Exploring Computer Science Students' Learning of Sensor-Driven Mobile App Design: A Case Study

Joon Suk Lee

Department of Engineering and Computer Science
Virginia State University
Petersburg, Virginia, USA 23806
E-mail: jlee@vsu.edu

Kostadin Damevski

Department of Computer Science
Virginia Commonwealth University
Richmond, Virginia, USA 23284-3019
E-mail: damevski@acm.org

Hui Chen

Department of Engineering and Computer Science
Virginia State University
Petersburg, Virginia, USA 23806
E-mail: huichen@ieee.org

Abstract: Sensor-driven applications, implemented using modern mobile or gaming devices, have great potential in motivating computer science students. Recent industry trends toward including more sensors on devices such as mobile phones, which enable new applications in health monitoring, smart homes, and human safety, among others, indicate that the number of such sensor-driven applications will continue to rise. Via a study to learn the difficulties that a group of students face in designing such sensor-driven applications, we uncover a set of instructional principles for instructors to follow in using sensor-driven applications in classrooms. Our findings include that (1) exposing students to sensor data earlier helps improve self-efficacy; (2) focusing on extracting overall patterns from sensor data rather than understanding specifics of physical quantities is beneficial; and (3) good sensor data visualization is beneficial to design, but bad visualization can confuse students.

Keywords: sensor-driven applications, case study, qualitative research

1 Introduction

Sensor data processing is increasingly permeating mainstream software development via a variety of popular hardware devices, such as modern mobile phones, fitness trackers, smart watches, and gaming consoles, which are packaged with many types of sensors including accelerometers, gyroscopes, proximity sensors, location sensors, and others. Sensor-driven software applications rely on such sensors for their primary functionality. Such sensor-driven applications have been developed for a range of purposes, such as health monitoring, smart homes, driving safety, personal security, gaming and entertainment.

Applications based on sensor-rich hardware devices have the benefit of easily engaging many students. Researchers have developed educational interventions to prepare students for the specific challenges of these platforms. Issues such as system and application reliability, sensor data lag, and mutiple threading were mentioned as relevant computing concepts to be emphasized by educators [1, 2, 3]. Based on the assumption that collecting and visualizing the sensor data would help students develop intuition, form heuristics, and develop a more robust algorithm, a teaching model with a set of proposed tools for collecting and visualizing sensor data has also been proposed [4].

While previous research has identified factors such as the indeterminacy in sensor data and complexity in development environment as possible causes of the challenges associated with teaching sensor-driven application development [4], the accounts were provided as a rough sketch of an issue that needs to be scrutinized more throughly. Simply put, there has not been enough study that tries to examine and understand the challenges that students face in designing sensor-driven applications. Sensor data is unique in its nature. It is typically noisy. It is constantly streaming. And it is subject to the impact of various factors (e.g., sampling rates) which make it unique and unfamiliar to most computer science students. By understanding the difficulties that students face in designing such applications, we can target instruction to effectively help students and improve student learning outcomes, especially under the constraints of the already packed computer science curricula.

In this paper, we report on a study of students designing a sensor-driven application. The intent of the study is to (1) understand the difficulties students face in designing sensor-driven applications, (2) examine effective and ineffective common strategies that students use in designing such applications and (3) validate and verify the teaching model proposed in [4] to determine the effect of observing and visualizing the sensor data stream when designing appropriate sensor data processing algorithms.

The study provides a characterization of the difficulties that students face when designing sensor-driven applications. Many of the difficulties we found are unique to such applications, such as the dynamics between understanding the physics behind sensed physical quantities and extracting useful overall patterns from time-series of sensor data. Other difficulties, found also in other computer science domains, include the contention between the necessity of visualizing data and the misleading effect of poorly crafted visualizations, as well as the challenges conducting early and frequent evaluation and assessment of algorithmic solutions. These observations lead to a set of recommendations to instructors who are interested in teaching sensor-driven applications.

The rest of the paper is organized as follows. Section 2 provides an overview of the related work and a brief comparison between this study and related studies from other engineering domains. In Section 3 we describe the study that we conducted. We analyze the collected data and present a taxonomy of difficulties that students encounter in designing sensor-driven applications in Section 4, and analyze the effects of exposing students to the

sensor data and to the data visualization in Section 5. We discuss the implications of our study in Section 6.

2 Related Work

The ubiquity of smart phones, tablet computers, and smart watches has created both a need and an opportunity for teaching sensor-driven application development to a broader set of computer science students [1, 5]. This development in our field has motivated several prior efforts to bring such sensor-rich mobile platforms to many aspects of education. These include a broad range, from teaching K-12 students (e.g., [6]) to college students, (e.g., [7, 8]), and from teaching introductory computer science courses [9] to teaching in depth a specific area of computing such as operating systems, computer graphics, computer networks, gaming, or modeling (e.g., [10, 11, 12, 13]).

Teaching sensor-driven applications has a number of challenges, such as the uncertainty inherent in sensor data, and the complexity of both platforms and algorithm design [4]. In many domains of engineering, sensors and sensor data have long been at the core, for instance, in robotics and automatic control. Educational examples include teaching students to work with wireless sensor networks or robots (e.g., [14]). In addition, in contrast to teaching sensing and control system design in engineering curricula whose focus is often on hardware system design and on signal processing of sensor data [15], teaching sensor-driven applications in computing curricula requires designers to extract overall patterns to support "high-level" decision making, such as, to infer whether a user is under stress using voice data collected from an audio sensor [16], to turn users' location sensor data into textual description that establishes semantics of users activity [17], and to build an inference model to estimate human queue length and in turn measure service and waiting times [18]. Such sensor-driven applications require knowledge and skills not only in processing sensor data, but also in extracting features from data. The application designers need to draw on knowledge and skills from multiple disciplines or areas of computing and engineering to extract features by building comprehensive analysis and processing workflows, which is a difficult task.

