implement a data model that specifies what data should be stored in databases or files on the FCS. These queries include:
    a. For a specified location, for a specified time range, for a specified sensor type, return the time-series of values archived, or an aggregate over those values (average, standard deviation, count, X^{th} percentile).
    b. Aggregate queries over a portion of a building, or over an entire building, or over a part of campus, such as the average temperature over a specified time range, or the average temperature during weekends, or the average temperature over a specified time-of-day range (say, midnight to 6 am) over a sequence of days.
    c. Which sensors on campus (or a specified location) have reported anomalous behavior over a specified time range?

The IHTFP Task Force has decided to enlist 6.033 (i.e., *you*!) to design this system. Your design should specify the queries supported by your system and show how your system's data model supports them. The richer the set of supported queries, the better, but use your judgement to avoid excessive complexity.

Assumptions

You may assume that the user's mobile device and app are trusted and will not maliciously attempt to tamper with or suppress data or configuration information. You may also assume

that the system supports “over-the-air” reprogramming of the sensor nodes using a method supplied to you.

You may **not** assume that a user’s mobile device is always connected to the network. A user may turn their phone off, lose service, etc., at any time.

The intended scale of the system is the entire MIT campus. MIT Facilities is responsible for 158 buildings covering a total indoor area of 12 million square feet. You may assume the average density of sensor nodes is one every 120 square feet.

Assume that each sensor’s data is recorded using a 32-bit timestamp (accurate to 1 second) and a 16-bit value.

Please note that the specifications provided here are not necessarily complete. You may ask questions of the staff to understand the requirements better. If you need to make any assumptions not stated in this document, please state them, and if necessary, briefly justify why they are reasonable.

As is customary in many real-world projects, it is likely that new requirements will emerge after you have started working on your design. A good design will have the modularity and simplicity to accommodate mid-course changes.

Additional Requirements

Since the start of this project, experiments have shown that the clocks on the sensor nodes drift and are not accurate. The IHTFP Task Force has added the requirement that the system should use the phone and/or FCS to set the sensor node's clock whenever possible.

Furthermore, the task force has requested an additional feature: Any mobile device in the vicinity of a sensor should be able to retrieve the most recent value recorded by the sensor, if one is available, and show it to the user on request regardless of authorization.

If you are unable to design a system that meets these additional requirements, your design report (and presentation) should explain what parts of your system break down in the face of these requirements and how.

2. Your Job

Your job is to design the system described above.

Design Considerations

While designing the system, you should consider the following questions:

- Naming: how should each sensor's region, major and minor IDs be named? How would you support location-specific queries?
- Sensor node query interface: What query interface to retrieve data does your system support? Bear in mind that your goal isn't to build a database on the sensor node, but primarily to ensure that as much of the sensor's stored data gets delivered to the FCS.
- How will the system detect and handle duplicate pieces of data that may be delivered by multiple phones?
- How can the system attempt to ensure that all logged data is delivered to the FCS? Note that the delivery of *all* data is not a strict requirement, but your system should attempt to reduce the amount of data missing.
- How can the system ensure that anomalies are prioritized over non-anomalous data meant for the historical record?
- How will the sensors decide when to remove data from its local storage? What should a sensor do when new data is logged? (As a general rule, we expect older data to be less useful or valuable than new data, unless the old data has recorded an anomaly and that has not been delivered successfully to the FCS.)
- The IHTFP task force would like each sensor node to last four years or more before its battery runs out and the node needs to be replaced. The appendix below defines the sensor node's power consumption parameters. Your design should propose (via calculation) a battery capacity for each sensor node (in milliAmp-hours, mAh) to achieve this four-year lifetime goal. As a practical matter, the battery capacity ought to be under 1600 mAh.
- Bear in mind that there is an energy cost on the sensor node to initiating and maintaining a connection from the mobile device to the sensor node. The energy cost on the mobile device is small and may be ignored, but you should prevent bugs or malicious behavior of a mobile device to drain the battery of a sensor node down. For example, if you're in your room for 8 hours, remaining connected continuously is a bad idea. Think about how you would design the duration and periodicity of connections to achieve the desired sensor node lifetime (and you get to pick a battery capacity as well). Your solution should incorporate appropriate mobile device behavior, but should not assume that all mobile devices are correctly implemented (i.e., a sensor node should defend itself from running its battery down too fast).
- Mobile devices like smartphones and tablets also have sensors: can you think of ways in which you can use a smartphone's sensors to augment the information from the sensor nodes that might help your system or provide new features? If so, what sensors would you use, and what are the trade-offs in using them? One example might be position sensors, but unlike GPS outdoors, indoor location on mobile devices today isn't too accurate. You can get building-level accuracy and accuracy to within a couple of floors of a building, but no better than that reliably.
- Should the server infrastructure identify users' mobile devices when they are used to deliver data? Or, would you treat them as anonymous devices? What are the trade-offs between these choices?

