

# 2023 STiCS littleBits Session



<https://trickel.org/thomas/skc/littleBits.html>

## **2023 Spring Technology in the Classroom Showcase**

### **littleBits**

- [2023 Spring Showcase littleBits Slide Deck](#)
- [Code Kit Core](#)
- [Introduction: Hello World slides](#)
- [Fuse](#)

### **Useful Links**

- [Code Kit Core Lessons](#)

# littleBits Code Kit



# littleBits – Code Kit Core

## ○ What To Expect

20+ engaging hours of CSTA- and NGSS-aligned curriculum. Easy to implement in classes with students grades 3-8



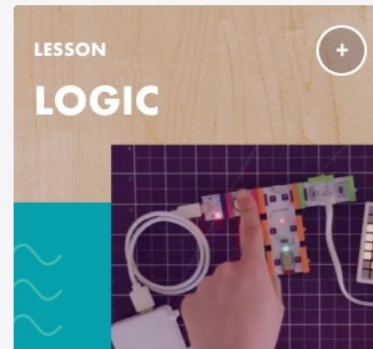
11 Lessons



Explore littleBits basics and inventing with the Code Kit app.&nbsp;



The loops lesson teaches students how to use loops to create



Use conditional logic to program rules into games. The Logic lesson teaches



The variables lesson teaches students how to create variables



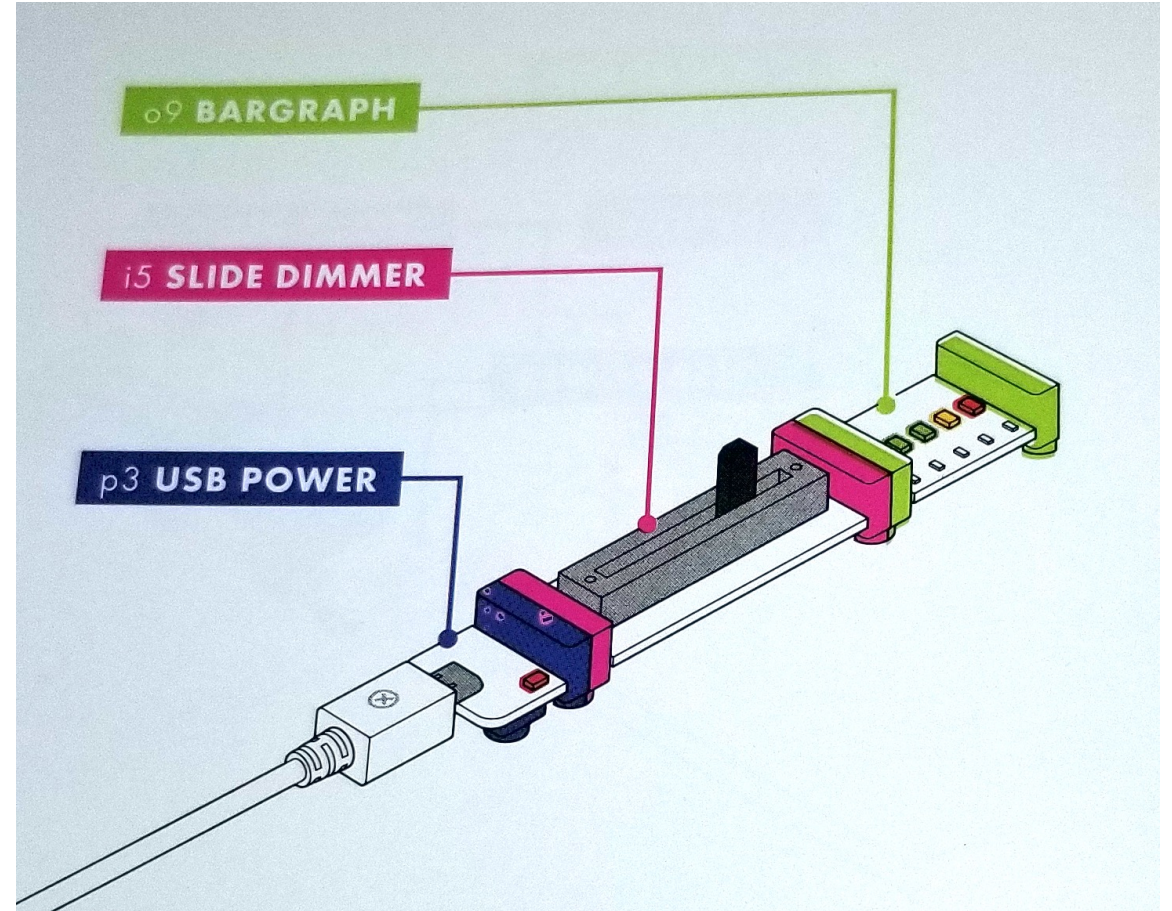
# littleBits – Code Kit Core Hello World Lesson Slides