This paper further extends and supports the above work by examining a set of difficulties that computer science students face in designing sensor-driven applications. By conducting a qualitative user study, we first systematically survey and unveil types of difficulties associated with sensor-driven application development instead of enumerating them based on our (educators') casual observations. By providing findings that are grounded in the collected data, we aim to eliminate any possible misalignments between educators' perceived and students' experienced difficulties (e.g. [19]) shows how educators' beliefs can differ from student data). In addition, we investigate whether and how *the development process and a set of supporting tools and programs* described in [4] help students.

3 The Study: Methodology

This exploratory study is centered around students designing a sensor-driven application to classify human activities. From the study, we investigate the types of difficulties that students face in designing sensor-driven applications. Here, design refers to algorithmic design, i.e., what algorithm can classify user activity correctly.

Table 1 Summary of participants' previous programming experience

Participant	Gender	CS Courses	Programming Languages
P1	F	1	C/C++, Java, Arduino
P2	F	5	Java
P3	F	10	C/C++, Java
P4	M	0	Matlab, Java, Python
P5	M	5	C/C++, Matlab, Java, C#
P6	M	5	C/C++, Matlab, Java, JavaScript, C#
P7	F	2	C/C++, Matlab, C#
P8	M	12	C/C++, Matlab, C#
P9	M	15	C/C++, Matlab, Java, Python
P10	M	7	C/C++, Matlab
P11	M	16	C/C++, Matlab, Java, JavaScript, C#, Python

3.1 Perspective

The work presented in this paper is a *phenomenologically-informed case study*. Case study research is "a qualitative approach in which the investigator explores a bounded system (a case)..., through detailed, in-depth data collection involving multiple sources of information (e.g., observations, interviews, audiovisual material, and documents and reports), and reports a case description" (p.73, [20]). In this study, we explore a particular group of students engaged in sensor-driven application design. Through detailed, in-depth data collection involving multiple sources of information (e.g., questionnaire data, recorded interviews data, submitted solutions), we too aim to deliver an "intensive, holistic description and analysis" (p.46, [21]) of the kinds of difficulties our students face during the design process. By taking a phenomenological [1] stance, this study intends to investigate the *essence*([22]) of students' lived experience([23]) of designing a sensor-driven application.

Our study is an *intrinsic case study* in its nature, since the purpose of our study is not to present the *theory* of learning difficulties associated with sensor-driven application building, but is "undertaken because of an intrinsic interest in" (p.445, [24] quoted in [21]) exploring different kinds of hardships inherent in sensor-driven application design and development.

One can also view our study as a *contextual inquiry* since our investigatory methodology is "an immersive, contextual method of observing and interviewing that reveals underlying (and invisible) work structure" (p.46 [25]) of students working on a sensor-driven application development. The end-product of our research can then be viewed as a *case study* ([21]).

The exploratory study is centered around students designing a sensor-driven mobile application to classify human activities. From the study, we investigate the types of difficulties that students face in designing sensor- driven applications. Here, design refers to algorithmic design, i.e., what algorithm can classify user activity correctly.

3.2 Participants

We conducted the study in the summer of 2014 over two days with 12 participants recruited from the College of Engineering and Technology at our institute. Participants were junior or senior-level undergraduates or first-year master's students, whose ages ranged from 18

to 25 (Mean Age = 21.45; Standard Deviation of Age = 1.97). One participant had to leave the study in the middle to attend a personal matter, so the participant's data therefore was excluded from the subsequent analysis. Participants were randomly assigned into pairs for the study.

All participants had prior programming experience. Participants on average had taken 7.1 computer science courses at our university. Ten participants said that they had taken introductory programming courses such as Intro to Programming in C++ or Intro to Programming in Java while the majority had also taken senior/graduate level courses such as Advanced Algorithms, Data Mining, Computer Simulation and Robotics. We had a 19-year-old student who had not taken any programming courses at our school, but stated that he had previous programming experience and could use three different programming languages — Java, Matlab and Python. He also stated that he had programmed on multiple platforms including Windows, Android, LEGO Mindstorms, and Arduino. Table I shows the participants, designated P1 to P11, their gender, the number of previous computer science courses taken at our university, and the lists of computer programming languages they said they were cable of using at the time of the study.

Four of the participants were Brazilian nationals whose native language was not English, but none had difficulties communicating in English.

3.3 Procedure - Human Activity Classification

The study was designed as a two-day long experiment. We asked the participants to design a sensor-driven application on day one, and interviewed them on day two.

On day one, participants were brought into a computer lab and randomly divided into groups of two. After obtaining written informed consents, the pairs, sharing a desktop computer, were asked to design a sensor-driven mobile application that automatically determines whether a user holding the device was standing, walking or running. We chose to adopt and use the Human Activity Classification application design task described in [13] since the task was compact enough to be a course module, but at the same time complex enough to be challenging to students who had never developed sensor-driven applications.

The authors instructed the participants that the pairs would complete the design in three phases following a given written description. In each phase, the researchers introduced a set of interventions to aid the participants.

In phase I, the participants were given a description of the intended application containing the necessary information to solve the problem. They were told that the target application should use data from the accelerometer sensor on a mobile phone, and that the accelerometer produces acceleration data in the X, Y and Z dimensions. Additional information given to the participants included the following: acceleration is the change of velocity over time; velocity is measured in meters per second (m/s); the acceleration is measured in meters per second squared (m/s^2); acceleration due to gravity is $\sim$9.8 m/s^2 and the mobile phone's accelerometer registers force in the direction of the ground, as well as human movements. The participants were also allowed to search the Web for further clues. Each group was given one hour to formulate and submit their solutions either in plain English or pseudo-code format on a project submission website.

