Challenge and Creativity: Using .NET Gadgeteer In Schools

Sue Sentance
Anglia Ruskin University
Chelmsford, UK
sue.sentance@anglia.ac.uk

Scarlet Schwiderski-Grosche
Microsoft Research Cambridge
Cambridge, UK
scarlets@microsoft.com

ABSTRACT

This paper reports on a study carried out in secondary schools in the UK with students learning to use .NET Gadgeteer, a rapid prototyping platform for building small electronic devices [32]. A case study methodology has been used. Some of the students involved in this four-month-long project had some prior background in computer programming whereas for others this was completely new. The teaching materials provided a two-phase model of learning: an instruction phase followed by a creative phase, the latter utilising a bricolage approach to learning programming [30]. The aim of the pilot was to generate an interest in building devices and stimulate creativity. The research found that the tangible nature of the .NET Gadgeteer modules helped to engage the students in becoming creative, and that students valued challenges with which they were not usually presented within the curriculum.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer Science Education,Curriculum

Keywords

Secondary education, .NET Gadgeteer, high schools, computer programming, creativity, bricolage

1. INTRODUCTION

Microsoft .NET Gadgeteer[1] is a platform that enables rapid prototyping of small electronic gadgets and embedded hardware devices. It combines the advantages of object-oriented programming, solderless assembly of electronics using a kit of hardware modules, and the quick fabrication of physical enclosures using computer-aided design. The fact that .NET Gadgeteer covers a variety of sophisticated computer science and engineering skills, but requires minimal prior knowledge, makes it especially suitable for education. .NET Gadgeteer has great potential in schools as it can be used to teach students simple electronics and computer programming as well as computer-aided design. Moreover, it is very motivating for young people to be able to build their own gadgets.

A research project was initiated to investigate whether .NET Gadgeteer has the potential to be used in schools. This was designed to elicit students' and teachers' perceptions of the platform, and suggest directions for further research and development. A case study methodology was used, as defined by Stake [26], whereby different sources were drawn together to give an accurate description of how the pilot use of this platform was perceived by both teachers, students and researchers.

This paper reports on the findings of this first study. In the paper, we will give an outline of the current situation with regard to Computer Science (CS) education in schools in the UK which provides some motivation for the current project. A brief description of .NET Gadgeteer will help to set the scene and the next section of the paper will focus on the methodology and findings of the research. The findings will be discussed and then further work suggested.

2. COMPUTER SCIENCE EDUCATION IN SCHOOLS

2.1 Background

A recent report by the Royal Society, an influential academic body in the UK, begins with the statement *"The current delivery of Computing education in many UK schools is highly unsatisfactory"* [29, p.1]. Prior to this report, there had been an increasing awareness of the need for more Computer Science in schools in England and Wales. The Computing At School group[2] (CAS) has been very active since 2008 in advocating the need for more Computer Science in the curriculum and supporting teachers on the ground [6].

Since the publication of the Royal Society report, the government has started to implement change in the curriculum; more qualifications are being launched to feed the need of schools and pupils to study this subject within the curriculum. The Royal Society states that *"Computer Science is sufficiently important and foundational that it should be recognised as a high status subject in schools, like mathematics, physics or history"* [29, p.34]. These developments in the UK mirror those that have been happening elsewhere

[1]http://netmf.com/gadgeteer

[2]http://www.computingatschool.org.uk

in the world, for example, as described by Wilson et al. in the USA's *Running on Empty* report produced by the CSTA [33], and as implemented in Israel [10].

It is apparent that there are many schools keen to offer Computing-related content to students aged 11-14 and also school qualifications in Computing/Computer Science to students aged 14-18. Within this climate, providing engaging resources and vehicles for learning Computer Science is very timely.

As the emphasis grows on secondary Computing education, more research in this area is needed. It is important to look at the pedagogical approaches to bridge the gap between computer literacy and Computer Science as well as what should be taught within the curriculum [3]. **Constructivist learning** theories applied to Computer Science emphasise the active, subjective and constructive character of knowledge, placing students at the centre of the learning process [13]. Specifically, constructivist learning is based on students' active participation in problem-solving and critical thinking regarding a learning activity which they found relevant and engaging [11]. As a learning theory, it has profoundly influenced the teaching of programming [2]. **Experiential learning** stems from constructivism and is a term which can be used to describe the design of activities which engage learners in a very direct way. It describes the process of engaging learners in an authentic experience in which they can make discoveries and experiment with knowledge at first hand. Through reflection, students construct new knowledge and ways of thinking about themselves, leading to deeper learning.

Research to date in secondary computing education has investigated a range of approaches to structuring the curriculum [27, 10, 1], whilst other research has focused on tools and environments that may motivate and engage young people in the classroom such as Scratch[17], Alice[5], Greenfoot[15], and Kodu[16]. It is important, however, to address the approach to learning that underlies these tools or environments. Scratch, for example, builds on work in Logo and on the constructionist ideas of Papert [17]. Papert used the term **constructionism** which is a combination of 'constructivism' and 'construction' [21]. This is particularly relevant for .NET Gadgeteer which involves physically constructing devices in an exploratory way.

2.2 Engaging students

There have been many discussions in the literature about how to engage students with Computer Science and suggestions for reasons why students do not have a positive attitude to the subject, for example [25, 9, 34, 4]. Downes and Looker suggest that the more IT students use at school, the more they are likely to take up computing-related subjects when they are given a choice [8]. Brinda, Puhlmann and Schulte discuss how to introduce Computer Science by working from what students already know from their ICT education and making Computer Science relevant to their own experience [3]. Pollock and Harvey integrated a range of pedagogical approaches in order to engage students more effectively and demonstrated the effectiveness of collaborative work and reflection on learning [22]. Collaborative working is especially important in our .NET Gadgeteer trials. Cutts, Esper and Simon discuss how to offer all students Computing education in relation to "what a computer can do and how one can interact with it" [7] by giving students an understanding that computers are deterministic, precise and comprehensible. This understanding can be gained without necessarily learning to program. With .NET Gadgeteer, we hope that we can engage all students in terms of a better understanding of how the devices and technology all around us works.

Schulte and Knobelsdorf look at attitudes towards Computer Science using a biographical approach, and note the differences between those that regard themselves as *insiders* and *outsiders* [25]. They recommend that teachers *"should intertwine introduction to CS (e.g. learning programming - a design activity) with learning professional use . . . a major problem is to teach another world-image of CS"* [p.37]. The interest of students in real devices and current technology makes .NET Gadgeteer quite engaging in this regard, as users can relate it to the professional world of developing devices.