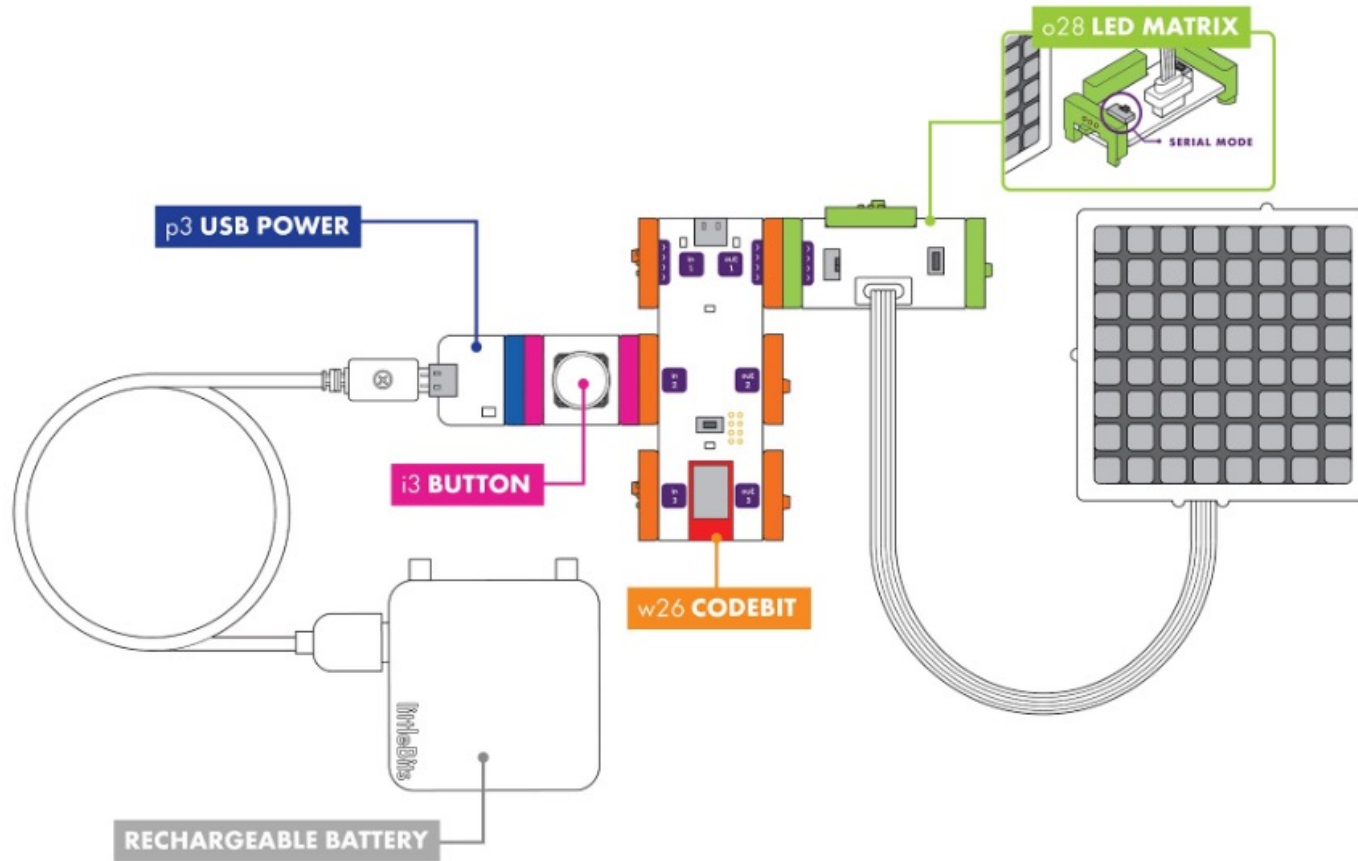
- [trickel.org/thomas/skc/littleBits.html](http://trickel.org/thomas/skc/littleBits.html)