In phase II of the study, the participants were given another hour to revise their phase I solutions. Unlike phase I, however, in phase II each group was given a mobile phone with a sensor data collecting application written by the authors [2]. By providing students with the

Table 2 Summary of study phases and interventions.

phase I	Participants were presented with the description of a sensor-driven mobile application.
phase II	Participants were instructed to use a sensor data collecting application to examine sensor data stream
phase III	Visualizations of pre-collected data were provided to the participants to aid their design

application for sensor data collection and visualization, we intended to study the effects of the students' exposure to the sensor data purported in [4].

In addition, the participants were given instructions to: (1) collect accelerometer data using the provided application, (2) download the collected data off the phones, (3) import the data into a spreadsheet application, and (4) plot and visualize the collected data in the spreadsheet application. The instructions were also available as a short video that the participants could review while performing their work.

In phase III, the groups were again given another hour to revise their phase II solutions. However, instead of providing mobile phones and asking groups to collect the sensor data, the researchers collected sensor data using the same application on a phone beforehand and made the prepared data available to students in a spreadsheet. The authors marked the data in advance with tag information (i.e., standing, walking, and running) on different data segments and created three separate plotted graphs for each segment representing either a standing, walking, or running state that a user was in. The files were provided to each group to review and interpret. The phases and accompanying tasks of the study are summarized in Table 2.

Video recordings of each group's activities from all three phases were collected as well as the solutions the pairs submitted at the end of each phase. Additional drawings and notes the pairs produced in each phase were also collected for analysis.

3.4 Procedure - Survey

During the initial sign-up process, participants were directed to an online survey site and asked to fill out questionnaires about demographics, prior programming experiences and programming self-efficacy measures. Using Bandura's guidelines [26], the researchers created five sets of self-efficacy measures to assess students' general self-efficacy levels in programming easy projects, moderate projects, hard projects, sensor-data-driven application development projects, and GUI (Graphical User Interface) application development projects. Self-efficacy measures were designed not only to assess students' initial perceptions of difficulties pertinent to the different types of programming tasks, but also to understand the perceived difficulty level of sensor-driven application development in comparison with general and GUI programming tasks.

Day one of the study lasted 3.5 hours. After each phase, participants were asked to fill out online questionnaires about their problem solving activities. The questionnaires asked participants to describe how they solved the given problem, to list and state difficulties they experienced in solving the problem, and to delineate how and why they were able/unable to overcome the difficulties. The questionnaires also included a self-efficacy measure on sensor-driven application development.

3.5 Procedure - Interview

After carefully reviewing participants' responses to the survey questions, the authors conducted semi-structured interviews with each group on day two, and asked what the pairs did on day one, how they drove their design, what their retrospective evaluations of their own solutions were, and what different kinds of difficulties they faced. Each interview lasted 30 to 40 minutes. The interview sessions were video recorded for analysis.

3.6 Data Analysis

3.6.1 Initial Coding Analysis

The data analysis was done in multiple iterations. In an initial analysis, the authors went through the questionnaires and interview data multiple times, conducted *open and axial coding* [27], and developed a three-layered hierarchical coding scheme in order to understand the kinds of challenges students encounter in sensor-driven software design. That is, we went through the questionnaire data to first extract and develop 25 thematic codes solely based on the participants' stated accounts of their experienced difficulties. We then went back to the questionnaire data and annotated and extracted 52 different categories of factual statements about how participants had tried to solve the problem. After reviewing these 52 thematic categories, we then extracted 9 more inferred difficulty codes. For instance, when a participant stated that his team found z values the most salient and therefore used only z values to estimate and classify human activity, it was clear that the pair drew a hasty conclusion in interpreting accelerometer data probably by just looking at one set of experimental sensor data. Because factors such as the orientation and tilt angle of the device as well as gravity can continuously change x, y, z values in accelerometer data, a better way to handle acceleration data is to use a square root of the sum of all three axes squared as shown in [13]. Hence we coded such a response as an *inferred* difficulty.

The final coding scheme consisted of 34 difficulty codes under 5 top-level categories and 8 second-level categories, encompassing explicitly observable as well as inferred difficulties students exhibited during all three phases. The authors used *affinity diagraming* [25] to organize and visualize the coded data.

3.6.2 Contextual Inquiry

In order to develop deeper understandings of what students tried to do when designing the solution, and to explore the types as well as the possible causes of difficulties they experienced, the researchers reviewed video recordings of the problem solving sessions as well as the interview data in multiple iterations, identifying and transcribing the key moments. In conjunction with video data analysis, the researchers studied submitted solutions and transitory outputs such as the drawings and notes students produced. Since the responses to the questionnaires were sometimes too terse to yield complete understandings and some other times too vague to produce sensible interpretation, the researchers had to iteratively make changes to the coding scheme while reviewing interview and session data. Figure 1 illustrates our research methodologies.

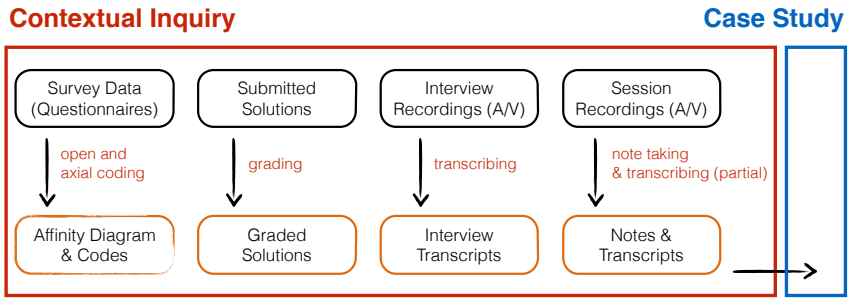


Figure 1: Research Methodologies

4 Taxonomy of Difficulties

The five main categories of difficulties were *basic skill set related* ($\mathbf{D_{basic}}$), *problem-solving skill set related* ($\mathbf{D_{problem}}$), *task-specific skill set related* ($\mathbf{D_{task}}$), *experimental procedure related* ($\mathbf{D_{experimental}}$), and *incidental* ($\mathbf{D_{incidental}}$). Table 3 gives an overview of the five categories of difficulties and their sub-categories.

