Meet MicroCode: a Live and Portable Programming Tool for the BBC micro:bit

KOBI HARTLEY, Lancaster University, UK
ELISA RUBEGNI, Lancaster University, UK
LORRAINE UNDERWOOD, Lancaster University, UK
JOE FINNEY, Lancaster University, UK
THOMAS BALL, Microsoft, US
STEVE HODGES, Lancaster University, UK
ERIC ANDERSON, Microsoft, US
PELI DE HALLEUX, Microsoft, US
JAMES DEVINE, Microsoft, UK
MICHAŁ MOSKAL, Microsoft, US

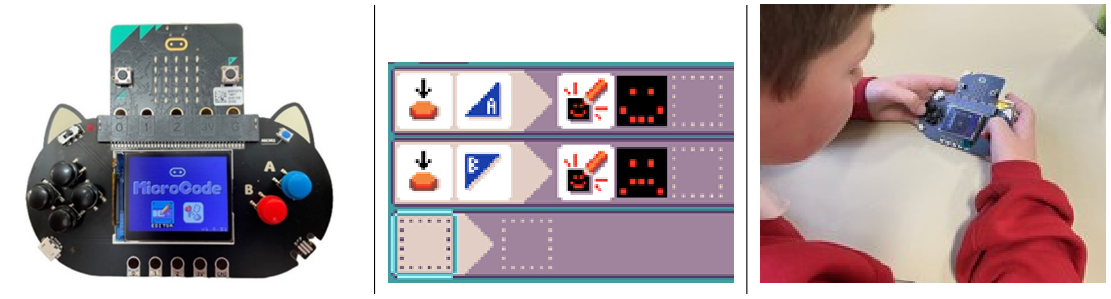


Fig. 1. The MicroCode physical computing system. On the left, micro:bit V2 inserted into a MicroCode shield and running MicroCode; in the centre, a screenshot of the MicroCode editor displaying the "Happy/Sad" program; at right, a student using MicroCode.

Physical computing has emerged as an effective approach to introducing computing and coding to students. One of the most popular enabling tools is the BBC micro:bit, well-known for its positive impact on teaching programming and driving engagement in the classroom. We extend these benefits by developing a new approach to coding with micro:bit: MicroCode. Unlike other experiences, MicroCode couples the micro:bit with a low-cost handheld accessory to enable live and portable programming via an on-device visual programming language; no separate host computer is needed. We present the design of MicroCode and the findings of a study in which we interviewed five primary school teachers and 60 children aged 10-11 working with MicroCode. The outcomes of the study show that MicroCode raised children's engagement and stimulated the development of a strong sense of agency on coding activities, while teachers felt empowered to adopt situated and cross-curricular learning approaches.

CCS Concepts: • **Human-centered computing** → **Empirical studies in HCI**; • **Social and professional topics** → **K-12 education**.

Additional Key Words and Phrases: child-computer interaction, physical computing education, micro:bit.

ACM Reference Format:
Kobi Hartley, Elisa Rubegni, Lorraine Underwood, Joe Finney, Thomas Ball, Steve Hodges, Eric Anderson, Peli de Halleux, James Devine, and Michał Moskal. 2018. Meet MicroCode: a Live and Portable Programming Tool for the BBC micro:bit. In *Delft '24: ACM Symposium on Neural Gaze Detection, June 03–05, 2023, Woodstock, NY*. ACM, New York, NY, USA, 24 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The UK, like many other countries, has integrated computer science and computational thinking into the school curriculum over the last decade. As a result, various methodologies for incorporating programming into computer science classes have emerged. Among these has been the convergence of physical computing [26] and constructionist learning pedagogy [51]. This emerging educational approach promotes a hands-on, tangible and experiential approach to learning that empowers and motivates children to turn their ideas into meaningful digital creations. Programming is an essential part of the physical computing experience. It often takes place with the physical computing device tethered to a classroom computer such as a laptop, desktop or tablet, which hosts the programming environment. The device must be connected to the host computer to update the program it's running, and even portable host computers such as laptops and tablets are relatively bulky and therefore unlikely to be taken outside the classroom. Updating the program on the physical computing device may take sufficiently long that it breaks a student's flow and focus. And even if it's fast, the child has to continually transition their focus of attention back-and-forth between the host (during programming) and the device (for testing). Physical computing is, by its very nature, a highly iterative process embodied in the physical environment, so it's important to allow children to edit and evaluate their code quickly, easily and in the context of their project. The established paradigm described above – where a physical computing device is tethered to a separate host computer – can limit the ways in which teachers and students work, and what they imagine is possible. This leads to a gap between the potential of physical computing and the technologies that currently enable it.

The goal of the MicroCode project is to enable students to program a variety of applications for the micro:bit without the need for a separate computer to host the programming environment. We take advantage of the micro:bit V2, introduced in 2020, by slotting it into a display shield accessory, see Figure 1 left. The micro:bit V2's increased memory and speed, coupled with the shield's D-pad, buttons and color display, allows the micro:bit to host the programming environment *and* run user code at the same time.

Research on programming environments for novices show that successful early adoption is based on the ease of use, familiarity, and understating of the conceptual model [8, 17, 60]. Thus, the design of the MicroCode programming language and editor draws heavily on Kodu [43], an established visual programming environment for young learners around 5–11 years old that originally used an Xbox game controller for programming. Figure 1 shows a simple MicroCode program consisting of two "When-Do" rules. Together, the MicroCode software and hardware provide a live, portable and visual programming environment for the micro:bit V2 that runs on the same micro:bit as the user's code. When the micro:bit is unplugged from the shield, as long as it is powered (e.g. with a battery or via USB power), the user's current program will continue to execute. This flexibility and portability provides many benefits and enables the creation of educational activities drawing on situated learning [11], where children can practice and apply their knowledge across multiple contexts. We believe teachers can leverage these aspects to organise activities across both indoor and outdoor environments – blending physical engagement with learning activities.

To better understand the broad value of the portable programming experience for physical computing that MicroCode provides, we carried out a study across three different schools in the UK. This research helped us to answer three main research questions:

- RQ1: In what ways does MicroCode add value across the broader curriculum beyond coding?
- RQ2: How does MicroCode create engagement and a sense of agency with young children?
- RQ3: How does MicroCode enable young children to tackle introductory programming tasks?

Our study included two phases. In the first phase we interviewed five teachers of children in Key Stages 1 and 2 (ages 4-11) to better understand where MicroCode could add value. We discussed how MicroCode might support their teaching needs and help them achieve their curriculum goals. In the second phase, we observed children using both MicroCode so that we could understand the benefits and challenges of using it. Phase 2 was conducted in two primary schools with Key Stage 2 primary school children in Years 5 and 6 (aged 10-11).

From our study we learned that MicroCode supports micro:bit-based learning at primary level very effectively, stimulating a high level of engagement and agency in children. We discovered how different aspects of MicroCode support learning; in particular, the ease of updating the micro:bit's program "on the go" opens up the micro:bit to new contexts, such as outdoor data collection. Teachers were inspired by MicroCode and envisaged new ways of using physical computing to achieve school curriculum goals, including those beyond computing-related subjects.

Our work has three main contributions:

- we introduce MicroCode, a live and portable programming experience for the micro:bit;
- we collect and present evidence about the benefits and shortcomings of MicroCode; and
- we reflect on how MicroCode can support children's experience and teachers' curriculum objectives.

The rest of this paper is organized as follows. Section 2 reviews related work, including the BBC micro:bit, MakeCode, and Kodu. Section 3 describes the design of the MicroCode language and editor. Section 4 provides an overview of our study design, data collection, and analysis; Section 5 presents the results of our analysis and Section 6 discusses our findings. Section 7 concludes the paper and Section 8 reviews how study participants were recruited. The Appendix provides more details about the survey given to children and sample programs presented by the teachers.

2 RELATED WORK

In this section we review the extensive body of theoretical work and empirical research studying the benefits of teaching and learning with physical computing in the classroom, live programming, and physical computing toolkits, with particular attention to primary school ages.

2.1 Creating a meaningful learning experience with physical computing

Physical computing involves the creation and use of tangible, embedded computing devices that can sense and respond to their environment [26]. Physical computing devices often combine a variety of sensors and actuators with a programmable microcontroller unit (MCU) that can run from batteries. These devices allow students to engage in hands-on construction and activities where the device is used interactively in the environment for a variety of purposes, ranging from fitness and environmental sensing to physics experiments and art projects.

A wealth of literature on learning theories has shown the benefit of using physical computing to create a meaningful learning experience by engaging students "in creating something that is meaningful to themselves or to others around them" [57]. The manipulation of tangible objects and embodied interactions has been shown to support a hands-on approach to skill development. A key example of this is the field of physical computing, which "involves combining software and hardware to build interactive physical systems that sense and respond to the real world" [25].

Physical computing is often connected to the concept of experiential learning [40]: learning by doing or, more specifically, a process where knowledge is learned from understanding and transforming information [12]. Requiring a student to create interactive objects and systems in this way involves critical thinking and can foster imagination and creativity [75], as well as triggering intrinsic motivation [54] and engagement [65]. Within this context we agree with Boekaerts that engagement is a result of motivation [6]. Indeed, students' engagement is an equally important factor in improving long-term learning outcomes, as discussed by Kearsley [39]; it also encourages on-going interaction with educational content or systems [50] which of course has further long-term benefits. This has been demonstrated in a range of projects such as the design of IoT tangibles in primary schools [22] and engaging children with AI programming [33]. Collaboration is a key learning goal across the school curriculum.

Recent advances in the use of technology within the classroom have both reinforced and facilitated this goal [8, 10, 20, 34, 37]. Studies have shown how tangible physical computing can have a positive effect on soft skills [25, 26, 28], which can be enhanced by ensuring a consistent distribution of tasks, an essential element for supporting collaboration [35, 59]. In addition, it is paramount to promote agency within learning; to encourage independence and confidence when children are problem-solving. Concepts of agency relate to active learning, and thus back to engagement [18]. Again, recent computing research has shown that technology can significantly improve educational outcomes [38, 44, 77]. It's clear that all these factors are interrelated and Szabo et al. [68] discuss some of these relationships and their relevance to the evolving technology landscape.

