IoT Maker: Moving Beyond Basic Microcontrollers by Creating Complex Electro-Mechanical Devices Through Live Programming for Youth

ANONYMOUS AUTHOR(S)

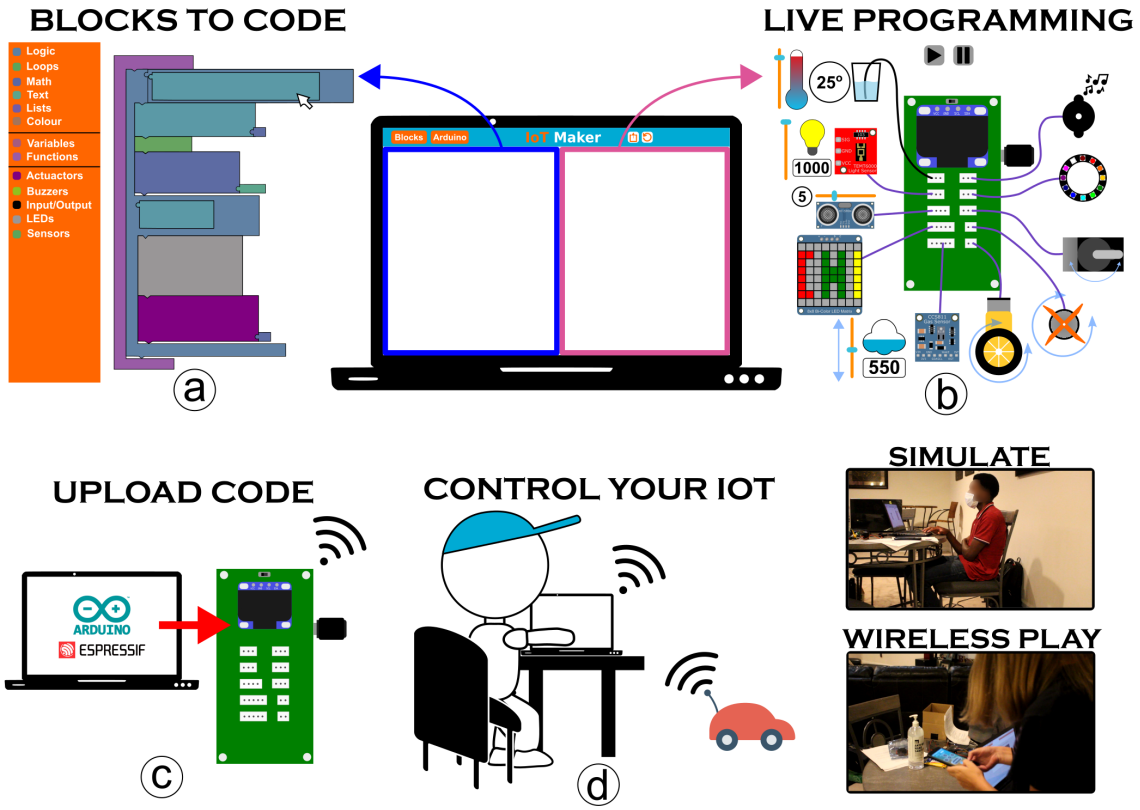

Figure 1: *IoT Maker is a web application that uses live programming to simulate the functionality of various electronic devices. Users can (a) drag and drop blocks of code into the programming environment and (b) watch their code being executed in real-time on the screen, while interacting with the various sensors via sliders, buttons, and color pickers. Once the user has a sufficient understanding of the functionality of their code, they can (c) upload the code to our customized iBoard, and (d) connect their phone, tablet, or computer to the iBoard, and control the electronics via WiFi.*

Microcontrollers have become a popular tool to use when designing both basic and complex electro-mechanical devices. Block-based

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2020 Association for Computing Machinery.
Manuscript submitted to ACM

programming interfaces are an ideal environment to program physical computing devices, lowering the barrier of entry for novice programmers. The advent of live programming has given rise to a more engaging and thorough process of programming such devices, but leaves much to be desired due to primitive functionalities or high barriers of entry. To understand the impact of live programming on designing complex electro-mechanical devices, we developed a plug-and-play electronics module and an accompanying block-based live programming environment. We conducted a pilot study with 8 experts in making electro-mechanical devices to elicit critical User Experience criteria, followed by a comparative user study with 12 students (age=11-18). The results show that our system helps reduce the barrier to entry for young users to create complex electro-mechanical devices in a comparatively short period of time.

CCS Concepts: • **Human-centered computing** → **Open source software**; **User interface toolkits**.

Additional Key Words and Phrases: Maker Culture, Block Programming, Live Programming, Internet of Things

ACM Reference Format:
Anonymous Author(s). 2020. IoT Maker: Moving Beyond Basic Microcontrollers by Creating Complex Electro-Mechanical Devices Through Live Programming for Youth. *J. ACM* 37, 4, Article 111 (August 2020), 20 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The prevalence of varying types of physical computing kits both inside and outside of the classroom is a testament to the growth and expansion of the field of physical computing over the years [7, 11, 28]. What is more, research has shown that incorporating physical computing devices into both formal and informal educational settings has given students a heightened sense of self-efficacy, gives way for access to new knowledge domains, offers new design and architecture paradigms, and engages youth by presenting new opportunities for learning [7, 11, 35]. Despite the large strides in the areas of physical computing and physical computing devices, many researchers are still confident that the apex of these fields is yet to be attained.

Over the past decade, physical computing devices have amassed a plethora of physical, electrical, and programmatic features that help make the experience of using one of these devices better [3, 7]. A feature that is rapidly being added for use with physical computing devices is the advent of block-based programming languages. Block programming editors have grown in popularity for physical computing devices because of their easy-to-use interfaces; both novices and experienced users can use them to help build large amounts of code by connecting several, color-coded blocks together in a logical way. Block-based programming is easier to use for novice programmers because it eliminates common pitfalls that novices run into when programming physical computing devices (e.g. memorization of library names, language syntax, data type conversion, etc.). Even though this method of programming is beneficial to the novice, it can be more arduous for seasoned programmers creating larger, more complex projects. While there are many physical computing devices with their own programming environments, Arduino [5] is among the top choice for novice users attempting their very first micro-controller project. Unfortunately, Arduino does not offer its users a block programming experience. However, the lack of such a feature has sparked a series of open-source projects aiming to do just that. Blocklyduino [8] is one project that has attempted to bring a block-based programming environment to the Arduino platform. With further advancement in this area, Live Programming [25] and Live Coding [30] have become a popular feature to include in both traditional computing environments, as well as physical computing ones. In particular, Live Programming in physical computing has been explored to make the process of building physical computing devices even more engaging and convenient. Live programming allows users to edit and debug code in real time through visual interactive feedback. For example, live programming environments have been added to the Microsoft® MakeCode platform to "bring computer science to life for all students," with "both block and text editors for

learners at different levels," [15]. Additionally, platforms like Scratch have begun to provide its community with live programming capabilities to combine the digital and physical worlds [26].

While live programming features bode well for helping students debug their code, the ultimate goal is to be able to create a working, physical prototype. The emergence of the Internet of Things (IoT) has invoked new ways for humans to interact with physical computing devices [4]. This is achieved by offering advanced WiFi, Bluetooth, Radio Frequency, and other wireless networking/communication protocols that are accessible through downloadable libraries and (sometimes) special hardware. Companies like Arduino [5] provide low-cost "shields" that can be attached to many of their Micro-Controller Units (MCU) to achieve this and have recently begun exploring IoT in their Education modules online [36]. Meanwhile, Espressif [18, 19] has features like these built into their hardware. While IoT solutions are rising in popularity, it is often difficult to implement for novice users because it requires a broad understanding of programming and networking concepts that novice programmers don't traditionally receive.