Table 3 Taxonomy of difficulties.

Main Category	Sub Category
$\mathbf{D_{basic}}$	inability to understand problems
	inability to articulate solutions
$\mathbf{D_{problem}}$	inability to evaluate their own solutions
	working with an iterative process
$\mathbf{D_{task}}$	declarative knowledge related
	procedural knowledge related
$\mathbf{D_{experimental}}$	time constraints
$\mathbf{D_{incidental}}$	broken phone

Basic skill set related refers to difficulties associated with fundamental skills such as an ability to read and understand the given problem, or to formulate and present solutions in a comprehensible way.

In phase II, for instance, when we distributed mobile devices to the participants, we also provided a short tutorial to show the participants how they could (1) use the mobile devices to collect sensor data, (2) transfer the data to their computers, and (3) open the data file with a spreadsheet application. We then asked the participants to revise their phase I solutions.

While five groups used the mobile devices to design and carry out small experiments to collect sample standing, walking and running sensor data in order to revise their solutions, one group sat in front of their computer throughout phase II and followed the steps in the tutorial without conducting an actual experiment. When the post-phase II questionnaire asked the groups whether they were able to finish the given task—to revise their phase I solutions using the tools given to them—this group answered "yes" and explained how they were able to collect the data, transfer it to the computer and open the file.

However, when asked to tell how they used the experimental data to revise their previous solution, this group said the following during the interview.

> *we kinda made a mistake because we did like, for question three, when you then gave us information, we actually did that for the number two but accidentally*

We believe what they meant is that they expected to have the sensor data pre- planted on the phone for them so they could just follow the instructions to download the file. By examining the data they collected while sitting in front of the computer, they revised their solutions and said "*We determined plot X was standing, plot Y was walking and plot Z was running*" on the submitted solution. From these facts, we can safely tell that this group clearly misinterpreted both verbal instructions and handed-out written instructions given by the authors.

From these facts, we can safely tell that this group clearly misinterpreted both verbal instructions and handed-out written instructions given by the authors. Even though it would also be possible to argue that better formulated instructions could have helped this group, the fact that five other groups had no problem interpreting the instructions shows that the instructions alone were not at fault. Previous educational research has shown that the lack of fundamental skills such as reading comprehension is one of the main causes of classroom under-performance [28]. In our case, we had a group who had difficulty understanding the given instructions and the problem.

Several groups had difficulties writing out their solutions using pseudocode format, even when they had a clear idea of how to solve the problem. In addition, on multiple occasions we had to probe the pairs to reformulate and re-articulate their solution several times during the interview in order to gain full comprehension of their design. We weren't able to definitively identify what caused this type of difficulty. Poorly formulated instruction sets and interview questions can surely cause the problem. We cannot ignore the possibility of the particular experimental setting in which our students were asked to perform the task causing the problem. It might also actually be the students' lack of fundamental skills as previous research suggested [28], or any combination of these factors. In any case, acknowledging that such a difficulty can exist, especially when the the task and the instructions for sensor-driven application development are unfamiliar to many students, is indispensable in teaching sensor-driven application.

In our case, we found that inabilities to understand problems and to properly articulate solutions were the kinds of fundamental skills that some participants lacked. For instance, we observed that many groups were not able to write out their solutions using pseudocode format, even when they had a clear idea of how to solve the problem. Moreover, some participants were not even able to clearly articulate their solution to the researchers during the interviews on day two. During the interview, the researchers had to repeatedly ask students to re-phrase or re-state their solutions multiple times in order to gain full comprehension of their design and solution. While this type of difficulty is not just relevant to sensor-driven application development, acknowledging that such a difficulty can exist, especially when the the task and the instructions for sensor-driven application development are unfamiliar to many students, is indispensable in teaching sensor-driven application.

Problem-solving skill set related difficulties are the ones germane to general problem-solving skills. One of the most common problems the participants experienced in phase I was rooted in the *inability to evaluate their own solutions*. The participants invariably expressed their frustration over not being able to test and generate "reliable" solutions. After looking at the sensor data, the participants did not express such a concern in phase II.

In various programming classes, we teach our students to write automated test cases as a part of the development process so that they can test their solutions by manipulating expected input and output values. Yet, unlike many of the problems our students have previously experienced, the sensor-driven application development requires testing the solutions against empirical data sets. In that sense, not being able to test out the solution in phase I was a problem inherent in the given instruction, not in the students' lack of certain abilities.

Another kind of *problem-solving skill set related* difficulty was *working with an iterative process*. Even though most real life problem solving tasks require longitudinal investigation of the problems, and determined commitment and effort to iteratively generate and refine solutions, some students stated that working on the same problem to refine already formulated solutions was one of the main difficulties.

While research has shown the importance of formative assessment over summative evaluation [29], it is disputable to say that students are more used to solving problems iteratively than to providing one-off solutions. Indeed, especially in introductory programming classes, many class projects or exercise problems are summatively evaluated. Although not unique to sensor-driven application development, this kind of difficulty calls for our recognition of the importance of formative assessment as well as the careful design of problem solving activities for teaching sensor-driven application.

We also have to note that the reporting of such a difficulty might have risen from the fact that students had to work on the problem solving activity for 3.5 hours, which exceeds a typical block of time allocated for computer science classes at our school.

Task-specific skill set related code captured difficulties particular to sensor-driven application development. The majority of difficulties students experienced during sensor-driven application design belongs to this category.

