Primary School Students Programming with Real-Time Environmental Sensor Data

Hussel Suriyaarachchi
Augmented Human Lab
The University of Auckland
Auckland, New Zealand
hussel@ahlab.org

Paul Denny
School of Computer Science
The University of Auckland
Auckland, New Zealand
paul@cs.auckland.ac.nz

Juan Pablo Forero Cortés
Augmented Human Lab
The University of Auckland
Auckland, New Zealand
juan@ahlab.org

Chamod Weerasinghe
Augmented Human Lab
The University of Auckland
Auckland, New Zealand
chamod@ahlab.org

Suranga Nanayakkara
Augmented Human Lab
The University of Auckland
Auckland, New Zealand
suranga@ahlab.org

Figure 1: A) Our plug-and-play sensor toolkit tailored for use by young children; B) Seamless access to exploring physical phenomena and integrating real-time data with Scratch; C) Students engaging in free exploration with the toolkit.

ABSTRACT

Programming is now introduced as an essential skill at a very young age, often through block-based programming environments. Interaction with programs created using such platforms typically occurs through the use of keyboard and mouse. Incorporating environmental data as real-time input to programs through physical computing devices can deliver an engaging programming experience that fosters creativity. However, using such devices with block-based environments often requires significant technical configuration, which can be challenging in the classroom, especially for very young learners. In this paper, we describe our design, development and use of a sensor toolkit and a companion Scratch extension that is very easy for young students to use. The toolkit allows students to focus on constructive exploration while providing teachers with an easy way to incorporate sensor-driven programming lessons in a time-constrained classroom setting. We deployed our toolkit in a primary school classroom with 19 students aged between 7 and 9, and two teachers who were introducing sensor-driven programming for the first time. We report how these young students used our toolkit to control actions in their programs, their post-session views of programming as an interesting and creative task, and the teachers' perspectives of using the toolkit with young learners.

ACE '22, February 14–18, 2022, Virtual Event, Australia

ACM ISBN 978-1-4503-9643-1/22/02…$15.00
https://doi.org/10.1145/3511861.3511871

CCS CONCEPTS

• **Social and professional topics** → *Computing education*; • **Hardware** → *Sensor devices and platforms.*

KEYWORDS

Scratch, Sensors, Kiwrious, Real-time, Creativity, Block-based programming, Physical Computing, Computational Thinking, Computer Science Education

ACM Reference Format:
Hussel Suriyaarachchi, Paul Denny, Juan Pablo Forero Cortés, Chamod Weerasinghe, and Suranga Nanayakkara. 2022. Primary School Students Programming with Real-Time Environmental Sensor Data. In *Australasian Computing Education Conference (ACE '22), February 14–18, 2022, Virtual Event, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511861.3511871

1 INTRODUCTION

Computing is now commonly taught at the primary school level, with many countries developing and adopting national curricula [8, 14, 15, 43]. As an increasing number of children learn to code, there has been a surge of interest in developing approaches and tools that are suitable and effective for students of various ages. Indeed, a recent survey of computing educators by Denny et al. revealed a strong desire for finding tasks that are both engaging and effective in computing classrooms [12].

Block-based programming environments, like Scratch [39] and Blockly [18], are very popular at the primary school level, as they provide helpful visual cues to students when constructing programs [52]. Scratch is designed to support learners between the ages of 8 and 16 and its simplified counterpart, ScratchJr, targets learners between 5 and 7[1]. Statistics show that the age distribution for students new to Scratch peaks at 12, with relatively few students between the ages of 7 and 9[2].

A wide range of physical computing devices have also made their way into classrooms to provide an engaging experience for young learners. Popular examples include the BBC micro:bit [45] and Lego Mindstorms [27], both of which provide options for drag-and-drop block-based coding. Scratch also provides an extension framework, allowing third-party developers to design blocks that can be used to interface with external hardware. Sentance et al. describe the benefits of physical computing in the classroom, highlighting natural opportunities for collaboration and for unleashing student creativity [44]. Indeed, Geldreich et al. found that 'creating' was the most common theme expressed by primary school students when asked to talk about programming [19], a phenomenon the authors attribute to the recent growth of the maker movement [42].

However, especially for very young learners, programming physical computing devices using block-based environments can present practical challenges. From the teachers' perspective, Tyrén et al. [49] highlight a number of concerns for classroom deployment, including the need for some devices to be flashed prior to use, and the difficulty in asking a classroom of young students to follow technical configuration instructions. Such concerns are particularly problematic when teaching sessions are limited in time. Flannery et al. also report that young children using Scratch often struggle to enter sensible parameter values into variable blocks [17], which can be problematic when specific values are needed to correctly control hardware. Physical computing tools need to be easy for young students to use, so that they can focus on constructive exploration and creativity, and easy for teachers to deploy and support in time-constrained classrooms.

In this practitioner paper, we describe our design, development and deployment of a sensor toolkit that provides real-time environmental data as input to Scratch programs. We have designed the toolkit to be very easy to use, essentially plug-and-play, to simplify classroom deployment. To promote student creativity, we support a range of sensors that measure different physical phenomena, including humidity, surface temperature, total volatile organic compounds, visible light, and conductance. We see this connection to the environment as critical to promoting student interest. As Sheehan notes in his recommendations for the design of programming environments for children [46], the important thing is to provide an environment that has points of contact with children's current understanding. While the concepts of programming may be new and unfamiliar to novices, interaction with the physical environment through sensors draws on students' everyday experiences.

We describe our development of the extension to integrate the sensors with Scratch, and our design of the custom Scratch blocks to be suitable for use by young students. We report our experience deploying the toolkit for the first time in a short 60-minute class session, alongside the two teachers who regularly teach the class, with 19 students between the ages of 7 and 9 working in pairs. We explore the following questions:

(1) Which sensors do students prefer, and how do they use the sensor data to control the action in their programs?
(2) Upon completing the session, do students view programming as interesting, and do they see it as a creative endeavour?
(3) What opportunities do teachers see in using the toolkit for programming instruction for young children?

Given the importance of teacher acceptance for successful adoption of technology in the classroom, we explore the perceptions of teachers with no prior experience using the toolkit.

2 RELATED WORK

Recent advances in low-cost hardware and the development of suitable programming environments for children have led to greater use of physical computing in primary school classrooms [23]. In their seminal work on constructionism, Papert and Harel argue that learning happens most readily when students are consciously engaged in constructing a 'public entity', which is real and visible to others [36]. Physical computing lends itself to such constructionist learning, as it involves the combination of software and hardware to build interactive systems that sense and respond to the real world [5]. In addition to the development of technical skills, tangible and physical computing environments can have very positive effects on student motivation, collaboration and active learning, as students work together in visible ways [24, 32, 44, 48].