2.3 Using tangible environments

Besides .NET Gadgeteer, other tangible devices are available which enable students to write programs using actual hardware components. These include Lego Mindstorms[3], the Scratch Pico Board[4], Open University‘s SenseBoard [23] and Arduino[5]. All these hardware kits have different features but offer the same experience of hardware in addition to software, thus broadening the exposure to the way that technology works. The Raspberry Pi[6] is now available which also has the appeal of being tactile and exposed; this has generated a lot of enthusiasm, although it is a computer rather than a means to build devices such as the other devices described here.

Marshall [18] and Horn et al. [12] both describe how tangible environments can have a very positive effect on collaborative and active learning, as they enable students to share work together in a very visible way. They also utilise concrete physical manipulation which can facilitate more effective or natural learning. Working with physical devices can encourage an exploratory or **bricolage** approach, as discussed next.

2.4 The bricolage approach

The concept of **bricolage** was introduced by Levi-Strauss in *The Savage Mind* [14]. It refers to a science of concrete development as an alternative to abstract planning, and was applied to the area of computer programming by Turkle and Papert [30]. It represents a mode of learning based on 'try-it-and-see-what-happens' [2], of which Ben-Ari is rather critical claiming that *"A student who exclusively uses such techniques is ultimately not qualified to work on the software of embedded and operating system, which requires the ability to create and test abstract hypotheses"* [2]. However, Stiller [28] successfully used a bricolage approach when teaching programming by encouraging students to build on previous programs, based on a pedagogy of incremental problem-solving. The intention of this project was to use a bricolage approach when introducing .NET Gadgeteer to students.

[3]http://mindstorms.lego.com
[4]http://wiki.scratch.mit.edu/wiki/PicoBoard
[5]http://www.arduino.cc/
[6]http://www.raspberrypi.org/

file=example.eps, height=1.3in, width=2.5in

Figure 1: .NET Gadgeteer modules

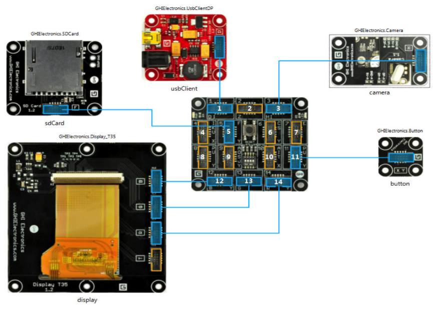


Figure 2: Designer view of a .NET Gadgeteer device

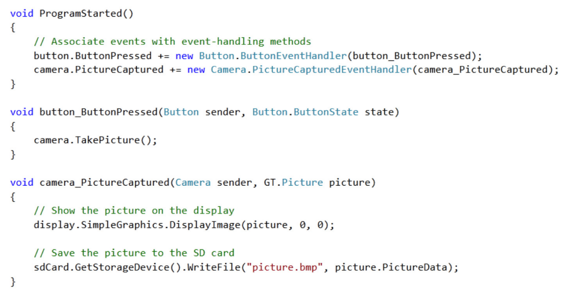
3. .NET GADGETEER

.NET Gadgeteer is a platform for creating your own electronic devices using a wide variety of hardware and a powerful programming environment [32]. .NET Gadgeteer hardware consists of an ever increasing range of mainboards and modules. The environment is open source and therefore, new modules can be developed by any enthusiast. Students with little or no Computing background can build robot-like devices made up of components that sense and react to their environment using switches, displays, motor controllers, and more. Components are plugged into a mainboard and subsequently programmed to make them work together (see Figure 1).

.NET Gadgeteer originated at Microsoft Research Cambridge, UK. It was designed as a tool for researchers to make it faster and easier to prototype new kinds of devices. For example, a digital camera can be built in about half an hour. One of the motivations for the Gadgeteer project has been to provide "*both a low threshold for entry – allowing non expert developers and designers to quickly sketch and construct functional devices – together with a high ceiling that allows experienced users to create sophisticated and capable devices that can be used in practice*" [31]. This is ideal for education where there is a need to stretch and challenge youngsters to achieve their highest potential. Since then, the platform has proven to be of interest to hobbyists and for secondary and tertiary education. Microsoft Research has launched .NET Gadgeteer as open source software/hardware, and .NET Gadgeteer kits are now available from a variety of hardware vendors.

A starter .NET Gadgeteer kit consists of a mainboard, and various modules including a camera, joystick, buttons, LEDs, potentiometer, Ethernet port, and touch-sensitive display. In this study, the FEZ Spider Starter Kit from GHI Electronics[7] was used. In addition, there are many other sensors and other modules that can be added separately. Devices can be constructed by connecting modules with cables and then programming it with respect to the events triggered when using the device (for example, a `ButtonPressed` or `PictureCaptured` event in case of a digital camera). The programming language used is **Visual C#** with support for **Visual Basic .NET** coming soon. Figure 1 shows the .NET Gadgeteer hardware with a range of modules connected together. Figure 2 shows the graphical **Gadgeteer Designer** that is used to generate code for each module being used. Figure 3 shows the programming environment in Visual C#.

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


```
void ProgramStarted()
{
    // Associate events with event-handling methods
    button.ButtonPressed += new Button.ButtonEventHandler(button_ButtonPressed);
    camera.PictureCaptured += new Camera.PictureCapturedEventHandler(camera_PictureCaptured);
}

void button_ButtonPressed(Button sender, Button.ButtonState state)
{
    camera.TakePicture();
}

void camera_PictureCaptured(Camera sender, GT.Picture picture)
{
    // Show the picture on the display
    display.SimpleGraphics.DisplayImage(picture, 0, 0);

    // Save the picture to the SD card
    sdCard.GetStorageDevice().WriteFile("picture.bmp", picture.PictureData);
}
```

Figure 3: Example Visual C# code

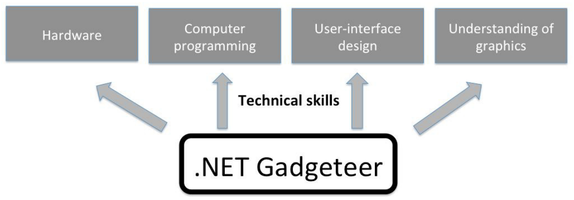


Figure 4: Aspects of learning with .NET Gadgeteer

Programming a .NET Gadgeteer device involves learning to program in Visual C#, but there are other skills involved in designing and implementing a gadget. Connecting the input and output modules to the mainboard will give students more of an understanding of hardware, and there is the option to teach students the underlying electronics of the board. A so-called *extender board* can be used to attach third-party hardware. In addition, the design of the case and the interaction of the user with the device will involve an understanding of user-interface design and physical form factor. Being able to program the screen will also teach students an understanding of graphics objects. Overall, the experience of using .NET Gadgeteer will cover a variety of aspects of the CS school curriculum, as shown in Figure 4.