# littleBits

- Snap together
- Color coded
  - Blue – Power
  - Pink – Input
  - Green – Output
  - Orange – Connections
- Complexity hidden



# Hello World



# Hello World

The image shows the Fuse littleBits code editor interface. At the top, there is a purple header with the 'fuse littleBits' logo on the left, a 'Create New' button in the center, and a 'Getting Started?' button on the right. Below the header, the interface is split into two main sections. On the left is a vertical sidebar with a 'Build' button and a 'Code' button. Below these are several category buttons: 'DISPLAY' (purple), 'TEXT' (light blue), 'INPUT' (pink), 'OUTPUT' (green), 'TIMING' (orange), 'SOUND' (red), 'LOOPS' (dark green), 'LOGIC' (dark blue), 'MATH' (purple), 'VARIABLES' (dark red), and 'FUNCTIONS' (dark blue). The main workspace on the right is a grid with a light gray background. A green 'ON START' block is attached to the top of the grid. Below it, an orange 'WAIT FOR IN2 TO BE on' block is connected. Underneath that, a light blue 'SEND SCROLLING TEXT "HELLO WORLD" TO ROUND MATRIX' block is connected. The text 'HELLO WORLD' is enclosed in a white rounded rectangle within the scrolling text block.

# Hello World

- Make Hello World display each time the button is pressed

The screenshot displays the Fuse LittleBits code editor interface. At the top, there is a purple header with the 'fuse littleBits' logo on the left, a 'Create New' button in the center, and a 'Getting Started?' button on the right. Below the header, the interface is divided into a left sidebar and a main workspace. The sidebar contains a 'Build' button and a 'Code' button, followed by a vertical list of category buttons: DISPLAY (purple), TEXT (light blue), INPUT (pink), OUTPUT (green), TIMING (orange), SOUND (red), LOOPS (dark green), LOGIC (blue), MATH (purple), VARIABLES (dark red), and FUNCTIONS (dark blue). The main workspace features a grid background and a green 'ON START' block. Attached to the 'ON START' block is an orange 'WAIT FOR IN2 TO BE on' block, and below it is a light blue 'SEND SCROLLING TEXT "HELLO WORLD" TO ROUND MATRIX' block. A plus sign icon is visible at the end of the scrolling text block.

