Project Sundial:
Using Networked Sensors to Detect Ambient Light

Miranda Lee
University of California, Los Angeles

Summer Undergraduate Program in Engineering Research at Berkeley
SUPERB, Summer 2002

Faculty Advisor: Dr. Ruzena Bajcsy
Graduate Mentor: Barbara Hohlt

Electrical Engineering and Computer Sciences
College of Engineering
University of California at Berkeley
Project Sundial: Using Networked Sensors to Detect Ambient Light

Miranda Lee
Faculty Advisor: Dr. Ruzena Bajcsy
Graduate Mentor: Barbara Hohlt
SUPERB-IT, UC Berkeley, Summer 2002

Abstract

An understanding of the capabilities of the Mica mote light sensors is currently limited. To determine how much information data from the light sensors reveals about the environment, we performed a series of experiments designed to test the sensitivity of the light sensors to the time of day, the presence/absence of close objects, and different colors. Measuring the ambient light alone, we discovered that the light sensors can provide an approximate gauge of the time of day (i.e., morning, noon, evening). Depending on the object size, there is a definite range within which an object’s presence can be detected by the sensor motes. Beyond that range, the object cannot be clearly detected. The light sensors can also distinguish between different colored environments. Finally, using a six-sided sensor, we can determine the direction of a light source. We discuss these findings in this paper, as well as the limitations of the mote light sensors.

1. Introduction

In this project, we use tiny, wirelessly networked sensor devices called motes to detect ambient light, and we generate a graph of the light intensity over time. Since the amount of sunlight on a given day is correlated with the time of day, data from the light sensors provides a rough measure of time, almost like a natural clock or sundial. For this reason, we have named this project, Sundial. In addition to the time of day, we want to determine whether the motes can provide us with other useful information about the environment.

Because sensor motes are a fairly new technology, their capabilities have not yet been fully explored. This project is in part an effort to discover the strengths and weaknesses of these devices, and to assess their potential for various applications. If we can show that sensor data from the motes provides accurate, reliable, and substantial information about the environment (e.g., we can predict the time of day from the light intensity), the sensor motes can be used in a wide range of sensing applications, including energy-efficient motion detectors or surveillance instruments and other force/field detectors.

This paper is organized as follows. Section 2 describes background information about networked sensor devices, or the hardware part of the project, which is being developed by Kris Pister’s research group at UC Berkeley. We also

---

1 A sundial is a device consisting of a shadow stick centered on a plate that displays the time of day. The time is determined from the shadow cast on the plate by the stick. Because the length and position of the shadow depends on the intensity and direction of light from the sun, the time displayed is a function of sunlight.
discuss TinyOS, the software architecture for the sensor devices, which has been the focus of David Culler’s research group. Section 3 covers the objectives of Project Sundial, as well as the experiments we conducted. There were two basic experimental setups. The first setup involved an application called Chirp, which programs a sensor mote to periodically send messages over the radio containing light sensor readings. The second setup used an application called Getsix, which is a modification of Chirp designed to gather data from six sensor inputs. Finally, we discuss the results and conclusions of our experiments in sections 4 and 5.

2. Background

2.1 Networked Sensor Devices

Networked sensors are devices that combine sensing, computation and communication all in a tiny package, along with a power supply. Sensing is accomplished through various sensors (e.g., temperature, light, and acceleration sensors) and computation is handled by a microprocessor. Communication can take various forms: wired, short-range RF, infrared, or optical. The sensors that we use in this project communicate with each other via a 916.50 MHz radio, which allows them to form wireless networks. [1, 2]

Networked sensors are particularly interesting because if successfully implemented, they can be used in a variety of applications – both for sensing and for actuation. Their small size enables them to be placed almost imperceptibly in any location. They can be attached to objects or scattered throughout the environment. Because networked sensors have self-contained power sources (either batteries or solar energy), they are also autonomous. Wireless communication allows the sensors to send information either to neighboring sensors or to a base station sensor (which communicates to all other motes and the computer) and to receive information back. Together, these characteristics enable the sensors to form ad hoc networks, or self-arranging and adapting distributed systems.

