Create your Own Computer Companion

**Year level band:** 7-8

**Description:**
In this lesson, students will build a pop-up computer companion that pops up from behind the screen whenever a sound is heard. The companion’s voice is programmed using Scratch. This project introduces students to the littleBits kit and the Makey Makey bit and discusses how they can be used to build digital systems.

This lesson is a transition from visual programming to general purpose programming. It explores decomposition, branching and iteration as well as the concepts of systems thinking and integrating circuitry with the Makey Makey, and a computer program. This is a good precursor for arduino lesson plans.

**Type:** Visual programming, systems thinking

**Resources:**
- littleBits Rule Your Room Kit
- littleBits Rule Your Room Invention Guide (included with the kit)
- Mac or PC with latest version of Scratch installed
- Spare 9V batteries
- Phillips head screw driver and one screw

**Prior Student Learning:**
A basic understanding of circuits is useful.

An understanding of general programming concepts - input and output, algorithms, loops and debugging.

An understanding of how littleBits work and the various types of bits, including the Makey Makey bit.

If the class is not familiar with littleBits, the companion lesson plan, called **Introduction to littleBits and Makey Makey** should be used instead of this lesson.

<table>
<thead>
<tr>
<th>Digital Technologies Summary</th>
<th>This activity further explores the idea of a digital system composed of littleBits circuitry connected to a Makey Makey, connected in turn to a computer, where a Scratch program receives input from the Makey Makey.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Band</td>
<td><strong>Content Descriptors</strong></td>
</tr>
</tbody>
</table>
Design the user experience of a digital system, generating, evaluating and communicating alternative designs (ACTDIP028)

- determining the factors that influence proposed solution ideas, for example identifying limitations on the arm and the motor and the weight they can lift together
- Identify features that make a HID interface easy to use and incorporate these into their own design
- Identifying similar digital systems and their user interfaces, assessing whether user interface elements can be re-used.
- Presenting and comparing alternative designs to a solution for a problem, for example presenting alternative design mock-ups to the class

Design algorithms represented diagrammatically and in English, and trace algorithms to predict output for a given input and to identify errors (ACTDIP029)

- Flowcharts present the branching algorithm for the Scratch program, identifying key sprites and decomposition their behaviour into key blocks

Implement and modify programs with user interfaces involving branching, iteration and functions in a general-purpose programming language (ACTDIP030)

- Decomposition the project into key Scratch sprites (corresponding to parts of the art display) and decomposing their behaviour into instructions
- Using the input from the Makey Makey and branching to determine the behaviour of the Scratch program

<table>
<thead>
<tr>
<th>7-8</th>
<th>Design the user experience of a digital system, generating, evaluating and communicating alternative designs (ACTDIP028)</th>
</tr>
</thead>
</table>
|  | - determining the factors that influence proposed solution ideas, for example identifying limitations on the arm and the motor and the weight they can lift together  
  - Identify features that make a HID interface easy to use and incorporate these into their own design  
  - Identifying similar digital systems and their user interfaces, assessing whether user interface elements can be re-used.  
  - Presenting and comparing alternative designs to a solution for a problem, for example presenting alternative design mock-ups to the class  
  
|  | Design algorithms represented diagrammatically and in English, and trace algorithms to predict output for a given input and to identify errors (ACTDIP029) |
|  | - Flowcharts present the branching algorithm for the Scratch program, identifying key sprites and decomposition their behaviour into key blocks  
  
|  | Implement and modify programs with user interfaces involving branching, iteration and functions in a general-purpose programming language (ACTDIP030) |
|  | - Decomposition the project into key Scratch sprites (corresponding to parts of the art display) and decomposing their behaviour into instructions  
  - Using the input from the Makey Makey and branching to determine the behaviour of the Scratch program  
  
|  | The particular elements of Critical and Creative Thinking addressed by this content description  
  - Inquiring – identifying, exploring and organising information and ideas  
  - Generating ideas, possibilities and actions  
  - Analyzing, synthesising and evaluating reasoning and procedures  
|  | Inquiring – identifying, exploring and organising information and ideas  
  - Identify and clarify information and ideas  
  - Organise and process information  
|  | Generating ideas, possibilities and actions  
  - Consider alternatives  
  - Seek solutions and put ideas into action  
  - Imagine possibilities and connect ideas  
|  | Analyzing, synthesising and evaluating reasoning and procedures  