# Hello World

- Make Hello World display each time the button is pressed



Blink Light



# Blink Light



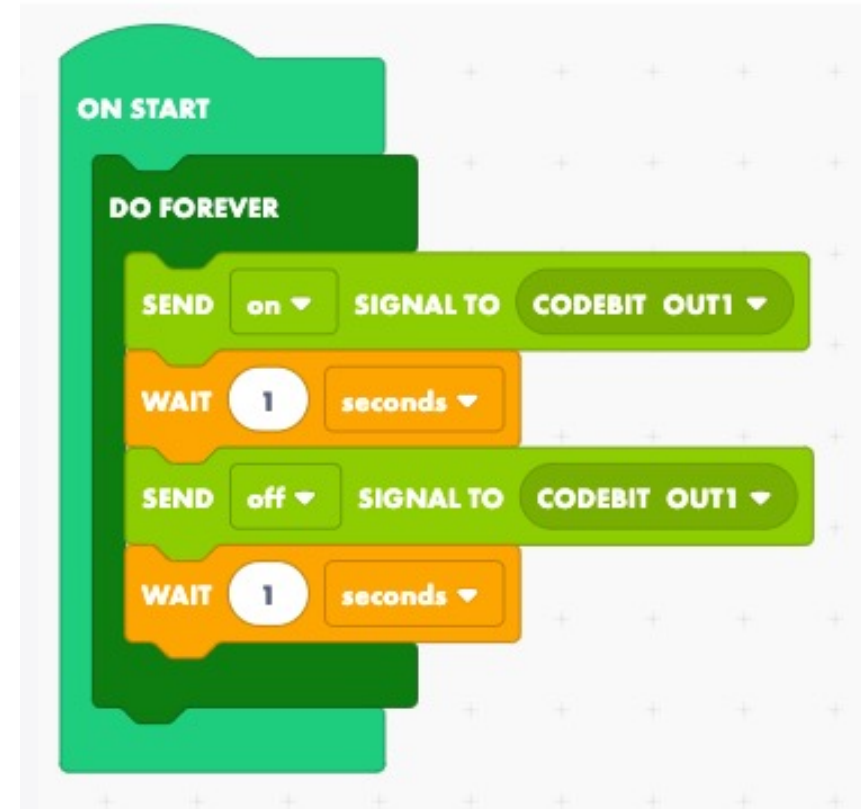
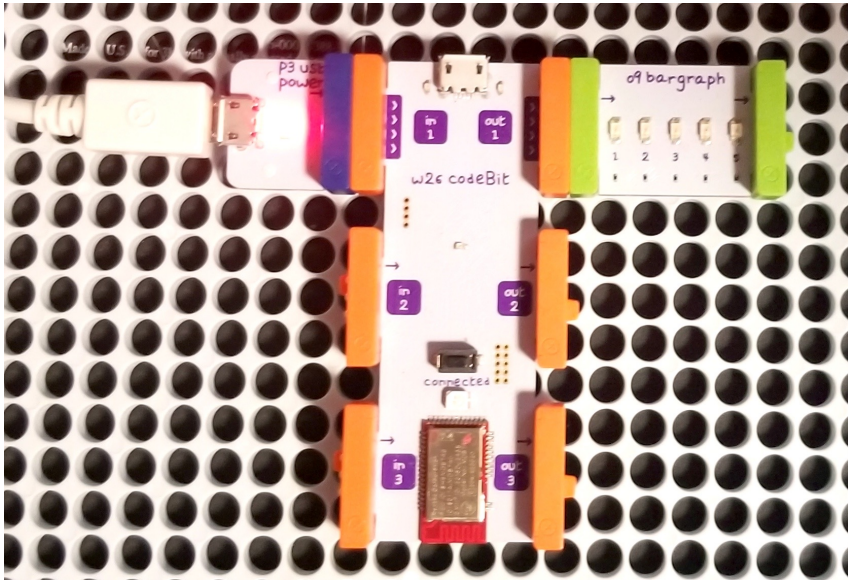
- Hardware
- Software

# Blink Light – v1

- Same Hardware

The screenshot displays the Fuse IDE interface for a 'Blink Light' project. The top navigation bar features the 'fuse littleBits' logo and a 'Create New' button. Below this, a sidebar on the left contains tabs for 'Build' and 'Code', and a vertical menu with categories: 'DISPLAY', 'TEXT', 'INPUT', 'OUTPUT', 'TIMING', and 'SOUND'. The main workspace shows a block-based program structure. It begins with an 'ON START' block, followed by a 'DO FOREVER' loop. Inside the loop, the sequence of blocks is: a 'SEND' block with a red circle icon and the text 'COLOR IMAGE TO ROUND MATRIX ON OUT1', a 'WAIT 1 seconds' block, a 'SEND' block with a black circle icon and the text 'COLOR IMAGE TO ROUND MATRIX ON OUT1', and another 'WAIT 1 seconds' block.