Many applications have been proposed for networked sensors. One team at UC Berkeley led by Kris Pister is working on reducing the sensors down to the millimeter scale, almost to the size of dust particles. They call the microsensors ‘Smart Dust’, and they hope to deploy thousands of these miniature sensor-computers into the environment for various applications, including: meteorological, geophysical, and planetary measurements; examinations of insects or other small animals; and stealthy observation of hostile areas. [2] The Smart Dust team’s effort is part of a universal trend in computer science, in which the focus of research has been migrating toward the goal of ‘ubiquitous computing’. Ubiquitous computing refers to the widespread and deep integration of computers throughout the environment so that computers are everywhere, but are effectively invisible.

One of the biggest challenges in implementing Smart Dust is power efficiency. A characteristic of any sensor device is that as size decreases, or as the battery/power supply (the largest component) shrinks, the energy capacity must also decrease. The energy limitations place constraints on the number of sensors that a mote can use, the amount of
computation it can perform, and the distance that it can communicate. As a result, the sensor devices have limited resources and have to be extremely power efficient.

So far, the Smart Dust team has developed several prototypes of the wireless sensor device that use off-the-shelf components and are about one to two inches in size. These larger sensor devices are called motes. A mote is another name for a small particle or speck. Several versions, or platforms, of this prototype have been made, including the disk-like Dot mote, the mid-sized Rene mote, and the larger Mica mote. These platforms differ in their sensing and computational capabilities. For example, the Mica mote can sense light, temperature, magnetism, acceleration, and sound, while the Dot mote can only detect light and temperature. We use the Mica mote in all of the experiments for this project.

The Mica mote consists of five major subsystems: processing, radio frequency (RF) communication, power regulation, input/output (I/O), and secondary storage. An 8-bit microcontroller with 128 Kbytes of flash program memory and 4 Kbytes of system RAM handles most of the processing, while a 916.50 MHz transceiver and antenna perform most of the RF communication functions. Power is regulated by a DC-DC converter, which provides a constant 3.0V supply and operates off a pair of AA batteries. Finally, a 51-pin connector provides an I/O interface for different sensor- and programming- boards, and a 4 Mbit external flash chip provides secondary storage. [3, 1]

A programming board is used to connect the mote with the computer. To install programs into the Mica mote, we fit the 51-pin connector of the mote into the connector on the programming board, and we plug the parallel port of the programming board into the computer. The serial port of the programming board is used for transferring data between the mote and the computer. We connect the programming board to the serial port of the computer when we want to send information from the mote to the computer or vice versa. (See Figures 1 and 2.)

Using the 51-pin connector, we can also attach different sensorboards onto the Mica mote (Figures 3 and 4). The basic sensorboard contains a temperature and a photo (light) sensor (as well as a prototyping area with empty pads), while the Mica sensorboard contains, additionally, a microphone, a buzzet/sounder, an accelerometer, and a magnetometer. As we can see, the sensors occupy very little space. The largest component of the mote is the battery. Therefore, to reduce the device size, we would need to shrink the battery, which would decrease the energy capacity.
Consequently, to use these devices effectively, we need good software, or a system architecture that efficiently manages the hardware. With the right software, we can program the motes to perform a wide range of applications.

**Figure 3 (left)**: Basic sensorboard.  
**Figure 4 (right)**: Mica sensorboard.

### 2.2 TinyOS: Tiny Operating System

Researchers in David Culler’s group at UC Berkeley solved the software challenge posed by the motes. They designed an efficient and flexible operating system called Tiny Operating System, or TinyOS, that meets the requirements of the hardware. In particular, TinyOS efficiently handles multiple and simultaneous information flows while operating within the memory, processor and power constraints, and its modular structure supports a diversity of applications.