4. CASE STUDY

The case study was designed as an observation of how students and teachers used .NET Gadgeteer and their experiences.

4.1 Aims and objectives

The aims of the case study were to investigate the potential of .NET Gadgeteer to consider which age group, which type of lessons, and which type of activities would be suitable for learning with the use of this tool. Specifically, the following research questions were addressed:

1. Does .NET Gadgeteer yield an engaging and motivating environment to work with in schools?

2. Are the initial teaching materials sufficient for students in lower and upper secondary schools to build .NET Gadgeteer devices?

3. Could .NET Gadgeteer be used to support student learning in Computer Science in school?

4. Is .NET Gadgeteer most suitable in schools as an extra-curricular activity or could it have a place in the main curriculum (in England and Wales)?

4.2 Participants

In this study, .NET Gadgeteer was used for the first time with secondary school students. Eight local schools in Cambridgeshire (UK) volunteered to be involved. The schools were a mixture of age 11-16 and age 11-18 schools, state schools (seven) and private schools (one), mixed schools (six), girls' schools (one) and boys' schools (one). The teachers were initially trained in the use of .NET Gadgeteer and provided with lesson plans and example projects. Munson et al. [20] point out that in order to engage students, it is also necessary to support teachers. We devised this pilot project with an initial learning session for teachers at Microsoft Research Cambridge, for them to cascade to pupils. Teachers introduced .NET Gadgeteer to their schools in the form of after-school or lunchtime clubs. The ages of the students attending the club ranged from 11 to 15 year, with one school choosing to use .NET Gadgeteer with an older group of 17 year olds. The students worked in groups of three to four with one .NET Gadgeteer kit per group. There was no requirement, given the existing ICT curriculum, for the schools to have taught programming before and it was acknowledged that it would not be possible to teach significant amounts of C# programming in the course of the pilot.

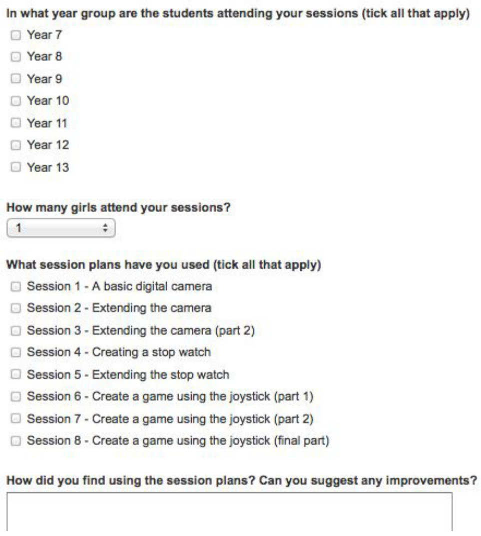
The materials developed consisted of eight lesson plans with approximately one hour's teaching material in each. The materials included learning objectives, aims of the session, a starter session as well as the main session in several steps, and extension material. The lesson plan contained all the instructions for carrying out the tasks. The session plans had aims and outcomes, in terms of programming skills to be learned, but the approach taken was actually to get devices working by following through instructions.

The students were loaned .NET Gadgeteer kits for four months. Teachers planned to run either one after-school club or one lunchtime club each school week. The plan was for them to run around ten hours of activity.

4.3 Methodology

The students' progress with .NET Gadgeteer was monitored by observation visits and their experiences evaluated at the end of the project. A case study methodology was used [24], taking the form of a **collective case study**, whereby a variety of sources were used to give an overall picture of how .NET Gadgeteer could be used in schools. An online questionnaire was designed to give teachers the opportunity to give feedback in their own time. This was available via a web link and had a mixture of open and closed questions. Teachers were also interviewed informally about their experiences of the project. An extract of the teachers' questionnaire is shown in Figure 5.

To collect verbal responses from students about their experiences, short interviews were held with students who attended at an event associated with the pilot. This represented a sample of 16 out of 84 students who took part in the pilot. The students volunteered to be interviewed, and had the option of speaking to the researcher individually or in pairs. This made it less stressful for the participants, although the sampling was therefore not of a representative group of the participating students. The interviewer asked each of the students the following four questions:


In what year group are the students attending your sessions (tick all that apply)

- ☐ Year 7
- ☐ Year 8
- ☐ Year 9
- ☐ Year 10
- ☐ Year 11
- ☐ Year 12
- ☐ Year 13

How many girls attend your sessions?

1

What session plans have you used (tick all that apply)

- ☐ Session 1 - A basic digital camera
- ☐ Session 2 - Extending the camera
- ☐ Session 3 - Extending the camera (part 2)
- ☐ Session 4 - Creating a stop watch
- ☐ Session 5 - Extending the stop watch
- ☐ Session 6 - Create a game using the joystick (part 1)
- ☐ Session 7 - Create a game using the joystick (part 2)
- ☐ Session 8 - Create a game using the joystick (final part)

How did you find using the session plans? Can you suggest any improvements?

Figure 5: Extract from teachers' questionnaire

1. What sort of things have you worked on in your .NET Gadgeteer sessions?

2. What has been your favourite part of working with .NET Gadgeteer and why?

3. Would you continue to use .NET Gadgeteer if you had a kit at school or home?

4. Has working with .NET Gadgeteer increased your understanding of how computers and electronic devices work?

The researcher filmed a demonstration of their gadget if they had made one. These interviews were transcribed and coded and key themes drawn out of the resultant data. The questions were designed to generate a range of data about engagement with the project.