Although a variety of hardware-based tools, including robots, programmable toys and microcontrollers, have been used in computing classrooms for decades, age-appropriate tools are needed for very young students. Younger children can find programming difficult when the instructions are too low-level and thus programs require significant decomposition, or when the instructions lack visible outcomes and immediate feedback [17]. In their guidelines for designing construction kits for children, Resnick and Silverman advocate for a 'low floor and wide walls', indicating that it should be very easy for students to get started, while providing flexibility for creating sophisticated projects [40]. Similar work by Sheehan presents several recommendations for the design of programming environments for children aged between 6 and 10, including that the environment should be usable in a group situation, to facilitate collaboration, and provide high-level instructions that map to the changes students want to see or control in their programs [46].

[1] https://scratch.mit.edu/parents/
[2] https://scratch.mit.edu/statistics/

2.1 Sensors as input devices

A common method of interaction with running programs is direct input via the keyboard or mouse. Block-based environments suitable for young learners, like Scratch [39], provide 'Events' blocks that respond to such input by triggering scripts to run when a particular key is pressed or if the mouse clicks on a sprite. Although such user-initiated inputs are intuitive for a student who is already using the mouse and keyboard, programs are limited to responding only to direct user actions. Sensors that measure physical phenomena can provide real-time streams of data that serve as dynamic input to programs, and can be particularly motivating for children. For example, Wang et al. designed a tangible programming tool for children, called T-Maze, that involves the construction and solution of simple mazes [11]. The tool incorporates temperature and light sensors that provide input to a simulation program, and students interact with these sensors to solve the mazes. In an evaluation with children aged between 5 and 9, almost all of them showed particular interest in the sensors, and "once they saw the sensors, they could not help trying to trigger them" [50]. Physical sensors can also use design aesthetics to convey their purpose in a meaningful way for young students. For example, the KIBO robotics kit uses an ear-shaped design for a sound sensor, and an eye-shaped design for a light sensor, allowing children to leverage their existing symbolic knowledge when interacting with the physical world [3].

Microcontroller kits are a popular solution for enabling the connectivity and use of generic sensors as program inputs. One such example is the Arduino [33], which supports a variety of sensors through onboard digital pin attachments. However, students often encounter difficulties when using microcontroller-based tools as they require knowledge of wiring, pin configuration and serial communication. Prerequisites and barriers such as these exist with the Arduino and have been explored in prior work by Booth and Stumpf [6], and Heo [22]. In 2015, the BBC launched the micro:bit [1, 44], a microcontroller kit tailored for educational use. With built-in sensors, it avoids the complexities of Arduino-like devices and offers superior ease of use when programming. However, there remain some documented pitfalls for classroom deployment of the micro:bit, such as its need to be flashed prior to use and the difficulty of guiding young students through certain procedures such as Bluetooth pairing [49]. With the rapid growth of computing as a subject at the primary school level, toolkits must be reliable, simple and quick to deploy in time-constrained classrooms, especially for teachers who may lack confidence with digital technology [26].

2.2 Block-based programming and creativity

Block-based programming environments provide clear visual cues for connecting blocks to help students construct programs [52]. Across a wide variety of approaches, there is substantial evidence for the value of such environments for developing computational thinking skills in young learners [10, 21, 47, 53]. Deschryver and Yadav further argue that fostering computational thinking skills can help to promote creative thinking [13]. In general, block-based programming is particularly suitable for young learners as the complexity of syntax is removed. This allows students to focus on being creative while avoiding the common difficulties relating to text-based error messages [2]. Recently, frame-based programming editors have been proposed to ease the transition, as students mature, from block to text-based programming [7, 38].

Scratch is currently the most widely used block-based programming platform [51], catering primarily for children between the ages of 8 and 16. ScratchJr is a simplified version of Scratch, designed to support younger learners from ages 5 to 7[3]. In describing the design of ScratchJr, Flannery et al. outline several challenges that young children encounter when using Scratch [17]. These include decomposing programs into low-level instructions that lack visible feedback, and determining appropriate parameter values for blocks that allow arbitrarily large inputs. ScratchJr thus employs high-level instructions, and input-constrained blocks, which students typically use to animate the movement of sprites or characters on the screen. Bers argues that such environments are ideal for supporting new pedagogies that teach computational concepts, while supporting personal expression and fostering creativity [4]. 'Motion' blocks, which are used to control the movement of sprites and characters for creative story-telling, tend to be the most commonly used blocks across all age ranges using ScratchJr [37].

Adapting block-based programming environments to accommodate hardware input, such as environmental sensor data, usually involves installing offline software components to support the new functionality. For example, integrating supported devices into Scratch requires initial set up via utility software[4]. In addition, appropriate blocks must be available within the environment for reading sensor data. Scratch does not provide such blocks by default, requiring the development of custom environments that are modifications of Scratch (e.g. [16]), or the use of the Scratch extension framework which allows custom blocks to be loaded into the editor alongside standard Scratch blocks. This latter approach is the one we adopt in the current work.

In 1972, Seymour Papert presented a grand vision for the use of technology in education in which students were empowered to create exciting projects by "providing them access to computers with a suitably clear and intelligible programming language and with peripheral devices capable of producing on-line real-time action" [35]. In this work, we explore the artefacts that students create, and their views on creativity in programming, when using a sensor-based physical toolkit for measuring real-time environmental phenomena within a block-based programming environment.

3 SENSOR DESIGN & DEVELOPMENT

We developed a toolkit consisting of five sensor units that plug directly into the USB port and measure various aspects of the environment, including humidity, surface temperature, total volatile organic compounds (tVOC), visible light, and conductance. Our choice of sensors was informed by prior work that has explored the use of sensor technology in science classrooms for teaching scientific inquiry skills and computational thinking [9, 20]. Since such phenomena occur universally and are easily manipulatable, they are ideal for use within the confines of a classroom. In this section, we outline the technical specifications of the hardware and describe our aesthetic design choices for the sensors.

[3] https://www.scratchjr.org/

[4] https://en.scratch-wiki.info/wiki/Scratch_Link

Figure 2: A) Temperature; B) Air Quality; C) Light, D) Conductance, E) Humidity. 1) Micro-Controller Unit, 2) AR Marker, 3) Analog Module Conductivity Sensor, 4) SHTC3 Humidity Sensor, 5) MRT311 Temperature Sensor, 6) SGP30-2.5K VOC Sensor, and 7) SI1133-AA00-GMR Light Sensor

3.1 Enabling plug-and-play support

Considering the plug-and-play requirements, we designed the sensors to be powered directly from a standard Micro-USB port @5V. The sensors implement a USB-CDC(ACM) (Communications Device Class, Abstract Control Model) interface to emulate a serial port on the host computer. The Web Serial interface provides direct access to this port through the host computer's web browser, achieving a device-agnostic approach for communicating with our sensors. The communication is straightforward; once initialised the sensor will start transmitting data packets, each 26 bytes in size. The packet consists of a header, an identifier byte denoting the sensor type, 2 to 16 bytes of data depending on the sensor type, and a footer. Additionally, each sensor exposes two more USB interfaces, a Device Firmware Update interface for updates in the field, and a WebUSB interface. The WebUSB capability enables the web browser to automatically open a pre-defined URL stored in the sensor upon its connection. Thus, users experience a seamless transition from connecting the sensors to using them, as visualised in Figure 3.