# Blink Light – v2



# Switch Controlled Light

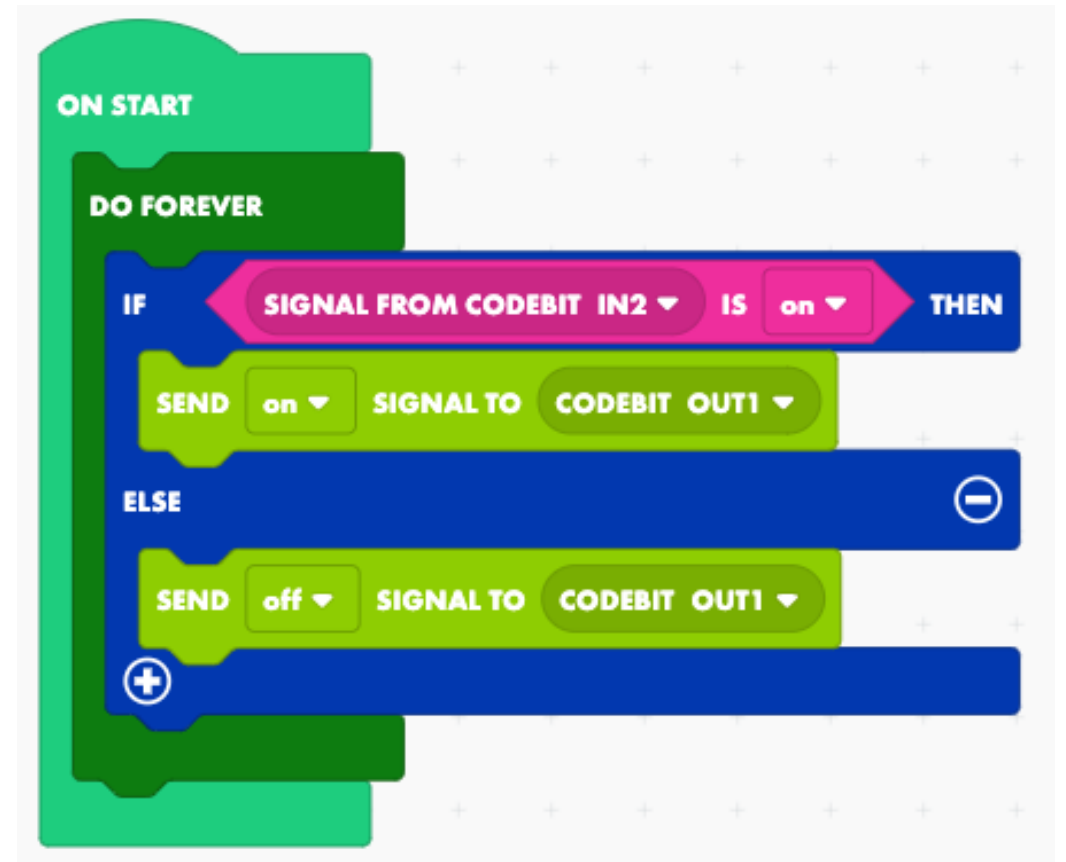
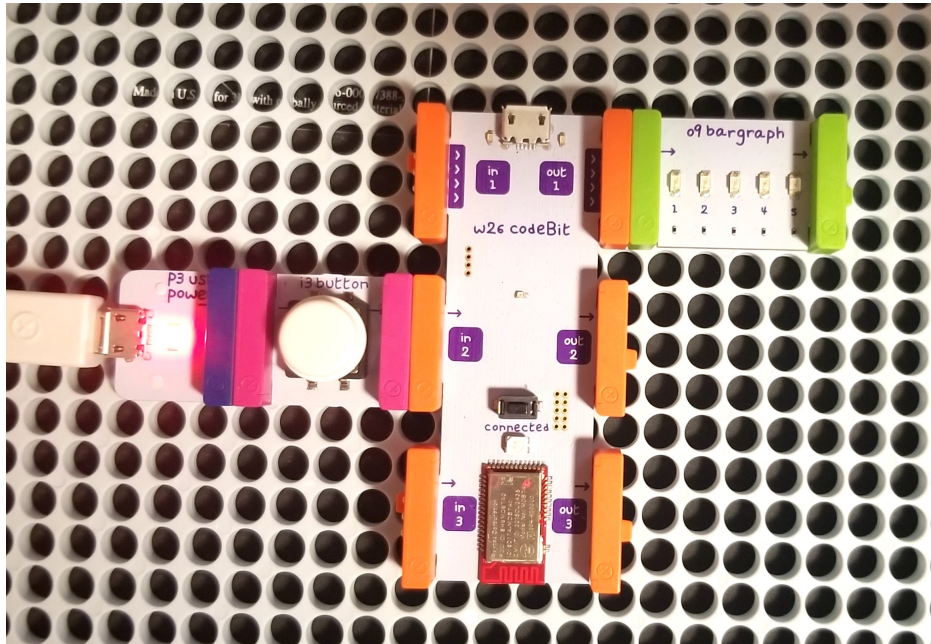