The short interviews were followed by an in-depth focus group at one of the pilot schools. Two girls and two boys were invited to participate in the focus group which was conducted by the first author with a teacher present at all times. A focus group has many advantages when seeking to find out attitudes as the one-to-many dialogue, perhaps including a difference of opinion, may allow other issues to be raised which may not arise in an interview where there is no opposing view. The focus group was planned such that a teacher was present and the students all knew each other. It included more in-depth questions about the students' engagement with technology and computer programming more generally, with the intention of providing an elaboration of the points made in the short interviews and in the teacher questionnaires. The questions were based around the following themes: experiences of the .NET Gadgeteer project, learning new technologies, and aptitudes and difficulties with learning to program. The focus group discussion

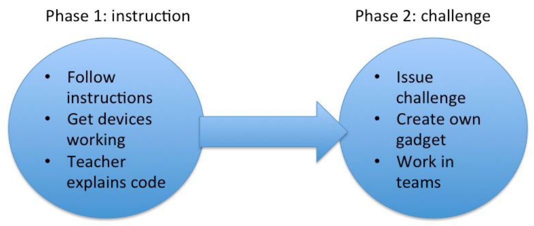


Figure 6: Pedagogical model

was carried out by the first author, and recorded, transcribed and then coded using TAMS analyser[8].

4.4 Pedagogical approach

The pedagogical approach taken with the teaching materials was based on a two-phase model whereby the first phase involved instruction and the second phase involved presenting students with a challenge. The students needed some instruction on how to use .NET Gadgeteer and the instructions led them through the implementation of three devices. Teachers utilised the materials differently, some setting student tasks based on the materials, and others letting the students work through the materials at their own pace. Each teacher had their own different style of teaching and this was noted in observations made by the first author at visits to each of the schools. Despite the teacher-differences, there was a common approach to teaching which led into the challenge phase of the project. Students were encouraged to create their own device for the end of the project. In order to investigate how students were able to learn with the 'try-it-and-see' bricolage approach, students were encouraged to be as inventive as they could in coming up with their own ideas for devices. Figure 6 shows the rationale behind the simple two-phase pedagogical model adopted.

5. RESULTS

Table 1 shows the schools, numbers of students attending sessions, and the number of session plans they managed to complete, although some schools ran more sessions and some students used some of their own time to complete projects.

After completing the session plans, students then worked on their own projects in Phase 2 of the pilot, applying what they had learned to their own ideas. Students worked in teams of between two and five students and developed a range of small devices. Students also enjoyed building the housing for the gadgets, which in some cases was quite sophisticated using moulded plastic, and for other gadgets, just as effective, using polystyrene or cardboard. Examples of the gadgets developed by students are shown in Figure 7.

Several schools ran more than ten sessions although engagement between the eight schools varied. The number of students attending at each school varied from three to 24 students. None of the schools completed all of the eight session plans provided, indicating that the material took longer to cover and that the difficulty level may have been underestimated. We originally aimed the project at Y9 students (aged 13-14) but gave teachers the freedom to select appro-

[8]http://tamsys.sourceforge.net/

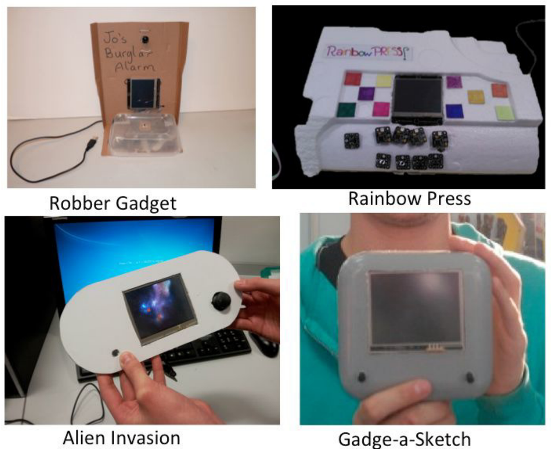


Figure 7: Examples of devices created by students with .NET Gadgeteer: "Robber Gadget" takes a picture of a burglar when a precious item is taken off the sensor platform; "Rainbow Press" is a reaction game; "Alien Invasion" is a shooting game; "Gadge-a-sketch" is a drawing device.

Table 1: Participants in the project

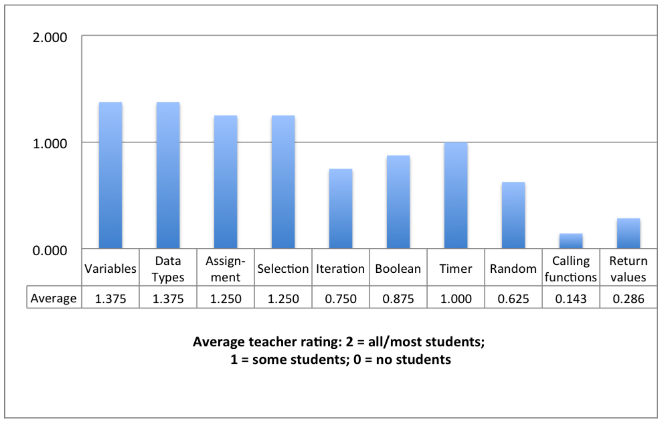
School	Number sessions	Number students	Sessions (out of 8)	Lowest year group	Highest year group
A	10	8	5	Y9	Y10
B	4	6	4	Y7	Y13
C	4	24	3	Y7	Y8
D	5	6	6	Y11	Y13
E	>10	10	6	Y8	Y11
F	>10	3	7	Y9	Y10
G	>10	10	6	Y7	Y11
H	7	18	4	Y9	Y9

(Y7 = age 11/12 (equivalent to 6 in USA);
Y13 = age 17/18 (equivalent to 12 in USA)

priate students as school environments differ. This had the benefit of giving us some examples of projects from students from Y7 to Y13, demonstrating the wide potential of .NET Gadgeteer.

Teachers were asked to rate how they thought that the students had mastered key programming concepts. It was not expected that novice programmers would be able to achieve a thorough understanding of Visual C# programming in just 10 hours of sessions, and we had no prior hypothesis about how much students in such a mixed age group at different schools with different teaching styles would learn. However, it was encouraging to see that most schools felt that their students had an understanding of assignment, data types, variables and selection as a result of the pilot. Figure 8 shows the relative acquisition of programming concepts as rated by their teachers.

Teachers were broadly positive about the achievements of their pupils and the motivation provided by .NET Gadgeteer. One teacher commented that *"It was fun and really nice to have things to touch and build. Some students were very engaged in the whole thing"* (School H, teacher).



	Variables	Data Types	Assign-ment	Selection	Iteration	Boolean	Timer	Random	Calling functions	Return values
Average	1.375	1.375	1.250	1.250	0.750	0.875	1.000	0.625	0.143	0.286

Figure 8: Acquisition of programming concepts (average across 8 schools)

There were a few technical problems experienced by teachers as the platform had never been tested before on school networks. These are discussed in Section 5.4.