3.2 Hardware development of the sensors

We tried to maximise the intuitiveness and usability of the system by creating five specialised sensor modules (Figure 2). The electronics of each sensor type were designed to measure one physical property. Every sensor incorporates an ARM Cortex-M0+ Microcontroller Unit (MCU) @48MHz with an integrated USB 2.0 Interface (i.e., ATSAMD11D14A-MUT). In the event of an electrostatic discharge or short-circuit, both the sensor and the user's device are protected by transient voltage suppression diodes (i.e., USBLC6-2SC6) at the USB data lines and a ressetable Fuse @8V/500mA (i.e., 0ZCJ0050FF2G) tied to the power line. All the sensors are compliant with the AS/NZS CISPR-32:2015 regional standard for electromagnetic compatibility. The circuits of the *Humidity*, *Conductance*, *Temperature* and *Light* sensors are powered through a linear regulator @3V3/300mA (i.e., MIC5504-3.3YM5-TR) tied to the USB power pin @5V. The *VOC* sensor uses a linear regulator @3V1/300mA (i.e., TLV70231DBVR). The *Light* sensor incorporates an ambient optical sensor @450/525/625nm (i.e., SI1133-AA00-GMR). The *VOC* sensor features a digital multi-pixel gas sensor (i.e., SGP30-2.5K) with 0.2% accuracy in both the H_2 and Ethanol signals. The *Temperature* sensor offers a contact-less measurement $[-30\,to\,100C^{\circ}]@0.5C^{\circ}$ accuracy through a thermopile infrared sensor (i.e., MRT311). The *Humidity* sensor readings are enabled by a digital relative humidity(RH) sensor @2% RH accuracy (i.e., SHTC3). Finally, the *Conductance* sensor incorporates a custom analog circuit consisting of a voltage divider with switchable resistors, reverse and over-voltage protection, and a RC low pass-filter.

Figure 3: A standard Micro-USB cable makes for a simple connection between sensor and host device. Upon connection, the sensor type is automatically detected and data readings become available in the web browser.

3.3 Form factor design

Each sensor has a unique shape reminiscent of the natural phenomena being measured (e.g., the Light sensor is shaped as a tiny sun, and the Temperature sensor is shaped as a thermometer). This aesthetic design helps to convey the purpose of each sensor to the user, improving accessibility for young children [3]. Additionally, the unique shape in combination with the unique AR markers printed on each sensor (e.g. see Figure 2:2) provide an opportunity in future work to develop mixed-reality applications. Finally, the back of each sensor is populated with a glossy and colourful sticker that helps in identifying the sensor and contributes to a consistent look and feel.

4 SOFTWARE IMPLEMENTATION

The technical architecture of the Scratch platform consists of a Node.js backend called the Scratch Virtual Machine (VM) and a React-based Graphical User Interface (GUI). The Scratch VM is responsible for defining the operations of all blocks and handles the execution of programs. The interface required to create such programs is provided by the Scratch GUI that enables visualising tasks performed in the VM.

4.1 Scratch Extension

Scratch extensions[5] are a standard part of the infrastructure provided by Scratch to accommodate new features and hardware into the platform. Through extensions, custom blocks can be instantly loaded into the Scratch editor and then used alongside standard Scratch blocks to create programs. We developed a Scratch extension by modifying the Scratch VM to include a new Javascript file specifying our sensor blocks and their behaviour.

With the latest version of Scratch being an entirely web-based application, classrooms can easily access the programming environment using a browser without the need for any further installations. However, custom Scratch extensions are not permitted to be directly included on the official Scratch website. We achieved a workaround to this obstacle by deploying a clone of the Scratch platform containing our extension. Similar to the official website, this clone is accessible through the browser at our custom URL[6] and identical in functionality.

4.2 Establishing Sensor Connectivity

Using the Web Serial API offered in Chromium-based browsers, we implemented the two blocks illustrated in Figure 4A that enable the Scratch platform to read data from our sensor toolkit. The *Connect* block is used to discover any connected sensors, obtain user permissions to connect with them and establish a communication channel. We included a USB device filter in the JavaScript code for this block to avoid recognising any devices that do not belong to our sensor kit. As sudden sensor disconnections may lead to unsafe program execution in the Scratch VM, we included internal USB event listeners to compensate for such scenarios.

Once connected, the sensors commence their transmission protocol described in Section 3.1. The task of initiating data retrieval and processing this raw byte data are handled asynchronously through the *Read Forever* block. This block runs a threaded process that continuously performs appropriate byte manipulations to update the internal variables representing the sensor values. Thus, both the *Connect* and *Read Forever* blocks need to be used in unison to set up and maintain sensor connection and communications.

4.3 Monitoring Environmental Data

Figure 4C shows the collection of five Scratch reporter blocks, each corresponding to a particular sensor. These blocks directly mirror the state of the internal variables containing the respective live stream of sensor data. Since reporter blocks are Scratch's equivalent of traditional variables, they can be placed within appropriate Scratch blocks to perform various operations. However, especially

[5]https://en.scratch-wiki.info/wiki/Scratch_Extension
[6]https://play.kiwrious.com/

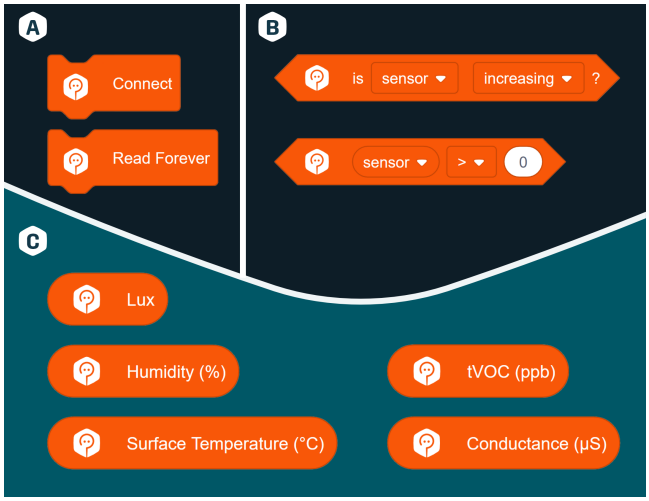

Figure 4: Scratch blocks which provide the capability to A) Establish sensor connectivity; B) Monitor environmental data; C) React to changes in the real world.

for younger children, these blocks offer a helpful way to simply monitor live sensor data and enable quick experimentation with sensors before building programs.

4.4 Reacting to Changes in the World

We created two variations of Scratch boolean blocks, as seen in Figure 4B, that trigger responses to changes in physical conditions.

Increasing/Decreasing Block. Young children often find defining numerical thresholds challenging and unappealing when programming [17, 34]. We addressed this concern by implementing a block triggered by an increase or decrease in sensor data. Students simply specify their desired trigger condition (increase/decrease) by selecting an appropriate option from the dropdown menu within the block. The comparison of numerical values needed to indicate this directional change were performed internally through JavaScript and thus hidden from the user. We adjust the precision of sensor values prior to performing this comparison to account for the small natural fluctuations of sensor readings in realistic environmental conditions. Thus, relying on the gradient of change in sensor data improves a student's programming experience as it fosters a more intuitive association with the natural world.