MicroCode is grounded upon and leverages these theoretical approaches, aiming to provide innovative features that create new opportunities to support teaching through physical computing.

2.2 Live Programming

The concepts of live coding and live programming has been explored by several scholars [64] and are summarised as follows [56]: live programming establishes a cause-effect chain by immediately reflecting the impact of a change to a program on the program execution, with the aim of making programming more comprehensible and accessible. In general, the concept of liveliness in the context of coding and programming is connected to the idea of "creating an impression of changing a program while it is running" [56]. The literature cites several advantages and the pedagogical potential of live programming [58] in computing education. One of the main advantages of live programming is that students receive immediate feedback on the effect of manipulating parameters, readily seeing how program execution changes [72]. This immediacy of feedback facilitates comprehension and learning of program behavior.

Live programming has proved to be particularly helpful for novices and less experienced programmers [31]: students feel in control, can explore the design space, and can directly assess the effect of their changes. This can, in turn, improve the self-direction of students [7], increase their engagement [4] and show them how to apply programming concepts [66]. Like physical computing, live programming is based on the constructivist framework [42] supporting discovery [21] and active learning [77].

We build on previous work on live programming for physical computing, investigating how adding portability can change the student's experience.

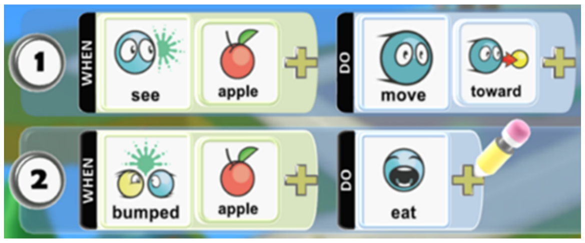


Fig. 2. A Kodu program with two "When-Do" rules.

2.3 Portable programming for physical computing

The educational benefits of physical interfaces have led to many experiences in the last few decades which do not require interaction with a companion or host computer [45, 78]. Hereafter we refer to this approach as portable programming for physical computing. For example, Perlman's 'Action Box' [53] supported programming-by-example of a robot turtle, an idea that was subsequently picked up in the CurlyBot system [19] and by Topobo [52]. The 'Action Box' associates actions the turtle might make, such as 'move forward' or 'turn to the right', with buttons that must be pressed in sequence.

This approach is still used by modern devices such as the popular Bee-Bot mobile robot [23], a standalone device with buttons mounted on its back. In these examples, there is no persistent, visible, or editable representation of the program. Another approach explored by Perlman was the 'Slot Machine' [53] which used physical tokens to represent commands for a turtle robot to obey in sequence. A similar principle is reflected in projects and products that use tangible but passive blocks to construct programs, such as Strawbies [30], and the Quetzal and Tern programming languages [29]. The Ozobot Evo is a small turtle-like robot that can be programmed in a 'pen-and-paper mode' to follow a colored line drawn on the paper. This has proven successful in the primary school classroom [24], as well as with blind and low-vision children [48]. roBlocks [63], which evolved into Cubelets [62], takes a slightly different approach to encoding the behavior of a robotic device. In this case a variety of small cubes may be snapped together magnetically (and electrically) to define both the form and function of the resulting self-contained assembly. A similar approach has been adopted for non-robotics applications too; examples include Electronic Blocks [76], Tangible Programming Bricks [45, 46] and Project Bloks [5]. Finally, environments such as LittleBits [3], Circuit Stickers [27] and LightUp [9] provide tangible, tactile approaches to building with electronics where the resulting functionality of a construction is determined by its physical configuration.

Many devices integrate a screen to support students while they are programming. An example is Lego Mindstorms, which is based around a standalone device with a small screen, buttons, and ports for connecting sensors and motors. Its user interface allows the creation of very simple programs that contain a sequence of actions, with looping also possible, as described in the Lego NXT User Guide [1]. An example from the research community is TileCode [2] which extends this on-device programming approach with a simple editing paradigm that runs on MakeCode Arcade gaming handhelds, allowing users to create new games without a host computer if they wish. TileCode supports games based on a tile grid and uses graphical rewrite rules to define the evolving game behavior. TileCode was evaluated in a family-based study and used both web-browser-based and standalone on-device coding [16]. Note that the focus of the study in [16] was how families learn together rather than a comparison of web-based vs device-based programming experiences. Although

[1]https://www.generationrobots.com/media/Lego-Mindstorms-NXT-Education-Kit.pdf

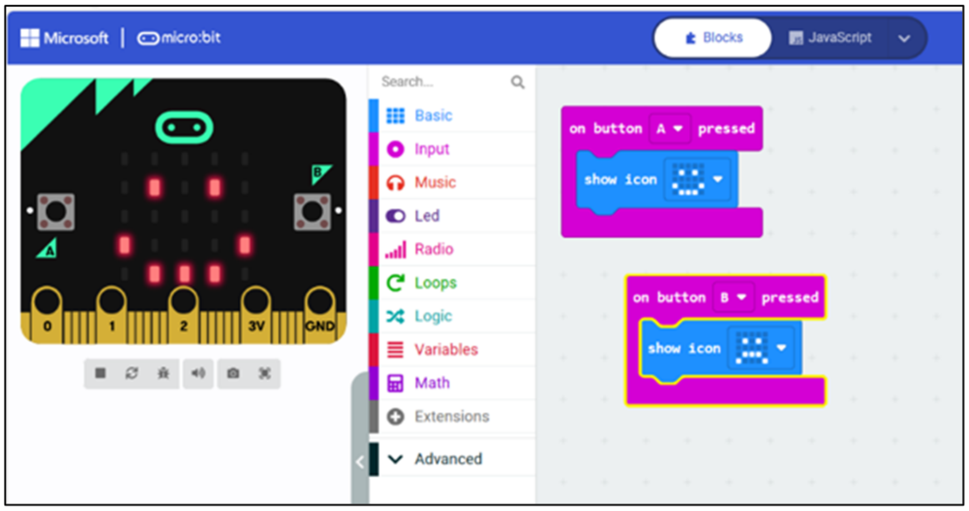


Fig. 3. MakeCode for the micro:bit, with "Happy/Sad" program.

not a physical computing experience, Kodu was inspired by programming models for describing robot behavior and makes extensive use of 3D graphics rendering to provide a virtual world that is modifiable, programmable, and engaging [13, 43, 67], Figure 2 shows two examples of When-Do rules from Kodu.The user can edit the program using an Xbox controller or PC keyboard. Per Coy, "One unexpected benefit of using the [Xbox] controller on the PC is that children immediately pick it up and feel like they are playing rather than learning. In fact, they are doing both" [13].

We build on the rich history of portable programming for physical computing, bringing together a familiar game-controller style accessory and a built in screen, with the popular micro:bit physical computing device.

2.4 The BBC micro:bit and MakeCode

The BBC micro:bit is a popular physical computing device for introducing computers and coding to students that debuted in 2016 [1]. Started by the BBC and partners with an initial deployment of one million micro:bits in the UK, the micro:bit was then taken global via the non-profit Micro:bit Educational Foundation, with over 9 million micro:bits distributed worldwide to date [1, 61]. Micro:bit has shown value in its accessibility, both in terms of its easy learning curve and its affordability and availability [32, 41, 71]. Indeed, the educational benefits of micro:bit in schools have been proven, e.g., [74]. The micro:bit has a programmable MCU, a 5x5 matrix of LEDs, two buttons, several sensors, and wireless communications capabilities. Programming environments that support the micro:bit include Microsoft MakeCode [14], MicroPython, Scratch, and Swift Playgrounds. These have many excellent features and have proven to be useful in the classroom setting but are all constrained by the fact that they need a separate host computer, as discussed above. Many educators use Microsoft MakeCode as a starting point for the micro:bit [17, 36, 47, 73, 74]. MakeCode is a web app that allows users to program the micro:bit (V1 and V2) visually via the Blockly framework and with text via JavaScript or Python [14]. Figure 3 shows a screenshot of the MakeCode editor with one of the popular starter programs, "Happy/Sad". MakeCode has proven popular in the classroom, because the drag-and-drop block-based visual coding interface it provides is generally accepted as being more suitable for young learners than text-based coding [17]. It is simple to use and understand, and some users are already familiar with related systems like Scratch

[36, 47]. While MakeCode provides an excellent experience, it does require a host computer running a modern web browser. As highlighted above, liveliness, tangibility and portability are important elements in student engagement and the reliance on a host computer reduces all three of these even when using a physical computing paradigm.

MicroCode was designed to complement MakeCode, supporting physical computing with the micro:bit without compromising liveliness, tangibility and portability. In this way we hope to give students an exciting new experience while simultaneously providing teachers with more flexibility as they consider what ages, contexts of use and learning outcomes they want to address.

3 MEET MICROCODE

The goal of MicroCode is to make live and portable programming of the micro:bit possible in a low-cost and practical way. When the micro:bit is slotted into a MicroCode shield, as shown in Figure 1, the MicroCode software provides:

- a portable and visual programming experience (see Figure 1);
- live programming with instantaneous program execution after every edit;
- cursor-based navigation and editing with help text;
- access to most of the micro:bit V2 hardware features.

With MicroCode we would like to make the task of coding as tangible and hands-on as the act of interacting with the resulting program when it runs. The MicroCode programming model is based on Kodu, as previously reviewed, using When-Do rules composed of visual tiles. To provide a low friction and familiar route for teachers, MicroCode was designed to support many of the existing micro:bit and MakeCode projects published by the Micro:bit Educational Foundation (MEF).[2] At the time of writing, the set of features included in MicroCode allows the creation of 28 of the MEF's 43 beginner projects, 11 of the 29 intermediate projects and 5 of the 15 advanced projects.