The two subcategories are *declarative knowledge related* and *procedural knowledge related* difficulties. The term *declarative knowledge* typically refers to a problem solver's knowing factual information about a task, while the term *procedural knowledge* refers to a problem solver's knowing how to perform a task [30]. In this study, we use the term *declarative knowledge related* difficulties to indicate challenges originated from the problem solver's lack of fundamental knowledge of Physics, Mathematics, or Sensors.

When asked during the interview to describe what the team did on day one, many stated that they first tried to recall what they had learned in Physics class. For instance, the following example shows how one of the participants tried to recall a formula for calculating the velocity from the acceleration data.

> *(I tried to) remember bunch of formulas from a physics class to figure out how to get certain things*
> *when you were given us uh equations for velocity and accelation*
> *I'm like*
> *one is derivative of the other then*
> *second devrivate is makes you get uh accerlation*
> *...*

The source of these kinds of difficulties can be easily traced back to not having (or not remembering) enough knowledge of physics, hence we coded them as *declarative knowledge related*. Some task-specific difficulties students expressed during the study included *not knowing how to calculate velocity based on accelerometer data,* or *not knowing*

what gravity data is. Such difficulties can typically be moderated by providing necessary information to students.

On the other hand, when students experienced difficulties associated with *interpreting acceleration data to determine state information* or with *interpreting plotted graphs* they had to utilize and process existing knowledge to formulate a new set of *know-how* knowledge [31] in order to overcome these difficulties. In this study, we observed that even when students had all the information necessary to solve the given task, they did not know how to process the information at hand in a useful way. We coded such difficulties as *procedural knowledge related*.

Procedural knowledge is often associated with building abstractions over fragmentary information, or extracting features from raw data. In this study, providing factual information alone was usually not sufficient enough to reduce difficulties associated with *procedural knowledge*.

In phase I, half the *task-specific skill set related* difficulties were *declarative knowledge related* and the other half *procedural knowledge related*. However, having observed the data in phase II, the participants started reporting less of what we would call *declarative knowledge related* difficulties. In phase II and phase III, majorities of the *task-specific skill set related* difficulties belonged to *procedural knowledge related*.

That is, at first, most groups started to solve the problem with various and often incorrect assumptions about the accelerometer data or how they may use the data in their design. These groups were able to reformulate, refine, and rectify their answers after observing the data collected in phase II.

However, *procedural knowledge related* difficulties existed in all three phases.

For example, one of the pairs tried to generate a computational model for Human Activity Classification in phase II and III. The computational model as shown in Figure 2 implies a state model with standing, walking and running states. In this model, it is only possible to go from standing state to walking and then to running, and from running to walking and then to standing. This model is conceptually sound since in order for a person to run, he or she needs to first move through walking speed, and only then can the person begin running by accelerating more. Although this model sounds logical, one needs to understand that while sensor data is continuously streaming, the data can be noisy, and only a fixed number of readings can be done per second (sampling rate). In an empirical state model, one should be able to go from any of the three states to any other as shown in [13]. Understanding the possible discrepancy between the conceptual model and the empirical model requires more than just seeing the sensor data. One needs to be able to extract necessary information from the raw data and draw an abstraction over it.

This shows that providing students with actual sensor data could curtail a number of ineffectual trials and meaningfully reduce *declarative knowledge related* difficulties. But helping students with *procedural knowledge related* difficulties demands additional interventions such as teaching students how to extract necessary information from the raw data to form profound understandings of sensor data.

Lastly, we also want to note that on some occasions, *declarative knowledge related* and *procedural knowledge related* difficulties were not clearly distinguishable. Some problems the participants encountered belonged to both sub-categories. For example, almost all groups first tried to set and define threshold values of users' velocity that distinguished walking from running. Yet, finding a set of threshold values requires not only *declarative knowledge* but also *procedural knowledge* about sensor data.

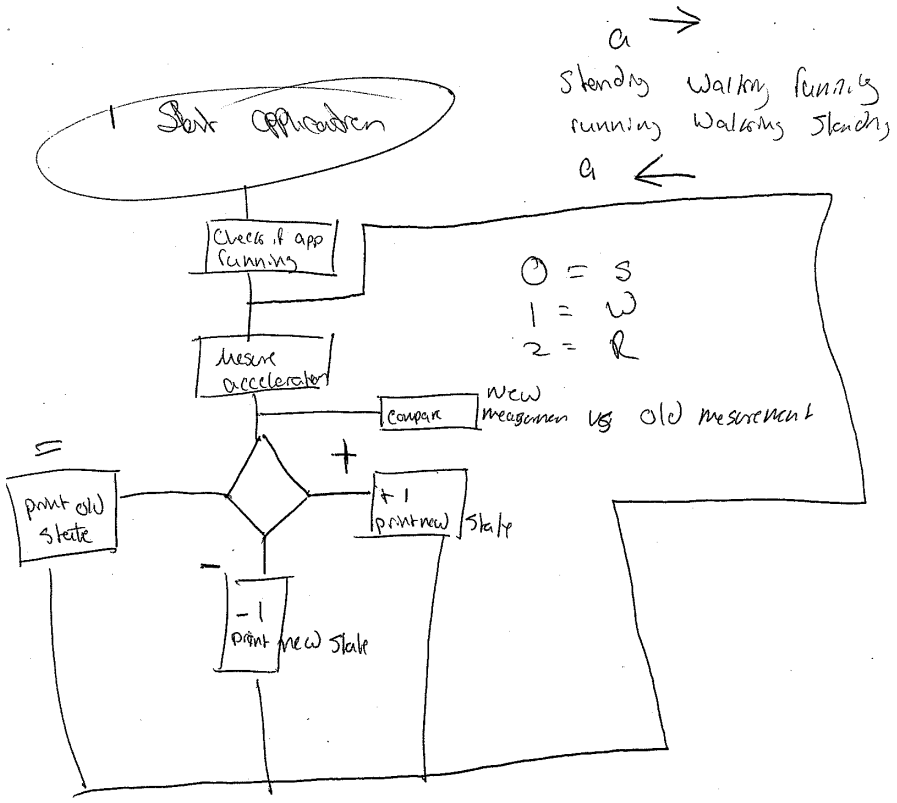


Figure 2: A sample student work: State Model

Experimental Procedure Related code captured difficulties created by our particular experimental setting. For instance, one of the challenges that the participants encountered in the study was *time constraints*. Three groups reported that they did not have enough time to generate proper solutions. We considered *time constraints* an extraneous factor that the participants were not able to control and categorized it under *Experimental Procedure Related*.