Raw Sensor Value Comparison Block. Creating a trigger based on the numerical value of a sensor measurement usually involves dragging and dropping a sensor reporter block inside a Scratch operator block. We consolidated the operations of these blocks into a single boolean block. In addition to improving usability by reducing the number of user tasks [31], the comparison of values was handled differently to a regular operator block. As sensor reporter blocks may return strings such as 'Not Connected', exceptions were needed to prevent incorrect boolean evaluations. Compared to the 'increasing/decreasing' block, this block is slightly more complex to use as it requires entry of a suitable threshold value, yet it offers greater control over sensor data that could lead to a broader exploration of environmental changes when programming (i.e. the 'wide walls' advocated by Resnick and Silverman [40]).

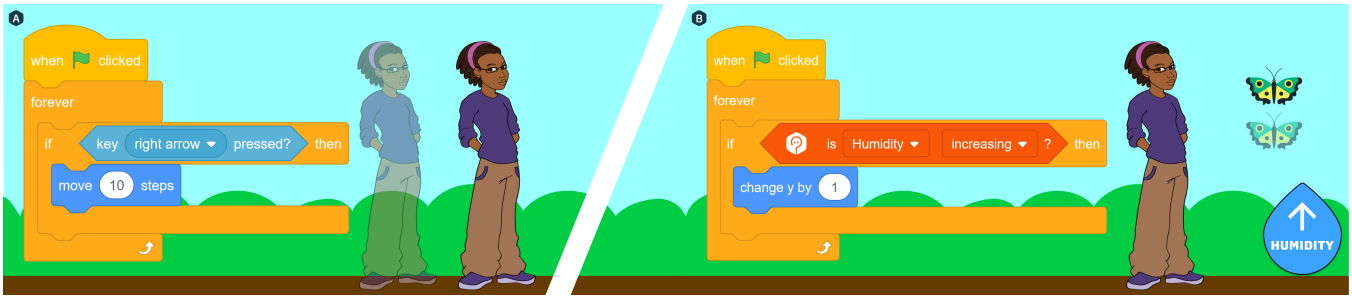

Figure 5: A) Starter project for providing the fundamentals of Scratch programming. The sprite moves to the right when the right arrow key is pressed; B) Sensor-controlled sprite where the butterfly moves upward when humidity increases.

5 DEPLOYMENT

5.1 Participants

We evaluated our tool in a classroom lesson conducted as part of an afterschool STEM initiative aimed at young children. The session was led by two teachers, both of whom were highly proficient with Scratch but having no prior experience using our tool in the classroom. Nineteen students (10 boys, 9 girls) aged between 7 to 9 years from various primary schools in the country attended the class.

All students were enrolled in a program called "Tinker Club"[7] which primarily focused on non-coding based learning. At the time of the study, students were three weeks (an equivalent of 3 hours of learning) into their curriculum. Most of their activities involved hands-on tasks for understanding practical concepts around physics, chemistry and biology, and their real-world application, e.g., how bridges overcome gravity. Despite the absence of Scratch in this program, all students stated they had previously used Scratch elsewhere (at home or school). Six students claimed to be regular users, while the others rarely used Scratch. In addition to their minimal experience with Scratch, this was also the students' first time programming with environmental data.

5.2 Methodology

The study was performed in one of the weekly in-person classes hosted by the education program provider. The entire duration of the class (60 minutes) was utilised, with the study serving as the lesson taught that week. It consisted of a 10-minute brainstorming activity, 10-minute introduction, 35-minute open-ended Scratch coding activity and 5-minute feedback session.

5.2.1 Brainstorming Activity (10 mins). The focus of this segment was to leverage on the everyday experiences of students to inspire the notion of working with environmental data. Drawing on the program's theme of exposing students to "how stuff works", the teachers opened a conversation to discuss examples of sensors and where students may have seen them in use. Following this, the teachers introduced the sensor toolkit to the class and explained the various aspects of the physical world that they could measure. Students were encouraged to think aloud and draw on whiteboards to ideate applications of sensing these phenomena.

5.2.2 Introduction to Programming with Sensors (10 mins). As illustrated in Figure 5, our introduction revolved around a starter project which provided students with a fundamental understanding of Scratch and programming with sensors. Basic Scratch blocks and their functions were explained to the class while creating the sprite shown in Figure 5A. Together, these blocks make the sprite move five steps to the right on pressing the keyboard's right arrow key.

We then proceeded to demonstrate how students could explore using sensors to control the behaviour of their programs. Sensor blocks along with regular Scratch blocks were combined to introduce a new sprite (butterfly) as seen in Figure 5B. When executed, the sprite will move upwards if an increase in the humidity level is detected (e.g. by blowing on the humidity sensor). Thus, students were shown how easy it was to build interactive programs in Scratch that respond to changes in the real world.

5.2.3 Open-Ended Coding Activity (35 mins). The introduction was followed by a 35 minute free-exploration period, guided by the Use-Modify-Create pedagogy [30, 41]. Prior work by Iskrenovic-Momcilovic et al. [25] recommended adopting a pair-programming approach for younger primary school students involved in coding Scratch. Thus, students were grouped into pairs before commencing the activity. We ensured that the six students who were relatively confident in using Scratch were placed in different pairs. One of these students, however, requested to work on their own.

All students had complete freedom in deciding what projects they created and what sensors they used. We also indicated that the example project was available for them to download and modify if desired. Both teachers were present to offer active support during the session. Additionally, students were also provided with helpful guides on programming with Scratch and using the sensors.

5.2.4 Data Collection (5 mins). At the end of the coding activity, groups were asked to submit their programs to a Padlet board[8]. We administered a questionnaire via Google Forms to explore how students perceived programming following our activity. All questionnaire items were adopted from the survey instrument devised by Kong et al. [28], which specifically aims to evaluate the attitudes of primary school students towards programming. Questions pertaining to the interest and creativity subscales of the survey were selected for inclusion. Students responded to these questions on a 5-point Likert scale. The questions, and a summary of the data collected, are shown in Table 2. Furthermore, both teachers were asked to submit a short reflection of their classroom experience by the end of the day.

