PhoneIoT for Teaching 'Internet of Things': Smartphones to Promote Accessible, Engaging, and Authentic Experiences

Devin Jean, Vanderbilt University, devin.c.jean@vanderbilt.edu
Shuchi Grover, Looking Glass Ventures, shuchigrover@acm.org
Akos Ledeczi, Vanderbilt University, akos.ledeczi@vanderbilt.edu
Brian Broll, Vanderbilt University, brian.broll@vanderbilt.edu

Abstract: We rely on a vast network of devices that communicate autonomously to provide many of the services we use every day. However, the enabling technologies behind the Internet of Things (IoT) are often not taught in K-12 classrooms, in part due to the need for hardware. But most teens in the United States have smartphones. Thus, we introduce *PhoneIoT*, a mobile app that allows students to access their smartphones programmatically over the Internet. *PhoneIoT* supports access to live sensor data from the device and controlling a customizable display on the phone's screen. *PhoneIoT* allows students to learn the fundamental concepts of distributed computing and networked sensing using NetsBlox, a simple but powerful extension of the Snap*!* block-based programming environment. Because both *PhoneIoT* and NetsBlox are free and open-source, instructors are able to teach these advanced computer science topics even remotely without extra hardware.

Introduction

More and more, the programs and tools we use every day rely heavily on Internet connectivity to provide complex services. For example, Google Maps uses the location of connected smartphones (even if the app is not open) to approximate traffic statistics by keeping track of the number of phones in a given area. This is an example of distributed sensor networks, which are becoming increasingly important in our daily lives. Such sensor networks are the *Internet of Things (IoT)*, an umbrella term to describe a network of internet-connected sensors and devices that can be used to gather data or perform actions remotely, and potentially on a large scale. Despite being such a ubiquitous reality in modern computing, there are few opportunities for K-12 students to have hands-on experience with these topics. There are existing curricular activities as well as out-of-school clubs and makerspaces that engage students with tangible computing technologies like Raspberry Pi and Micro:bit, which can both act as or interface with simple sensors and actuators over USB or Bluetooth. However, they rely on local connectivity and are disconnected from the ubiquitous internet and IoT. Lastly, due to the cost of hardware and the logistics of keeping track of it, as well as a lack of teachers familiar with these technologies, few schools offer IoT experiences in schools.

A concurrent reality is that close to 90% of teenagers in the US already own a personal smartphone, and this number is only expected to grow (Sandler, 2022). Mobile devices such as these come with a wide variety of sensors including an accelerometer, gyroscope, microphone, camera, and GPS location service. Additionally, they are fully Internet-capable out of the box, allowing external software to easily communicate with them. Because of this, smartphones present a cost-effective opportunity for introducing students to distributed computing ideas such as the Internet of Things by using their own devices. By using everyday devices instead of relying on expensive or specialized hardware, students are able to engineer and examine the world around them through more practical, authentic, and engaging hands-on projects.

In this paper, we describe *PhoneIoT*, our free smartphone technology innovation and associated curricular activities for high school computer science (CS) students, designed to allow students to connect to and interact with their smartphone over the internet. *PhoneIoT* leverages NetsBlox, an extension of the block-based programming environment Snap*!* (Harvey et al, 2013) that makes networked communication and web data easily accessible (Broll et al., 2017).

Design principles & related work

***Easy entry point to hitherto-inaccessible, advanced networking ideas*:** The list of existing tools which allow students to create their own standalone mobile apps from a block-based programming environment is small but growing. This includes programming tools such as Thunkable (Siegle, 2020), App Inventor (Wolber, Abelson, & Spertus, 2011), and Pocket Code (Wolfgang, 2014). Some of these tools (Thunkable, for instance) are similar to *PhoneIoT* in that they provide access to some Internet-based resources (similar to NetsBlox services) and live sensor data from the phone. However, these are all strictly app development tools supporting projects local to the device, unlike *PhoneIoT* which exposes the device to the internet for students to manipulate in a distributed environment, thus making distributed computing and networked sensing accessible to students. For instance, with *PhoneIoT*, students could easily create a single server-like project that runs in the browser and connects to several phones to implement a distributed chat app. This feature is significant, given recent findings of a meta-analysis of (only 12 existing) K-12 IoT curricula, which concluded that K-12 physical computing/IoT experiences are restricted to the IoT 'sensor layer' and few-to-none expose students to the crucial underlying 'networking layer' due to its complexity and lack of tools (Abichandani et al., 2022).