5.1 Interviews with students

Approximately 85 students used .NET Gadgeteer in school as part of this pilot and 50 attended the end-of-school pilot event. Of these, 16 were interviewed during the day (14 boys, two girls). The comments from the student interviews and the focus group clearly identified four emerging themes. These were around the following features of the programme:

- Challenge and difficulty
- Creativity and freedom
- The tactile nature of .NET Gadgeteer
- The concept of "real programming"

These will be discussed in turn.

5.1.1 *Challenge and difficulty*

Without being prompted, students commented on the fact that they had found working with .NET Gadgeteer quite challenging at times. For example, one student said "*I enjoyed it but it was hard. And it was a challenge*"(School C, male), and another found it "*. . . a big learning curve really*" (School G, male). One student tried to explain why some other students from his school found the programming difficult: "*. . . but it took a great deal of effort, and not everyone's ready to . . . you know, actually step up to the challenge*" (School B, male). However, the level of challenge seemed to be popular with several students: "*. . . on Gadgeteer it's so much better because it's harder, but that's the good of it you know*" (School B, male). One young student with some programming experience commented that it was "*. . . not too simple, so that you're not really not learning that much, but it's not too complex, that you're not getting any help with debugging*" (School G, male).

There were fewer comments to the extent that it was not too difficult, and these were from two students who had had extensive programming experience from working on projects at home and self-teaching. There was recognition from several students that they had learned a lot from their experience: "*. . . we never learnt code before at all, so it most of the things, well everything we learnt for Gadgeteer was stuff that was new*" (School A, male).

5.1.2 *Creativity and freedom*

The pilot was designed with a second phase whereby students were to design a gadget or device of their choice, with freedom limited only by the modules that were available in the particular .NET Gadgeteer kit they were working with. Students referred to the fact that they liked the freedom to be creative and that they liked the tactile nature of the modules: "*You're in control you can take an idea anywhere and use the hardware that's available to make it and without limited . . . so the key it's versatile*" (School A, male). Another student particularly liked the camera project, which could be extended in many ways: "*What I enjoyed most about .NET Gadgeteer is the creativity you can have and the challenge it poses . . . especially with the camera, I really enjoyed that. Also trying to build your own sort of gadgets, and that was, you could really use your imagination*" (School G, male). Another student commented on the freedom given to use .NET Gadgeteer as they wanted: "*You're allowed to be sort of creative and sort of like make anything so you weren't really limited to what you can make*" (School A, male). An older student, with experience of programming and the Arduino platform, was very impressed by the flexibility it gave him: "*. . . you just plug it together and it works, you don't have to figure out the circuits for yourself . . . , but it's brilliant, really fun*" (School D, male).

5.1.3 *The tactile nature of .NET Gadgeteer*

Six students made comments to the extent that the tangible nature of the modules made it exciting to build and program devices. For example, one of the boys commented that "*. . . you don't just like have it all as pictures on the screen, you actually have the stuff that you can put together . . .*" (School C, male). During the event, the students were proud to be able to demonstrate the devices that they had made to other students, and their achievement was more visible in its physical form. Another student commented that developing a physical gadget meant that they could build something that had a purpose: "*You actually have something you could hold and manipulate, that's, it's sort of feeling a practical use for your programs, instead of just doing it for the sake of it*" (School G, male). Having both aspects of development, with the hardware and the software, also appealed to another student: "*You need to program it, you need to put it together, and it just makes it a whole lot better*" (School B, male).

5.1.4 *Doing "real programming"*

Students made reference to the fact that they knew that Visual C# was a programming language used in industry by professionals, and that they were using a tool that felt very 'adult'. For example, one student commented that "*This was much better because it was more professional and, you could do a lot more with it*" (School G, male). The comment made was in comparison to Scratch, which is the environment with which many of the students were already familiar. Another student compared .NET Gadgeteer to his experience with Scratch also: "*.NET Gadgeteer . . . sort of like got proper programming if you know what I mean, . . . with Scratch you've got words already set out for you and you can sort of like make building blocks . . . but with .NET Gadgeteer* *you've definitely got to have that more adult aspect of it and the programming ...*" (School G, male).

The pilot had encouraged students to consider doing more programming in the future. Some students reflected that their views on Computer Science as a subject had changed as a result of the engagement with .NET Gadgeteer: "*... I definitely want to take Computing and I'm looking to take a job inside Computing as well when I'm older*" (School G, male). A young student mused about the possibilities that might open for him if he pursued a career in Computer Science: "*It would be a nice career to take on ... and so I'd really like to be working at a place like Silicon Valley, it would be a really nice opportunity*" (School G, male).

Overall, the interviews elicited mostly positive comments about .NET Gadgeteer. The results of the focus group discussion will be reported next.

5.2 Focus group findings

This group of students included three from Y10 and one from Y9 (this translates to the beginning upper secondary school in Europe and the beginning of high school in the USA). The focus group was held in School A with two boys and two girls who had participated in the project. They were asked similar questions to the short interviews initially and similar themes emerged, for example, relating to freedom and creativity: "*I liked designing for the competition, the fact that we got to design our own product*" (School A, male). The students also reported that they liked the tactile nature of the kits, and that the first project was quite accessible: "*... like the ease of it, it was quite easy to integrate the bits, for instance making the camera was quite simple*" (School A, male). Other comments were made that mirrored the content of the short interviews already reported on. They commented that .NET Gadgeteer made the concept of the technologies they use more accessible "*because essentially we had many of the bits of an iPod Touch, a camera and a touch screen, and you can get the audio jack and stuff, to give you an idea that you could perhaps make something like that and that these technologies weren't far away*" (School A, male). Students also discussed some of the key aspects that they felt appealed about Computer Science to young people, what helped them to learn, what qualities were useful in learning programming and why other students might not succeed. The first quality to be suggested was to be open-minded: "*Open-minded ... because it's like some people if they don't want to learn something then they most probably won't listen to it*" (School A, female). The student went on to explain that the stereotype of being "weird and nerds" could put people off doing Computer Science if they were not open-minded.