[7] http://www.mycreatelab.com/

[8] https://padlet.com/KiwriousPlay/49fsmsyic2yhrfr1

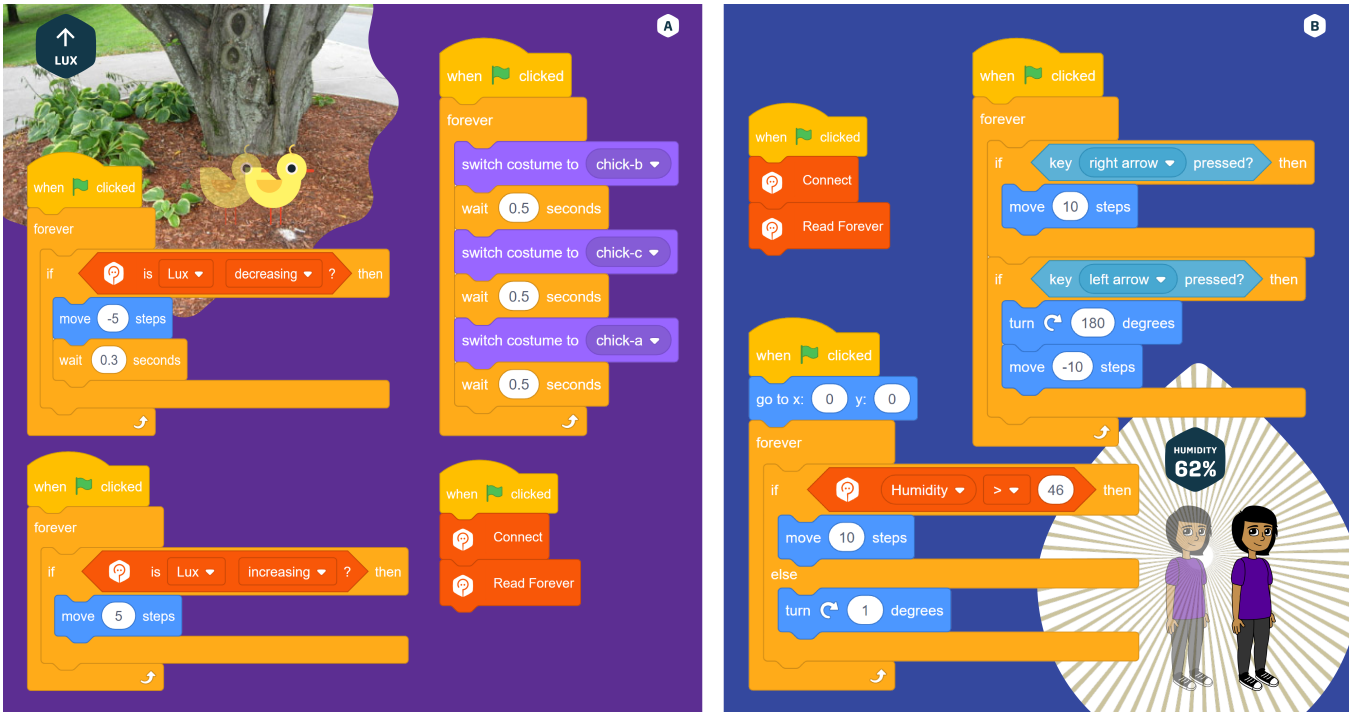

Figure 6: A) Program created by a student pair using the light sensor, where the sprite moves to the left or right based on an increase or decrease of the detected lux value; B) Program created by a student pair using the humidity sensor, where the humidity level influences how the character is controlled through the keyboard.

6 FINDINGS

This deployment represents the first use of our toolkit in a real classroom. Our evaluation is guided by the three questions listed in Section 1. For the first question, we are interested in knowing which of the sensors students chose to use when writing their programs and how they made use of the sensor data. For the second question, we analyse the questionnaire data collected at the end of the session to understand how students view programming as an interesting and creative activity. Finally, for the third question, we reflect on the teachers' feedback regarding the usability of the toolkit in the primary school classroom.

We begin our analysis by decomposing all submitted programs into their constituent blocks. Analysing the usage of individual blocks in Scratch programs is a common strategy, and has been used recently to measure creativity in student projects [29]. Table 1 illustrates the block elements used across all nine submitted projects (P1 – P9).

6.1 Project Analysis

Exploring different physical phenomena was of interest to the students, with all five of our sensor types being used across the nine projects submitted. As seen in Table 1, the Light, Humidity and Temperature sensors were used more frequently among projects than the Conductivity and VOC sensors. Since changes in conductance and tVOC may be challenging to induce without additional material or knowledge, students preferred to work with sensors to which they could better relate. Manipulating light, humidity and temperature is relatively straightforward (e.g. by covering, holding and blowing on the corresponding sensors).

When allowed free exploration, students explored the information provided by their sensor of choice to provide an initial context that guided how they constructed their Scratch programs. All projects employed a feedback mechanism-based approach where changes in the environment caused subsequent action in the program. As seen in the example programs shown in Figures 6A-B, sensor data was primarily purposed for controlling sprites in response to a change in the environment. For instance, Figure 6B illustrates a student pair leveraging the humidity of their breath to move and rotate a sprite by blowing on the sensor, or waving the sensor in the air.

To handle incoming sensor data, students needed to use appropriate programming constructs to achieve their desired functionality. We observed students following a standard procedure of using an event listener to trigger the sequential execution of a sensor-based conditional logic (selection) and action inside an infinite loop (iteration). Table 1 indicates the recurring use of the *Green Flag Clicked*, *Forever*, *If*, and *If-Else* Scratch blocks as they were essential for enabling this mechanism. The program shown in Figure 6B illustrates the handling of sensor data using the raw sensor value comparison block. The conditional argument is constructed by providing a sensible humidity parameter (> 46%) in their sensor logic. Only three student pairs attempted sensor logic this way (see Table 1). In contrast, Figure 6A shows the more popular method for working with sensor data. In this case, the project used the Light sensor, and an increase or decrease in lux moved the sprite to the left or right. The popularity of our scaffolded 'increasing/decreasing' blocks aligns with guidelines described by Flannery et al. for providing young children with input-constrained blocks [17].

Table 1: The different block elements used by all nine student pairs (P1 – P9) when creating their Scratch projects.

	Block	P1	P2	P3	P4	P5	P6	P7	P8	P9
Constructs	Green Flag									
	Forever									
	Single If									
	If Else									
Sensor Type	Humidity									
	Lux									
	Surface Temperature									
	Conductance									
	tVOC									
Sensor Trigger	Raw Value Comparison									
	Increasing/Decreasing									
Visual Changes	Move									
	Go to x, y									
	Change y by									
	Turn									
	Change Size									
	Say									
	Say for									
	Switch Costume									
	Next Costume									
User Interactivity	Key <x> Pressed									
Synchronisation	Wait									

Blocks evoking various visual changes for sprites were very common in the student programs. Table 1 shows that students often related a change in sensor data to the movement of sprites on the screen. *Move*, *Turn*, and *Go To* were popular examples of such blocks, and this mirrors the findings of Portelance et al. where 'Motion' blocks were the most common type of blocks used by young learners using ScratchJr [37]. In addition to movement, changes to the appearance of sprites through costume switches were also used by some students. Although rare, advanced blocks such as 'wait' blocks for introducing delays in programs and 'sensing' blocks that listen to keypresses were also present. Thus, while all students successfully used sensor data in their programs, the data was used in a variety of ways.