- How does your system support the addition of new sensors to the infrastructure? You may assume that all sensor nodes are added only by facilities personnel.
- How would your system identify whether a sensor has been moved around to another location? You may assume that people don't maliciously move these around, but that a small fraction of them may be moved accidentally.
- How would your system identify malfunctioning sensors?
- How would you entice or incentivize MIT users to participate in your system?
- Using mobile phones as a key part of the system means that it will be important to have something that's easily evolved, ported, or reimplemented as those platforms evolve. Your design should minimize complexity of the mobile device components, keeping this in mind.

Components of the Design

Your design should describe how the following four components are designed, how they interact with each other to accomplish the six elements described above, and how they address the considerations laid out above.

1. **Sensor node software:** The software in the sensor node that keeps track of its data and eventually communicates with a participating mobile device.
2. **Communication protocols between:**
    - the FCS and the sensor nodes (by which we mean the message format and what the fields mean): note that these communications are done via the mobile device, but the *end-to-end* message semantics are between the FCS and the sensor nodes
    - the mobile device and the sensor node (by which we mean the "query" protocol between the mobile device and sensor node to retrieve data, and the "command" protocol to configure new thresholds or change other aspects of the sensor node's operation). The messages that phones use to connect to the sensors are defined by the BLE spec, but it is up to you to define the sensors beaconing behavior as well as how it responds to phones initiating connections.
    - the mobile app and the FCS (the message format used by the mobile app to send data to the FCS and receive data from it)?
3. **The FCS software:** the way in which it stores data (files, databases, etc.), its interface to the mobile app, its communication with the sensor nodes, its handling of sensor data arriving via the mobile app, and its interface to facilities administrators to configure sensor nodes.
4. **The mobile app:** what are its components and modules? What does it do in the background and what does it do in the foreground? What data does it store and how does it help the FCS and sensor nodes communicate with each other?

Extensions to the Design

If your design is complete and you are looking for an extra challenge, then extend it to support the scenario when the mobile device is not trusted. What damage could a malicious mobile device or mobile user cause? How would you overcome these problems?

Appendix: BLE

BLE is a low-power communication protocol. For the purposes of this project, on average, the radios have a range of about 30 meters (100 feet). BLE is a point-to-point protocol between a *peripheral* and a *central* node. In the envisioned system, the sensor node is the peripheral and the mobile device is the central node. The peripheral *advertises* its existence periodically using beacons, which include a 128-bit service identifier, a 16-bit major identifier, and a 16-bit minor identifier. To ensure a sufficient battery life, the sensors are configured to broadcast their advertisements once per second. The mobile devices initiate connections to the sensor nodes; the sensor nodes cannot. As mentioned earlier, at most one active connection can be active at a time on a sensor node, but the mobile device may have hundreds of BLE connections active concurrently.

Before installing a sensor node, the system administrator may configure the three fields of its advertisement to whatever your design suggests. You may assume that there is no error made here. In practice, the Each sensor node has a 48-bit globally-unique identifier set by the factory that makes the hardware device (this identifier is not part of the advertisement).

BLE uses short data packets, up to 20 bytes in size. The details of the link-layer protocol are not relevant for this project, but it does not guarantee perfect reliability. As a user walks around, the link-layer protocol allows for a data rate of 4 kbytes/s.

Sensor node energy considerations: The sensor node runs on a 3 Volt battery. When advertising at 1 Hz, the average current drawn is 25 μA, measured over the entire second. It takes 2 seconds for the mobile device to establish a connection after the mobile devices initiates one, after which it can transmit at the BLE data rate of 4 kbytes/s until it disconnects. During the connection setup, as well as when a connection is active and transmitting data, the current drawn is 1 mA (1000 μA). Most of the power consumed by a sensor nodes is from BLE communication; you may assume that the rest of the processing and sensing consumes negligible power (this assumption is realistic).