Each of these individual areas – live programming, block-based programming languages, and IoT – provide tremendous opportunities to educate novices (especially youth) some fundamental concepts in the areas of electronics, circuitry, computer programming, et. al. However, at the intersection therein lies a gap which, if addressed properly, can prove to give novice users the ability to go beyond what basic MCUs can typically do. To explore this intersection, we developed IoT Maker, an IoT creation platform that utilizes live programming, an integrated block-based programming language, and a lightweight companion app for mobile/desktop that pairs wirelessly with a customized circuit board that we call the iBoard. IoT Maker extends access to a wide range of sensors and output components through a set of custom blocks that otherwise need special libraries and require the user to go through a lot of documentation. Our system invokes the concept of Low Floors and Wide Walls from Resnick and Silverman [32] by reducing the barrier of entry for novices that lack the prerequisite electronics and programming knowledge, while raising the level at which novices typically enter this area. Using IoT Maker, users can stack blocks to build functions for their IoT devices, which can be simulated in real-time for the users on-screen. First, users select a set of electronics from our Electronics Bank to control and plug them into our specially designed iBoard. Once a user has a sufficient understanding of how their code will function, they can upload that code to the iBoard via an Over-The-Air (OTA) WiFi updating protocol. Finally, users control the electronics with our companion app by selecting a User Interface (UI) element and the function(s) that they want to control. IoT Maker allows users to go beyond the capabilities of a basic MCUs by giving access to advanced electronic components. We offer three main contributions in this paper:

(1) The design rationale for a system that allows novice users to prototype electro-mechanical IoT devices, which were extracted from 4 experts in physical computing devices, electronics, and physical prototyping,
(2) The IoT Maker system, which provides users with a plug-and-play electronics kit, a block programming environment, a live programming simulation tool, and a companion app to control everything via WiFi,
(3) The results from a study conducted on 15 youth (age 11-18), which shows that our users were able create more complex electro-mechanical devices than they would have without our system.

2 RELATED WORK

IoT Maker is motivated by prior work in the areas of block programming environments, physical computing devices, and real-time simulation tools using live programming. Each of these areas is reviewed below.

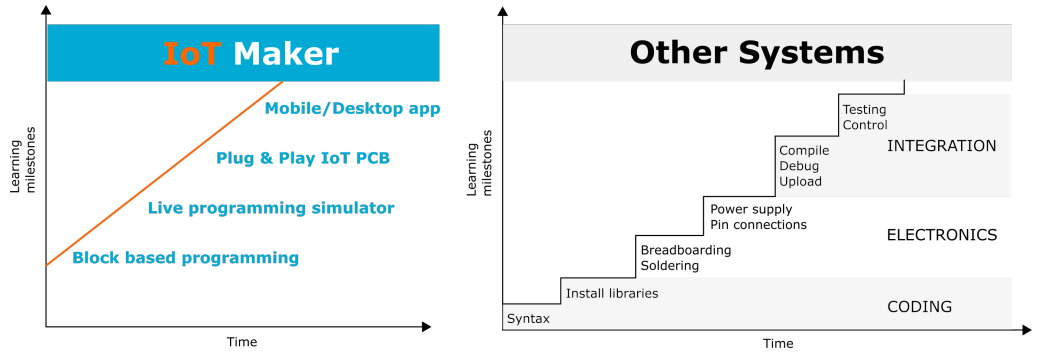

Figure 2: *Learning curve comparison between IoT Maker and other systems. Other systems go through a discrete set of skills that user needs to learn while IoT Maker eliminates the number of prerequisites and offers a continuous learning curve through its interrelated subsystems. This reduces the time required for learning.*

2.1 Robotics and Tangible Computing Devices

Computer programming has been at the forefront of teaching young students how to control electronic devices for nearly half a century [31]. The idea of programmable bricks – electronics blocks with small computers in them – was revolutionary in the sense that it gave way for a new frontier of learning. Students no longer had to stare at a computer screen and watch something virtual happen as a result of their programming practices. Now, learners can perceive some tangible operation happening to this brick after their programming was completed. Physical computing devices sit at the intersection of mechanics, electronics, and programming; manufacturers of these devices have been trying to make the experience of using their devices better by integrating one – or all – of these areas. For example, Paulo Blikstein [7] has discussed the underlying design principles of multiple generations of physical computing kits that have evolved over the years. Arduino and Raspberry Pi [2] gained popularity and have become among the most preferred platforms for physical computing. These platforms offer enhanced flexibility through their different shields, break-out boards, and other attachments. Higher level programming languages with component specific libraries can be compiled in their respective Integrated Developer Environments (IDE). Given the plethora of applications for these devices, there is no wonder why hobbyists, industries, and even Universities have begun to design interfaces that make using these devices easier.

Like industries, prior work suggests that researchers aim to understand how different populations could utilize these devices and what impact these devices make on their lives. One such work is COBRIX, which is designed to enable visually impaired humans to construct physical code with LEGO bricks to solve various computing problems [3], but lacks the ability to troubleshoot until after you've tested the final configuration. On the other hand, CODAL is a platform much like the aforementioned Microsoft® MakeCode that offers innovative engineering features for physical computing [6]. Non-experts can use these systems to program devices that can send and receive data to/from other IoT devices, but building on top of this knowledge is not easy as the electronics can become quite convoluted as more electronics are introduced. Glenn, et. al. designed a storytelling system that gives their users access to a design space for novices to design, build, program, and interact a physical computing device with its virtual AR characters [20]. However, the feedback loop on properly programmed devices is quite large as users aren't able to see what they've programmed on their device until after all devices and AR content is set-up and ready to go. Lastly, Modkit provides its users with a simple interface that helps users visualize how to build the electronics for several physical computing

devices, including the Arduino Uno, and a block programming environment to help quickly generate code to control those devices [29]. Modkit takes the programming a step further by utilizing the Arduino Command Line Interface (CLI), which permits the simulation of electrical components on the actual device rather than on a computer screen. Modkit excels at these, but carries the same inability to easily scaffold upon prior experiences due to complex wiring diagrams and use of more advanced electrical components.

Although the realms of science and technology is intriguing and interconnected, robotics is a field that encompasses many of these topics at the same time. The intersection of mechanics, programming, and electronics creates a space that is difficult for those that do not possess prior knowledge to be successful in. Kits such as LittleBits [27], VEX Robotics [40], Lego Mindstorm [21, 22], and Cubelets [43] center their product experience around reducing the barrier to entry and difficulty of the learning process. One way they achieve this is by focusing more on the ease of assembling the physical connections of the computing devices. The press fit design in Lego bricks and the wireless magnetic connectors in LittleBits enhance the hands-on interaction with the components. These kits center around creativity of the users, keeping the functionality of the MCU at the forefront, and the intricate connections hidden. Moreover, prior research studies [35] have also linked increases in student self-efficacy with their involvement in competitions centered around these types of kits, such as the VEX Robotics Competition [33, 41]. These platforms are widely popular for all these reasons and serve their purpose well. IoT Maker separates itself by providing a physical computing device called the iBoard, which boasts a design for plug-and-play electronics, a large library of electrical components for users to choose from, and a simulation tool built into the block programming environment for users to test and debug their code before they ever upload it to the iBoard.

Across the array of commercial toolkits, however, there is a huge amount of overlap between the available hardware and software designs, which creates large redundancies in the types of learning opportunities available within each kit. For example, many kits including simple inputs/output devices like motors, light sensors, etc. that target learning of simple computing and mechanical motion objectives with few exceptions that target higher level physics and electro-mechanical learning objectives [34]. Our kit, IoT Maker, by contrast is designed to target more complicated electro-mechanical engineering and computing objectives to address this gap in prior research and development.

2.2 Live Programming Simulation

Live programming, not to be confused with Live coding, is not a new field of research; however, it is starting to gain higher traction in Physical Computing environments [11]. According to Click Nilson, live coding is described as *"the art of programming a computer under concert conditions, [and] can enable a novel engagement with the notions of algorithm, and the mapping from code to musical resultant"* [13, 30, 42]. Although music and live coding isn't the main focus of IoT Maker, the act of "liveness" is very obviously attractive for learning – you do something and receive nearly immediate feedback that shows you the affects of what you did. This powerful notion is displayed in numerous places, which Kubelka, et. al explore in their review paper [25]. We discussed earlier how Microsoft aims to use live programming to help make physical computing more accessible, but several industries like Google® are also adding some live programming features to it's Chrome Web Development kit [37]. Apple and Facebook have joined in with their respective Swift and React platforms, which give developers options to see what will happen when they implement certain features to their software in real-time [17, 38]. Large corporations, researchers, and small software developer alike are looking into live programming as a way to enhance the programming experience. How could this be applied to other areas, such as IoT creation? And in what ways can live programming help with the creation of an electro-mechanical device?