The students agreed between them a set of qualities that might make students good at programming, or working with .NET Gadgeteer, as follows:

- Open-mindedness
- Common sense
- Problem-solving skills
- Patience
- Imagination
- Intuition

Common sense was defined by the student as "*just being able to look at something, and then work out, kind of maybe use your brain to go through what the computer is doing, the way the computer is looking at it. There's a lot of people can't do that. I think they find it a lot more difficult to program*" (School A, male). From the student's point of view (this student had taught himself some programming already) this was common sense, whereas another might see this as logical thinking. This was an interesting example of a student with experience perhaps underestimating what was difficult to a non-programmer [25]. The two girls were more able to empathise with the issues that other students might have in learning to program.

The notion of patience being a key to success in programming was mentioned at other times in the focus group and in one of the short interviews: *"Problem-solving skills and patience, because when you're learning it, you have to have a lot of patience in you because it might take some time to learn everything, you can't exactly rush through it otherwise you won't learn it properly"* (School A, male).

Students also reported on the sources that they used to assist them when they had problems and needed help with their work in computing programming. They demonstrated that they knew where they could look for help on the Internet: "*Looking at other people's code*" (School A, male), and also: "*I read tutorials and find the answer on the Internet*" (School A, male). The two girls were more reliant on teachers and friends to help them with their coding: "*Mainly the teachers who help me try and learn and help me go through it so it sticks in my head a bit more*" (School A, female) and: "*Just having people explain it to you and if you go wrong having people explaining exactly where you went wrong and how you're supposed to do it right, and things like that would be quite helpful. Having someone that can really explain it to you*" (School A, female).

This illustrates some of the gender differences, as discussed briefly below.

5.3 Girls and .NET Gadgeteer

In this study, we had not particularly focused on comparing the responses from different gender groups in our design, and we had a much smaller number of girls in the project than boys. However, the comments from girls (two in the short interviews and two more in the focus group) suggested some surprise that they had managed to be successful with .NET Gadgeteer. Girls reported that prior to using .NET Gadgteer "*I really thought I wouldn't be able to do anything*" (School G, female). Another girl admitted that it hadn't really appealed to her: "*... just thought it was all a bit, sort of hard, just like why would I need to know*" (School E, female). The focus group girls were more confident: "*Well, maybe a little bit overwhelming at first but once you understand what it all means it's not that difficult once you know what you're doing*" (School A, female).

The girls reported more confidence after they had finished creating their gadgets with .NET Gadgeteer: "*Yeah, it's definitely opened up more about what I know about programming*" (School E, female). Another of the girls said that she was more confident since building her (highly innovative) gadget: "*I've really surprised myself, and I'd have to say, I've grown in confidence as well*" (School G, female).

One of the teachers (School C), in an interview, commented on what girls enjoyed about the programme: "*The* *girls, I think they were just more interested in getting the camera working, and they all wanted to be able to take pictures of themselves, which kind of gives them that sense of ownership*".

With such a small number of girls, we can only note these gender differences; intuitively, though, it seems as if working with this platform could appeal widely to both genders, as gadgets with a socially useful function could be developed over time that are more appealing to girls. The next section addresses some of the limitations and problems we found with .NET Gadgeteer on this first use in UK schools.

5.4 Problems with .NET Gadgeteer

In terms of their dissatisfaction with the project the main complaint from students was about lack of time to complete their projects: " *. . . we had after-school clubs every Wednesday, but towards the end we ended up doing it at lunchtimes as well*" (School G, male). Another student reported spending much time on his .NET Gadgeteer trying to complete a project: " *. . . mainly just in lunchtimes, then Thursday and Wednesday I've been going after school, and every opportunity possible really . . . I've been trying to get things finished*" (School E, male).

Teachers also would have liked more time to complete projects, indicating that working with .NET Gadgeteer can encourage longer-lasting project work than we had originally envisaged: "*I would have loved to take it further if we had had more time, and to see the possibilities and how far our students would be able to take it. What could our students create if they had another term? How advanced could our students take it? What new systems, concepts could they interpret and understand to produce bigger and better systems? Just more time for students to play and to see the real potential of the gadgets they create*" (School A, teacher).

An issue for teachers was some of the technical difficulties they had had at the outset when the pilot started. These schools were some of the first public users ever of .NET Gadgeteer, so some locked-down school networks posed a challenge initially. This links to the fact that the students felt that they did not have enough time to build what they wanted to. For our research project, we had set these time constraints. For future work with .NET Gadgeteer it is likely that a full academic year would be needed to really benefit and learn, as with any new skill. .NET Gadgeteer was very new when the project started, as the teachers received some of the first kits that were shipped over to the UK. The students were using the .NET Gadgeteer environment before the schools had had time to test it properly on their systems: "*I still have not been able to load the required software onto the school system, and have had to work using a teacher laptop that I had to battle 8 weeks for to get admin rights to load the programs onto it*" (School C, teacher). These teething problems have now been eradicated as indicated by this teacher: " *. . . there were some minor glitches with a mismatch of software and hardware at the start*" (School G, teacher).

The teachers also wanted more materials, as identified here: "*Some form of 'student book' to allow them to track and take notes might help*" (School H, teacher). Another teacher wanted more model projects to inspire students: "*I would have quite liked a very advanced gadget to have been created and code provided, not for students to see code, but so we could easily show some of the high potential that could be achieved using Gadgeteer systems*" (School A, teacher). Finally, one teacher felt that their Y7 students were too young and that the older students got more out of the experience: "*Great ideas and all the students were enthusiastic despite often making very little progress. Our mistake - we opened it to all age groups and while the younger students were very keen, they lacked the concentration and basic programming experience to get as much from it as the older students did*" (School G, teacher).

6. DISCUSSION

The results of this study of .NET Gadgeteer in secondary schools demonstrate that students were engaged by .NET Gadgeteer, although they did find it difficult. This is backed up by their teachers' comments. Challenge is seen as a positive feature as reported by the students in the short interviews; many students referred to this although it was not specifically elicited. There were some very able students within the pilot groups for whom the opportunity to use .NET Gadgeteer provided them with a stimulating resource that they very much enjoyed. However, these students had volunteered to be part of the pilot groups, and more investigation is needed to see how less motivated students experience the platform.

Students automatically seemed to compare .NET Gadgeteer with the Scratch programming environment, although again this was not mentioned by the interviewer, presumably because that is the experience that students had had so far and as such the only experience with which they were able to compare .NET Gadgeteer. Their comments implied that, after using Scratch, they felt that .NET Gadgeteer allowed them to write their own functions and have more freedom to customise their own programs. What .NET Gadgeteer offers after having learned Scratch is an introduction to a real programming language and this was popular with students. The introduction to programming provided by Scratch, if incorporated in a teaching programme at school, will help them prior to moving forward with .NET Gadgeteer.