MicroCode was implemented using MakeCode Arcade [49], a MakeCode editor for creating 1980's style retro arcade games for the web and for dedicated gaming hardware. Arcade games are based on a 160 x 120 color display, a four-way D-pad or equivalent buttons, additional 'A' and 'B' buttons, and an MCU with enough memory and speed to drive the display and run the game. While the micro:bit V2 (128kB RAM, 64MHz CPU) more than meets the memory and speed requirements for MakeCode Arcade, it doesn't include the necessary screen or enough buttons. This shortcoming was addressed through the use of a MicroCode shield, an accessory which the micro:bit V2 can be slotted into, as shown in Figure 1. KittenBot, a manufacturer of educational hardware including accessories for the micro:bit, now manufactures and sells this shield and this is what we used in our classroom studies. Several other manufacturers now make Arcade shields for the micro:bit V2.

3.1 Programming language and program execution

The MicroCode language has events and actions that model the hardware sensors and output mechanisms of the micro:bit, as reviewed below. Other features, such as variables, numbers, math, and program pages are elided from the description. Figure 4(a) shows the event tiles that represent the micro:bit's hardware sensors for detecting input. Under each event tile are the filter tiles associated with that event. For example, the program of Figure 1 uses the 'press' event along with filters for the 'A' and 'B' buttons of the micro:bit. The icons provide a direct visual link between a program and hardware features, which is not as evident in the block-based code of MakeCode (see Figure 3). Figure 4(b) shows six action tiles for generating output via the micro:bit's 5x5 LED display, speaker and radio. The first two action tiles

[2] https://microbit.org/projects/make-it-code-it/

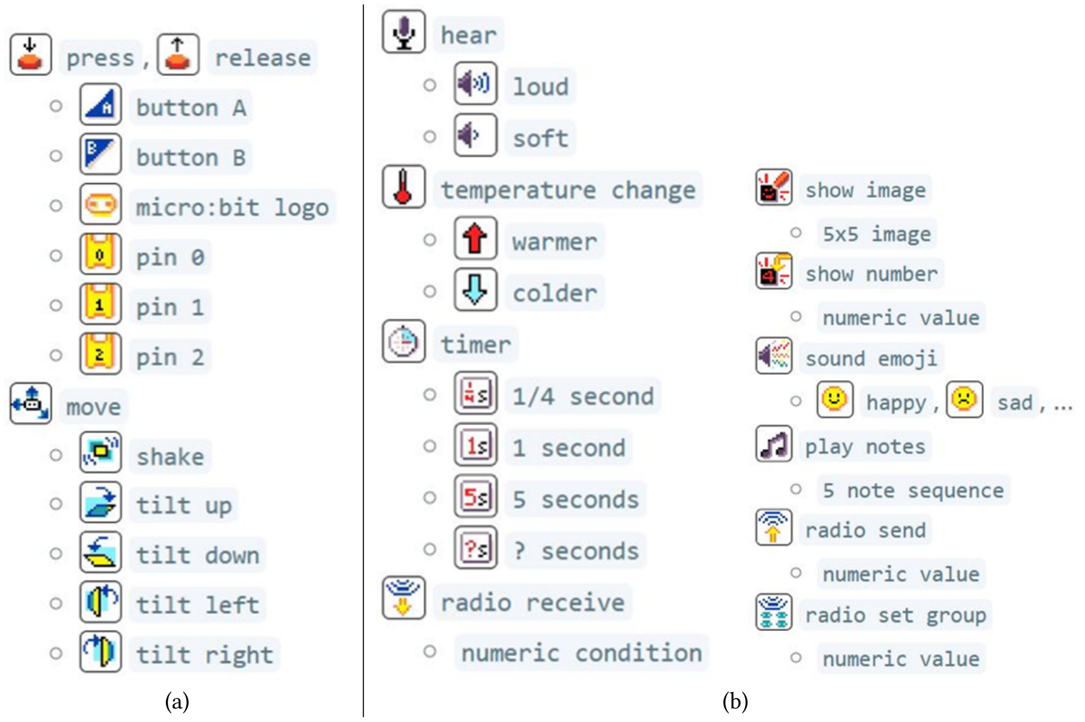


Fig. 4. MicroCode programming tiles for (a) events and (b) actions. Images were designed to represent the physical hardware.

show a 5x5 image or a number on the display; editors for creating 5x5 images and note sequences are discussed later. The next two tiles play one of a gallery of sound emojis, or a sequence of musical notes defined by the user. The final two tiles send a message over the micro:bit's radio and set a radio group number for separating radio messages into different channels.

MicroCode programs execute immediately after every edit. Incoming events are serialized and for each event, any Do section which matches that event (per its When section) executes in serial order from the top to the bottom of the page. Some actions, such as playing a sound or animating the screen, execute asynchronously; in this way rules appear to execute simultaneously upon reception of a matching event.

3.2 Program creation

Figure 5 shows how to create the first rule in Figure 1 with MicroCode, starting with an empty program. The MicroCode editor has a blue cursor in what we call a skeletal rule, see upper left of Figure 5(a). The user can move between nearby tiles using the D-pad of the MicroCode shield. The skeletal rule has two "placeholder" tiles, one for the sensor in the When section (in beige) and one for the action of the corresponding Do section (in dark grey). In Figure 5(a) the cursor is on the sensor placeholder and the user has pressed the A button, activating the When menu, which displays all the available sensor inputs and allows the user to select among them. Figure 5(b) shows the program after the selection of the 'press' sensor. Note that as soon as a rule is no longer skeletal, the editor automatically adds a new skeletal rule to the program. The When section of the first rule has expanded to make space for a filter, and the available filters for

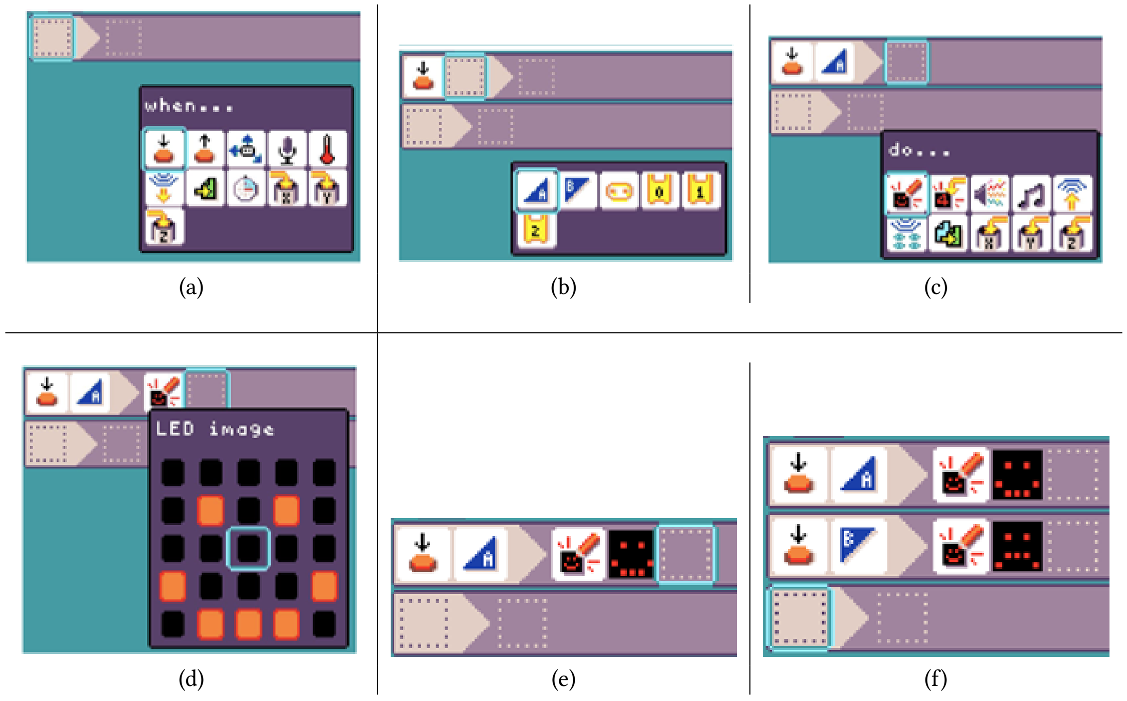


Fig. 5. (a-d) Editing actions to create a When-Do rule; (e) resulting rule; (f) addition of second rule.

the press event are shown in the menu. Figure 5(c) shows the program after the selection of the micro:bit's A button filter, which completes the When section – no further filters are allowed in this case. The cursor is now on the action placeholder and the Do menu displays all the available actions. Figure 5(d) shows the program after the show image action has been selected, which brings up a specialized editor that lets the user define a 5x5 image. Figure 5(e) shows the complete When-Do rule and Figure 5(f) shows the completed program with a second rule, implementing the basic logic of the "Happy/Sad" program from Figure 1.

4 THE STUDY: OVERVIEW

We undertook an exploratory study in which we assessed the potential for MicroCode to enable a live programming physical computing class in a primary school. Considering that this is the first study "in the context" our research was directed to investigate the potential of MicroCode and understanding the pedagogical design space for teaching physical computing in class. Thus, in a first phase we involved the teachers in a deep reflection on technology-based activity in class. Then in a second phase, we assess MicroCode in class with children. The two phases were designed to inform our three main research questions, considering the MicroCode's nature of being portable, based on visual programming, and enabling a live coding experience:

- RQ1: In what ways does MicroCode add value across the broader curriculum beyond coding?
- RQ2: How does MicroCode create engagement and a sense of agency with young children?
- RQ3: How does MicroCode enable young children to tackle introductory programming tasks?

Table 1. Participants in Phase 1.

Teacher	Year	Subjects	Age range
T1	Y6	English and computing	10-11
T2	Y5	Computing	9-10
T3	Y5	Computing	9-10
T4	Y2	Computing	6-7
T5	Reception/Y1	Computing	4-6

Table 2. Participants in Phase 2.

School	Year	Age	F	M	Total
A	Y6	11	17	14	31
B	Y5	10	13	16	29
Total			30	30	60

4.1 Participants and Recruitment

We recruited teachers by leveraging our UK network from previous research projects. We received replies from eight schools and selected participants based on their availability for engagement according to the project timeline, with consideration for their knowledge of micro:bit (expert and non-expert). In all cases teachers were asked to participate by the schools' directors, contacted by us via email, and formally agreed to participate in the study by signing a consent form.