6.2 Student Perceptions

The responses to the 7 items on the questionnaire are shown in Table 2. In general, the Likert scores show high agreement with the statements across both scales. With respect to interest, students indicated strongly that they were curious about programming and found it fun and interesting. There was a noticeably weaker sentiment regarding attraction to programming. Although the items on the questionnaire were taken from a survey designed for primary school students and validated by experts [28], as the students were very young, their lack of exposure to 'programming activities' may be the reason for this indifference.

With respect to creativity, responses strongly indicated that students viewed programmers as creative people and recognised the importance of being creative when programming. Although the session was limited in time, the ease of using the sensors enabled students to create a wide variety of programs with respect to the sensors chosen and the programming blocks used. Across all of the groups, all of the different sensors were used in some way to form novel programs. The high visibility of the projects in the classroom setting may have positively impacted students' perceptions of creativity.

6.3 Teacher Perceptions

Both teachers who attended the session highlighted the simplicity and robustness of our toolkit for enabling programming with sensors in Scratch. They frequently mentioned terms such as "stable" and "easy to follow" when describing the classroom deployment. Our design considerations to realise simple but powerful sensor blocks were emphasised in one teachers' account of events: *"Programming has been made easy with all the relevant coding blocks in one place. Thus, students can start quickly, a much-needed factor"*. These comments, and our observations of the deployment, suggest that our toolkit is easy to use in practice and avoids some of the common technical pitfalls relating to the classroom deployment of physical computing devices [49].

Teachers saw the opportunity to draw connections to the real world as an exciting way to evoke enthusiasm and creative thinking for students in the programming task. At the start of the session, they noticed that *"the kids lit up, on discovering the possibility to create hardware-based expressions with Scratch"*. The impact of using sensor data to support usage of Scratch blocks was acknowledged in a teachers' observation: *"The kids were very excited to be able*

Table 2: Questionnaire responses to the survey investigating interest in programming and creative self-efficacy. Responses on a 5-point Likert scale: *SD:* Strongly disagree, *D:* Disagree, *N:* Neither agree nor disagree, *A:* Agree, *SA:* Strongly agree.

Question		SD	D	N	A	SA
Interest in programming	Programming is interesting	0%	0%	0%	30%	70%
	I am curious about the content of programming	0%	0%	0%	40%	60%
	I think the content of programming is fun	0%	0%	0%	0%	100%
	I am very attracted to computer programming activities	0%	10%	40%	20%	30%
Creativity	It is important to be creative when you are programming	0%	0%	0%	0%	100%
	I would like to design things using programming	0%	0%	0%	50%	50%
	Computer programmers are creative	0%	10%	0%	30%	60%

to modify the parameters (of blocks) with things that are in their immediate environment". Referring to the student project illustrated in Figure 6B, the teacher commented, *"loved what some of the kids did by conceptualising a game that is sensor-enabled! - Blow to raise the humidity over a certain level, resulting in a sprite moving a few steps."*

The students in our evaluation were at the very low end of the recommended age range for Scratch. One of the key principles of Scratch is its 'low floor', referring to the ease with which learners can get started. We have found that our sensor toolkit maintains this low floor, even for the youngest students for whom Scratch is recommended. One teacher stated, *"Overall, the session was very engaging even for the younger age group. I see good potential here. The kids definitely wanted (to do) more"*. Furthermore, teachers also appreciated the design of the session in having students work in pairs. They noted that *"collaboration between the kids worked well"* and were *"surprised by the kids' desire to share their projects"*. Consequently, all teachers found our toolkit a viable approach for motivating students to learn programming and were eager to use it in the future.

7 CONCLUSION

In this paper, we describe the design, development and evaluation of a tool that allows young learners to easily incorporate real-time sensor data into their visual programming environment. Evaluation of this in a classroom setting with 19 young learners (aged 7–9) revealed that the sensors were popular with students and all five of our sensor types were used in the student projects. Students used the real-time sensor data mostly to evoke visual changes in sprites, consistent with prior studies, and they found our scaffolded blocks that were triggered by an increase or decrease in sensor data particularly useful. At the end of the session, students reported that programming is an interesting activity that requires creativity. Feedback from the two teachers involved in the session was very positive as they saw our toolkit as an exciting prospect to generate enthusiasm and creativity in a time-constrained classroom. Given that students were keen to express their creativity and teachers saw the ease of integrating our toolkit into classroom teaching, we see great value in providing plug-and-play sensor availability as a native feature in block-based programming environments.

ACKNOWLEDGMENTS

This work was supported by the Assistive Augmentation research grant under the Entrepreneurial Universities (EU) initiative of New Zealand. We would like to thank MyCreate Lab, Singapore for their support in making this work possible.

REFERENCES

[1] Jonny Austin, Howard Baker, Thomas Ball, James Devine, Joe Finney, Peli De Halleux, Steve Hodges, Michał Moskal, and Gareth Stockdale. 2020. The BBC Micro:Bit: From the U.K. to the World. *Commun. ACM* 63, 3 (feb 2020), 62–69. https://doi.org/10.1145/3368856

[2] Brett A. Becker, Paul Denny, Raymond Pettit, Durell Bouchard, Dennis J. Bouvier, Brian Harrington, Amir Kamil, Amey Karkare, Chris McDonald, Peter-Michael Osera, Janice L. Pearce, and James Prather. 2019. Compiler Error Messages Considered Unhelpful: The Landscape of Text-Based Programming Error Message Research. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education* (Aberdeen, Scotland Uk) *(ITiCSE-WGR '19)*. Association for Computing Machinery, New York, NY, USA, 177–210. https://doi.org/10.1145/3344429.3372508

[3] Marina Umaschi Bers. 2018. Coding, playgrounds and literacy in early childhood education: The development of KIBO robotics and ScratchJr. In *2018 IEEE Global Engineering Education Conference (EDUCON)*. 2094–2102. https://doi.org/10.1109/EDUCON.2018.8363498

[4] Marina Umaschi Bers. 2019. Coding as another language: a pedagogical approach for teaching computer science in early childhood. *Journal of Computers in Education* 6, 4 (2019), 499–528. https://doi.org/10.1007/s40692-019-00147-3

[5] 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* (New York, New York, USA) *(IDC '13)*. Association for Computing Machinery, New York, NY, USA, 173–182. https://doi.org/10.1145/2485760.2485786

[6] Tracey Booth and Simone Stumpf. 2013. End-User Experiences of Visual and Textual Programming Environments for Arduino. In *End-User Development*, Yvonne Dittrich, Margaret Burnett, Anders Mørch, and David Redmiles (Eds.). Springer Berlin Heidelberg, Heidelberg, 25–39. https://doi.org/10.1007/978-3-642-38706-7_4

[7] Neil C. C. Brown, Amjad Altadmri, and Michael Kölling. 2016. Frame-Based Editing: Combining the Best of Blocks and Text Programming. In *2016 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*. IEEE Computer Society, USA, 47–53. https://doi.org/10.1109/LaTiCE.2016.16

[8] Neil C. C. Brown, Sue Sentance, Tom Crick, and Simon Humphreys. 2014. Restart: The Resurgence of Computer Science in UK Schools. *ACM Trans. Comput. Educ.* 14, 2, Article 9 (June 2014), 22 pages. https://doi.org/10.1145/2602484