# Switch Controlled Light

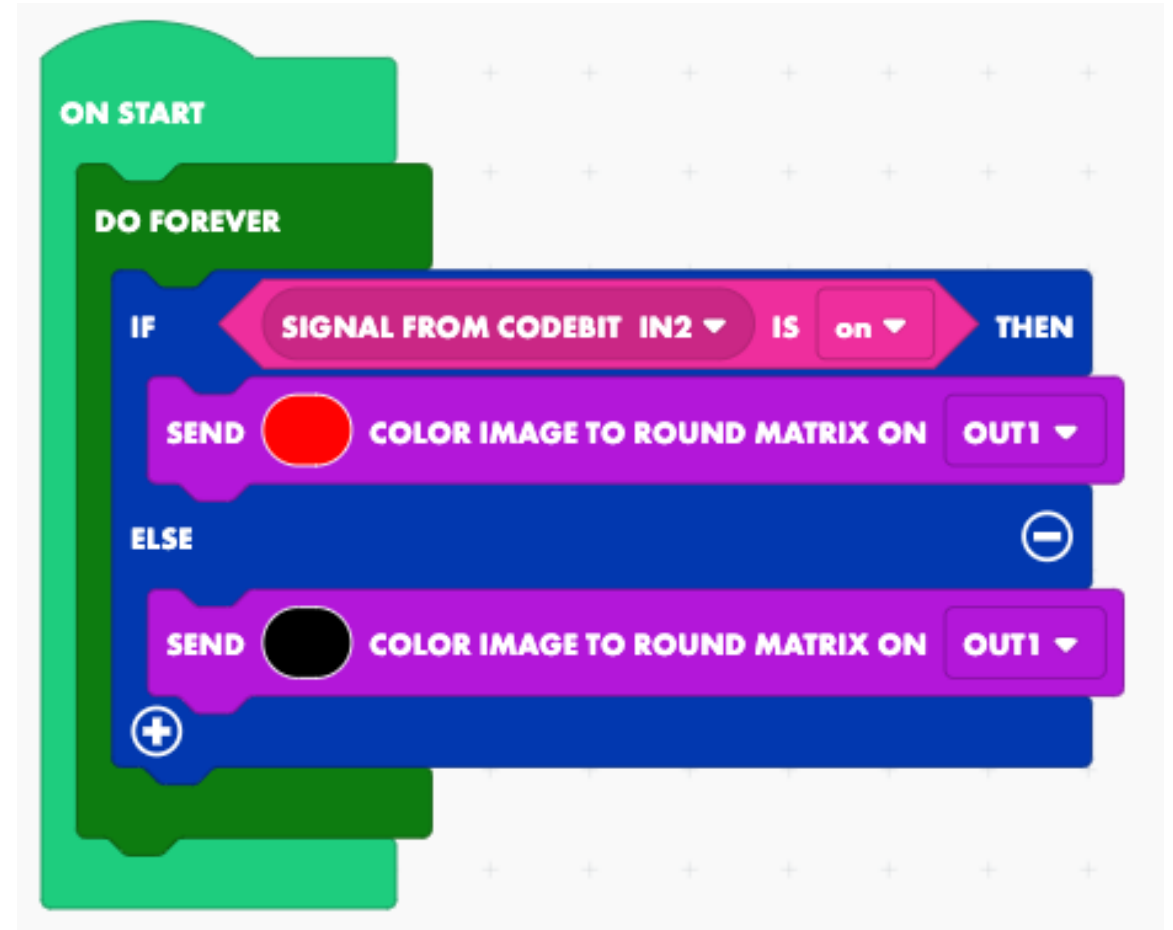
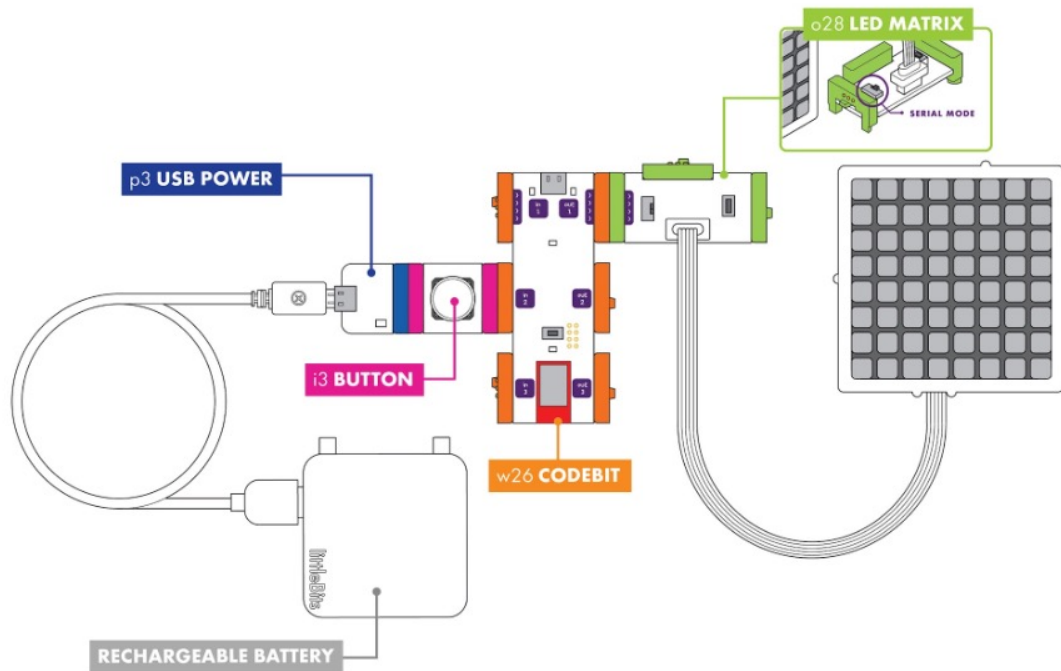