<table>
<thead>
<tr>
<th>Element</th>
<th>Summary of tasks</th>
</tr>
</thead>
</table>
| **Learning hook** | We are going to use a Makey Makey bit to connect our littleBits prototypes to a computer program. First we need to explore what Bits we have and how they work and interact.  

If you haven't done so before, you may want to go through the littleBits basics with the class (p2-3 in the Invention Guide): Anatomy, Color-coded categories, Magnets, Order of Bits. Similarly, ask students explore the bits in the kit so they know their features and functions. Pages 6-11 introduce all the available Bits and how they can interact.  

If students have not used Makey Makeys before, introduce them to the Makey Makey. This YouTube tutorial might be useful:  

https://www.youtube.com/watch?v=-X3hb__YynM  

Ask students to think about how the computer greets them when it realises their presence. Ask them to think about the ways in which we let the computer know we want to use, when the computer is idle: mouse click, keyboard.  

What if we could use sound to make some alerts pop up? What if these alerts were physical objects that pop up from behind the computer let's say. |
| **Achievement Standards** | Students plan and manage digital projects to create interactive information. They define and decompose problems in terms of functional requirements and constraints. Students design user experiences and algorithms incorporating branching and iterations, and test, modify and implement digital solutions. They evaluate information systems and their solutions in terms of meeting needs, innovation and sustainability. They analyse and evaluate data from a range of sources to model and create solutions. |
| **Learning Map (Sequence)** | • Students explore the components of the littleBits Rule Your Room Kit |
● Students create a physical prop that will pop from behind the computer, when a sound is heard – a picture of a cat or a monster, or a stick figure are good examples
● Students build the littleBits circuitry that react to the sound (i.e., using a sound sensor) and move the prop using a robotic arm
● Students write, test, and debug a Scratch program activated by the Makey Makey
● Students reflect on their work and make suggestions for improvements or extensions

**Learning input**

1. Ask students to look at their Makey Makey bit and highlight the input and output connections. Remember the colour coding - input pink - output green.
2. Ask students to select the bits that they would need for this project. In this case power, sound sensor, bar graph, servo motor, connected to the Makey Makey
3. Explain that inputs can be digital or analogue. Digital has two states (on/off, high/low, or true/false) but analogue can take values in between. In our kits the button is digital and the dimmer is analogue - and they need to use the appropriate pins.
4. Students draw the circuitry and connections, showing the input/output flow, as well as the type of input: digital/analogue.
5. Once they are sure the circuitry is correct, students build the circuit using littleBits. Follow the steps here for full instruction (http://littlebits.cc/projects/invent-the-ultimate-computer-companion).

6. Students attach servo arm to the motor and stick the arm behind the laptop or computer screen, like in the figure:
7.

8. Students stick the pop-up cardboard character to the robotic arm, using blue tac or duct tape.

9.

10. Students decompose the way the digital system works into the key components, highlighting the where the computation lies, as well as the connections between the systems and where code needs to be written and for what:
    • The circuitry detects the sound in the vicinity of the computer
- If the sound is detected, the circuitry moves the arm for the companion to be shown
- If a sound is detected, the Makey Makey bit click is activated, and a signal is passed to the computer -- **connection** between the circuitry and the Scratch program
- At the same time, if the Makey Makey click is activated, the Scratch program plays a sound and a counter is increased.