[9] Jiashuo Cao, Samantha W. T. Chan, Dawn L Garbett, Paul Denny, Alaeddin Nassani, Philipp M. Scholl, and Suranga Nanayakkara. 2021. Sensor-Based Interactive Worksheets to Support Guided Scientific Inquiry. In *Interaction Design and Children* (Athens, Greece) *(IDC '21)*. Association for Computing Machinery, New York, NY, USA, 1–7. https://doi.org/10.1145/3459990.3460716

[10] Yu-Hui Ching, Yu-Chang Hsu, and Sally Baldwin. 2018. Developing Computational Thinking with Educational Technologies for Young Learners. *TechTrends* 62, 6 (2018), 563–573. https://doi.org/10.1007/s11528-018-0292-7

[11] Zhen Liu Danli Wang, Tingting Wang. 2014. A tangible programming tool for children to cultivate computational thinking. *The Scientific World Journal* Article 428080 (2014), 1–10. https://doi.org/10.1155/2014/428080

[12] Paul Denny, Brett A. Becker, Michelle Craig, Greg Wilson, and Piotr Banaszkiewicz. 2019. Research This! Questions That Computing Educators Most Want Computing Education Researchers to Answer. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (Toronto ON, Canada) *(ICER '19)*. Association for Computing Machinery, New York, NY, USA, 259–267. https://doi.org/10.1145/3291279.3339402

[13] Michael D. DeSchryver and Aman Yadav. 2015. Creative and Computational Thinking in the Context of New Literacies: Working with Teachers to Scaffold Complex Technology-Mediated Approaches to Teaching and Learning. *Journal of Technology and Teacher Education* 23, 3 (2015), 411–431.

[14] Caitlin Duncan and Tim Bell. 2015. A Pilot Computer Science and Programming Course for Primary School Students. In *Proceedings of the Workshop in Primary and Secondary Computing Education* (London, United Kingdom) *(WiPSCE '15)*. Association for Computing Machinery, New York, NY, USA, 39–48. https://doi.org/10.1145/2818314.2818328

[15] Katrina Falkner, Sue Sentance, Rebecca Vivian, Sarah Barksdale, Leonard Busuttil, Elizabeth Cole, Christine Liebe, Francesco Maiorana, Monica M. McGill, and Keith Quille. 2019. An International Comparison of K-12 Computer Science Education Intended and Enacted Curricula. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '19)*. Association for Computing Machinery, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.1145/3364510.3364517

[16] Aamir Fidai, Mary Margaret Capraro, and Robert M. Capraro. 2020. "Scratch"-ing computational thinking with Arduino: A meta-analysis. *Thinking Skills and Creativity* 38 (2020), 100726. https://doi.org/10.1016/j.tsc.2020.100726

[17] Louise P. Flannery, Brian Silverman, Elizabeth R. Kazakoff, Marina Umaschi Bers, Paula Bontá, and Mitchel Resnick. 2013. Designing ScratchJr: Support for Early Childhood Learning through Computer Programming. In *Proceedings of the 12th International Conference on Interaction Design and Children* (New York, New York, USA) *(IDC '13)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/2485760.2485785

[18] Neil Fraser. 2015. Ten Things We've Learned from Blockly. In *Proceedings of the 2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond) (BLOCKS AND BEYOND '15)*. IEEE Computer Society, USA, 49–50. https://doi.org/10.1109/BLOCKS.2015.7369000

[19] Katharina Geldreich, Alexandra Simon, and Elena Starke. 2019. Which Perceptions Do Primary School Children Have about Programming?. In *Proceedings of the 14th Workshop in Primary and Secondary Computing Education* (Glasgow, Scotland, Uk) *(WiPSCE'19)*. Association for Computing Machinery, New York, NY, USA, Article 1, 7 pages. https://doi.org/10.1145/3361721.3361728

[20] Alexandra Gendreau Chakarov, Quentin Biddy, Jennifer Jacobs, Mimi Recker, and Tamara Sumner. 2020. Opening the Black Box: Investigating Student Understanding of Data Displays Using Programmable Sensor Technology. In *Proceedings of the 2020 ACM Conference on International Computing Education Research* (Virtual Event, New Zealand) *(ICER '20)*. Association for Computing Machinery, New York, NY, USA, 291–301. https://doi.org/10.1145/3372782.3406268

[21] Jamie Gorson, Nikita Patel, Elham Beheshti, Brian Magerko, and Michael Horn. 2017. TunePad: Computational Thinking Through Sound Composition. In *Proceedings of the 2017 Conference on Interaction Design and Children* (Stanford, California, USA) *(IDC '17)*. Association for Computing Machinery, New York, NY, USA, 484–489. https://doi.org/10.1145/3078072.3084313

[22] Gyeongyong Heo. 2019. Implementation of an Arduino Compatible Modular Kit for Educational Purpose. *Journal of the Korea Institute of Information and Communication Engineering* 23, 5 (2019), 547–554. https://doi.org/10.6109/jkiice.2019.23.5.547

[23] Steve Hodges, Sue Sentance, Joe Finney, and Thomas Ball. 2020. Physical Computing: A Key Element of Modern Computer Science Education. *Computer* 53, 4 (2020), 20–30. https://doi.org/10.1109/MC.2019.2935058

[24] Michael S. Horn, R. Jordan Crouser, and Marina U. Bers. 2012. Tangible interaction and learning: the case for a hybrid approach. *Personal and Ubiquitous Computing* 16, 4 (2012), 379–389. https://doi.org/10.1007/s00779-011-0404-2

[25] Olivera Iskrenovic-Momcilovic. 2019. Pair programming with scratch. *Education and Information Technologies* 24, 5 (2019), 2943–2952.

[26] Michail Kalogiannakis and Stamatios Papadakis. 2019. Pre-service kindergarten teachers' acceptance of 'ScratchJr' as a tool for learning and teaching Computational Thinking and science education. *Journal of Emergent Science* 16 (2019), 31–34.

[27] F. Klassner and S.D. Anderson. 2003. LEGO MindStorms: not just for K-12 anymore. *IEEE Robotics Automation Magazine* 10, 2 (2003), 12–18. https://doi.org/10.1109/MRA.2003.1213611

[28] Siu-Cheung Kong, Ming Ming Chiu, and Ming Lai. 2018. A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education. *Computers & Education* 127 (2018), 178–189. https://doi.org/10.1016/j.compedu.2018.08.026

[29] Anastasia Kovalkov, Avi Segal, and Kobi Gal. 2020. Inferring Creativity in Visual Programming Environments. In *Proceedings of the Seventh ACM Conference on Learning @ Scale* (Virtual Event, USA) *(L@S '20)*. Association for Computing Machinery, New York, NY, USA, 269–272. https://doi.org/10.1145/3386527.3406725

[30] Irene Lee, Fred Martin, Jill Denner, Bob Coulter, Walter Allan, Jeri Erickson, Joyce Malyn-Smith, and Linda Werner. 2011. Computational Thinking for Youth in Practice. *ACM Inroads* 2, 1 (2011), 32–37. https://doi.org/10.1145/1929887.1929902

[31] I Scott MacKenzie. 1992. Fitts' law as a research and design tool in human-computer interaction. *Human-computer interaction* 7, 1 (1992), 91–139.