To handle the high levels of concurrency (i.e., many operations running at once within a limited amount of space), TinyOS takes an *event-driven* approach to programming. That is, events/commands are triggered by other events/commands. For example, sensor data gathering is set-off/initiated by a clock tick (or an instant of the microprocessor’s internal timing device). Event handlers are called *components*, and programs or applications consist of graphs of components, or *component graphs*. Component graphs were designed to be a metaphor for ‘wiring’ components together to create a working program. [4]

Three basic files are required to specify a component graph or program: the component file (.comp), the source file (.c) and the description file (.desc). These files are written in the C programming language and make extensive use of preprocessor macros. *Component files* specify the interface of the program, and are similar to header files in that they contain declarations and prototypes of functions. Unlike header files however, component files also list the functions that the component requires. *Source files* specify the implementation or functionality of a component, and *description files* link different components together. In addition, the compiler automatically generates a header file (.h) each time a program is compiled.

A component graph can have several layers. Application level components sit on the top layer, while components that directly control the hardware are at the bottom of the component graph. Components execute three kinds of procedures: *events, commands, or tasks*. *Events* are procedures carried out by components that are located higher on
the graph than the requesting component. When a component calls (or signals) an event, it makes an upcall. The component that carries out the event is said to handle the event. On the other hand, commands represent procedures performed by components below a requesting component. When a component calls (or uses) a command, it makes a downcall. Components that perform the command are said to accept the command. (See Figure 5.) Finally, tasks are procedures that run asynchronously with respect to events and allow us to simulate concurrency. Tasks perform the primary work in a program. [4]

![TinyOS Component Diagram](image)

**Figure 5:** TinyOS Component

### 3. Project

#### 3.1 Purpose

This project has gradually evolved over the past eight weeks, but the general goal has been to determine how much information data from the light sensors reveals about the environment, and how accurately the measurements reflect reality. Among other things, we are interested in the sensitivity of the light sensors to the time of day, the presence/absence of close objects, the distance from objects, and different colors. If we can show that light intensity data from the sensors provides accurate information about these and other features of the environment, the sensors could potentially be used in a wealth of useful sensing applications.

This project was motivated by the desire for highly distributed, energy-efficient information systems. Such systems are of special interest to the Center for Information Technology Research in the Interest of Society (CITRIS) which has been seeking technological solutions to a variety of economic, educational, environmental, health and safety problems in California. If successful, the networked sensors could be the foundation for the massive, societal-scale information systems envisioned by CITRIS.

#### 3.2 Setups

Two main questions naturally arise as we determine the capabilities and limitations of the sensor motes:

- What does the light sensor data reveal about the environment?
- How significant is the data gathered from these devices?
To answer these questions, we devised several small experiments, using two basic setups: Chirp and Getsix.

**Chirp setup**

In the first setup, we programmed one mote with an application called **CHIRP**, which periodically sends out radio messages containing light sensor readings, and we attached to it a Mica sensorboard. A second mote was programmed as a base station using the **GENERIC_BASE_HIGH_SPEED** application, and was connected to the computer via the serial port. A Serial Forwarder program on the computer then read the data from the serial port and a Visual Basic program opened up the Winsock to process the data. The Chirp setup is shown in Figure 6.

![Figure 6: Chirp setup](image)

We can understand how the **CHIRP** application works by looking at the component graph. The **CHIRP** component graph has three layers. The top layer consists of the **MAIN** component, which is present in all applications. The program starts when **MAIN** initializes the **CHIRP** component (using the command **CHIRP_INIT**). **CHIRP** in turn initializes the **CLOCK** (using the command **CLOCK_INIT**) and the **GENERIC_COMM** component (using **SUB_INIT**). An LED flashes on and off, and then **MAIN** commands **CHIRP** to start (**CHIRP_START**). When **CHIRP** starts, it issues a command to the photo sensor to collect data (**GET_DATA**). At the next clock tick (**CLOCK_EVENT**), another command is made to get data from the photo sensor. At each data event (**DATA_EVENT**), **CHIRP** commands **GENERIC_COMM** to send the message over the radio (using **SUB_SEND_MSG**) and to turn the radio power on (**SUB_PWR**). When a message is sent, **GENERIC_COMM** sends a flag notifying that the message was sent (**SUB_MSG_SEND_DONE**). The program continues in this manner with each clock tick. (See Figure 7.)
In addition to the component graph, the program code also defines the structure of the message packet. Message packets from the Mica motes contain 36 bytes, the first four of which comprise the header (bytes 1-4). The header is defined in a separate file called MSG.h and is generally fixed, while the data payload (bytes 5-34) is application specific. In CHIRP, the first byte of the data payload (byte 5) contains the ID number of the sending mote and the next two bytes (bytes 6 and 7) store the light sensor readings. Light sensor readings take values between 0 and 1024 (10-bit sensitivity), and hence require two bytes for storage. The integer values represent the quantized output values of an ADC channel, which converts the output voltage (of a voltage divider formed by the photoresistor and a load resistor) into an integer. The remaining data bytes are zeros, which conveniently provides a mechanism for us to filter out erroneous packets: if a zero byte is nonzero, we know that the packet contains errors. (See Figure 8.)