- Hardware
- Software

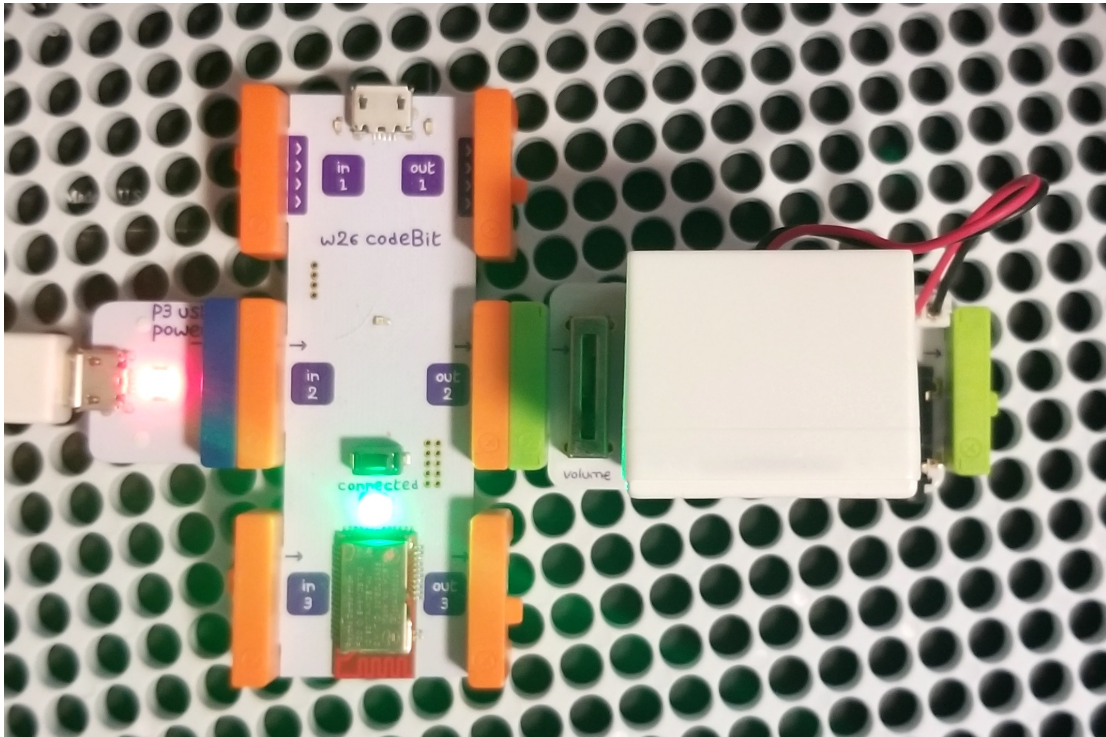
# Slide Switch Controlled Light - v1



# Slide Switch Controlled