Although a challenge, students seem to have enjoyed working together in groups and creating something of their own. The production of a physical device to demonstrate was a motivating factor. Observations of students working revealed that they worked well in groups in the main, with teachers being flexible about sizes and make-up of groups. Within groups, students assigned themselves different roles, with some having more responsibility for the design of the case or user-interface, with others writing more of the code. This is one of the outcomes of the project that we had not expected. To develop their own gadget, students were necessarily involved in both problem-solving and some independent research. They had to experiment with different modules to find out exactly how they worked. The wider skills they gain through engagement in these activities have a positive effect on developing their personal, learning and thinking skills.

The pedagogical model that we used encouraged a bricolage approach. We wanted to provide some instruction, but then to inspire children to experiment with modules and devices on their own and come up with their own devices. For some of the children, though, the gap between our Phase 1 and Phase 2 was too great and the teachers did not obviously have the background or sufficient training in the platform to support them. We need to build on the incremental prob-

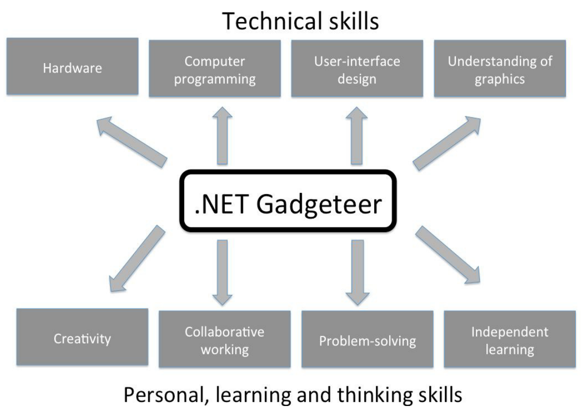


Figure 9: Wider skills development with .NET Gadgeteer

lem solving as used successfully by Stiller [28] and provide smaller gaps in future materials, to maintain both teacher and student confidence.

We have not specifically tested the students' understanding of programming principles in this project. Meerbaum-Salant carried out a study whereby she measured the understanding of key programming concepts of students who had participated in a course using the Scratch programming environment [19]. She reports that *"the bottom-up programming habit is clearly encouraged by the characteristics of the Scratch environment and is in line with Papert's philosophy of constructionism . . . and with bricolage"* [p.171] and concludes that *". . . we are disturbed by the habits of programming that we uncovered. These habits are not at all what one expects as the outcome of learning computer science"* [p.172]. It is not clear whether she is blaming the Scratch environment in being too "easy" or the constructionist and bricolage approach in her concerns about long-term effects of an exploratory approach to programming. We would hope to be able to show that an exploratory, and bricolage-centred, pedagogical approach can be both motivating and teach good programming habits, in the context of a real programming language that works behind a motivating hardware environment.

As the platform is still new and materials are still under development, we have not yet been able to replicate this level of analysis of learning for .NET Gadgeteer. However, although a bricolage approach seems implicit in 'gadget-building', it may not be as bottom-up as the Scratch programming that Meerbaum-Salant describes. Where students assemble their gadget before programming, they have put together all the modules they need for the project and then can break down the programming into events. This provides more of a top-down, but also very concrete experience of program development. More research is obviously needed to discover if this is effective in the long-term acquisition of key concepts. This will involve the development of a suitable research instrument to measure learning with .NET Gadgeteer.

Figure 9 adds the development of the wider personal, learning and thinking skills to that shown in Figure 4. This illustrates some of the potential wider skills that can be gained by students working on .NET Gadgeteer in groups on their own projects.

7. CONCLUSIONS AND FURTHER WORK

In this paper, we have presented our experiences of using .NET Gadgeteer in schools with students of various ages and backgrounds over a four-month period. Our findings have revealed that students are very engaged by it, and are inspired to build devices for which they can see a real purpose. Through analysis of data from students and teachers, we have uncovered some key features of .NET Gadgeteer that provide motivation and interest: challenge, freedom to explore, tangibility and exposure to the real world of Computing.

We are partway towards answering our research questions:

1. *Is .NET Gadgeteer an engaging and motivating environment to work with in schools?* We have found that students reported positively about using .NET Gadgeteer for a range of reasons, as described in this paper.

2. *Can students in lower and upper secondary schools use .NET Gadgeteer to build devices with the initial teaching materials developed?* Students have been successful in building devices; however, the teachers indicated that the materials need to be developed further.

3. *Could .NET Gadgeteer be used to support student learning in Computer Science in school?* With the changes in the curriculum in the UK, involving the introduction of more Computer Science in secondary schools, .NET Gadgeteer provides an environment that is both engaging and encourages experiential learning. Working with .NET Gadgeteer has the potential to develop students' ability to work collaboratively. Programming skills can be developed through a mixture of instruction and an exploratory approach to learning.

4. *Is .NET Gadgeteer most suitable in schools as an extra-curricular activity or could it have a place in the curriculum (in England and Wales)?* The research revealed that working through the materials and creating the devices was quite intensive and required more time than available in an after-school club. Further work is needed to write longer courses that would work with curricular in UK schools and within examined courses.

The .NET Gadgeteer platform is still in its infancy as it was only launched in August 2011. At the time of writing, more modules are being developed by an increasing number of manufacturers.

The pilot project was successful in its aims and objectives which were to establish that .NET Gadgeteer can be a useful tool for teachers and students in the classroom. Its tangible nature engenders curiosity and creativity, and motivates students to explore, bricolage-fashion, how to program their device. The physical nature of the platform encourages a learning-by-doing experiential approach to learning and in addition lends itself to collaborative working, whereby students with a range of skills and abilities can support and learn from each other.

As an initial pilot project with a new platform, this research has highlighted particular areas of further work. In future studies using .NET Gadgeteer, we would like to investigate how students' perception of the importance of challenge and creativity links to the development of secure and robust programming understanding. It is thus proposed that further studies with .NET Gadgeteer focus on the facilitation of the acquisition of certain programming concepts. The two-phase pedagogical model utilised in the pilot project will be developed further to provide more staged support, whilst retaining an exploratory and experiential approach.

8. ACKNOWLEDGMENTS

With thanks to Steve Hodges, Steven Johnston, James Scott and Nicolas Villar of Microsoft Research, and to all the participating teachers and students for their assistance and enthusiasm during the pilot project. The first author would like to thank Microsoft Research for supporting her work.