Once the base station receives a packet from the Chirp mote, a Java program called Serial Forwarder connects to the serial port and allows for communication between the base station and computer. Because the Serial Forwarder is socket-based, it can be accessed by different applications using the Winsock. For example, we can create a Visual Basic program that opens up the Winsock and collects data from the base station for processing. Dan Steingart wrote a VB Listener/Logger program that opens up the Winsock, collects packets, parses each packet into 36 bytes and graphs selected byte values. Since light sensor readings from Chirp occupy two bytes of the data packet, we modified his code slightly to calculate the light intensity measurement from the two bytes and to plot the intensity versus packet number. To calculate the light intensity from the two data bytes, we multiplied the first byte (which represents the two most significant bits of the measurement) by 255 (or the largest 8-bit binary number) and added

![Figure 7: Chirp component graph](image)

![Figure 8: Chirp packet structure](image)
the last 8 bits stored in the second byte. Figures 9 and 10 show screen captures of the Serial Forwarder and Visual Basic program interfaces, respectively.

Getsix setup

The second setup added a little more complexity. Instead of the Mica sensorboard, a customized basic sensorboard with six light sensors (which could be arranged in a cube configuration) was attached to the sensor mote. We needed a six-sided sensor because we wanted to collect data about ambient light from all six directions in 3-D space. Each of the six sensors could then constitute a ‘pixel’ in the light data, like in an image. The sensor mote was programmed with an application called GETSIX (written by Nate Ota), which periodically sends packets over the radio containing light sensor data from all six sensors. (See Figure 11.) In both setups, we set the clock tick rate to be approximately one tick per second. Like the Chirp setup, a second mote was programmed as a base station, which received the message packets from the sensor mote wirelessly. The Serial Forwarder was then used to extract the data from the serial port, and a Visual Basic program processed the data.
Because the Getsix program is modeled after Chirp, their component graphs look very similar. Instead of a PHOTO component however, the Getsix program uses a SENSEALL component to collect data from six ADC outputs. (See Figure 12.)

![Figure 12: Getsix component graph.](image)

The packet structures of both applications are also similar. However, instead of a single sensor reading, the Getsix message packet contains six sensor readings, stored in 12 bytes. To reduce the variation in readings among the different sensors, we reduced the sensitivity of the sensors from 10-bit to 8-bit. As a result, the light sensor readings take values from 0 to 255, and only one of the two data bytes for each sensor contains information from that sensor. (See Figure 13.)

![Figure 13: Getsix packet structure.](image)

A big challenge in assembling the Getsix setup was constructing a sensorboard with six sensors. To construct the sensorboard, we used the prototyping area provided by the basic sensorboard. We removed the thermistor and the existing photocell, and soldered on six new photoresistors, along with six 2.2kΩ load resistors. Each photoresistor was placed in series with a load resistor to create a voltage divider. To ensure that the output voltage increased with increasing light intensity, we placed the photoresistor in between the power supply and the load. The load resistor was grounded on one end and connected to the ADC on the other.
Essentially, the Getsix sensorboard consisted of six voltage divider circuits. To avoid confusion, we color-coded the wires corresponding to each sensor, using the resistor color code (i.e., brown = sensor 1, red = 2, orange = 3, yellow = 4, green = 5, blue = 6). To build the circuits, we started by soldering a wire to the analog ground pad. We then soldered one end of each of six 2.2k\(\Omega\) resistors to the grounded wire. The free ends of these resistors were then soldered to colored wires (each approximately 4 inches long) coming from six ADC pads. We also soldered 4-inch long colored wires to six PW pads. Finally, we attached the photoresistors to the wire leads from the ADC and PW pads. Because the photoresistors were attached to the ends of wires, they could be arranged in different positions. For the Getsix experiments, we positioned the sensors so that they were orthogonal to each other.