Accessible & engaging experiences: *PhoneIoT* is designed to be quick to set up and simple to use. "Low floor, high ceiling" has been a crucial guiding principle for the creation of novice programming environments for K-12 learners dating back to Papert's (1980) work on children and LOGO. A low floor entry point is suitable for all students and a high ceiling supports the curiosity of all learners. Block-based programming in general, and in particular, Snap! (a Scratch derivative), have been designed for providing gentle introductory programming experiences to K-12 learners, and have especially benefited minoritized students and those with less preparatory privilege and prior programming knowledge (Goldenberg et al., 2018; Weintrop et al., 2019). Even students who have never used NetsBlox before can learn the basics and build their first distributed app in one class period.

Interactive & Customizable**:** Unlike other tools like Sensor Fusion and Cumulocity (Hendeby et al., 2017; Srirama et al., 2017) which allow for reading sensor values remotely in real time but restrict users to only using the device as a sensor hub, *PhoneIoT* promotes interactivity through a customizable phone display which affords engagement in event-based graphical inputs and remote controllers (see below). Additionally, tools such as Cumulocity are complex and expensive due to running in fee-based, specialized cloud environments.

Promoting creativity, authentic engagement**:** Since the phone is a personal, easily accessible device, *PhoneIoT* provides excellent opportunities for developing projects that promote creative, authentic engagement. For example, the phone's dedicated step counter sensor opens up a world of possibilities for 'quantified self' data science activities that promote personally and culturally relevant projects and artifacts (Lee et al., 2021).

Broadening participation through physical computing and interdisciplinary connections**:** Recent research highlights several benefits associated with physical computing, including increased motivation for students (especially from diverse backgrounds) because working with sensors is tangible and affords interdisciplinary projects (Sentence & Childs, 2020). We ensure broad accessibility through making the *PhoneIoT* app freely available on both Android and iOS, and designed to cover as many phone models and versions as possible.

Protecting Student Privacy**:** To address justifiable student privacy concerns related to providing convenient access to sensors, such as the camera, microphone, and location, *PhoneIoT* proactively (a) prohibits any network communication when the screen is turned off or the app is put into the background, (b) limits the functionality of some sensors, such as the microphone so that *PhoneIoT* cannot be used for eavesdropping (c) prevents direct access to the camera; users must explicitly click an image display (with appropriate optional settings) to take an image from the camera and store it in the display after confirmation, and (d) password-protects each request to the device and passwords automatically expire after 24 hours.

PhoneIoT design: NetsBlox, access to sensor data, & custom interactivity

NetsBlox enables two key features: 1) accessing web-based services, which reach out into the Internet to effect changes or collect and return information, and 2) message passing, which allows two NetsBlox projects running anywhere on the internet to communicate and exchange data. Using these simple networking primitives, students can design powerful applications such as a live weather map or chat server, or remotely control virtual or physical robots (Brady et al., 2022). *PhoneIoT* taps into these existing concepts to provide two core features to students' projects: 1) the ability to access live sensor data from the device, and (2) the ability to configure and control an interactive custom display on the phone. When the app is opened, simply pressing the "Connect" button connects the device to the NetsBlox server. Also shown on the screen are the device ID and password, which are needed for a user's NetsBlox project to connect to the device through the server (Figure 1a).

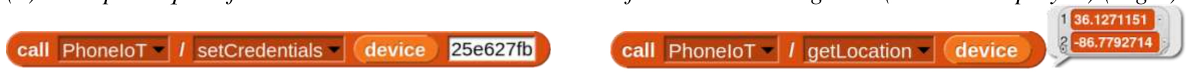
Students can access **sensor data** from *PhoneIoT* in one of two ways. The first (somewhat simpler to introduce to students), is by sending an explicit request to the device through the *PhoneIoT* service in NetsBlox. For example, Figure 1b shows how this can be used to instantly tell the current device location. Explicit requests are convenient if sensor values are only occasionally needed: for instance, when the project is first started. However, it is often the case in practice, both in the classroom and in industry, that live sensor values are needed continuously; *PhoneIoT*'s second sensor access mode does exactly this. The *listenToSensors* function in the *PhoneIoT* service can be provided a list of sensor and update period pairs; it then requests the mobile device to send a message to the student's NetsBlox project with the specified sensor data every time the update period elapses. Figure 2 shows how to request and receive location updates every two seconds. Note that this *streaming* technique has the added benefit of automatically breaking the values up into separate variables like "latitude" and "longitude" rather than receiving a list as in Figure 1b. With these two simple techniques, students can immediately begin accessing all of *PhoneIoT*'s supported sensors including the accelerometer, location sensor, and microphone, the magnetic field sensor, gyroscope, orientation sensor, or step counter, in their programs.