9. REFERENCES

[1] O. Astrachan, J. Cuny, C. Stephenson, and C. Wilson. The CS10K Project: Mobilizing the Community to Transform High School Computing. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, pages 85–86, New York, NY, USA, 2011. ACM.

[2] M. Ben-Ari. Constructivism in Computer Science Education. In *Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education*, pages 257–261. ACM, 1998.

[3] T. Brinda, H. Puhlmann, and C. Schulte. Bridging ICT and CS - Educational Standards for Computer Science in Lower Secondary Education. *Proceedings of ITICSE'09, Paris, France*, July 6-9 2009.

[4] L. Carter. Why Students with an Apparent Aptitude for Computer Science Don't Choose to Major in Computer Science. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, pages 27–31. ACM, 2006.

[5] S. Cooper, W. Dann, and R. Pausch. Alice: a 3-D Tool for Introductory Programming Concepts. *J.Comput.Sci.Coll.*, 15(5):107–116, Apr. 2000.

[6] T. Crick and S. Sentance. Computing at School: Stimulating Computing Education in the UK. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*, Koli Calling '11, pages 122–123, New York, USA, 2011. ACM.

[7] Q. Cutts, S. Esper, and B. Simon. Computing as the 4th R: a General Education Approach to Computing Education. In *Proceedings of the 7th International Workshop on Computing Education Research*, ICER '11, pages 133–138, New York, NY, USA, 2011. ACM.

[8] T. Downes and D. Looker. Factors that Influence Students' Plans to Take Computing and Information Technology Subjects in Senior Secondary School. *Computer Science Education*, 21(2):175–199, 2011.

[9] A. Fisher and J. Margolis. Unlocking the Clubhouse: the Carnegie Mellon Experience. *SIGCSE Bulletin*, 34(2):79–83, June 2002.

[10] O. Hazzan, J. Gal-Ezer, and L. Blum. A Model for High School Computer Science Education: the Four Key Elements that Make It! In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, pages 281–285. ACM, 2008.

[11] F. T. Hofstetter. *Multimedia Literacy.* Irwin/McGraw-Hill, Boston, 2nd edition, 1997.

[12] M. S. Horn, R. J. Crouser, and M. U. Bers. Tangible Interaction and Learning: the Case for a Hybrid Approach. *Personal Ubiquitous Computing*, 16(4):379–389, Apr. 2012.

[13] M. Kordaki, M. Miatidis, and G. Kapsampelis. A Computer Environment for Beginners' Learning of Sorting Algorithms: Design and Pilot Evaluation. *Computers and Education*, 51:708–723, 2008.

[14] C. Lévi-Strauss, J. Weightman, and D. Weightman. *The Savage Mind.* University of Chicago Press, 1966.

[15] M. Kolling. The Greenfoot Programming Environment. *Transactions on Computing Education*, 10(4):14:1–14:21, Nov. 2010.

[16] M. MacLaurin. Kodu: End-User Programming and Design for Games. In *Proceedings of the 4th International Conference on Foundations of Digital Games*, FDG '09, New York, USA, 2009. ACM.

[17] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond. The Scratch Programming Language and Environment. *Transactions on Computing Education*, 10(4):16:1–16:15, Nov. 2010.

[18] P. Marshall. Do Tangible Interfaces Enhance Learning? In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, TEI '07, pages 163–170, New York, NY, USA, 2007. ACM.

[19] O. Meerbaum-Salant, M. Armoni, and M. Ben-Ari. Habits of Programming in Scratch. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, ITiCSE '11, pages 168–172, New York, NY, USA, 2011. ACM.

[20] A. Munson, B. Moskal, A. Harriger, T. Lauriski-Karriker, and D. Heersink. Computing at the High School Level: Changing What Teachers and Students Know and Believe. *Computing Education*, 57(2):1836–1849, Sept. 2011.

[21] S. Papert. *Mindstorms: Children, Computers, And Powerful Ideas.* Basic Books, 1993. 92053249.

[22] L. Pollock and T. Harvey. Combining Multiple Pedagogies to Boost Learning and Enthusiasm. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, ITiCSE '11, pages 258–262, New York, NY, USA, 2011. ACM.

[23] M. Richards, M. Petre, and A. K. Bandara. Starting with Ubicomp: Using the Senseboard to Introduce Computing. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, SIGCSE '12, pages 583–588, New York, NY, USA, 2012. ACM.

[24] R.Stake. *The Art of Case Study Research.* London, Sage, 1995.

[25] C. Schulte and M. Knobelsdorf. Attitudes Towards Computer Science-Computing Experiences as a Starting Point and Barrier to Computer Science. In *Proceedings of the 3rd International Workshop on Computing Education Research*, ICER '07, pages 27–38, New York, NY, USA, 2007. ACM.

[26] R. E. Stake. *The Art of Case Study Research.* Sage Publications, 1995. 95004979.

[27] C. W. Starr, D. Bergman, and P. Zaubi. The Development and Implementation of a Context-Based Curricular Framework for Computer Science

Education in High Schools. In *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education*, pages 283–287. ACM, 2009.

[28] E. Stiller. Teaching Programming Using Bricolage. *J.Comput.Sci.Coll.*, 24(6):35–42, June 2009.

[29] The Royal Society. Shut Down or Restart? The Way Forward for Computing in UK Schools. Technical Report January 2012 DES2448, The Royal Society, 2012.

[30] S. Turkle and S. Paper. Epistemological Pluralism and the Revaluation of the Concrete. *Journal of Mathematical Behaviour*, 11(1):pp 3 – 33, 1992.

[31] N. Villar, J. Scott, and S. Hodges. Prototyping with Microsoft .NET Gadgeteer. In *Proceedings of the 5th International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '11, pages 377–380, New York, NY, USA, 2011. ACM.

[32] N. Villar, J. Scott, S. Hodges, K. Hammill, and C. Miller. .NET Gadgeteer: A Platform for Custom Devices. *Proceedings of Pervasive 2012, Lecture Notes in Computer Science*, 2012.

[33] C. Wilson, L. A. Sudol, C. Stephenson, and M. Stehlik. Running on Empty: The Failure to Teach K–12 Computer Science in the Digital Age. Technical report, CSTA, 2010.

[34] S. Yarosh and M. Guzdial. Narrating Data Structures: the Role of Context in CS2. *J.Educ.Resour.Comput.*, 7(4):6:1–6:20, Jan. 2008.