We conducted Phase 1 in three schools and in the UK Key Stages 1 and 2.: from Reception (4-5 years old) to Year 6 (10-11 years old). By including all the school years where computing is part of the curriculum, we hoped to uncover the most appropriate age groups for MicroCode.

Phase 2 happened in two of the three Phase 1 schools with the support of two teachers, one of whom had also participated in Phase 1. We selected children in Years 5 (Y5) and 6 (Y6) as the target group for this phase, noting that children of these ages can articulate their thoughts well and provide insightful feedback on their experiences. Our participants were all 10 or 11 years old. In Phase 2 the teachers asked children and their parents/guardians about their willingness to participate and when confirmed, they signed a consent form. Both teachers and children participated on a voluntary basis and could withdraw from the study at any time. In Phase 2, 60 children we engaged over a period of 6 months: School A (31, Y6 children; all 11 years old, F=17, M=14) School B (29, Y5 children; all 10 years old, F=13, M=16).

4.2 Study Procedure: an overview of phases 1 and 2

In Phase 1, we investigated the teachers' perspectives to understand how MicroCode might be used in class by interviewing five teachers with different levels of experience in using micro:bit (two experts and one beginner). We asked teachers about the potential for MicroCode to help them with the implementation of the school curriculum and their learning objectives. This phase took place prior to the assessment of MicroCode in class and informed our planning for Phase 2. In particular, data collected from teachers' interviews allowed us to better understand the school context and design a suitable activity for Phase 2. Data collected in Phase 1 naturally fed into our analysis addressing RQ1. In Phase 2 we explored the children's experience with MicroCode via an in-classroom exercise in which the children used a micro:bit with the MicroCode software and shield. By observing children interacting with MicroCode, we aimed to gain a deeper understanding of the challenges of and opportunities for MicroCode. This classroom assessment helped us address RQ2 and RQ3. More specifically, RQ2 is addressed by using data about engagement and sense of agency that emerged from both the survey and the notes from observation, while RQ3 is addressed using data from observational notes that help us understand whether the MicroCode's ease of use and transparency could be valuable for novices. In Phase 2 each lesson was organized as follows:

- Introduction to micro:bit with MicroCode (10 min).
- Coding activities (30 min).
- Wrap up and questionnaire filling (20 min).

The tasks were selected in agreement with the teachers based on the micro:bit lesson "Emotion Badge" . From a programming perspective, the children create a simple reactive program – more specifically, their code must react to the A and B buttons to draw different images on the 5x5 micro:bit display. The teachers' sample programs are available in Appendix B and the input controlling output is shown in Figures 1 and 3. Children performed the activity in pairs, assigned by teachers based on their abilities. The lesson was led by the teachers and observed by three researchers who provided the content and the technology. Teachers led the activity in class by showing the students how to use the technology and helping them when required.

4.3 Data collection and analysis

During two phases of the study we collected multiple data that were analyzed separately and afterwards triangulated to provide answers to our research questions. Overall, the audio of the interview and the observational notes (taken by the authors) were recorded, transcribed, and analyzed following a thematic inductive approach in parallel and blind [70] using "NVivo 12" and "ATLAS.ti 23". All analyses were performed individually by three researchers, and upon an inter-coder agreement on discrepancies, results were merged into the final outcome as per [69]. We calculated the simple percentage agreement which resulted in a statistic of ~ 85% agreement between all coders and of ~ 87% with data collected respectively in Phase 1 and 2.

In Phase 1 we ran semi-structured interviews with our five teachers to explore their teaching practices and learn how they might make use of MicroCode. All teachers interviewed taught all subjects. T1 and T2 are subject specialists in computing who have taught with micro:bit before, while T3-5 are not specialists and have never used micro:bit. T4 had done little teaching of computing at all. T1 also participated in Phase 2. Each session involved one teacher, one micro:bit expert and two people with a HCI research background (hereafter "the researchers"). One of the researchers facilitated a one-hour discussion, during which the teachers were shown videos of MicroCode and asked how they would use it to achieve curriculum goals for each age group. After this, the teachers were asked about their activities in class and the topics taught, the technology setup in their classrooms, and what the main challenges in implementing their school curriculum were. The group then brainstormed about the potential use of MicroCode in class and what curriculum goals could be enabled with the micro:bit. By looking at the curriculum subjects, and with the aid of the HCI researcher, the teachers envisioned how the micro:bit could be used to achieve these objectives.

***A total of 125 quotations were analyzed, 28 codes generated and grouped into 7 themes**: 1. Curriculum focus on STEM-related concepts, 2. challenges with using PCs, 3. Situated learning, 4.Computing in-the-wild, 5. development and practice of soft-skills, 6. Computing to extend learning, 7. cross-curriculum concepts. Data was anonymized and we refer to the teachers using their pseudonyms, i.e. (Tn).*

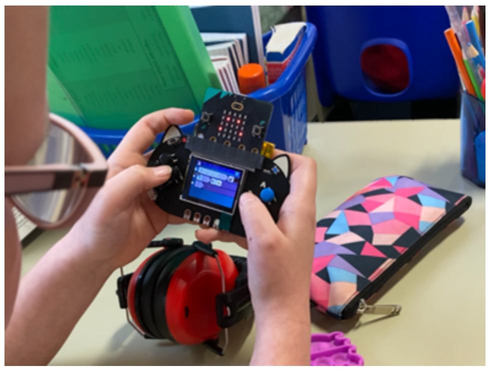


Fig. 6. Two images of the study run in one school, on the right a child who is using MicroCode for the first time.

In Phase 2 we worked with two schools (School A and B), one class in each (Y6 and Y5 respectively), taught by two teachers, one of whom participated in Phase 1. Data were collected using a questionnaire for evaluating students' self-assessment of their experience (see Appendix), and took observational notes during the tasks, including quotes from children. The children's questionnaire, adapted from [15], consisted of 14 statements about the experience, rated on a scale of 1–4 (1: completely disagree, 4: completely agree). Designed to capture specific aspects of the user experience, these are detailed in Appendix A. The questionnaire was filled out at the end of the activity in class and subsequently analyzed following the previously established methodology of "Giggle Gauge": a modified quartile scale to account for skewed positive responses in children [15].

The interpretation guidelines for Giggle Gauge advise averaging the responses of all children for each question and the scores are then grouped as follows: <3.0 is low engagement, 3-3.6 is average engagement and >3.6 is high engagement. Averaging responses for the questionnaires can provide indication of general sentiment which is useful for early iteration and rapid development [55]; we are interested here in pinpointing sentiments towards specific aspects of the user experience [13].

A total of 250 quotations were analyzed, 36 codes generated and grouped into 6 themes*: 1. Level of agency, 2. Motivation and Engagement, 3. Focus on content creation, 4. Relation to other subjects, 5. Easily to handle, 6. Autonomy and freedom. Data was anonymized and we refer to the children, observers/researchers, and teachers using pseudonyms, i.e. (IDn, On and Tn).*

4.4 Threats to validity

One potential limitation of the study is the limited number of participants (60 children and 5 teachers). However, a small sample allowed us to have a deeper understanding of their thoughts, needs, and wishes, including the contextual factors that influenced participants' perceptions and interactions with the technology. Moreover, the sample is representative of the target population as the children and teachers were consistent in terms of demographics, experience using micro:bit and computing skills. The data collection was conducted in a natural and ecologically valid setting, providing an rich dataset that enabled us to better understand the complexity of implementing a learning experience using MicroCode in a classroom. The rigor and validity of the research approach were further supported by triple-blind thematic data analysis and the triangulation of data collected through diverse methods: a standardized survey, ensuring consistency in data collection procedures, the use of a mixed methods approach (interview, survey, participatory observation), and the integration of the qualitative findings. By employing appropriate data analysis techniques (e.g. three researchers analyzing data separately in blind mode) we reduced the risk of biases.

5 RESULTS

5.1 RQ1: In what ways does MicroCode add value across the broader curriculum beyond coding?

Overall, the five teachers believed there was value in using MicroCode to raise children's interest in learning via physical computing. They envisioned concrete examples of how they could use it in the classroom to support their teaching goals. Regarding the challenges of teaching computing in general, teachers mentioned that **most computing curricula mainly focus on teaching STEM-related concepts (Theme 1).** These range from hands-on skills such as learning how to code to more abstract concepts, such as online safety: "*My specific target that I like to drive is the safety aspect of it [technology]. … using it [computing lessons] to extend learning and making sure they know how to stay safe when they're playing, especially gaming at home.*" (T3). All teachers were very excited about the potential of MicroCode to extend learning in new ways by **overcoming the challenges they currently face with the PC (Theme 2)**, e.g.: "*[the current laptops] are all plugged in and wired, as if they were desktops … and they're not mobile.*" (T2). Example benefits outlined by teachers included being able to **contextualize and provide situated learning (Theme 3)**: *"Just linking it [concepts taught in computing class] into reality in real-life learning when you're just using a computer [is very challenging]"* (T3), and being able to use **computing on-the-go and in-the-wild (Theme 4)** : "*As I was playing with it [MicroCode] I was thinking, 'Oh, wouldn't it be great if you could fit one up in a little hedgehog tunnel?' Check the next day, and 'Oh, have we had any hedgehogs? No, but you've had three neighborhood cats.*" (T5).

Teachers related how they could use MicroCode in computing lessons to **foster the development and practice of soft skills (Theme 5)** such as collaboration, empathy, and imagination. Moreover, they found that MicroCode provides an opportunity to organize activities in class that could **increase students' interest in using computing to extend learning (Theme 6)**. For example, one teacher described how MicroCode could be used to build STEM skills on top of concepts that they have worked on in **other subjects**: "*So [they could code an interactive scene or simple game] changing the background, making the character do something e.g., having a sound... and then linking it to what we're doing [in other subjects] like prehistoric Britain. We might link it to dinosaurs.*" (T2) and "*In terms of the curriculum, as it stands, there's loads of the computing stuff that it does and so you would be able to use it with [other subjects, like science]*" (T5).