Iot Maker explores these questions by pushing the limits of what is possible for novice users. IoT Maker separates itself from these other platforms because of its intersection of the areas covered in this section of the paper. In the following sections, we describe the System Overview of the IoT Maker system, as well as the workflow for using IoT Maker.

3 SYSTEM OVERVIEW

Our system aims to combine multiple aspects of physical computing by blending them into three sub-systems. The web application that is made of block-based programming platform and a visual simulator that allows users to see their code execute on the screen before they can actually put things together, the iBoard that allows users to simply plug and play a wide variety input and output devices and a companion mobile/desktop application that allows users to control their devices wirelessly. Each of these sub-systems has been elaborated in the following sub-sections.

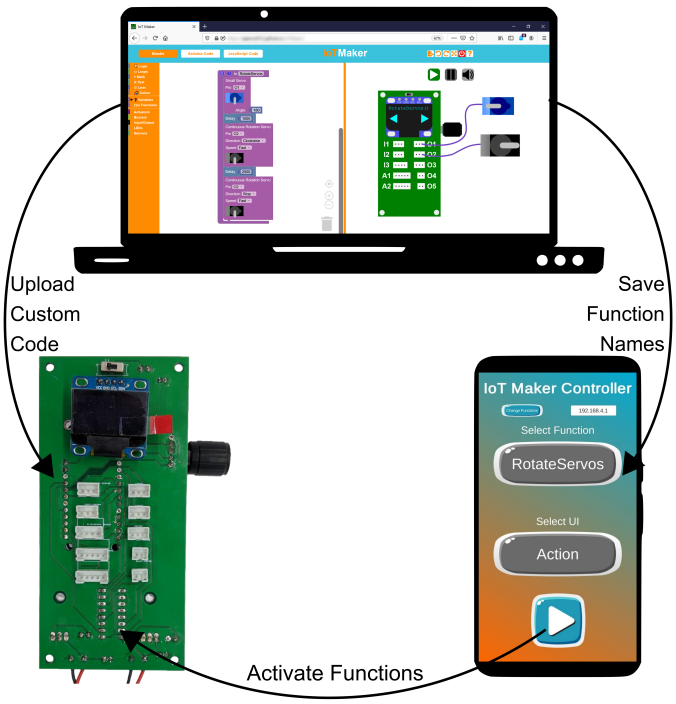

Figure 3: *Users generate code for their IoT devices using our block-based programming environment. Users can upload their code to the iBoard and the custom Function Names are then read by the mobile app. Using the mobile app, users can connect to and activate their custom functions on the iBoard.*

3.1 Block Programming Interface

The first step to creating a device with our system is to decide which components the user desires and what functions the device will be capable of. This construction takes place on our IoT Maker website which is a combination of a graphical programming interface with a simulator tool. The programming panel utilizes Google Blockly [1] - an open source project that allows users to create syntactically correct code by placing and linking code blocks. With the foundation for code creation set by Blockly, the generated code output needed to be compatible with our Espressif (ESP32) boards; so we modified an updated version of the aforementioned BlocklyDuino [8, 9] web application to create a compliant language. To use the block panel, makers can select various components (from our IoT Maker toolkit) and place them in custom functions. Each component has its own unique block for determining pin-placement and functionality. The blocks inherently prevent the user from connecting a device to a wrong pin in the iBoard. This is further reinforced by the design iBoard with different number of pins for different devices. These features prevent any connection errors and thus avoid any frustrating experiences for the user. We have a mixture of Input and Output featuring various attributes and hidden features that require special emphasis on each block. This required us to design each block from the ground up so that it mimics the experience if one were to manually program one of these components (such as including the ability to

modify the picture on an LED directly on the block). As users are moving blocks around, they can see the constructed ESP32/Arduino code generated in realtime by switching to the Arduino tab. Even though our system tailors to novice programmers, if users have prior experience with Arduino programming, they are able to directly edit the code in this window if they so desire. Since our boards use numerous custom libraries and components, each block placed will create all the necessary helper methods and header file links in the code so users do not need to worry about setting up and compiling an external library by themselves. After the user has set up a draft of their device, they are able to simulate how the device functions using the simulator panel.

3.2 Live Simulator

One of the barriers we aimed to tackle with our system is being able to instantly test complex micro-controllers without having to compile and upload code to a real board. This is important as this whole process can be time consuming and inconvenient in the event that components are missing or unavailable at the time of programming. The inclusion of a live simulator allows users to get an idea of what their code outputs as well as allows for quick adjustments that would otherwise require complete re-compilation of code. It also increases accessibility of hardware programming since users do not need to own or be near their own hardware to see what their functions can do. Our simulator is inspired by Microsoft's micro:bit ecosystem [15] where users can test simple circuits and electrical components, however, our goal was to expand on this idea as micro:bit is limited in number of components and gets very intimidating for new users with the amount of wires and tech displayed. As users place components in the programming panel, they will be rendered in the simulator in the same pin position that they would on the actual board. Once the user is satisfied with their functions and components, they can load up a function on the simulator and hit the play button. The simulator will then evaluate the desired block of code and render animations for output components and controllers for output components based on the code logic. Controllers for input components (such as temperature readings or RGB sensing) are interactable modules that allow users to simulate real-world conditions. Each change to an input will cause the function to re-evaluate since the boards constantly check every function in an infinite loop. The user is able to pause and play the simulator whenever they choose and any changes made in the programming panel will instantly update on the simulator. In the same fashion that our custom programming blocks generate Arduino code, they also generate Javascript code (specific to the simulator) which lets intermediate/expert users read and understand the flow of events that the simulator is evaluating. The visual simulator also acts as a visual guide for the users when they assemble the components to the iBoard. Since the users have confirmed the functionality of their devices visually on the simulator they are now more confident while building the devices.

3.3 iBoard

The electronic subsystem of IoT Maker is comprised of the iBoard and a repository of Input Output devices. Most hobby/DIY electronics kits available in the market require prerequisite knowledge of electronics and power systems. The iBoard bypasses these prerequisites and allows user to quickly build complex electro-mechanical devices. This is achieved by the inherent design of the PCB that ensures the correctness in pin connection and power supply while allowing users to simply plug and play the desired components. The iBoard incorporates a Wi-Fi capable micro-controller unit (MCU), Adafruit HUZZAH32 ESP32 Feather Board [18], and several in/output ports to help provide a modular and connected experience. Furthermore, the iBoard is compatible with a wide range of components - actuators, sensors, LED modules and other electronic components that support interaction with heat, light, sound, gas, color and motion. This gives the users the freedom to exercise creativity and imagination and build a wide variety of projects (see Figure 4).

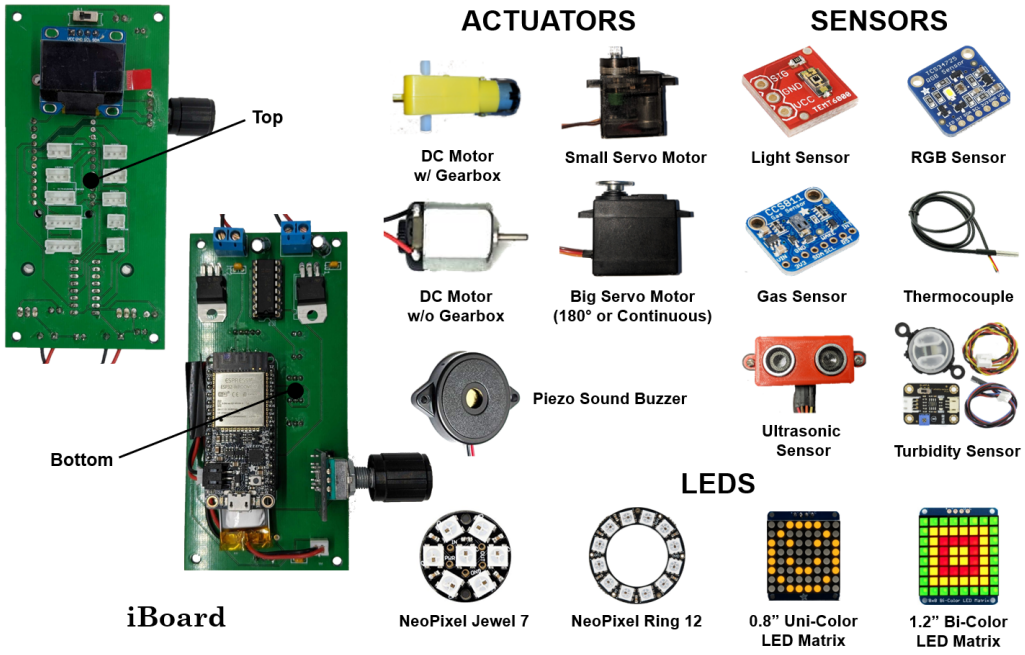

Figure 4: *iBoard and the Electronics Repository. 15 peripheral devices offering a variety of functionality*