Incidental code captured a case in which one of the groups received a faulty phone. When the group reported the problem to the researchers, we exchanged the broken phone with a new, working one. When this group reported the incident as a difficulty, we categorized it as *incidental*.

Table 4 shows the frequency distribution of the difficulty codes across all three phases.

Overall, students tended to report a greater number of difficulties in Phase I and fewer in the later phases. These numbers alone cannot show that students experienced fewer difficulties as they proceeded into the later phases. A more plausible explanation would be that students became more and more fatigued as they progressed into the later session and therefore wrote less on the questionnaires. However, the fact that many of the *task-specific skill set related* difficulties reported in all three phases were relatively distinctive in all phases, while *basic skill set related* and *problem-solving skill set related* codes reported were very much repeated in all three phases is worth noting.

Table 4 Frequency Distribution: Difficulty Codes Over Three Phases

Difficulty Code	Phase1	Phase2	Phase3
$\mathbf{D_{basic}}$	11	6	2
$\mathbf{D_{problem}}$	6	2	2
$\mathbf{D_{task}}$	21	17	13
$\mathbf{D_{experimental}}$	8	2	1
$\mathbf{D_{incidental}}$	0	4	1

Table 5 Initial self-efficacy levels.

Participant	Easy Project	Moderate Project	Hard Project	Sensor Data Project	GUI Project
P01	9.1	6.8	5.4	3.7	3.7
P02	10	9.7	9.3	9.1	9.2
P03	9.6	7.6	7.3	5	9.8
P04	9.5	6	3	5	5
P05	10	10	9.1	7.1	9.5
P06	7.7	7.1	5	6.6	6.6
P07	8.4	9.1	9.2	8.8	8.8
P08	8.9	7.7	8	8.4	8.7
P09	9.7	9.3	7.8	8.3	8.5
P10	5.7	5	5	5	5
P11	9.5	8.9	8.1	9.5	7.8
Average	8.92	7.93	7.02	6.95	7.51

As mentioned earlier, we were able to observe that the interventions we applied in each phase had greater impacts on reducing *task-specific skill set related/declarative knowledge related* difficulties than any other types of difficulties.

In the next section, we explore the impacts of the interventions in more detail.

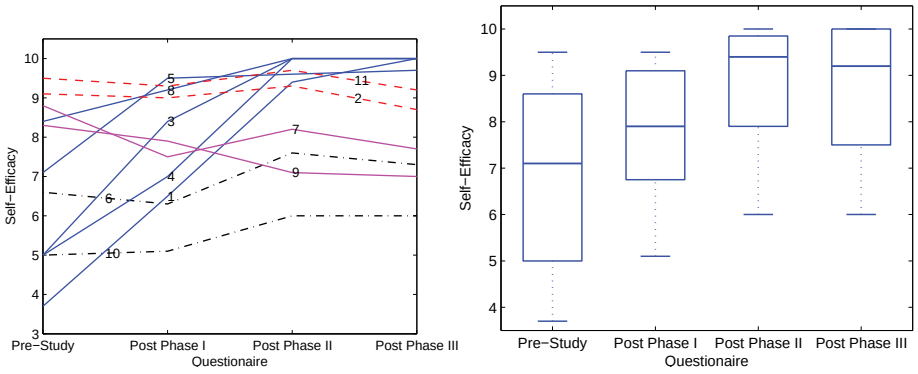
5 Effects of Interventions

Our exploratory study investigated the effects of providing students with data collection tools and visualization examples in designing a sensor-driven mobile application. After categorizing student difficulties in creating sensor-driven applications, we went back to the coded data sets in order to investigate whether and how the interventions we deployed in each phase helped students.

Students are apprehensive of developing applications based on sensor data, but exposure to the data itself improves self-efficacy. At the outset of the study, we measured students' programming self-efficacy levels across five different types of projects (easy projects, moderate projects, hard projects, sensor data projects, and GUI projects). For each type of programming project we asked students to rate how certain they were to complete 10% of the given type of projects on a scale of 1 (cannot do at all) to 10 (highly certain can do), 20% of projects and so forth up to 100%, according to Bandura's self-efficacy assessment guidelines [26]. Students initially perceived sensor-driven application development to be the hardest. As the study progressed in 3 connected phases, we repeatedly asked students about their self-efficacy in developing sensor-driven applications. The average self-efficacy levels for each participant following each phase of the study are shown in Figure 3.

Initial confidence levels for different types of projects are shown in Table 5.

As each phase of the study introduced a new intervention, we were interested in the change in self-efficacy following each phase, which is visible in Figure 3b. For phase I,



(a) The trend of each participant

(b) A box-and-whiskers plot

Figure 3: Each participant's self-efficacy in developing sensor-driven applications at different state of the study.

we considered the change from the initial assessment. Our analysis indicates that while the average students' self-efficacy increased from the initial assessment to post phase I by 12.03%, many students (5 out of the 11 assessed participants) decreased their self-efficacy scores in this period. In other words, trying to solve the assigned problem in phase I significantly increased some students' confidence levels while it slightly reduced the confidence of others. The change in self-efficacy after phase II became more consistent and positive for 10 out of 11 participants, reflecting the overall positive impact of students' observing the sensor data. Finally, self-efficacy remained mostly high and unaltered after phase III, where participants where nearly evenly split between a very small positive or negative change in efficacy. This phase did very little to move students' already established self-efficacy levels.