Building on this, teachers shared how they envisioned how, by using MicroCode, physical computing can become a knowledge vehicle **to tie together cross-curriculum concepts (Theme 7)** beyond "*the most direct links (math, science, design and technology)*" (T3) or "*to develop a water meter or an electric guitar*" (T1). T3 expressed interest in conducting an activity that could be related to climate change with MicroCode. And finally, T1 and T3 proposed the idea of connecting physical computing and science: "*I'm instantly thinking science. Yes. Temperatures, tilts. If you can measure the tilt, you could do that in like, friction. Like see how tilted the... Yeah, but the bodies and how fast the car goes*" (T3). The Y2 teachers believed that "*Y2 would be perfect for this device*" (T4). Indeed, MicroCode was well perceived by beginners "*for someone like me who has no experience teaching coding at all. I felt like I could grasp that*" (T4). Regarding this age group they also mentioned that MicroCode could be a useful for cross-discipline work "*Well certainly math. There's a lot of math with position and direction going on.*"(T4).

Table 3. Results from the engagement survey given to children.

Question	Component	Averaged Response	Level of Engagement
1	Aesthetic and sensory appeal	3.56	Moderate
2		3.03	Moderate
3	Challenge	3.13	Moderate
4		3.3	Moderate
5	Endurability	3.71	High
6		3.65	High
7	Feedback	3.2	Moderate
8		3.2	Moderate
9	Interest	3.8	High
10		3.6	High
11	Novelty	3.71	High
12	Perceived user control	3.42	Moderate
13		3.52	Moderate
14a	Learning goals	3.3	Moderate
14b		3.33	Moderate
14c		3.5	Moderate
14d		3.3	Moderate

5.2 RQ2: How does MicroCode create engagement and a sense of agency with young children?

To address this question, we used two sets of data: the user experience survey filled in by children at the end of the activity in class, and the researchers' notes on the activity.

5.2.1 Engagement Survey. The results from the engagement survey will be discussed in-line with the components that each question was designed to elicit – following the Giggle Gauge survey, adapted from [15]. Some examples of statements which children scored include: '*I liked how MicroCode looked'* and *'I had control over what I was doing with MicroCode'* which were designed to evaluate the *aesthetic/sensory appeal* of MicroCode and *perceived user control*, respectively. The results are summarized in Table 3 and more detail including the specific questions appears in Appendix A. Children's responses to all factors investigated indicated either high or moderate engagement, so overall, they clearly had a positive experience. MicroCode had a strong, positive reception with respect to endurability, interest and novelty. For the remaining components, all were found to elicit moderate level of engagement. The results for perceived user control indicate that users felt a sense of agency over their MicroCode experience.

5.2.2 Observational Data. We report the main findings of our thematic analysis of the data via representative quotes from children and researchers that were transcribed by the researchers during the observation. Our intent is to provide an overview of children's experiences in using MicroCode in the classroom. We have anonymized children's quotes and report them by using an identification number (IDn); similarly for the researchers who observed the activity (On). Bold statements below refer to the themes that emerged from the analysis.

With MicroCode, children demonstrated a **high level of agency (Theme 1):** they felt able to define their own activities, pursue their own objectives, and felt ownership of the contents they produced. They clearly owned the use of the tool and the tasks: "*This is our music, listen!*" (ID25) and "*I like how there's more choices, you can design your own* *thing*" (ID19). Indeed, children had more time to explore and create as reported by one of the observers: "*More kids are taking the lead and changing their designs (sending hearts instead of smiles)*" (O2).

Overall children were highly **motivated and engaged (Theme 2)** in using MicroCode, with a high level of awareness: "*I loved this lesson because I have not experienced this type of coding"* (ID7). They were extremely excited about learning a new tool that allowed them to express themselves: "*It's fun to make the different drawings and different sounds*" (ID2). The level of children's **agency and engagement** in accomplishing the tasks was demonstrated also by the fact that some of them finished earlier and immediately asked to do more advanced activities:

"*I loved this lesson. I want one of them and I could be on it for hours! I just wish you could change the colour of the lights, if there were other buttons to mess with... But I still LOVE IT. Thank you for doing this lesson with us. I really enjoyed using the device.*" (ID7)

Moreover, children were engaged physically which contributed to create a high sense of ownership towards the tool: "*I want to bring the controller with me. I need to see my code. . . I don't want to leave it behind"* (ID8), "*With this, you can hold it and work with it better when moving around"* (ID4).

MicroCode allowed children to be **focused on content creation (Theme 3):** *"I liked MicroCode because you could do lots of really cool things like making music and saying things"* (ID6), *"I like how you can make the sounds"* ID9, and "*I am good at coding; I could do a heart even if I am not good at drawing a heart.*" (ID11). Aligned with that, they perceived **how the lesson relates to other subjects (Theme 4)** and they appreciated that it: "*made maths more fun.*" (ID3). Children resonated with the physicality of the device, especially the fact that they could **easily handle it and move it around (Theme 5)**:*"Easy to handle"*(ID2*), "I like how you can just shake it"* (ID5)*, "With this, you can hold it and work with it better when moving around"* (ID4).

Among the most welcomed features of MicroCode was **the autonomy and freedom (Theme 6)** of being disconnected from a computer. One observer noted that "*students traveled around the school in pairs, collecting the temperature from different areas"* (O3). Children also remarked that *"Removing the [host] computer is good"* (ID24) and *"It was really fun making patterns and I loved the bigger screen and I think the battery pack is cool for coding because, you can make it and use the MicroCode portable*" (ID17).

5.3 RQ3: How does MicroCode enable young children to tackle introductory programming tasks?

To answer this research question we report on examples of ease-of-use that emerged from the study as these are essential for understanding whether MicroCode would be suitable for introducing children to programming tasks.

Our analysis highlighted that children quickly understood the MicroCode conceptual model. Multiple aspects contributed to this. First, the **transparency of the user interface** and **intuitive nature** of the visual programming environment allowed children to easily understand how to interact with the device: *"It was simple: I clicked on movement then I saw tilt down, and instead of deciding to put an image, I put a number, and now it you tilt it down there's a number on the screen" (ID17).* It also helped children understand the function of each tile and command: *'Students broadly say the pictures helped with the programming'* (O2). Children liked it as they could **see the icons and the workflow**: "*The resolution of the screen was good. Easy to read and understand icons"* (ID10)*,* "*I made a whole sentence at the top, super easy you just need to click on the icons to make the letter*" (ID18).

The MicroCode interface stimulates a **sense of familiarity** with other coding tools children may have used such as Scratch, and facilitated their interaction with it: "*I like how you can make images like in Scratch*" (ID9). Children perceived MicroCode in terms of **block programming languages**: "*[MakeCode] is like Scratch; I preferred [MicroCode] as it's like a game controller"* (ID15)*, "It was more like Scratch than I thought, but it was fun doing something different!"* (ID6).

In addition to this **familiarity of the MicroCode interaction model**, children readily understood the MicroCode physical controls due to their **similarity** to game controllers, which suggested to them the correct way to use the device: "*It looks like an Xbox Controller!"* (ID14), "*It looks like a Gameboy"* (ID7). On the other hand, we must also note that **familiarity can occasionally be misleading,** suggesting ways of use that are wrong both in relation to coding or physical interaction: "*I find it confusing because it's a bit harder than Scratch"* (ID9), and an observer noting that *"one person was trying to use its screen as a touch screen initially"* (O1).

Overall, MicroCode appeared to allow children to shift their attention from "**learning to code" to "creating content"**: *"I liked this MicroCode because you could do lots of really cool things like making music and saying things."* (ID6). With MicroCode, learning to code became a transparent part of the process of creating content, and the latter was the focus: *"I liked this MicroCode because you could do lots of really cool things like making music and saying things"* (ID6).

6 DISCUSSION: MEANINGFUL LEARNING WITH MICROCODE

In this section, we discuss the lessons learned and the implications of our findings for the design of physical computing activities in the classroom. By triangulating data collected during our interviews and evaluation it became clear how MicroCode could be used in a physical computing class.

6.1 Portability and the broader curriculum

In the literature it is well known that the manipulation of tangible objects and embodied interactions supports a hands-on approach to skill development and frees up the student's valuable cognitive resources during problem-solving [43]. Similarly, receiving immediate feedback from actions is a valuable aspect of live programming [72] and has a positive impact on discovery [21]. As mentioned in the related work section, the mobile/portable aspect of MicroCode is unique in the panorama of physical computing tools for education [45, 78]. The flexibility of MicroCode's portable programming environment offers numerous benefits and facilitates the development of educational activities rooted in situated learning [11], wherein children can apply and practice their knowledge across various contexts. With MicroCode children can create, view, and edit code with the MicroCode shield; this extends the hands-on approach of physical computing by allowing them to "**code on the go**" e.g. *"With this, you can hold it and work with it better when moving around"* (ID4). Thus, its portability enabled children to perform tasks while freely moving within the school space, seamlessly transitioning between indoor and outdoor settings. This approach also encouraged the design of more immersive and active assignments, also fostering the development of kinesthetic abilities.

Moreover, MicroCode's portability has a relevance also to **the broader curriculum**. In the interviews teachers showed how MicroCode could be beneficial for teaching a broad spectrum of subjects, beyond computing and technology classes.