Apart from being optimized for functionality, the iBoard is also designed for its form and usability. Designed to primarily provide a plug-and-play experience, several design optimizations were performed to enhance the user-friendliness of the board based on Gestalt's principles of perceptual organization [14]. The principle of proximity states that objects which are close together in space tend to be grouped together when visually inspected; whereas, the principle of continuity states that objects that are arranged in straight or curved lines tend to be grouped together. In accordance with these principles, the input and output ports were grouped separately in straight lines, in order to help the user distinguish between similar I/O male pin headers on the iBoard. The non-interactable electrical components, such as the MCU and power regulators, were placed on the rear side of the iBoard, whilst the interactable components, such as the input output ports and OLED screen, were on the front. The iBoard gives users access to 10 I/O ports. While 5 ports are used for input and sensor devices, the other 5 ports are used for output and actuator devices. In order to facilitate the use of different types of input components, whilst keeping things simple, three different types of input pin headers were used; 3, 4 and 5-pin input headers. While the 3-pin and 4-pin inputs connect to devices that supply simple analog/digital data to the MCU, the 5-pin inputs utilize the I2C communication protocol. The use of I2C communication protocol allows control of both 5-pin devices with a single pin on the MCU, which frees up GPIO pins for other uses and devices. With so many different components, power distribution was a key design concern, and hence, we optimized power delivery such that the power load is split across three different power sources. The MCU, OLED display and rotary encoder are powered by a 3.7-volt Lithium-Polymer battery that is tucked away in between the MCU and the PCB surface. The power drawn by the devices connected to the I/O ports is split evenly between two 9-volt Lithium-Ion batteries that power the iBoard, and this allows for the iBoard to operate with all pins and components being used at once. The wide variety of in/output ports can also allow intermediate/expert users to plug in other devices that are not included in our electronics repository at this stage.

3.4 Mobile App

IoT Maker's subsystems are all designed with ease of use in mind as a major design priority, and this focus continues to the mobile application that allows users to start their custom functions in a manner preferable to them. Unity's 3D application development suite [39] allows for multi-platform app development, and hence was chosen as the software through which the mobile control application was made. While the visual elements of the application are optimized for mobile devices, it is also capable of being used on personal computers. Upon startup, it reads in a text file generated by the IoT Maker Web application that contains the names of all the user created functions and populates its function list. In an effort to supply the user with different possible methods to control their device, the app then presents four different ways in which they can control their custom functions; "Action", "Slider", "Directional Pad" and "Toggle". The "Action" option provides a button that runs a function upon button press, while the "Toggle" option allows the user to select the function that is called when the button is toggled on and the function that activates when it is toggled off. The "Slider" option presents an interactable slider which sends values between 0 and 180 to the iBoard, which enables the user to control certain devices and variables with variable input. The "Directional Pad" allows for the user to assign functions to the four directional buttons, each of which replicate the functionality of the "Action" button; this is particularly useful when the user creates a multi-function device such as a car or a garage. IoT Maker's iBoard communicates with the mobile application through User Datagram Protocol (UDP) packets, which utilizes a common Wi-Fi network and the IP Address of the iBoard to send basic commands and information between the two. The UDP protocol allows for quick communication with the iBoard, which in turn makes it very responsive to button presses and user input. Moreover, the IP Address of the board is editable in the application, making it possible to connect to other iBoards without restarting the app

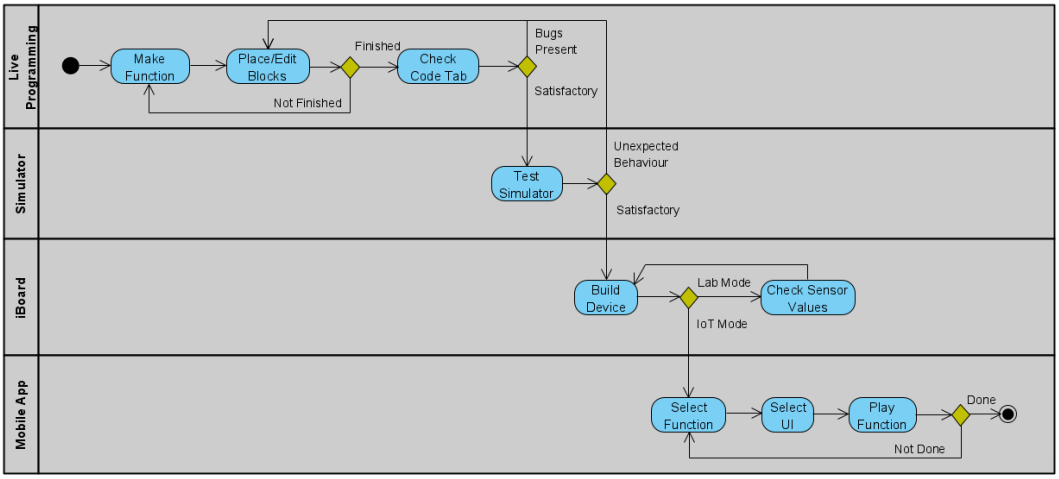

Figure 5: *A flow of events for a user in the IoT Maker System*

4 EXPERT STUDY

In order to gain insight on the impact of the IoT Maker system on the way it can be used to help both novices and experts create complex electro-mechanical devices, we sought counsel from several "Expert" users. Below, we describe

our inclusion criteria for our Expert users, our research questions, the methodology for the study, and results from the study.

4.1 Inclusion Criteria

We recruited our expert users from a group of graduate and professional students at a large Midwestern University. The students were selected based on their work and expertise, which were in any of these fields: (1) Programming Physical Computing Devices, (2) Electronics & Circuitry, or (3) Building & Using Electro-Mechanical Devices. Table 1 gives an overview of the Experts' experience level in those three areas. The study lasted approximately 90 minutes, and at the conclusion, each Expert was offered compensation in the form of a $20 Amazon gift card. Before starting the study, Experts were asked to fill out a pre-study survey that asked them a series of questions regarding their confidence levels in each of the above fields in which they indicated they were experts. The questions were adapted from the Internet Proficiency Scale developed by Eastin and LaRose [16]. Changes were made to be relevant to the field in which the Expert stated they are proficient, but not to change the meaning of the statement. In order to stay consistent with the prior work, all questions modified from the Internet Proficiency Scale are based on a 7-point Likert Scale, while the questions unrelated to it are based on a 5-point scale. All results are reported in section 4.4: Study Results.

4.2 Study Procedure

The purpose of our Expert Study was to gain a better understanding of how IoT Maker could be used to create more complex electro-mechanical devices. As a result, we aimed to answer the following Research Questions:

RQ1 What design considerations do we need to ensure that students aren't missing fundamental learning objectives by using our system?

RQ2 What foreseeable issues may arise for students in our target age group (11-18)?

RQ3 How could this system be attractive/usable by both novices and experts? What features are missing/should be omitted?

RQ4 In what ways can live programming help with the design of an electro-mechanical device? Does our system do a good job of this?

Guided by our Research Questions, we crafted the study to mimic that of the Youth Study so the experts could make more accurate comments on any User Experience (UX) struggles and pitfalls. Before the study, participants were given the option to conduct the study in three different ways.[1]

[1]These options were given to the participants to accommodate them and keep participants and researchers safe as a result of the COVID-19 Pandemic.