[32] Paul Marshall. 2007. Do Tangible Interfaces Enhance Learning?. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction* (Baton Rouge, Louisiana) *(TEI '07)*. Association for Computing Machinery, New York, NY, USA, 163–170. https://doi.org/10.1145/1226969.1227004

[33] David Mellis, Massimo Banzi, David Cuartielles, and Tom Igoe. 2007. Arduino: An open electronic prototyping platform. In *Proc. Chi*, Vol. 2007. 1–11.

[34] Timothy R Mickel. 2015. *Kids, coding, and connections: extending the ScratchJr programming environment to support wireless physical devices*. Ph.D. Dissertation. Massachusetts Institute of Technology.

[35] Seymour Papert. 1972. Teaching Children Thinking. *Programmed Learning and Educational Technology* 9, 5 (1972), 245–255. https://doi.org/10.1080/1355800720090503 arXiv:https://doi.org/10.1080/1355800720090503

[36] Seymour Papert and Idit Harel. 1991. Situating Constructionism. In *Constructionism*. Ablex Publishing Corporation, Norwood, NJ, 193–206.

[37] Dylan J. Portelance, Amanda L. Strawhacker, and Marina Umaschi Bers. 2016. Constructing the ScratchJr programming language in the early childhood classroom. *International Journal of Technology and Design Education* 26, 4 (2016), 489–504. https://doi.org/10.1007/s10798-015-9325-0

[38] Thomas W. Price, Neil C.C. Brown, Dragan Lipovac, Tiffany Barnes, and Michael Kölling. 2016. Evaluation of a Frame-Based Programming Editor. In *Proceedings of the 2016 ACM Conference on International Computing Education Research* (Melbourne, VIC, Australia) *(ICER '16)*. Association for Computing Machinery, New York, NY, USA, 33–42. https://doi.org/10.1145/2960310.2960319

[39] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. 2009. Scratch: Programming for All. *Commun. ACM* 52, 11 (nov 2009), 60–67. https://doi.org/10.1145/1592761.1592779

[40] Mitchel Resnick and Brian Silverman. 2005. Some Reflections on Designing Construction Kits for Kids. In *Proceedings of the 2005 Conference on Interaction Design and Children* (Boulder, Colorado) *(IDC '05)*. Association for Computing Machinery, New York, NY, USA, 117–122. https://doi.org/10.1145/1109540.1109556

[41] Jean Salac, Cathy Thomas, Chloe Butler, Ashley Sanchez, and Diana Franklin. 2020. TIPP&SEE: A Learning Strategy to Guide Students through Use - Modify Scratch Activities. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, NY, USA, 79–85. https://doi.org/10.1145/3328778.3366821

[42] Sandra Schön, Martin Ebner, and Swapna Kumar. 2014. The Maker Movement. Implications of new digital gadgets, fabrication tools and spaces for creative learning and teaching. *eLearning papers* 39 (2014), 14–25. Special edition 2014 "Transforming Education through Innovation and Technology".

[43] Deborah Seehorn, Stephen Carey, Brian Fuschetto, Irene Lee, Daniel Moix, Dianne O'Grady-Cunniff, Barbara Boucher Owens, Chris Stephenson, and Anita Verno. 2011. *CSTA K–12 Computer Science Standards: Revised 2011*. Technical Report. New York, NY, USA.

[44] Sue Sentance, Jane Waite, Steve Hodges, Emily MacLeod, and Lucy Yeomans. 2017. "Creating Cool Stuff": Pupils' Experience of the BBC Micro:Bit. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) *(SIGCSE '17)*. Association for Computing Machinery, New York, NY, USA, 531–536. https://doi.org/10.1145/3017680.3017749

[45] Sue Sentance, Jane Waite, Lucy Yeomans, and Emily MacLeod. 2017. Teaching with Physical Computing Devices: The BBC Micro:Bit Initiative. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education* (Nijmegen, Netherlands) *(WiPSCE '17)*. Association for Computing Machinery, New York, NY, USA, 87–96. https://doi.org/10.1145/3137065.3137083

[46] Robert Sheehan. 2003. Children's Perception of Computer Programming as an Aid to Designing Programming Environments. In *Proceedings of the 2003 Conference on Interaction Design and Children* (Preston, England) *(IDC '03)*. Association for Computing Machinery, NY, USA, 75–83. https://doi.org/10.1145/953536.953548

[47] M. P. Jacob Habgood Simon P. Rose and Tim Jay. 2017. An Exploration of the Role of Visual Programming Tools in the Development of Young Children's Computational Thinking. *The Electronic Journal of e-Learning* 15, 4 (2017), 297–309. http://www.ejel.org/volume15/issue4/p297

[48] Hussel Suriyaarachchi, Paul Denny, and Suranga Nanayakkara. 2022. Scratch and Sense: Using Real-Time Sensor Data to Motivate Students Learning Scratch. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1* (Providence, RI, USA) *(SIGCSE 2022)*. Association for Computing Machinery, New York, NY, USA, 7 pages. https://doi.org/10.1145/3478431.3499316

[49] Markus Tyrén, Niklas Carlborg, Carl Heath, and Eva Eriksson. 2018. Considerations and Technical Pitfalls for Teaching Computational Thinking with BBC Micro:Bit. In *Proceedings of the Conference on Creativity and Making in Education* (Trondheim, Norway) *(FabLearn Europe'18)*. Association for Computing Machinery, New York, NY, USA, 81–86. https://doi.org/10.1145/3213818.3213829

[50] Danli Wang, Cheng Zhang, and Hongan Wang. 2011. T-Maze: A Tangible Programming Tool for Children. 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, 127–135. https://doi.org/10.1145/1999030.1999045

[51] David Weintrop. 2019. Block-Based Programming in Computer Science Education. *Commun. ACM* 62, 8 (jul 2019), 22–25. https://doi.org/10.1145/3341221

[52] David Weintrop and Uri Wilensky. 2015. To Block or Not to Block, That is the Question: Students' Perceptions of Blocks-Based Programming. In *Proceedings of the 14th International Conference on Interaction Design and Children* (Boston, Massachusetts) *(IDC '15)*. Association for Computing Machinery, New York, NY, USA, 199–208. https://doi.org/10.1145/2771839.2771860

[53] LeChen Zhang, Jalal Nouri, and Lennart Rolandsson. 2020. Progression Of Computational Thinking Skills In Swedish Compulsory Schools With Block-Based Programming. In *Proc. 22nd Australasian Comp. Ed. Conf.* (Melbourne, Australia) *(ACE'20)*. ACM, USA, 66–75. https://doi.org/10.1145/3373165.3373173