Teachers possess backgrounds and competencies not only in computing but also in other disciplines. This is quite common among primary school teachers, enabling them to recognize potential interconnections between subjects. Thus, they also highlighted cross-curricular learning opportunities that could be enabled by physical computing. Regarding the latter, one teacher made a very good example, proposing to use MicroCode to increase students' awareness on sustainability and climate change concepts by monitoring the school temperature. This is a perfect example of a **cross-curricular activity** as it includes the subject of science (the measurement of the temperature), computing (the design and development of code), design and technology (problem solving, creative thinking), and relationships education (the activity was done in pairs to train children to learn how to be collaborative). This aligns well with the primary school practice of integrating different subjects within a thematic learning experience. Much existing research into the use of physical computing in educational contexts has focused on designing educational materials to support the teaching of STEM-related concepts, such as: assembling hardware elements to create interactive objects [19], learning to program video games and debug the code [54], and understanding how hardware and software come together with IoT [11]. In addition, teachers suggested that MicroCode could help to scaffold the **learning of 'abstract' concepts** such as sustainability, ethics, or interpersonal skills with **experiential educational activities**. This further aligns with the literature, as physical computing has been shown to facilitate the learning of abstract concepts and metaphors through their embodiment into physical activity [2, 33]. Unlike other tools, the portability offered by MicroCode allows children to move seamlessly within spaces and code within context, supporting situated learning. This broadens the range of activities that can be undertaken and greatly enhances the opportunity for teachers to create cross-curricular tasks which are beneficial for the children.

6.2 The conceptual model familiarity and intuitiveness lead to children tackling programming tasks.

In our study many of the children cited the **familiarity of the interaction model**. Initially relating to the physical similarity of MicroCode to widely used gaming controllers *"It looks like an Xbox Controller!"* (ID14). Previous work has shown that the micro:bit has an easy learning curve regardless of previous STEM knowledge [53, 54, 71]. But, by providing additional hardware features that leverage children's previous experiences, MicroCode further facilitates children's understanding of the conceptual model. Another quality where **familiarity of the interaction model** proved valuable was MicroCode's tile-based icon programming model, as children could leverage their previous experience with other beginner friendly **block-based programming** environments like Scratch and Purple Mash -*"It was more like Scratch than I thought, but it was fun doing something different!"* (ID6). Additionally, the programming model proved to be **intuitive** with children happy with the transparency of the experience in they could clearly see the programs they had created on the device. Previous work has shown how important promoting aspects like **familiarity** and reinforcing existing conceptual models are in teaching children programming concepts [8, 17, 60]. Building on previous work, the design of MicroCode aimed to provide children with a tool whose conceptual model was easy to understand, enabling them to interact with the content and perform tasks easily. Our findings demonstrated how MicroCode successfully fulfilled this intention, as children were able to quickly grasp its functions and effectively use it to tackling programming tasks just after a brief introduction from the teacher.

6.3 Promoting agency, engagement and active learning with MicroCode

The results from the engagement survey and observational data both show how MicroCode promoted several different qualities beneficial to the learning experience. Firstly, with respect to agency, there were many instances where children extended the tasks independently to create their own programs and explore the features that the micro:bit and MicroCode provide, a process through which they responded positively saying things like *"I like how there's more choices, you can design your own thing"* (ID19) and *"I like how you can make the sounds"* (ID9). In many instances this agency fostered confidence in the child's own programming skills: *"I am good at coding, I could do a heart even if I am not good at drawing a heart"* (ID23). These behaviors are effective in encouraging long-term learning outcomes of children, as well as providing a sense of ownership over their own achievements [39].

Our analysis also demonstrates that MicroCode facilitated active learning [77], with students working through concepts by actively experimenting and working through challenges faced. Moreover, children discussed potential solution, producing more contents and in some cases going beyond the assignment. Literature shows how this type of activities can stimulate the development of **critical thinking** skills, which are central to providing well-rounded student outcomes [75] as well as fostering creativity and wider engagement [54, 65]. On that point, MicroCode also promotes a **cognitive engagement**. As shown by the survey results children are more likely to return for repeated interactions (high level of endurability). This is also represented in observational data, where children extended their work beyond the set tasks. Moreover, the portability of the MicroCode shield with MicroCode enabled **physical engagement** and **sense of ownership**, *"I want to bring my controller with me"* (ID8). The small size of the hardware also encouraged the **distribution of tasks** among the children which also a sign of enabling a certain level of collaboration within children. In the findings we highlight also how these factors contribute to children being able to shift from **"learning to code"** to**"creating content".** MicroCode was flexible enough **to stimulate children's creativity in expressing themselves using** multimedia. This aligns with the literature that shows how physical computing, through requiring children to create objects and content, can foster imagination and creativity [75], and improve their intrinsic motivation [54] in addition to learning coding.

In our study, we have demonstrated how MicroCode enables these factors that are interrelated and can positively influence each other, aligning with recent research indicating the inter-connectedness of these learning theories [68]. This positive influence contributes significantly to creating a meaningful learning experience. Other learning tools enable similar experience and share similar traits enabled by MicroCode [8, 33, 65]. However, our literature review reveals a significant gap addressed by MicroCode, which incorporates live programming, visual coding, and portability. The uniqueness of MicroCode lies in its ability to bridge gaps in previous works and illustrate how these elements can be implemented in a unique artifact designed for children, enhancing a meaningful learning experience in primary education.

7 CONCLUSION AND FUTURE WORK

Since the introduction of computer science and computational thinking into school curricula, various methodologies have been implemented to design meaningful learning experiences for children. A significant challenge in recent decades has been merging the programming aspect of physical computing (typically done using a laptop or desktop) with the need to make activities lively and quickly observable in context.

Our research focuses on challenging this issue by converging programming physical computing [26] with constructionist learning pedagogy [51]. We aim to empower children to tackle introductory programming tasks by developing their own ideas and creating working systems, while also supporting teachers and educators in developing programming educational activities using physical computing. In our paper, we introduced MicroCode, a unique live and portable micro:bit programming experience that eliminates the need for a separate host computer. MicroCode promotes an educational experience that doesn't limit children to what they imagine is possible.

We present a study in which we collected and presented evidence about the benefits and shortcomings of MicroCode via a study; we engaged 5 teachers who work across ages 4-11, and 60 children who were 10 and 11 years old. By triangulating data collected in our user survey, our observation notes, and interviews with teachers, we saw how MicroCode can create real added value for both children and teachers in working with and learning about physical computing. The evidence we produced showed how MicroCode helped to increase the agency and the engagement of children. This positive impact is in part also due to the ease with which children can understand MicroCode's conceptual model, lowering the barrier to use. MicroCode was also flexible in supporting different contexts and showed a strong added value in "code-on-the-go" activities. These qualities unlock new learning opportunities and experiences that extend beyond STEM subjects and across the curriculum.

Leveraging these findings, future work will be addressed in two main directions: additional studies and improving the design of MicroCode.

We will run further fine-grained studies to capture more specific aspects of children's experiences using MicroCode. Following the teachers' wishes, we intend to conduct a study with younger children (Y2, aged 6). The teachers strongly believed that there is potential for MicroCode to be suitable for a younger age group as it is based on icons that could be easily understood by children that are still learning to read. This will allow us to explore if the icon-based programming model of MicroCode supports students whose literacy is still developing in a way that more traditional text-based programming models cannot.

Additionally, we will use the outcomes of the survey and the evidence from the analysis of the observation of children in class to improve the design of MicroCode. The survey provided a solid basis in terms of children's self-perception of MicroCode that, combined with the evidence from the observation, gave a detailed and fine-grained panorama of the experience that MicroCode could generate in a classroom context. Several design limitations of MicroCode emerged during the study. Firstly, the study highlighted that the design of MicroCode hardware does not really support two or more children using it, we wish to explore how a larger screen might promote easier collaboration between students. Secondly, physical computing often allows students to create new capabilities through the addition of modular hardware (such as new sensors, motors and actuators); this is known to further promote creativity within the learning experience.

8 SELECTION AND PARTICIPATION OF CHILDREN

Children and teachers were approached by the authors for recruitment from three local schools in the UK. The schools' directors asked the teachers to be involved on a voluntary basis and the parents or guardian's permission to involve their children was sought via a consent form. Only the children with a signed consent form participated in the study. Each participant could withdraw the study at any point and all the collected data will be deleted. The study has been approved by our University's ethical approval panel.

ACKNOWLEDGMENTS

We would like to thank all of the children, teachers, parents, and guardians that took part in this research. We are grateful to Hannah Whitham and Claudia Daudén Roquet for their help in the study. We also would like to acknowledge the Micro:bit Educational Foundation and the Microsoft MakeCode team for their support of this work.

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 UK to the world. *Commun. ACM* 63 (2 2020), 62–69. Issue 3. https://doi.org/10.1145/3368856

[2] Thomas Ball, Shannon Kao, Richard Knoll, and Daryl Zuniga. 2020. TileCode: Creation of Video Games on Gaming Handhelds. *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, 1182–1193. https://doi.org/10.1145/3379337.3415839

[3] Ayah Bdeir. 2009. Electronics as material: LittleBits. *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, 397–400. https://doi.org/10.1145/1517664.1517743

[4] Madhav Biros, David Ayyalasomayajula, Surya Dalal, and Nikunj Sharma. 2020. Teaching programming to the post-millennial generation: Pedagogic considerations for an is course. *Journal of Information Systems Education* 31 (6 2020). Issue 2. https://jise.org/Volume31/n2/JISEv31n2p96.html

[5] Paulo Blikstein, Arnan Sipitakiat, Jayme Goldstein, João Wilbert, Maggie Johnson, Steve Vranakis, Zebedee Pedersen, and Will Carey. 2016. Project Bloks: designing a development platform for tangible programming for children. *Position paper, retrieved online on* (2016), 06–30.