Expert #	Gender	Phys. Comp. Device Exp.						Electronics Exp					Electro-Mech. Device Exp.					
		Q1	Q2	Q3	Q4	Q5	Q6	Q1	Q2	Q3	Q4	Q5	Q1	Q2	Q3	Q4	Q5	Q6
1	M	6	5	6	5	6	4	7	5	5	6	5	6	5	6	5	5	5
2	M	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	6
3	M	6	5	6	5	5	5	6	5	5	5	5	–	–	–	–	–	–
4	F	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7

Table 1. Pre-Survey Results from the Expert Study. All demographic questions are based on a 7-point Likert Scale (1 - strongly disagree, 7 - strongly agree). Q1-6 can be found in Appendix A of this paper.

(1) Users could conduct the study in-person, as normal. Social distancing guidelines were put in place and masks were required of all in-person participants. All equipment was sanitized before and after each study.
(2) Users could conduct a hybrid virtual study over the Zoom video conferencing platform. Equipment would be packed up and shipped to participants, then configured for them at the beginning of the study. All equipment was sanitized before being shipped.
(3) Users could conduct the study completely virtual, where the participant would use the IoT Maker website to program and simulate the electronics. Since the iBoard would be in the possession of the researchers, we uploaded the code and showed the functioning circuit to the participant using a top-down projector.

Once the pre-survey was complete, the Experts were taken through a short tutorial of the IoT Maker website to highlight all of its relevant features. Next, the Experts were given a total of five Coding Challenges to complete throughout the duration of the study (See Figure 6 for details regarding the Coding Challenges). The Coding Challenges were distributed via a pseudo-code document with instructions on how to progress from one step to the next (See Supplemental Material for a copy of the pseudo-code document). The pseudo-code was written using a scaffolding teaching practice that is employed in Computer Science and Physical Computing Education [10, 45] so that users could build on what they learned as they progress.

Coding Challenges for Expert and Youth Studies

Coding Challenge Description	Targeted Concepts (Circuitry)	Targeted Concepts (Coding)	Targeted Concepts (Physical)	Components (IoT Maker)	Components (Arduino)
Build the circuit and code that makes a yellow DC Motor rotate CCW for 5 seconds, then CW for 10 seconds. Do the same with the small DC motors	Polarity & reversing polarity (intentionally) Connections	Time delays	N/A	iBoard, 1 yellow DC motor	Arduino, Yellow DC Motor, PNP Transistor (P2N2222A), Diode (1N4148), 220Ω Resistor, Wires
Build the circuit and code that turns the Neopixel LED white when the lights get dim. Create a Boolean variable that changes to true/false based on this.	Connections Input vs. Output	Conditional Statements Boolean Logic	Color Combinations	iBoard, Neopixel Jewel (7), Ambient Light Sensor	Arduino, Neopixel Jewel (7), Ambient Light Sensor 1-10kΩ Resistor, Wires
Use the RGB sensor to read the red, green, and blue values of an object and make a pattern on the small LED Backpack if there's a high red value, else turn off	Connections Input vs. Output	Conditional Statements RGB Values	I²C Communication	iBoard, Uni-Color Matrix, RGB Sensor Red Object	Arduino, Unicolor Matrix, RGB Sensor Red Object Wires
Build the circuit and code to iterate through a for loop to rotate a small servo from 0-180°, then back to 0	Connections Servo motors vs. DC motors	Loops Variables	Angles, PWM	iBoard, Small Servo	Arduino, Small Servo Wires
Build the car and attach the 2 DC motors. Create 5 functions (Forward, Backward, Left, Right, Stop). Add whatever functionality you want with 2+ more electrical components	Apply What You Learned	Apply What You Learned	Apply What You Learned	iBoard, 2 Yellow DC Motors, Extra Components	Arduino, 2 Yellow DC Motors, L293D Motor Driver, Extra Components, Wires

Figure 6: *Coding Challenges used for both the Expert and Youth Studies. The table shows a quick comparison between components that are needed to implement this coding challenge with IoT Maker vs. Arduino.*

Between Coding Challenges, users uploaded their code to the iBoard and utilized our IoT Maker Controller app to wirelessly control the code that they wrote. The final coding challenge was an open-ended project that asked users to create their own code for a car, which was designed by the researchers. After compiling and uploading their code to the iBoard, the participants used the Directional Pad on the IoT Maker Controller to operate the car with all the additional functionality that they programmed into it. At the conclusion of the study, participants were asked to fill out a post-study survey.

4.3 Data Analysis

We used a mixture of qualitative and quantitative data to assess our research questions. We analyzed the code that they had written, observational field notes taken by one of the researchers, and the pre- and post-survey data. The pre-survey gathered information about the participants, including demographics and prior relevant experience with Physical Computing Devices, Electronics & Circuitry, and Electro-mechanical Devices. The post survey asked participants questions related to the study, their experience using the system, and future experiences of students in our target age group. Both surveys used a mixture of closed-ended Likert scale questions [23] and free response questions (e.g. *In your opinion, what aspects of using the system might young students struggle with when using the system?* and *As an expert, how would you use this system?*).

4.4 Study Results

In this section, we discuss our key themes from the Expert Study and some common occurrences amongst participants related to using the IoT Maker website, the iBoard & Electronics, as well as the IoT Maker Controller. We report all names as pseudonyms with (gender, age) and all Likert Scale questions are reported with mean (M), median (m), and standard deviation (σ).

4.4.1 Participant Demographics : Based on the pre-survey results, we found that the Experts were generally familiar with live programming tools (M = 3.25, m = 2.5, σ = 0.473). As shown from Table 1, participants generally had rich experiences in programming physical computing devices, electronics/circuitry, or building/using electro-mechanical devices with high confidence on these devices.

4.4.2 Coding Challenges : During the Expert Studies, the participants were all engaged and enthusiastic to continue using the system throughout. Each Expert completed all 5 of the Coding Challenges in the allotted time with little-to-no issues. However, when asked where young students might struggle when using IoT Maker, Xavier (Male, 24) commented that, *"When things don't work out, they may not be able to find the problem easily using the current Blockly UI. Especially when the function has more variables, and logics,"*, while also stating that *"For experts, the provided functionalities may be limited. And when things turn to be complex, blockly may not be a good way to program,"* when asked if IoT Maker could be attractive/usable by both novices and experts.

Lewis (Male, 34) mentioned in his post-survey that *"I like the way it is. It has enough information in terms of features. Because of the tasks, I didn't use all the functionalities and capabilities so I don't really know if I will omit something,"* when asked if he would omit any features. Thus, it may be beneficial to require more time and more coding challenges to get a thorough understanding of the system and its features. Inevitably, after reviewing the data, we decided to stick with the Coding Challenges that were curated so as to not overload the young students in the next study.

Additionally, Experts were asked if the actions required to make the simulator function properly met their expectations, to which they all responded favorably (M = 4.5, m = 4, σ = 0.500). When asked how IoT Maker can help in the design of

electro-mechanical devices, 3 of our 4 Experts mentioned the importance of testing the written code without having to "burn" the code onto the iBoard first and 2 of them acknowledged how it helps make troubleshooting circuitry and debugging code easier. Lewis mentioned said, *"it won't help you to build the mechanical part, but it will give you an overview of all the elements you need to complete the task."* Finally, each Expert responded favorably when asked if the system does an overall good job of helping build electro-mechanical devices (M = 4.75, m = 4.5, σ = 0.433)

4.4.3 Learning Outcomes for Youth : Overall, the Experts believe the IoT Maker system to be suitable for teaching young students. Several experts pointed out the difficulty for learning electronics and computer programming at the same time, and indicated IoT Maker as a system that can be used to teach others, run simulations, and debug their code and devices. Despite the positive feedback regarding debugging code generated with IoT Maker, Xavier was very adamant about the lack of debugging tools found in Physical Computing software, including IoT Maker. He says, *"a more powerful debug tool properly integrated with the blockly programming UI. And, maybe the UI on the cellphone can also be integrated into the computer-based UI. Then the user will experience a more fluent and completed programming process, then move to the hardware to check the final result."* His comments are addressed in part during the next study with our Youth users.

4.4.4 Summary : Overall, experts gave very positive reviews for the system, as well as the UI design of IoT Maker. From these expert studies, we found the advantages of the IoT Maker system to be as follows:

- IoT Maker provides a platform to simulate a design project, removes part of the logic creation load from a project, and gives an overview of all the elements you need to complete the task.
- IoT Maker gives real-time output to help increase the design/debug/troubleshooting speed (more iterations).
- Has potential to do well at teaching novices how to program physical computing devices to control a wide variety of electronic hardware.