The second main feature of *PhoneIoT* is the **custom interactive display**. This lets students display information from NetsBlox graphically on the phone screen with labels, text fields, image displays, etc., as well as receive information from the user such as button presses, text entry, and joystick manipulation. The *PhoneIoT* block provides *addButton*, *addImageDisplay*, and *addJoystick*—functions that add a widget/element to the screen. These all typically take as input the location and the size of the new widget as well as an optional input which is a list of other values to set; these can control the orientation, font size, text color, background color, and other properties of the widget. Figure 3 shows an example of two widget-adding functions and the resulting phone screen; the image display has been filled with an image taken by the user's camera.

Importantly, one of the optional settings for a widget is an "event" (message type) to send to the student's project whenever the user interacts with the widget, such as pushing a button, moving a joystick, altering the textbox text, or updating the image. This allows instructors to cover important CS topics such as graphical interface design and event-based programming. This feature allows students to implement custom remote controllers for their NetsBlox projects. The example in Figure 3 is all the code needed to send images and text from a student's phone and display them on the NetsBlox stage running in the browser.

Figure 1

(a) Code required to initially connect to a PhoneIoT device ("device" variable holds the device ID) (Left) and (b) Example request for current device location in terms of latitude and longitude (result is displayed) (Right)

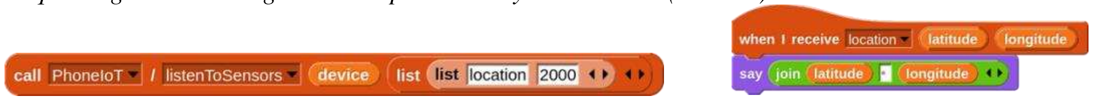


```
call PhoneIoT / setCredentials device 25e627fb
```

```
call PhoneIoT / getLocation device
1 36.1271151
2 -86.7792714
```

Figure 2

Requesting and receiving location updates every two seconds (2000ms)

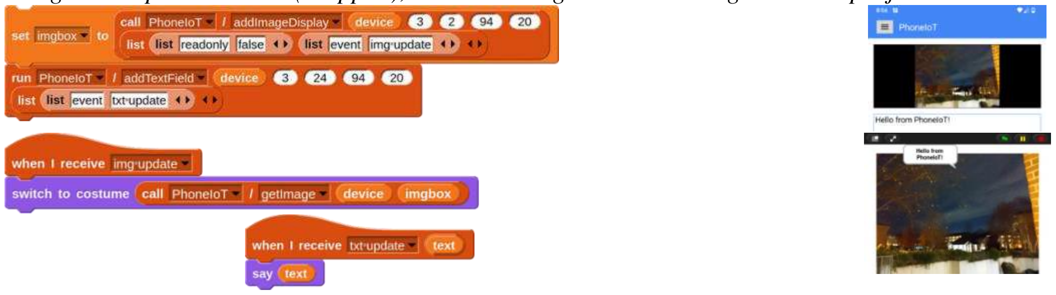


```
call PhoneIoT / listenToSensors device list list location 2000
```

```
when I receive location latitude longitude
say join latitude   longitude
```

Figure 3

Example code to make NetsBlox sprites display camera images and say messages from the phone. The top-right image is the phone screen (cropped); the bottom right is the running NetsBlox project.



```
set imgbox to call PhoneIoT / addImageDisplay device 3 2 94 20
    list list readonly false list event img-update
run PhoneIoT / addTextField device 3 24 94 20
    list list event txt-update
```

```
when I receive img-update
switch to costume call PhoneIoT / getImage device imgbox
```

```
when I receive txt-update text
say text
```

Preliminary evidence from pilot summer camp

Summer camp description:

PhoneIoT was used in a recent online summer camp to introduce students to distributed computing (DC) and IoT. After a brief introduction to NetsBlox and its DC features through a Weather app and MovieDB app, our project-based, hands-on curriculum advanced to multiple interesting *IoT* applications first involving robots, and then *PhoneIoT* apps such as: (1) streaming and plotting 3-axis acceleration data in real time; this project concept was eventually expanded from gathering raw data into a more engaging app where students recreated the classic labyrinth game with a twist: they could tilt their phone to control the ball on the NetsBlox stage; (2) creating graphical controls such as virtual buttons and touchpads, some of which were later used to implement remote controllers for games such as pong. All projects were done as iterative exercises: the instructors showed the *PhoneIoT* or NetsBlox feature and then students had to complete partially complete code. The capstone project was to turn their phone into a remote controller to command virtual robots (which had been previously introduced in the camp's robotics component) with custom controller layouts and behaviors. Students were free to use any

PhoneIoT features, resulting in creative original projects such as slider-based throttles for each wheel, single joystick-based steering/driving, and button-based controls for discrete commands (e.g., "turn left 10 degrees"). As the final challenge, students used their custom controllers to compete and drive their robot around an obstacle course in as short a time as possible. All students completed all tasks.