[6] Monique Boekaerts. 2016. Engagement as an inherent aspect of the learning process. *Learning and Instruction* 43 (2016), 76–83. https://doi.org/10.1016/j.learninstruc.2016.02.001

[7] Naomi R. Boyer, Sara Langevin, and Alessio Gaspar. 2008. Self direction; constructivism in programming education. *Proceedings of the 9th ACM SIGITE conference on Information technology education*, 89–94. https://doi.org/10.1145/1414558.1414585

[8] Anke Brocker, René Schäfer, Christian Remy, Simon Voelker, and Jan Borchers. 2023. Flowboard: How Seamless, Live, Flow-Based Programming Impacts Learning to Code for Embedded Electronics. *ACM Transactions on Computer-Human Interaction* 30 (2 2023), 1–36. Issue 1. https://doi.org/10.1145/3533015

[9] Joshua Chan, Tarun Pondicherry, and Paulo Blikstein. 2013. LightUp: An Augmented, Learning Platform for Electronics. *Proceedings of the 12th International Conference on Interaction Design and Children*, 491–494. https://doi.org/10.1145/2485760.2485812

[10] Patricia Charlton and Katerina Avramides. 2016. Knowledge Construction in Computer Science and Engineering when Learning Through Making. *IEEE Transactions on Learning Technologies* 9 (10 2016), 379–390. Issue 4. https://doi.org/10.1109/TLT.2016.2627567

[11] Paul Cobb and Janet Bowers. 1999. Cognitive and Situated Learning Perspectives in Theory and Practice. *Educational Researcher* 28 (1999), 4–15. Issue 2. https://doi.org/10.3102/0013189X028002004

[12] Bettina Conradi, Verena Lerch, Martin Hommer, Robert Kowalski, Ioanna Vletsou, and Heinrich Hussmann. 2011. Flow of electrons: an augmented workspace for learning physical computing experientially. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces* (Kobe, Japan) *(ITS '11)*. Association for Computing Machinery, New York, NY, USA, 182–191. https://doi.org/10.1145/2076354.2076389

[13] Stephen Coy. 2013. Kodu game lab, a few lessons learned. *XRDS: Crossroads, The ACM Magazine for Students* 19 (6 2013), 44–47. Issue 4. https://doi.org/10.1145/2460436.2460450

[14] James Devine, Joe Finney, Peli de Halleux, Michał Moskal, Thomas Ball, and Steve Hodges. 2018. MakeCode and CODAL: intuitive and efficient embedded systems programming for education. *Proceedings of the 19th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems*, 19–30. https://doi.org/10.1145/3211332.3211335

[15] Griffin Dietz, Zachary Pease, Brenna McNally, and Elizabeth Foss. 2020. Giggle Gauge: A Self-Report Instrument for Evaluating Children's Engagement with Technology. *Proceedings of the Interaction Design and Children Conference*, 614–623. https://doi.org/10.1145/3392063.3394393

[16] Stefania Druga, Thomas Ball, and Amy Ko. 2022. How families design and program games: a qualitative analysis of a 4-week online in-home study. *Interaction Design and Children*, 237–252. https://doi.org/10.1145/3501712.3529724

[17] Caitlin Duncan, Tim Bell, and Steve Tanimoto. 2014. Should your 8-year-old learn coding? *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*, 60–69. https://doi.org/10.1145/2670757.2670774

[18] Grace Fayombo. 2012. Active Learning: Creating Excitement and Enhancing Learning in a Changing Environment of the 21st Century. *Mediterranean Journal of Social Sciences* 3 (9 2012), 107 – 128. https://doi.org/10.5901/mjss.2012.v3n16p107

[19] Phil Frei, Victor Su, Bakhtiar Mikhak, and Hiroshi Ishii. 2000. Curlybot: designing a new class of computational toys. *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, 129–136. https://doi.org/10.1145/332040.332416

[20] Anna Gardeli and Spyros Vosinakis. 2019. ARQuest: A Tangible Augmented Reality Approach to Developing Computational Thinking Skills. *2019 11th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*, 1–8. https://doi.org/10.1109/VS-Games.2019.8864603

[21] Alessio Gaspar and Sarah Langevin. 2007. Active learning in introductory programming courses through student-led "live coding" and test-driven pair programming. In *International Conference on Education and Information Systems, Technologies and Applications, Orlando, FL.*

[22] Rosella Gennari, Alessandra Melonio, Mehdi Rizvi, and Andrea Bonani. 2017. Design of IoT Tangibles for Primary Schools. *Proceedings of the 12th Biannual Conference on Italian SIGCHI Chapter*, 1–6. https://doi.org/10.1145/3125571.3125591

[23] Yen Air Caballero González and Ana García-Valcárcel Muñoz-Repiso. 2018. A robotics-based approach to foster programming skills and computational thinking. *Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality*, 41–45. https://doi.org/10.1145/3284179.3284188

[24] Luisa Greifenstein, Isabella Graßl, Ute Heuer, and Gordon Fraser. 2022. Common Problems and Effects of Feedback on Fun When Programming Ozobots in Primary School. *Proceedings of the 17th Workshop in Primary and Secondary Computing Education*, 1–10. https://doi.org/10.1145/3556787.3556860

[25] Steve Hodges, James Scott, Sue Sentance, Colin Miller, Nicolas Villar, Scarlet Schwiderski-Grosche, Kerry Hammil, and Steven Johnston. 2013. .NET gadgeteer. *Proceeding of the 44th ACM technical symposium on Computer science education*, 391–396. https://doi.org/10.1145/2445196.2445315

[26] 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. Issue 4. https://doi.org/10.1109/MC.2019.2935058

[27] Steve Hodges, Nicolas Villar, Nicholas Chen, Tushar Chugh, Jie Qi, Diana Nowacka, and Yoshihiro Kawahara. 2014. Circuit stickers: Peel-and-Stick Construction of Interactive Electronic Prototypes. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1743–1746. https://doi.org/10.1145/2556288.2557150

[28] 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. Issue 4. https://doi.org/10.1007/s00779-011-0404-2

[29] Michael S. Horn and Robert J. K. Jacob. 2007. Designing tangible programming languages for classroom use. *Proceedings of the 1st international conference on Tangible and embedded interaction*, 159–162. https://doi.org/10.1145/1226969.1227003

[30] Felix Hu, Ariel Zekelman, Michael Horn, and Frances Judd. 2015. Strawbies. *Proceedings of the 14th International Conference on Interaction Design and Children*, 410–413. https://doi.org/10.1145/2771839.2771866

[31] Christopher D. Hundhausen and Jonathan L. Brown. 2007. What You See Is What You Code: A "live" algorithm development and visualization environment for novice learners. *Journal of Visual Languages & Computing* 18 (2 2007), 22–47. Issue 1. https://doi.org/10.1016/j.jvlc.2006.03.002

[32] James A. Jerkins, Yasmeen Rawajfih, and Cynthia L. Stenger. 2022. Exploring the impact of CS makers on teachers in Alabama. *Proceedings of the ACM Southeast Conference*, 201–205. https://doi.org/10.1145/3476883.3520219

[33] K Kahn, R Megasari, E Piantari, and E Junaeti. 2018. AI programming by children using Snap! block programming in a developing country. *Thirteenth European Conference on Technology Enhanced Learning* 11082.

[34] Sema A. Kalaian and Rafa M. Kasim. 2015. *Small-Group vs. Competitive Learning in Computer Science Classrooms.* 46–64. https://doi.org/10.4018/978-1-4666-7304-5.ch003

[35] Sara Kalantari, Elisa Rubegni, Laura Benton, and Asimina Vasalou. 2023. "When I'm writing a story, I am really good" Exploring the use of digital storytelling technology at home. *International Journal of Child-Computer Interaction* (2023), 100613.

[36] Filiz Kalelioglu and Yasemin Gulbahar. 2014. The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective. *Informatics in Education* 13 (4 2014), 33–50.

[37] Eva-Sophie Katterfeldt, Mutlu Cukurova, Daniel Spikol, and David Cuartielles. 2018. Physical computing with plug-and-play toolkits:Key recommendations for collaborative learning implementations. *International Journal of Child-Computer Interaction* 17 (9 2018), 72–82. https://doi.org/10.1016/j.ijcci.2018.03.002

[38] Eva-Sophie Katterfeldt, Nadine Dittert, and Heidi Schelhowe. 2015. Designing Digital Fabrication Learning Environments for Bildung. *Int. J. Child-Comp. Interact.* 5 (9 2015), 3–10. Issue C.

[39] Greg Kearsley and Ben Shneiderman. 1998. Engagement Theory: A Framework for Technology-Based Teaching and Learning. *Educational Technology* 38 (1998), 20–23. Issue 5. http://www.jstor.org/stable/44428478

[40] David Kolb. 1984. *Experiential Learning: Experience As The Source Of Learning And Development.* Vol. 1.

[41] Tiina Korhonen, Laura Salo, and Kati Sormunen. 2019. Making with Micro:bit: Teachers and Students Learning 21st Century Competences through the Innovation Process. *Proceedings of FabLearn 2019*, 120–123. https://doi.org/10.1145/3311890.3311906

[42] Ilya Levin and Dina Tsybulsky. 2017. The Constructionist Learning Approach in the Digital Age. *Creative Education* 08 (2017), 2463–2475. Issue 15. https://doi.org/10.4236/ce.2017.815169

[43] Matthew B. MacLaurin. 2011. The design of kodu. *ACM SIGPLAN Notices* 46 (1 2011), 241–246. Issue 1. https://doi.org/10.1145/1925844.1926413

[44] Gayathri Manikutty. 2021. My Robot can tell stories: Introducing robotics and physical computing to children using dynamic dioaramas. *2021 IEEE Frontiers in Education Conference (FIE)*, 1–9. https://doi.org/10.1109/FIE49875.2021.9637460

[45] TimothyS. McNerney. 2004. From turtles to Tangible Programming Bricks: explorations in physical language design. *Personal and Ubiquitous Computing* 8 (9 2004). Issue 5. https://doi.org/10.1007/s00779-004-0295-6

[46] Timothy McNerney. 2011. Tangible Programming Bricks: An approach to making programming accessible to everyone. (4 2011).