4. Results

4.1 Chirp Experiments

Time of Day

We conducted a number of experiments using the Chirp setup. The first experiment involved measuring the ambient light intensity as a function of time. To do this, we programmed a mote to collect light sensor data for each hour of a day between 7:30am and 9:00pm (from sunrise to sunset) and we graphed the light intensity over time. This graph could then be inverted and used to predict the time of day, given arbitrary light sensor readings.

We ran the experiment from a 7th floor dorm room with a single window. To avoid interference from hallway lights and activity, we closed the door. In addition, we turned off all lights, so that only ambient sunlight was measured (the only artificial light came from the laptop screen). The weather that day was cloudy in the morning and sunny at noon. To generate the graph in Figure 15, we collected approximately 100 packets at the beginning of each hour and took an average of the light intensities for each hour.

Looking at the graph, it is difficult to make general conclusions about the sensitivity of the light sensors to the time of day, because the experiment depends heavily on environmental conditions. For example, we could obtain completely different results on a rainy day. Also, buildings could shield some of the sunlight, resulting in misleading measurements. However, we can conclude from this experiment that the sensors at least give a rough measure of the
time of day (i.e., morning, noon, evening). Of course, we could modify the experiment to give a more precise measure of time. One way to do this is to use the six-sided sensor cube to take sensor readings from six different directions, from which we could deduce the position of the sun in the sky. Combining this data with information about the latitude, we could theoretically calculate the local time of day, just as from a sundial. Knowing the position of the sun in the sky, we could also determine the geographic orientation of the light sensor. We will discuss this experiment in a later section.

![Light Intensity vs. Time](image)

**Figure 15:** Graph of light intensity measured by a Chirp mote versus time.

**Distance from Occlusion**

In another experiment, we measured the light intensity as a function of distance from occlusion. Using a box (with approximate dimensions of 7” × 4.5” × 2”, large enough to completely cover a mote), we measured the light intensity as we moved the box away from the light sensor. Therefore, by distance from occlusion, we are referring to the distance between the mote and the box/covering. The point of this experiment was to determine how sensitive the sensor is to the proximity of an obstruction. In particular, we wanted to determine the limit beyond which having the box hang over the light sensor was the same as not having the box at all. This data could be useful if we wanted to add lenses to the motes.

Looking at the results in Figure 16, we can see that there is a definite distance beyond which the object is outside the detection range of the mote. In this case, using a 7” × 4.5” × 2” box, the range can be approximated around 6 inches.
At distances further than 6 inches, the presence/absence of an object of this size is difficult to distinguish using a mote. From the graph in Figure 16, we can also estimate the distance of an object away from the sensor, provided that the object is within the detection range. Such information could be particularly useful in motion detection applications; we could determine whether an object is moving closer to or farther from a sensor depending on its distance. It is important to note that the sensitivity range can differ for objects of different sizes. A tiny object for example, will probably be detectable only within very close distances, because smaller objects do not obstruct as much light as larger objects. This experiment was a one-dimensional sensitivity analysis of the sensor, in which the dimension in question was the distance of an object of a given size from the sensor. We could extend the dimensionality of this experiment by running similar tests on objects of different sizes.

![Graph of light intensity measured by a Chirp mote versus the distance from occlusion by a box.](image)

**Figure 16:** Graph of light intensity measured by a Chirp mote versus the distance from occlusion by a box.

**Color**

In the next experiment, we sought to answer the question: How well can the sensors discriminate between different colors? To test the sensitivity of the sensors to different colors, we lined the inside of a box with colored construction paper and placed the sensor mote inside the box with a single light source (a miniature flashlight that casts yellow light). The sensor mote was placed in one corner with the light sensor facing the (diagonally) opposite corner. The light source was placed in the opposite corner, facing the corner so that the only light directed toward the sensor was light that was reflected from the colored paper. Taking an average of the light sensor readings for
each color, we obtained a graph of the light intensity versus color. In the graph shown in Figure 17, the colors on the x-axis refer to the color of the paper that was used to line the inside of the box.