11. Students write the Scratch program that plays a sound once the left arrow is clicked (note connection of the littleBits to the Makey Makey has to happen to the left arrow).
12. Students explore other ways in which the Scratch program could be extended - e.g. a sprite can speak and be animated as a result of sound sensors being activated
13. Students explain how the digital system works and debug and test it in pairs.

<table>
<thead>
<tr>
<th>Learning construction</th>
<th>Students work in pairs to construct the circuit and write the Scratch program.</th>
<th>Students work in pairs to construct the circuit and write the Scratch program. The activity is about experimenting, trying new solutions, and debugging. Once students have successfully build the circuit, ask them to explore what other systems could be designed and connected. Encourage students to help each other - and look for help on the internet. Ask a friend. Ask Google. Then ask the teacher.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning demo</td>
<td>While students are working in groups, ask questions to give them the opportunity to demonstrate their thinking and understanding: What challenges have you faced in building this circuit? What other bits could you add to your circuit and how would you use them?</td>
<td>While students are working in groups, ask questions to give them the opportunity to demonstrate their thinking and understanding: What challenges have you faced in building this circuit? What other bits could you add to your circuit and how would you use them?</td>
</tr>
<tr>
<td>Learning reflection</td>
<td>• Remind students that littleBits is a prototyping platform but that the circuits we have been exploring exist in real-world products. • Can you think of any exciting products that could be created with this sort of technology? • What other components could be used as inputs or outputs? Consider everyday items that could be connected to the Makey Makey – see pages 24-25 in the inventor’s guide for suggestions. • Ask students to think about what other digital systems could be designed with littleBits, Makey Makey and computer programs: what real-world problems could they solve?</td>
<td>• Remind students that littleBits is a prototyping platform but that the circuits we have been exploring exist in real-world products. • Can you think of any exciting products that could be created with this sort of technology? • What other components could be used as inputs or outputs? Consider everyday items that could be connected to the Makey Makey – see pages 24-25 in the inventor’s guide for suggestions. • Ask students to think about what other digital systems could be designed with littleBits, Makey Makey and computer programs: what real-world problems could they solve?</td>
</tr>
</tbody>
</table>
Assessment:

- Observation of students building littleBits circuits
- Successful control of the robotic arm
- Successful synchronization of the robotic arm and the sound played by Scratch
- Teachers observe students creating their algorithms and debugging.
- Use questioning to elicit student understanding of the functions of littleBits and Makey Makey, the programming platform and their algorithmic thinking.
- You might take photos/videos of the students’ work to document their progress – or in the final presentations.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Quantity of knowledge</th>
<th>Quality of understanding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms</td>
<td>Pre-structural</td>
<td>Uni-structural</td>
</tr>
<tr>
<td>Programming</td>
<td>Circuit is built and Makey Makey is connected to computer.</td>
<td>Scratch code is written and input from the Makey Makey is captured.</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>No specific / technical terms used.</td>
<td>The terms input/output, code, or circuit may be used as a general description.</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

**Teacher/Student Instructions:**

- The servo mode should be adjusted to turn
  To adjust the sound sensor sensitivity
- Use the purple screwdriver to turn the sensitivity dial clockwise. This makes it more sensitive to sound.
- Make a sound to trigger the circuit! The bargraph should light up and the servo should move.

**CSER Professional Learning:**

This lesson plan corresponds to professional learning in the following CSER Digital Technologies MOOCs:

7 & 8 Digital Technologies: Next Steps
Unit 2 - Next Steps 7 & 8

**Further Resources:**

1. Information about the littleBits Rule Your Room Kit and some example projects: [http://littlebits.cc/projects](http://littlebits.cc/projects)
2. The littleBits Educator Guide, available online here: [https://d2q6sbo7w75ef4.cloudfront.net/littleBitsEducatorsGuide_FINAL.pdf](https://d2q6sbo7w75ef4.cloudfront.net/littleBitsEducatorsGuide_FINAL.pdf)

Author: Claudia Szabo
This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License. Computer Science Education Research (CSER) Group, The University of Adelaide.