Students can design effective sensor data collection experiments with little prior training. In phase III of the study, researchers handed out pre-collected, pre-tagged and pre-plotted data files to the groups in thinking that even though students were given tools to collect, tag and plot the sample data in phase II, some might still have difficulties planning out their data collection schemes, and might fail to map their data collection activities to the actual data points. That is, if one randomly stands, walks and runs without recording what he/she is doing, interpreting the collected data afterward can become a non-trivial task. In the same line of thought, researchers conjectured that providing pre-tagged data would notably help students.

In phase II, indeed as predicted, most groups did not use the tagging feature and some groups did not even plot the data. However, contrary to the researchers' initial intuition that providing tagged and plotted data would help students improve their solutions, neither noticeable improvements on students' submitted solutions nor increased levels of students' confidence were observed in phase III. Moreover, when we interviewed students on day 2, multiple students stated that the phase III data was not so much different from the data they had already collected in phase II.

Even though the majority did not use the tagging feature on the provided mobile devices to manually annotate activities onto the data during the experiments, five out of six groups did not have problems interpreting their data afterward. Apparently, utilizing

their cognition alone was enough to map their activities to the data. Those students were mentally tagging their data. During the day 2 interview, all groups were able to depict their data collection activities in great detail. While some conducted three distinct experiments to collect standing, walking and running data separately, many organized their activities either in a fixed order or with different durations for different states so that they could easily map their activities to the data.

Changing the focus of students' solution design from only using the physical quantities to extracting patterns from the data was beneficial. During phase I, the approach that the majority of students took to design their solutions was based on converting acceleration into velocity, and using velocity thresholds to classify human activity as standing, walking, or running. This approach, which was strongly based on physics, would cause difficulty dealing with the specifics of the accelerometer sensor itself, such as the overwhelming acceleration component due to gravity that it measures in any of the X, Y or Z dimensions, depending on the mobile device's orientation. The successful student groups, after observing the sensor data in phase II, focused instead on characterizing the difference in the accelerometer data collected during standing, walking or running. For instance, the students in three groups described their intention to identify proper "variation" in the acceleration data.

In general, sensor-driven applications have to take account of the artifacts induced by different types of sensors: noise, sampling rates, etc. While a basic knowledge of the underlying physics is valuable, the most important algorithm design decisions have to be data-driven and focus on overall patterns that the data exhibits over time.

Good visualization helped, while bad visualization hindered student designs. The data file provided in phase III included three plotted graphs, each representing three possible states: standing, walking and running. Despite researchers' expectation that having three distinct graphs for three states would help students to understand the data better and produce improved application designs, phase III data not only did not help students, but it also confused some of them. The biggest problem with the three distinct graphs was that each graph was drawn at a different scale. Three graphs with different scales perplexed some students and made them unnecessarily and counter-productively doubt their solutions. Many preferred to see all three stages in a single plotted graph. This experience indicates that even though data visualization helps, poorly designed visualization can actually hinder students' performance.

Ability to rapidly evaluate each proposed design is a key to success in sensor-driven application design. Examining the submitted solutions for all of the six groups after phases II and III, we observed that two groups completed the project to a satisfactory level, and designed an algorithm that is likely to work in practice. Both of these two groups exhibited a key pattern in their behavior: they tested their solutions against the collected data to obtain feedback on their accuracy. This behavior was not performed by any other of the remaining groups.

Before phase II of the study, as part of the instruction on how to collect accelerometer sensor data using a mobile device, the students were shown how to open the data in a spreadsheet application. The successful groups created another column in the spreadsheet where they implemented their proposed algorithm and observed the result. They could then compare the computed results with the ground truth of their experiment and determine the solution's effectiveness. This allowed these groups to iteratively improve their designs and reach much better solutions than they might have otherwise.

6 Implications and Conclusions

While the study described in this paper is small in scale and has several threats to its validity, including the specific application that was selected and specific interventions that were conducted, it introduces or underscores a few important points in teaching sensor-driven applications to computer science students. The single most important implication of the study is the importance of observing the sensor data itself instead of reading the description of it. The importance of observing the sensor data stream is more crucial to learning how to design sensor-driven software applications, compared to learning sensor applications in other engineering domains that emphasize data acquisition and signal processing. Sensor-driven applications that computer scientists often build require higher-level features extracted from the collected sensor data stream, which may also need to be transformed by signal processing filters, statistical methods or other means. In addition, an understanding of the underlying physics of the quantities measured by the sensors is valuable, but not as important as data analytical techniques that enable the extraction of relevant patterns from the sensor data stream.

Another implication of the study is that data visualization can be a hindrance if done improperly. We believe, although this is not something we were able to capture in the study, that students commonly do not possess the experience to produce visualizations that can aid their design, unless they are given some guidance from their instructor. Therefore we believe that the data visualization is an important tool in teaching sensor data application, but it needs to be carefully thought-out and designed.

Lastly, the ability to rapidly evaluate an algorithm improves student outcomes significantly. The data analytics community has long emphasized using cross- validation data sets to incrementally improve solutions. However, the modern sensor-driven application development ecosystem does not offer proper means for testing such application designs, and, therefore, it is important to teach students that prototyping is best performed offline, using some external means, such as a mathematical application (e.g. MATLAB, Octave, or a spreadsheet application), and that the simulation needs to be an integral part of sensor data application development.

Acknowledgments

This material is based upon work supported by the U.S. National Science Foundation under Grant No. 1044841 and Grant No. 1040254. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. National Science Foundation.

References

[1] Matthew H. Dabney, Brian C. Dean, and Tom Rogers. No sensor left behind: Enriching computing education with mobile devices. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 627–632, New York, NY, USA, 2013. ACM.