For the most part, these results agree with our expectations. For example, a white colored environment, which reflects all light, gives a higher sensor reading than the black colored box (since black absorbs all light). A possible reason why the sensor reading for the white paper was not the maximum value over all the sensor readings could be because the paper was not pure white.

![Light Intensity vs. Color (Yellow Flashlight)](image)

**Figure 17:** Graph of light intensity measured by a Chirp mote versus color of the surroundings using a yellow flashlight as the light source.

In general, at least two properties of the colored paper – its pigmentation and its texture – affect the absorption and reflection of light striking the paper surface. The amount of incident light from the flashlight that is absorbed depends on the pigmentation of the paper. Depending on the pigment, some or all wavelengths of light are absorbed, and the color that we see comes from the wavelengths of light that are not absorbed. For example, the red construction paper absorbs all wavelengths except those in the red range, which is why the color we see in the paper is red. The amount of reflection depends on the texture of the paper. Smooth surfaces, or surfaces that are made of particles of equal (or nearly equal) refractive index, will tend to reflect light at an intensity and angle equal to the incident beam. On the other hand, rough surfaces, or those that contain particles of different refractive indexes, will tend to scatter the incident light in different directions. [5] Looking at the construction paper that we used, we discovered that the white paper was fairly rough in texture (and therefore most likely scattered a great amount of the incident light), which further explains why the sensor reading did not register as high as we expected.
Another result we observed was that the yellow paper gave one of the highest sensor readings. This means that the yellow paper reflected a great amount of light, which is what we predicted, since the flashlight projected yellow-colored light. A fairly surprising result that we found was that the pink paper resulted in the highest sensor reading. This high sensor reading could be accounted for in part by the smooth surface of the pink paper, which tends to reflect a lot of light.

From the yellow paper/yellow flashlight experiment, we realized that the amount of incident light that was reflected depended partly on the color of the light. Therefore, our next task was to determine what effect the color of the light had on the light sensor reading for each of the paper colors. To test the effect of the light color, we repeated the experiment described above, using first a white LED, then a red LED and finally a green LED. The results of our experiments are shown below.

![Graph of light intensity measured by a Chirp mote versus color of the surroundings using various colored light sources.](image)

Figure 18: Graph of light intensity measured by a Chirp mote versus color of the surroundings using various colored light sources.

Looking at the graph, we can see that the light color indeed has an effect on the light intensity measured by the light sensor. For instance, the white paper reflected more light when using the white LED than when using the yellow flashlight. This was also true using the green light; i.e., more of the green light was reflected off of the white paper than the yellow paper. Comparing the data from the green LED with the other light colors, we also notice that, of all the light colors, the green LED resulted in the highest sensor reading when paired with the green paper. Intuitively, this makes sense; green paper reflects the wavelengths of light corresponding to green, so all of the light from the
green LED must have been reflected back to the sensor. On the other hand, red light is readily absorbed by the green paper, which explains why the sensor reading for the green paper using the red LED was so low. Likewise, green light is readily absorbed by red-colored paper, which explains why the sensor reading for the red paper using the green LED was so low.

True to our expectations, we also found that more light reflected back from the red LED using the warmer colored papers (pink, red, orange and yellow) than the cooler colors (green, light blue, dark blue, black). This was not true when we used the green LED, because green is a cool color. Therefore, we can conclude that the color of the incident light beam has a definite effect on the light intensity readings.

Nevertheless, we can identify a few trends that occur regardless of the color of the light. For instance, the pink paper seemed to reflect the most light for all cases. This most likely owed to the smooth, reflective texture of the pink paper. On the opposite end of the spectrum, the black paper always resulted in the lowest light intensity reading. Because black pigments tend to absorb all wavelengths of light, this was expected. We also noticed that the light sensors always measured higher intensities from the yellow paper for all of the light colors.

While these experiments with color tell us that the light sensors are sensitive to differences in color, they are not true reflections of the sensitivity of the sensor motes to colors under ambient light because we used flashlights and LEDs. However, we could modify these experiments to test the effect of different colors on the ambient light readings by instead placing colored objects around the sensors and measuring the reflections in the absence of flashlights and LEDs.