We took the information gathered from the Expert Studies and applied our findings to conduct our study with youth users in our target age group. This study is covered in the next section of this paper.

5 YOUTH STUDY

In order to gain insight on the impact of the IoT Maker system on the way it can be used to help novices users in our target age group (11 – 18) create complex electro-mechanical devices, we conducted several individual and group study sessions with 15 young people. Below, we describe our inclusion criteria for our Youth users, our research questions, the methodology for the study, and results from the study.

5.1 Inclusion Criteria

All of our Youth users were recruited via word of mouth and postings on our social media pages. Informed consent regarding the study was received from both parents (if under 18 years old) and students. We had 15 participants total (4 male, 11 female) across seven total workshops. The study lasted approximately 90 minutes, and at the conclusion, each Expert was offered compensation in the form of a $20 Amazon gift card. See Table 2 for participant demographics.

Before starting the study, participants were asked to fill out a similar pre-study survey to that of the Experts. The same questions regarding their confidence levels in each of the aforementioned fields were asked in order to compare and contrast the differences in self-reporting between Expert and Youth users. All results are reported in section 5.4: Study Results.

Pseudonym	Gender	Age	Phys. Comp. Device Exp.						Electronics Exp					Electro-Mech. Device Exp.					
			Q1	Q2	Q3	Q4	Q5	Q6	Q1	Q2	Q3	Q4	Q5	Q1	Q2	Q3	Q4	Q5	Q6
Vincent	M	16	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Donald	M	18	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Erica	F	15	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Kaitlyn	F	14	–	–	–	–	–	–	2	2	2	1	2	2	2	2	2	2	2
Kayla	F	11	4	5	5	4	4	4	4	4	3	4	4	–	–	–	–	–	–
Samantha	F	13	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Mica	F	18	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Brittany	F	18	5	2	5	3	3	2	–	–	–	–	–	–	–	–	–	–	–
Jennifer	F	17	3	2	3	2	2	2	2	2	2	2	2	3	2	3	2	2	2
Francine	F	18	2	2	4	3	2	2	4	2	2	2	3	–	–	–	–	–	–
Katy	F	17	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Victoria	F	13	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Garrett	M	18	–	–	–	–	–	–	4	3	1	5	2	–	–	–	–	–	–
Jayla	F	18	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
George	M	12	4	4	5	6	6	5	–	–	–	–	–	–	–	–	–	–	–

Table 2. Pre-Survey Results from the Youth Study. All demographic questions are based on a 7-point Likert Scale (1 - strongly disagree, 7 - strongly agree). Q1-6 can be found in Appendix A of this paper.

5.2 Study Procedure

The purpose of our Youth Study was to gain a better understanding of how IoT Maker could be used by Youth. As a result, we aimed to answer the following Research Questions:

RQ1 How does live programming help facilitate an understanding of simple and complex electro-mechanical devices?

RQ2 What effects does live programming have on Youth's ability to grasp programming concepts for physical computing devices?

RQ3 Why is it hard to live program complex electronics now?

RQ4 What does our board do to open the capacity and capability for live programming complex electro-mechanical devices?

Guided by our Research Questions, we designed our study to include both simple and complex electro-mechanical components. Before the study, participants were given the option to conduct the study in the same three ways that were given to the Expert users.[2]

For this study, rather than uploading the students written code to the iBoard after each Coding Challenge, we decided to pre-compile the code. This decision was made to decrease the time spent compiling during the study and afford the students more time to complete the challenges. It did not hinder the experience of using the system since the outcome of the Coding Challenges is the same with the exception of Coding Challenge 5. Further, we decided to make Coding Challenge 4 a bonus challenge rather than requiring students to complete it because during the Expert Study, we noticed

[2]These options were given to the participants to accommodate them and keep participants and researchers safe as a result of the COVID-19 Pandemic.

that none of the experts wanted to use the servo motor nor a `for loop` for the final Coding Challenge. This decision also gave students more time to iterate through and complete the final Coding Challenge. At the conclusion of the study, the students were asked to partake in a short interview with the researchers, as well as fill out a post-study survey.

5.3 Data Analysis

We used a mixture of qualitative and quantitative data to assess our research questions. We analyzed the code that the participants had written, transcribed the post-study interviews, and evaluated the pre- and post-survey data. The post-study survey asked participants questions related to the study and their experience using the system. Both surveys used a mixture of closed-ended Likert scale questions [23] and free response questions (e.g. *In your own words, what is different about IoT Maker vs. other programming software that you may have used in the past?* and *In what ways can live programming help with the design of an electro-mechanical device?*). The post-study interview asked four open-ended questions. A copy of the interview questions can be found in Supplementary Materials.

5.4 Study Results

In this section, we discuss our key themes from the Youth Study. Again, we touch on some common occurrences amongst participants related to using the IoT Maker website, the iBoard & Electronics, as well as the IoT Maker Controller. We also highlight interview comments and survey data from the students who used the system. For this study, **all** close-ended Likert-style questions are on a 7-point scale. We report all names as pseudonyms with (gender, age) and all Likert Scale questions are reported with mean (M), median (m), and standard deviation (σ).

Figure 7: *Participant (a) driving a car made during the coding challenge and (b) explaining the working of the car to their elder sibling*

5.4.1 Participant Demographics : Based on the pre-survey results, we found that the Youth were not very familiar with live programming tools (M = 2.133, m = 2, σ = 0.247). As shown from Table 2, most participants did not report having very many experiences (if any at all) with programming physical computing devices, electronics/circuitry, or building/using electro-mechanical devices with moderate or low confidence on these devices.

5.4.2 Coding Challenges : During the Youth Studies, every participant was able to complete the Coding Challenges during the 90 minute study with the exception of a group of 4 (Mica, Female, 18; Brittany, Female, 18; Jennifer, Female, 17; Francine, Female, 18). Most of our students had issues following along with the pseudo-code, but several students verbally commented that they believe it was their impatience and resistance to fully read the instructions that made them fall behind. In fact, students reported that the pseudo-code was clear and easy to follow step-by-step (M = 6.33, m = 7, σ = 0.943) and helped them understand the blocks more (M = 6.2, m = 7, σ = 1.166). However, Mica, Brittany, Jennifer, and Francine stated to the researchers that they would like to have seen more **visual** instructions that show up on the screen rather than the printed pseudo-code. Nonetheless, they were able to finish Coding Challenges 1-3

before the time was up and Brittany commented that she wished she was more focused so she could have finished all of the Coding Challenges. During her post-study interview, Erica (Female, 15) stated, *"Most of the time, the challenges just came from me, not completely reading the directions correctly so just going back over and reading things"*, which further eludes to the need for a different method of relaying instructions to participants.

5.4.3 Differences in Older & Younger Students :

One of the main differences we noticed during the study was that our youngest users were very engaged and interested – often times more than the older students. In particular,

5.4.4 Usability & User Experience : Through our Youth Studies, we were able to gather necessary data that shows that IoT Maker is a great tool for novices to learn to manipulate electrical hardware. When asked how easy it was to connect their electronics to the iBoard, every student responded favorably (M = 6.4, m = 7, σ = 0.712). Students commented on the lack of training that was required to actually use the iBoard (M = 6.53, m = 7, σ = 0.718), and how the easily connecting electronics helping them quickly bring their physical structures to life (M = 6.33, m = 7, σ = 0.869). Overall, our participants were glad that they didn't have to use a lot of knowledge about electronics to operate the iBoard (M = 6.067, m = 6, σ = 0.772) and thought that using blocks to program their devices was easy (M = 5.93, m = 7, σ = 0.929).

Although the plug-and-play electronics give way for a good user experience, the simulator that is embedded in the IoT Maker website provides a seamless transition between the virtual coding and the physical electronics. In their post-study survey, every student responded favorably when asked if they liked watching the code running on the computer screen (M = 6.33, m = 7, σ = 0.789) and if watching the code on the screen helped them visualize what would happen when they uploaded the code to the iBoard (M = 6.7, m = 7, σ = 0.718). In her pre-study survey, Erica self-reported that she didn't have any experience with Physical Computing Devices, Electronics, and Electro-Mechanical Devices. Yet, during her post-study survey stated, *"Live programming can help with the design of an electro-mechanical device because it allows the person programming to see exactly what works and does not work about their program in real time and work out problems quickly."*

5.4.5 Learning with IoT Maker : Finally, students reported that they learned a lot while using IoT Maker (M = 6.33, m = 6, σ = 0.699) and that they could use IoT Maker to create something more complex (M = 6.33, m = 7, σ = 0.869). This is evident by analyzing their post-study interviews where students like Kayla (Female, 11) and Donald (Male, 18) referred to their ability to quickly understand how to use the system and to build on top of the skills they learned to complete other challenges.