Feedback and results:

Nine students of color participated in the pilot camp (6 male, 3 female; 6 Asian, 2 Black, 1 Hispanic) from 9, 10, and 11th grades. Student engagement in the camp was very high. In the post-survey, students ranked rank the 2 distributed computing, 3 robot navigation, and 2 *PhoneIoT* projects from 1 (highest) to 7 (lowest). The *PhoneIoT* projects ranked the highest with average score of 2.2 for the tilting game and 3.2 for the remote controller app. There were significant average pre-to-post gains registered on 4-point likert scale survey questions on confidence, ability, and interest in CS. On the question on knowledge of how to build IoT applications, there was a significant gain on the mean score: 1.4 (pre) to 2.9 (post). Open-ended responses on students' camp experience suggests that their perceptions of computer science became more positive and the camp expanded their understanding of this vast and growing discipline. Sample responses were: a) "*Before, I did not have a lot of experience in "Internet of Things," but this camp provided an excellent introduction. It allowed me to understand how hardware (robots, phones, sensors, etc.) interact with code. Overall, it gave me a good understanding of the field and allowed me to think about it as a potential career option*." b*) I learned a lot about how hardware interacts with the software we make over the course of this camp*." c) "*Programming in general is very broad and we can expand it to many things in computer science.*" d) *I saw ways we use coding in real life, before I thought they couldn't go beyond a computer screen."* These positive findings from our pilot are encouraging and validate our approach and the design principles behind the *PhoneIoT* innovation to make the growing Internet of Things topic accessible and engaging to K-12 students. Our ongoing work involves expanding *PhoneIoT* activities to include more personal apps such as mapping one's walking or running route and overlaying points of cultural and community interest.

References

Abichandani, P., Sivakumar, V., Lobo, D., Iaboni, C., & Shekhar, P. (2022). Internet-of-Things Curriculum, Pedagogy, and Assessment for STEM Education: A Review of Literature, in *IEEE Access*, Vol.10.

Brady, C., Broll, B., Stein, G., Jean, D., Grover, S., Cateté, V., ... & Lédeczi, Á. (2022). Block-based abstractions and expansive services to make advanced computing concepts accessible to novices. *Journal of Computer Languages*, *73*, 101156.

Broll, B., Lédeczi, A., Volgyesi, P., Sallai, J., Maroti, M., Carrillo, A.,…& J.D., Lu, M. (2017). A visual programming environment for learning distributed programming. In Proceedings of SIGCSE, ACM.

Goldenberg, P., Mark, J., Harvey, B., Cuoco, A., & Fries, M. (2020, February). Design Principles behind Beauty and Joy of Computing. In *Proceedings of the 51st ACM Technical Symposium on CS Education*.

Hendeby, G., Gustafsson, F., Wahlström, N., & Gunnarsson, S. (2017). Platform for teaching sensor fusion using a smartphone. *International Journal of Engineering Education*, 33, 781–789.

Harvey, B., Garcia, D. D., Barnes, T., Titterton, N., Armendariz, D., Segars, L., ... & Paley, J. (2013). Snap! (build your own blocks). In *Proceedings of the 44th SIGCSE Conference* (pp. 759-759). ACM.

Lee, V. R., Drake, J., Cain, R., & Thayne, J. (2021). Remembering what produced the data: Individual and social reconstruction in the context of a quantified self elementary data and statistics unit. *Cognition and Instruction*, *39*(4), 367-408.

Sentance, S. & Childs, K (2020). X-ing Boundaries with Physical Computing. In S. Grover (ed). *Computer Science in K-12: An A-to-Z Handbook on Teaching Programming* (pp 250-28). Edfinity.

Siegle, D. (2020). There's an app for that, and I made it. *Gifted Child Today*. 43, 64–71.

Srirama S. N., Mass, J., Koppel, M. K., Sepp, A., & Avanashvili, S. (2017). Smartphone-based Real-time Sensing and Actuation with the Cumulocity Internet of Things Platform. 1–6.

Wolber, D., Abelson, H., Spertus, E., & Looney, L. (2011). *App inventor*. O'Reilly Media, Inc..

Wolfgang, S. (2014). Pocket code: a scratch-like integrated development environment for your phone. *Proceedings of the companion publication of the 2014 ACM SIGPLAN conference on Systems, programming, and Applications: Software Humanity*. 35–36.

Weintrop, D., Killen, H., Munzar, T., & Franke, B. (2019, February). Block-based comprehension: Exploring and explaining student outcomes from a read-only block-based exam. In *Proceedings of the 50th ACM technical symposium on Computer Science Education* (pp. 1218-1224).

Acknowledgments

We thank the National Science Foundation (#1949472, #1949488, & #1949492) for their support of this work.