4.2 Getsix Experiments

Calibration

Before we could use the six-sided sensorboard to run experiments, we first had to measure the degree of variation among the six light sensors. Like all electrical components, different photoresistors tend to have slightly varying characteristics. In this experiment, we arranged the six sensors so that they were all oriented upward, pointing toward the overhead lights. After collecting 20 packets, we averaged the light intensities from each sensor. Since the sensors received equal amounts of light under this configuration, we expected them to give equal light intensity measurements.

Looking at the graph in Figure 19 however, we can see that variations clearly exist between the sensor readings. On average, sensor 4 gave the highest light intensity measurement (142.75), while sensor 6 gave the lowest measurement (97). The remaining sensors reported light intensities in the neighborhood of 120. Because the six measurements averaged to approximately 120, we decided to normalize all the measurements to 120. To do this, we calculated the deviation of each measurement from 120. These results are tabulated in Figure 20. Assuming that the
discrepancies between the photoresistors are linear with respect to each other (i.e., measurements are simply shifted up or down in magnitude from the ‘true’ intensity), each future measurement could then be adjusted by the deviation factor to compensate for any variations. For example, if sensor 6 initially measured an intensity of 135, we can calculate its normalized intensity (with respect to 120) by adding the deviation factor of 23. This gives a normalized intensity of 158. Normalization, or what we will refer to as ‘calibration’, makes analysis of the data much easier by allowing us to compare sensor readings from the six sensors. Without calibration, we would not be able to justifiably compare the six sensors.

![Comparison of Light Sensor Readings for a Given Light Intensity](image)

**Figure 19**: Graph of average light intensity measured by six different sensors on a Getsix mote for a given light intensity.

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Measured Intensity</th>
<th>Deviation from 120</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>120</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>120</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>113</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>142.75</td>
<td>-22.75</td>
</tr>
<tr>
<td>5</td>
<td>126</td>
<td>-6</td>
</tr>
<tr>
<td>6</td>
<td>97</td>
<td>23</td>
</tr>
</tbody>
</table>

**Figure 20**: Results of calibration experiment.

Having quantified the variations between the six sensors and determined the corrections required to eliminate the variations, our next task was to use the six sensors to determine orientation and time of day.

**Time of Day**

To capture a more true measurement of ambient light over time (i.e., the direction of the light and the intensity distribution), we repeated the Chirp time-experiment using the six-sided sensorboard. We arranged the sensors in a
‘cube’ configuration so that each sensor occupied one face of a cube. This arrangement allowed us to monitor the ambient light intensity from all six directions in 3-D space. We placed the sensor mote on the window sill of a 7th floor dorm room facing North, next to a Chirp mote, and collected sensor readings from both motes approximately once at the start of each hour, from morning to evening (7:30am –8:30pm). As in the Chirp experiment, we closed the door and turned off all lights so that only ambient light from the window was measured. The six sensors were oriented as follows: sensor 2 pointed upward, sensor 6 pointed downward, sensor 3 faced the window, sensor 1 pointed away from the window, sensor 4 pointed to the left (parallel to the plane of the window), and sensor 5 pointed in the opposite direction to the right. The results of our experiment are shown below (Figures 22 and 23). Each data point represents the average light intensity over 20 packets.

As we can see, the six sensors measured similar curves (with varying levels of intensity), all of which were generally consistent with the graph generated by the Chirp mote. As expected, sensors 1 and 6 (which were respectively, pointed away from the window/light source and downward) gave lower readings than the other sensors. Sensors 2 and 3, which pointed upward and toward the window, respectively, both gave high measurements; this result also conforms to our predictions. We can also see that the light intensity increased as the day proceeded until 7:30pm when the sun began to set. This is because the sun moves from east to west, and sets on the west. As time passed during the day (and as the sun moved westward), the light levels increased, resulting in progressively higher readings.