During some of the group sessions, researchers noticed other students who were progressing through the Coding Challenges stopping to help other students when they were stuck. Students like Brittany (Female, 18) explained Coding Challenge 3 to her friends when asked to use her own words to tell us what the functions did.

5.4.6 Summary : Overall, we found the results indicate that IoT Maker is a great tool to help novices learn how to manipulate and navigate electronics, that it reduces the time to complete activities while offering time to improve upon iterations, and significantly reduces the barrier to entry for novice users. From these Youth studies, we found the advantages of the IoT Maker system to be as follows:

- IoT Maker is attractive and usable to young people who are interested in learning how to program physical computing devices.
- The plug-and-play iBoard makes programming physical computing devices easier and more enjoyable

- Live Programming is a great tool to assist novices with creating complex devices that would require more skill than they otherwise would possess; thus, lowering the barrier of entry **and** providing a higher entry point to this level of physical computing.

The results from both the Expert and Youth studies will be expanded upon in the next section.

6 DISCUSSION & FUTURE WORK

As a result of our Expert and Youth Studies, we found that the IoT Maker system succeeded in eliminating unnecessary barriers that to physical computing that typically hinder young people from being engaged with it. We discuss those barrier and some of our findings from the studies below.

6.1 Discussion

During the study, we noticed several key factors with both our Expert and Youth users. First of all, both demographics enjoyed using the system and made note of that in the post-study activities. Many of them attributed this to (1) the block-based interface, which allowed them to build large amounts of code in a short period of time, (2) the plug-and-play electronics, which give way quick and easy component swapping, and (3) the simulator, which gives an accurate depiction of the behavior of the actual electronics. Interestingly, though, the youngest students (age 11 and 12) were noted to be among the most engaged during the study. Initially, these two age groups weren't considered as a part of our target demographic; however, upon meeting with and speaking to the parents, they insisted on having their children be a part of the study. It is also worth noting that these two students self-reported themselves to be more knowledgeable on average in the three areas of expertise than the other students did. Based on our data, it seems as though this system could have a significant impact on younger students who are interested in learning to use Physical Computing Devices, and is something to look for in the future.

6.2 Limitations & Future Work

While we believe IoT Maker can be a launch pad to learn programming and electronics by exercising creativity and problem solving, we also acknowledge that there is room for improvement. In the following sub sections we discuss the limitations of our system, how to overcome those and what are some enhancements that we can implement to IoT Maker to improve its utility.

6.2.1 Software : The present version of our software outputs a .ino file that needs to be compiled alongside the rest of our code before it can be transferred to the iBoard. This process currently is an additional step and not an integral part of the software. In future iterations we plan to add this functionality to the software and allow the users to directly load the compiled code from the software over the air. While our system mainly covers various aspects of programming and electronics, the angle of mechanical design is under-explored. Adding a category of 3D printed connectors to assist in attaching electronics to raw materials (see Shape Structuralizer [12]). Primitive shapes as building blocks and simple machine components like gears and pulleys can add another level of creativity and constructionist thought to our platform in the future [24].

6.2.2 Hardware : IoT Maker hardware is comprised of the iBoard and the peripheral components. While a lot of thought has been put into the design for functionality aspect of the iBoard, as highlighted in the system overview section, a lot can be improved in the hardware to earn more points in design for usability. The iBoard is a bare PCB with all its

components exposed to the user. This not only exposes the iBoard to a potential accidental/environmental damage but can also lead to a functional failure induced by the user.Borrowing insights from prior works [34, 44], we propose to create an enclosure for the iBoard that will limit the interaction of the user only to the interfacing components on the board. Additionally, we see room for improvement with the connectors of the peripherals. IoT Maker being targeted mainly towards youth, we observed that the hardware handling experience gets fussy and clumsy with the small and tight connectors. To overcome this we intend to make some design changes in the connectors (see littleBits [27]). As already mentioned in the Software section, additional repository of mechanical elements on the simulator can be compounded with its physical parts which will add to the hands-on aspect of the system making it more engaging and creative.

6.2.3 Community : Our team plans to host IoT Maker on the web and make it public for all types of users. The platform is intended to allow users to create, collaborate and share projects. Additional enhancement to the system that our team aims towards is adding a developer feature that will enable other IoT developers to add new components and their corresponding animations using a modular piece of code. Thus, IoT Maker lays the foundation towards building a platform that seamlessly enables physical-digital interactions that find applications in a variety of areas.

7 CONCLUSION

This paper presented IoT Maker, an IoT creation platform that utilizes live programming, an integrated block-based programming language, and a lightweight companion app for mobile/desktop that pairs wirelessly with a customized circuit board that we call the iBoard. We presented prior work related to IoT Maker and showcased the workflow of the system. We tested the system with 4 expert users and 15 youth users in our target age range of 11 – 18 years old. Our results show that IoT Maker is a great tool to (1) teach novices how to program physical computing devices to control a wide variety of electronic hardware, (2) make programming physical computing devices easier and more enjoyable, and (3) assist novices with creating complex devices that would require more skill than they otherwise would possess; thus, lowering the barrier of entry **and** providing a higher entry point to this level of physical computing.

A PROFICIENCY SCALES FOR EXPERTS

During our Cognitive Walkthrough study, we adapted the Internet Proficiency Scale developed by Eastin and LaRose [16]. Changes were made to be relevant to the field in which the Expert stated they were proficient, but not to change the meaning of the statement. An example of our adaptation is below, and the rest of the proficiency scales were modified in a similar manner. Each question was evaluated on a 7-point Likert Scale.

- Understanding terms/words relating to developing a physical computing device,
- Describing functions of a physical computing device to others,
- Troubleshooting problems of a physical computing device,
- Explaining why a task is not working on a physical computing device,
- Using advanced skills when developing a physical computing device,
- Using a physical computing device

REFERENCES

[1] [n.d.]. *Blockly.* https://developers.google.com/blockly/
[2] [n.d.]. *Teach, Learn, and Make with Raspberry Pi – Raspberry Pi.* https://www.raspberrypi.org

[3] Jung-Hyun Ahn, Woonhee Sung, Sun Woong Lee, and Jang Hee I. 2017. COBRIX: A Physical Computing Interface for Blind and Visually Impaired Students to Learn Programming. In *Society for Information Technology & Teacher Education International Conference 2017*, Paul Resta and Shaunna Smith (Eds.). Association for the Advancement of Computing in Education (AACE), Austin, TX, United States, 727–733. https://www.learntechlib.org/p/177876

[4] Noah Apthorpe, Arunesh Mathur, Pardis Emami-Naeini, Marshini Chetty, and Nick Feamster. 2019-11-09. You, Me, and IoT: How Internet-Connected Home Devices Affect Interpersonal Relationships. In *Conference Companion Publication of the 2019 on Computer Supported Cooperative Work and Social Computing*. ACM, Austin TX USA, 142–145. https://doi.org/10.1145/3311957.3359489

[5] Arduino. 2020. Arduino - Home. https://www.arduino.cc/

[6] Thomas Ball, Judith Bishop, and Joe Finney. 2018. Multi-platform computing for physical devices via MakeCode and CODAL. In *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*. ACM, Gothenburg Sweden, 552–553. https://doi.org/10.1145/3183440.3183463

[7] Paulo Blikstein. 2013. Gears of our childhood: constructionist toolkits, robotics, and physical computing, past and future. In *Proceedings of the 12th International Conference on Interaction Design and Children - IDC '13*. ACM Press, New York, New York, 173–182. https://doi.org/10.1145/2485760.2485786

[8] BlocklyDuino. 2019-09-10. *BlocklyDuino/BlocklyDuino*. https://github.com/BlocklyDuino/BlocklyDuino original-date: 2012-10-12T13:48:09Z.