[47] Orni Meerbaum-Salant, Michal Armoni, and Mordechai (Moti) Ben-Ari. 2013. Learning computer science concepts with Scratch. *Computer Science Education* 23 (9 2013), 239–264. Issue 3. https://doi.org/10.1080/08993408.2013.832022

[48] Oussama Metatla, Sandra Bardot, Clare Cullen, Marcos Serrano, and Christophe Jouffrais. 2020. Robots for Inclusive Play: Co-designing an Educational Game With Visually Impaired and sighted Children. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–13. https://doi.org/10.1145/3313831.3376270

[49] Michal Moskal, Thomas Ball, Abhijith Chatra, James Devine, Peli de Halleux, Steve Hodges, Shannon Kao, Richard Knoll, Galen Nickel, Jacqueline Russell, Joey Wunderlich, and Daryl Zuniga. 2021. Web-based Programming for Low-cost Gaming Handhelds. *The 16th International Conference on the Foundations of Digital Games (FDG) 2021*, 1–12. https://doi.org/10.1145/3472538.3472572

[50] Heather L. O'Brien and Elaine G. Toms. 2008. What is user engagement? A conceptual framework for defining user engagement with technology. *Journal of the American Society for Information Science and Technology* 59 (4 2008), 938–955. Issue 6. https://doi.org/10.1002/asi.20801

[51] Seymour Papert and Idit Harel. 1991. Situating constructionism. *constructionism* 36, 2 (1991), 1–11.

[52] Amanda J. Parkes, Hayes Solos Raffle, and Hiroshi Ishii. 2008. Topobo in the wild. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1129–1138. https://doi.org/10.1145/1357054.1357232

[53] Radia Perlman. 2004. Using Computer Technology to Provide a Creative Learning Environment for Preschool Children. (10 2004).

[54] Mareen Przybylla and Ralf Romeike. 2018. Impact of Physical Computing on Learner Motivation. *Proceedings of the 18th Koli Calling International Conference on Computing Education Research*, 1–10. https://doi.org/10.1145/3279720.3279730

[55] Janet Read, Stuart MacFarlane, and Christopher Casey. 2009. Endurability, Engagement and Expectations: Measuring Children's Fun. *Interaction Design and Children* (9 2009).

[56] Patrick Rein, Stefan Ramson, Jens Lincke, Robert Hirschfeld, and Tobias Pape. 2018. Exploratory and Live, Programming and Coding. *The Art, Science, and Engineering of Programming* 3 (7 2018). Issue 1. https://doi.org/10.22152/programming-journal.org/2019/3/1

[57] Mitchel Resnick. 1996. Distributed Constructionism. *Proceedings of the 1996 International Conference on Learning Sciences*, 280–284.

[58] Anthony Robins, Janet Rountree, and Nathan Rountree. 2003. Learning and Teaching Programming: A Review and Discussion. *Computer Science Education* 13 (6 2003), 137–172. Issue 2. https://doi.org/10.1076/csed.13.2.137.14200

[59] Elisa Rubegni, Monica Landoni, Laura Malinverni, and Letizia Jaccheri. 2022. Raising Awareness of Stereotyping Through Collaborative Digital Storytelling: Design for Change with and for Children. *International Journal of Human-Computer Studies* 157 (2022), 102727. https://doi.org/10.1016/j.ijhcs.2021.102727

[60] Theodosios Sapounidis and Stavros Demetriadis. 2013. Tangible versus graphical user interfaces for robot programming: exploring cross-age children's preferences. *Personal and Ubiquitous Computing* 17 (12 2013), 1775–1786. Issue 8. https://doi.org/10.1007/s00779-013-0641-7

[61] Albrecht Schmidt. 2016. Increasing Computer Literacy with the BBC micro:bit. *IEEE Pervasive Computing* 15 (4 2016), 5–7. Issue 2. https://doi.org/10.1109/MPRV.2016.23

[62] Eric Schweikardt. 2010. Modular robotics studio. *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*, 353–356. https://doi.org/10.1145/1935701.1935784

[63] Eric Schweikardt and Mark D. Gross. 2006. roBlocks. *Proceedings of the 8th international conference on Multimodal interfaces*, 72–75. https://doi.org/10.1145/1180995.1181010

[64] Ana Selvaraj, Eda Zhang, Leo Porter, and Adalbert Gerald Soosai Raj. 2021. Live Coding: A Review of the Literature. *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, 164–170. https://doi.org/10.1145/3430665.3456382

[65] Sue Sentance, Jane Waite, Steve Hodges, Emily MacLeod, and Lucy Yeomans. 2017. "Creating Cool Stuff": Pupils' Experience of the BBC micro:bit. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 531–536. https://doi.org/10.1145/3017680.3017749

[66] Jia-Sin Tan Sheng-Rong Lin, Yu-Tzu Liou. 2016. Teaching by demonstration: programming instruction by using live-coding videos. *EdMedia 2016–World Conference on Educational Media and Technology*, 1294–1298. https://www.learntechlib.org/primary/p/173121/

[67] Kathryn T. Stolee and Teale Fristoe. 2011. Expressing computer science concepts through Kodu game lab. *Proceedings of the 42nd ACM technical symposium on Computer science education*, 99–104. https://doi.org/10.1145/1953163.1953197

[68] Claudia Szabo and Judy Sheard. 2023. Learning Theories Use and Relationships in Computing Education Research. *ACM Transactions on Computing Education* 23 (3 2023), 1–34. Issue 1. https://doi.org/10.1145/3487056

[69] David R Thomas. 2003. A general inductive approach for qualitative data analysis. (2003).

[70] David R Thomas. 2006. A General Inductive Approach for Analyzing Qualitative Evaluation Data. *American Journal of Evaluation* 27 (2006), 237–246. Issue 2. https://doi.org/10.1177/1098214005283748

[71] Ethel Tshukudu, Quintin Cutts, Olivier Goletti, Alaaeddin Swidan, and Felienne Hermans. 2021. Teachers' Views and Experiences on Teaching Second and Subsequent Programming Languages. *Proceedings of the 17th ACM Conference on International Computing Education Research*, 294–305. https://doi.org/10.1145/3446871.3469752

[72] Bret Victor. 2012. LEARNABLE PROGRAMMING - Designing a programming system for understanding programs. http://worrydream.com/LearnableProgramming/

[73] Maja Videnovik, Eftim Zdravevski, Petre Lameski, and Vladimir Trajkovik. 2018. The BBC Micro:bit in the International Classroom: Learning Experiences and First Impressions. *2018 17th International Conference on Information Technology Based Higher Education and Training (ITHET)*, 1–5. https://doi.org/10.1109/ITHET.2018.8424786

[74] Peter Vinnervik. 2023. Programming in school technology education: the shaping of a new subject content. *International Journal of Technology and Design Education* 33 (9 2023), 1449–1470. Issue 4. https://doi.org/10.1007/s10798-022-09773-y

[75] Jeannette M. Wing. 2006. Computational thinking. *Commun. ACM* 49 (3 2006), 33–35. Issue 3. https://doi.org/10.1145/1118178.1118215

[76] Peta Wyeth and Helen C. Purchase. 2002. Tangible programming elements for young children. *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, 774–775. https://doi.org/10.1145/506443.506591

[77] Nesra Yannier, Scott E. Hudson, and Kenneth R. Koedinger. 2020. Active Learning is About More Than Hands-On: A Mixed-Reality AI System to Support STEM Education. *International Journal of Artificial Intelligence in Education* 30 (3 2020), 74–96. Issue 1. https://doi.org/10.1007/s40593-020-00194-3

[78] Junnan Yu and Ricarose Roque. 2018. A survey of computational kits for young children. *Proceedings of the 17th ACM Conference on Interaction Design and Children*, 289–299. https://doi.org/10.1145/3202185.3202738

A DETAILS OF THE ENGAGEMENT SURVEY GIVEN TO CHILDREN

Question		Component	Avg. Response	Engagement
1	I liked how the micro:bit looked	Aesthetic and sensory appeal	3.56	Moderate
2	I liked how the micro:bit felt		3.01	Moderate
3	Using the micro:bit was hard, in a good way	Challenge	3.13	Moderate
4	Programming the micro:bit was hard, in a good way		3.3	Moderate
5	I would like to use the micro:bit again sometime	Endurability	3.71	High
6	I would like to run this activity again sometime		3.65	High
8	The on-screen blocks let me know when I did something		3.2	Moderate
9	I enjoyed using the micro:bit	Interest	3.8	High
10	I enjoyed programming with the micro:bit		3.6	High
11	I found lots of things to do with the micro:bit	Novelty	3.71	High
12	I had control over what I was doing with the micro:bit	Perceived user control	3.42	Moderate
13	I had control over what I was doing with the on-screen blocks		3.52	Moderate
14	During this activity I have learnt:	Learning goals	N/A	
14a	How to collaborate better with my classmates		3.3	Moderate
14b	How to program		3.33	Moderate
14c	How it is important to regulate temperature in a room to save energy		3.5	Moderate
14d	How it is important to save energy for climate change		3.3	Moderate

B SAMPLE PROGRAMS USED BY TEACHERS

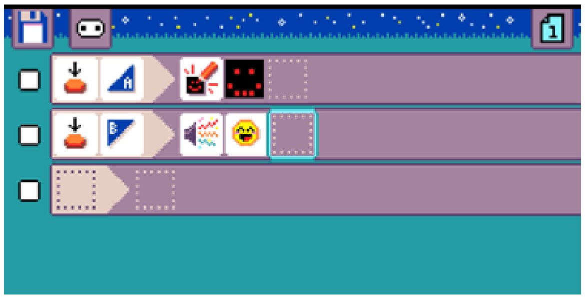
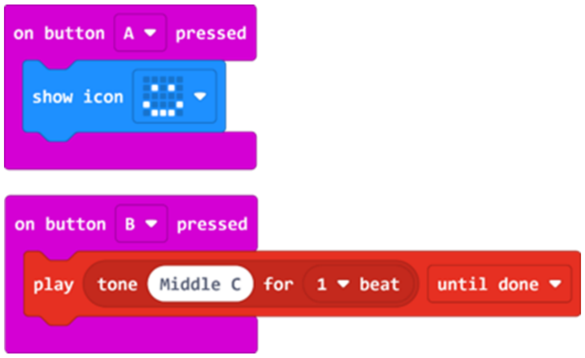


Fig. 7. A sample MicroCode program the teacher demonstrated to the students.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009



```
on button A pressed
    show icon
on button B pressed
    play tone Middle C for 1 beat until done
```

Fig. 8. A sample MakeCode program the teacher demonstrated to the students.