[2] Anna Förster and Mehdi Jazayeri. Hands-on approach to teaching wireless sensor networks at the undergraduate level. In *Proceedings of the Fifteenth Annual Conference on Innovation and*

Technology in Computer Science Education, ITiCSE '10, pages 284–288, New York, NY, USA, 2010. ACM.

[3] Sami Rollins. Introducing networking and distributed systems concepts in an undergraduate-accessible wireless sensor networks course. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, SIGCSE '11, pages 405–410, New York, NY, USA, 2011. ACM.

[4] Hui Chen and Kostadin Damevski. A teaching model for development of sensor-driven mobile applications. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, ITiCSE '14, pages 147–152, New York, NY, USA, 2014. ACM.

[5] Andrey Esakia, Shuo Niu, and D. Scott McCrickard. Augmenting undergraduate computer science education with programmable smartwatches. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, pages 66–71, New York, NY, USA, 2015. ACM.

[6] Krishnendu Roy. App inventor for android: Report from a summer camp. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, SIGCSE '12, pages 283–288, New York, NY, USA, 2012. ACM.

[7] Kelvin Sung and Arjmand Samuel. Mobile application development classes for the mobile era. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, ITiCSE '14, pages 141–146, New York, NY, USA, 2014. ACM.

[8] David Wolber. App inventor and real-world motivation. In *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, pages 601–606, New York, NY, USA, 2011. ACM.

[9] Ivaylo Ilinkin. Opportunities for android projects in a cs1 course. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 615–620, New York, NY, USA, 2014. ACM.

[10] Jeremy Andrus and Jason Nieh. Teaching operating systems using android. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, SIGCSE '12, pages 613–618, New York, NY, USA, 2012. ACM.

[11] Michael Werner. Teaching graphics programming on mobile devices. *J. Comput. Sci. Coll.*, 28(6):125–131, June 2013.

[12] Brad Richards and Benjamin Stull. Teaching wireless networking with limited resources. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '04, pages 306–310, New York, NY, USA, 2004. ACM.

[13] Kostadin Damevski, Badreldin Altayeb, Hui Chen, and David Walter. Teaching cyber-physical systems to computer scientists via modeling and verification. In *Proceeding of the 44th ACM technical symposium on Computer science education*, SIGCSE '13, pages 567–572, New York, NY, USA, 2013. ACM.

[14] T. Andrew Yang, Deepesh Jain, and Bo Sun. Development of emulation-based projects for teaching wireless sensor networks. *J. Comput. Sci. Coll.*, 24(2):64–71, December 2008.

[15] T.J. Cavicchi. Experimentation and analysis: Siglab/matlab data acquisition experiments for signals and systems. *Education, IEEE Transactions on*, 48(3):540–550, Aug 2005.

[16] Hong Lu, Denise Frauendorfer, Mashfiqui Rabbi, Marianne Schmid Mast, Gokul T. Chittaranjan, Andrew T. Campbell, Daniel Gatica-Perez, and Tanzeem Choudhury. Stresssense: Detecting stress in unconstrained acoustic environments using smartphones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 351–360, New York, NY, USA, 2012. ACM.

[17] Dan Feldman, Andrew Sugaya, Cynthia Sung, and Daniela Rus. idiary: From gps signals to a text-searchable diary. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pages 6:1–6:12, New York, NY, USA, 2013. ACM.

[18] Yan Wang, Jie Yang, Yingying Chen, Hongbo Liu, Marco Gruteser, and Richard P. Martin. Tracking human queues using single-point signal monitoring. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '14, pages 42–54, New York, NY, USA, 2014. ACM.

[19] Neil C.C. Brown and Amjad Altadmri. Investigating novice programming mistakes: Educator beliefs vs. student data. In *Proceedings of the Tenth Annual Conference on International Computing Education Research*, ICER '14, pages 43–50, New York, NY, USA, 2014. ACM.

[20] John W. Creswell. *Qualitative Inquiry and Research Design: Choosing Among Five Approaches*. SAGE Publications, 2012.

[21] S.B. Merriam. *Qualitative Research: A Guide to Design and Implementation*. Jossey-Bass higher and adult education series. Wiley, 2014.

[22] Michael Quinn Patton. *Qualitative Research & Evaluation Methods*. SAGE Publications, 2002.

[23] M. Van Manen. *Researching Lived Experience: Human Science for an Action Sensitive Pedagogy*. SUNY series, The Philosophy of Education. State University of New York Press, 1990.

[24] R.E. Stake. Qualitative case studies. In N. K Denzin and Y S. Lincoln, editors, *The Sage handhook ofqualitative research (3rd ed.)*, pages 443–466. Sage, Thousand Oaks, CA, 2005.

[25] B. Martin, B. Hanington, and B.M. Hanington. *Universal Methods of Design: 100 Ways to Research Complex Problems, Develop Innovative Ideas, and Design Effective Solutions*. Rockport Publishers, 2012.

[26] Albert Bandura. Guide for constructing self-efficacy scales. In Tim Urdan and Frank Pajares, editors, *Self-Efficacy Beliefs of Adolescents*, pages 307–337. Information Age Publishing, 2006.

[27] J.M. Corbin and A.L. Strauss. *Basics of qualitative research: techniques and procedures for developing grounded theory*. Sage Publications, Inc., 2008.

[28] P. R. Stevenson. Difficulties in problem solving. *The Journal of Educational Research*, 11(2):pp. 95–103, 1925.

[29] LTSN Generic Centre. *A briefing on key concepts: formative and summative, criterion and norm-referenced assessment*. Assessment series. Learning and Teaching Support Network, 2001.

[30] John Robert Anderson. *Cognitive Skills and Their Acquisition*. Carnegie Mellon Symposia on Cognition Series. L. Earlbaum Associates, 1981.

[31] Jane Roland. On "knowing how" and "knowing that". *The Philosophical Review*, 67(3):pp. 379–388, 1958.