![Light Intensity vs. Time](image)

**Figure 22:** Graph of light intensity measured by a Chirp mote versus time.
Aside from verifying the results collected by the Chirp sensor, the Getsix sensors can provide us with additional information, particularly about directionality and orientation. By directionality, we are referring to the location of a light source. For example, we can determine the position of the sun in the sky at a particular instant in time by comparing the readings from the six sensors. Because the sensor with the highest reading corresponds to the sensor receiving the most sunlight, we could deduce the direction of the sun’s spotlight (and hence the sun’s position) simply by comparing the six sensor readings. This information could be useful in global positioning applications. Because the sun’s position above a specific location at a specific time is unique, information about the sun’s position, when coupled with knowledge about the time of day, could be used to geographically locate a sensor. (For example, the sun sets at a particular time on the west.) In fact, it is this principle – that each combination of the time, the location, and the sun’s position is unique – that makes sundials possible. Using the Getsix data and knowledge about the geographic location and time of day, we can also abstract information about the orientation of the sensors. Thus, by simply arranging six sensors in a cube, we can collect much more information than we can by using a single sensor.

4.3 Challenges

One of the greatest challenges in doing this project was designing controlled and systematic experiments. Whenever we measure features of the environment, there is always a high degree of variation, and an experimental setup is rarely the same as the one after it, simply by virtue of natural fluctuations and inconsistencies. This was especially
true when we conducted the experiments to measure the time of day. Another difficulty we faced involved the
sensor motes themselves, which occasionally sent erroneous packets or failed to send packets at all. Many times,
these problems were easily resolved by reprogramming the motes.

5. Conclusion

In doing this project, we gained a deeper understanding of how networked sensors work. We also learned how to use
TinyOS, and how to construct circuits on the basic sensorboard.

Additionally, we gained many new insights into the capabilities and limitations of the motes through this project.
Specifically, we found that:

• Using a single light sensor, we can obtain a rough measure of the time of day. A more precise estimate of
  the time of day can be found by using a six-sided sensor. This is because the six-sided sensor provides
  information about the directionality of a light source, in addition to the light intensity.

• Measuring the light intensity as a function of the distance from an object (with approximate dimensions of
  7” × 4.5” × 2”), we found that the sensitivity range of the sensors to objects of that size is approximately
  six inches. Beyond six inches, the light sensor does not do a good job detecting the presence/absence of the
  object. When the object is sufficiently close however, the sensor can detect it, as well as approximate the
  distance between the object and the sensor.

• The light sensors can differentiate between different colors, and the color of the light has a definite effect
  on the sensor readings.

• We can use the light sensors to detect changes in an environment provided that the changes occur within
  the detection ranges. Otherwise, we can only detect global changes in the environment.

• The environment has a strong influence on the readings from the sensors, so if we wanted to detect changes
  in a given room, we would first have to calibrate the room.

It is clear that quite a bit remains to be done before we can ascertain the effectiveness of the sensor motes for
different sensing applications. Through this project however, we have identified some of the capabilities and
limitations of the sensor motes.

5.1 Future Directions

We could extend this project in a variety of ways to test different characteristics of the light sensors. One possibility
is to perform the distance of occlusion measurements using objects of different colors. From the data, we could
construct a table of the detection ranges for different colored objects. We could also create a table of detection
ranges for objects of different sizes. This information could be useful for motion detection applications.
Another practical experiment is to test the sensitivity of the light sensors to different numbers of occupants in a room. An equivalent, and simpler test, is to record the light intensity as a function of the number of stationary objects within the detection range. There are an infinite number of experiments that we could perform to learn more about the mote light sensors.

6. Acknowledgements

I would like to thank Dr. Ruzena Bajcsy for giving me the opportunity to work with CITRIS on this project, and for her guidance. Without the support and direction of Barbara Hohlt and Dan Steingart, I would not have been able to complete this project – many thanks to them for their generosity. I was also fortunate to have the help of the graduate students in Dr. Paul Wright’s lab, especially Russell Romero and Nate Ota. For their faithful support and assistance, I would also like to thank the SUPERB staff, especially Marie Mayne, Dr. Sheila Humphreys, Jeannie Lee, and Erika Tate. Finally, I would like to thank all the SUPERB students for their friendship and for giving me a great summer.

References