[9] BlocklyDuino. 2020. *BlocklyDuino-v2/BlocklyDuino-v2*. https://github.com/BlocklyDuino/BlocklyDuino-v2 original-date: 2020-02-19.

[10] C. Cabo. 2018. Effectiveness of Flowcharting as a Scaffolding Tool to Learn Python. In *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, San Jose, CA, 1–7.

[11] Lautaro Cabrera, John H. Maloney, and David Weintrop. 2019. Programs in the Palm of your Hand: How Live Programming Shapes Children's Interactions with Physical Computing Devices. In *Proceedings of the Interaction Design and Children - IDC '19*. ACM Press, Boise, ID, USA, 227–236. https://doi.org/10.1145/3311927.3323138

[12] Subramanian Chidambaram, Yunbo Zhang, Venkatraghavan Sundararajan, Niklas Elmqvist, and Karthik Ramani. 2019. Shape Structuralizer: Design, Fabrication, and User-driven Iterative Refinement of 3D Mesh Models. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*. ACM Press, Glasgow, Scotland Uk, 1–12. https://doi.org/10.1145/3290605.3300893

[13] Nick Collins, Alex McLEAN, Julian Rohrhuber, and Adrian Ward. 2003-12. Live coding in laptop performance. 8, 3 (2003-12), 321–330. https://doi.org/10.1017/S135577180300030X

[14] Stanley Coren and Joan S Girgus. 1980. Principles of perceptual organization and spatial distortion: the gestalt illusions. *Journal of Experimental Psychology: Human Perception and Performance* 6, 3 (1980), 404.

[15] Microsoft Corporation. 2020. Microsoft MakeCode Computer Science Education. https://www.microsoft.com/en-us/makecode

[16] Matthew S Eastin and Robert LaRose. 2000. Internet self-efficacy and the psychology of the digital divide. *Journal of computer-mediated communication* 6, 1 (2000), JCMC611.

[17] Keith Elliot. 2016. Swift Playground–Interactive Awesomeness. https://medium.com/swift-programming/swift-playgrounds-interactive-awesomeness-2a74143c233.

[18] Espressif. 2020. ESP32 Overview | Espressif Systems. https://www.espressif.com/en/products/socs/esp32/overview

[19] Espressif. 2020. ESP8266 Overview | Espressif Systems. https://www.espressif.com/en/products/socs/esp8266/overview

[20] Terrell Glenn, Ananya Ipsita, Caleb Carithers, Kylie Peppler, and Karthik Ramani. 2020. StoryMakAR: Bringing Stories to Life With An Augmented Reality & Physical Prototyping Toolkit for Youth. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376790

[21] The LEGO Group. 2020. LEGO. https://www.lego.com/en-us.

[22] The LEGO Group. 2020. LEGO Mindstorms. https://www.lego.com/en-us/mindstorms.

[23] Lynne Hall, Colette Hume, and Sarah Tazzyman. 2016. Five Degrees of Happiness: Effective Smiley Face Likert Scales for Evaluating with Children. In *Proceedings of the The 15th International Conference on Interaction Design and Children - IDC '16*. ACM Press, Manchester, United Kingdom, 311–321. https://doi.org/10.1145/2930674.2930719

[24] Idit Harel, Seymour Papert, and Massachusetts Institute of Technology (Eds.). 1991. *Constructionism: research reports and essays, 1985-1990*. Ablex Pub. Corp, Norwood, N.J.

[25] Juraj Kubelka, Romain Robbes, and Alexandre Bergel. 2018-05-27. The road to live programming: insights from the practice. In *Proceedings of the 40th International Conference on Software Engineering*. ACM, Gothenburg Sweden, 1090–1101. https://doi.org/10.1145/3180155.3180200

[26] Lifelong Kindergarten Group MIT Media Lab. 2020. Scratch micro:bit. https://www.scratch.mit.edu/microbit

[27] littleBits. 2020. *littleBits | Electronic Building Blocks for the 21st Century.* https://littlebits.com/?gclid=CjwKCAjwk93rBRBLEiwAcMapUXF9vGRXdv4AFdhlZspY-J9djVjr3eNKHPgadZn5CzT7xsPS7eHTchoChgwQAvD_BwE

[28] F. Liu, G. Tang, Y. Li, Z. Cai, X. Zhang, and T. Zhou. 2019. A Survey on Edge Computing Systems and Tools. *Proc. IEEE* 107, 8 (2019), 1537–1562.

[29] Amon Millner and Edward Baafi. 2011. Modkit: Blending and Extending Approachable Platforms for Creating Computer Programs and Interactive Objects. In *Proceedings of the 10th International Conference on Interaction Design and Children* (Ann Arbor, Michigan) *(IDC '11)*. Association for Computing Machinery, New York, NY, USA, 250–253. https://doi.org/10.1145/1999030.1999074

[30] Click Nilson. 2007. Live coding practice. In *Proceedings of the 7th international conference on New interfaces for musical expression - NIME '07*. ACM Press, New York, New York, 112. https://doi.org/10.1145/1279740.1279760

[31] M. Resnick, F. Martin, R. Sargent, and B. Silverman. 1996. Programmable Bricks: Toys to think with. *IBM Systems Journal* 35, 3.4 (1996), 443–452.
[32] Mitchel Resnick and Brian Silverman. 2005. Some reflections on designing construction kits for kids. In *Proceeding of the 2005 conference on Interaction design and children - IDC '05*. ACM Press, Boulder, Colorado, 117–122. https://doi.org/10.1145/1109540.1109556
[33] FIRST Robotics. 2020. FIRST Tech Challenge. https://www.firstinspires.org/robotics/ftc.
[34] Jasjeet Singh Seehra, Ansh Verma, Kylie Peppler, and Karthik Ramani. 2015. HandiMate: Create and Animate using Everyday Objects as Material. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction - TEI '14*. ACM Press, Stanford, California, USA, 117–124. https://doi.org/10.1145/2677199.2680570
[35] Gary A Stewardson, Trevor P Robinson, Joseph S Furse, and Michael L Pate. 2019. Investigating the relationship between VEX robotics and student self-efficacy: an initial look. *International Journal of Technology and Design Education* 29, 4 (2019), 877–896.
[36] Arduino Team. 2020. Get ready to Explore IoT with Arduino Education. https://blog.arduino.cc/2020/09/15/get-ready-to-explore-iot-with-arduino-education/
[37] The Good Chrome Development Team. 2020. Google Chrome Development Tools. https://developers.google.com/web/tools/chrome-devtools/.
[38] The React Development Team. 2017. React: A Javascript Livrary for Building User interfaces. https://facebook.github.io/react/.
[39] Unity. 2020. Unity Real-Time Development Platform | 3D, 2D VR & AR Engine. https://unity.com/
[40] VEX. 2020. VEX Robotics. https://www.vexrobotics.com/.
[41] VEX. 2020. VEX Robotics Competition. https://www.vex.com/competition.
[42] Adrian Ward, Julian Rohrhuber, Fredrik Olofsson, Alex McLean, Dave Griffiths, Nick Collins, and Amy Alexander. 2004. Live algorithm programming and a temporary organisation for its promotion. In *Proceedings of the README Software Art Conference*, Vol. 289. 290.
[43] Benjamin Wohl, Barry Porter, and Sarah Clinch. 2015. Teaching Computer Science to 5-7 year-olds: An initial study with Scratch, Cubelets and unplugged computing. In *proceedings of the workshop in primary and secondary computing education*. 55–60.
[44] Sang Ho Yoon, Ansh Verma, Kylie Peppler, and Karthik Ramani. 2015. HandiMate: exploring a modular robotics kit for animating crafted toys. In *Proceedings of the 14th International Conference on Interaction Design and Children - IDC '15*. ACM Press, Boston, Massachusetts, 11–20. https://doi.org/10.1145/2771839.2771841
[45] Junling Zhang and Jing Liu. 2018. Construction of Scaffolding Instruction Mode for Mblock for Arduino Maker Course Based on Design Thinking. In *Proceedings of the 2nd International Conference on Computer Science and Application Engineering* (Hohhot, China) *(CSAE '18)*. Association for Computing Machinery, New York, NY, USA, Article 147, 6 pages. https://doi.org/10.1145/3207677.3278031