Light – v2



# Sound



# Switch Controlled Sound

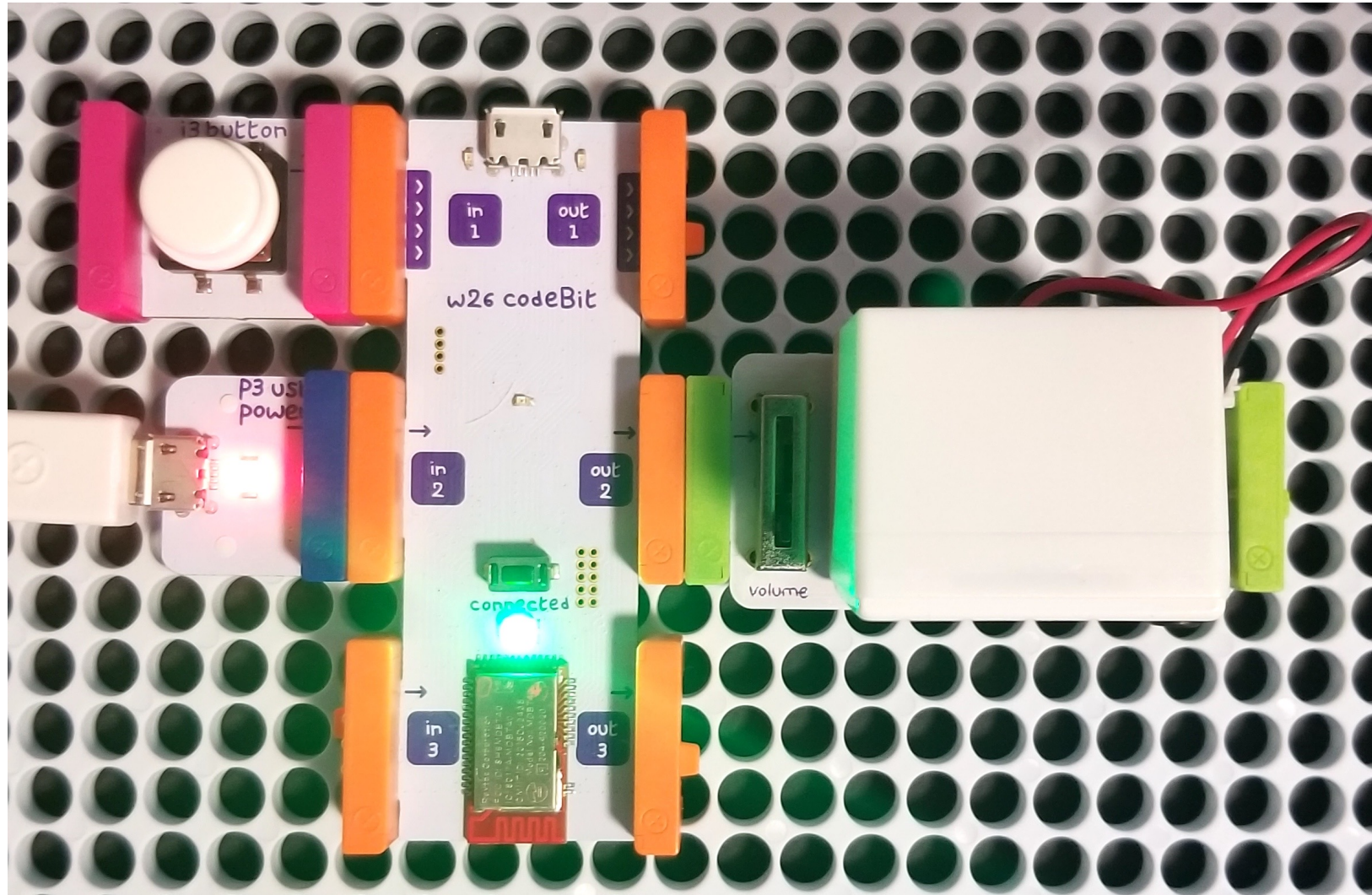


# Switch Controlled Sound



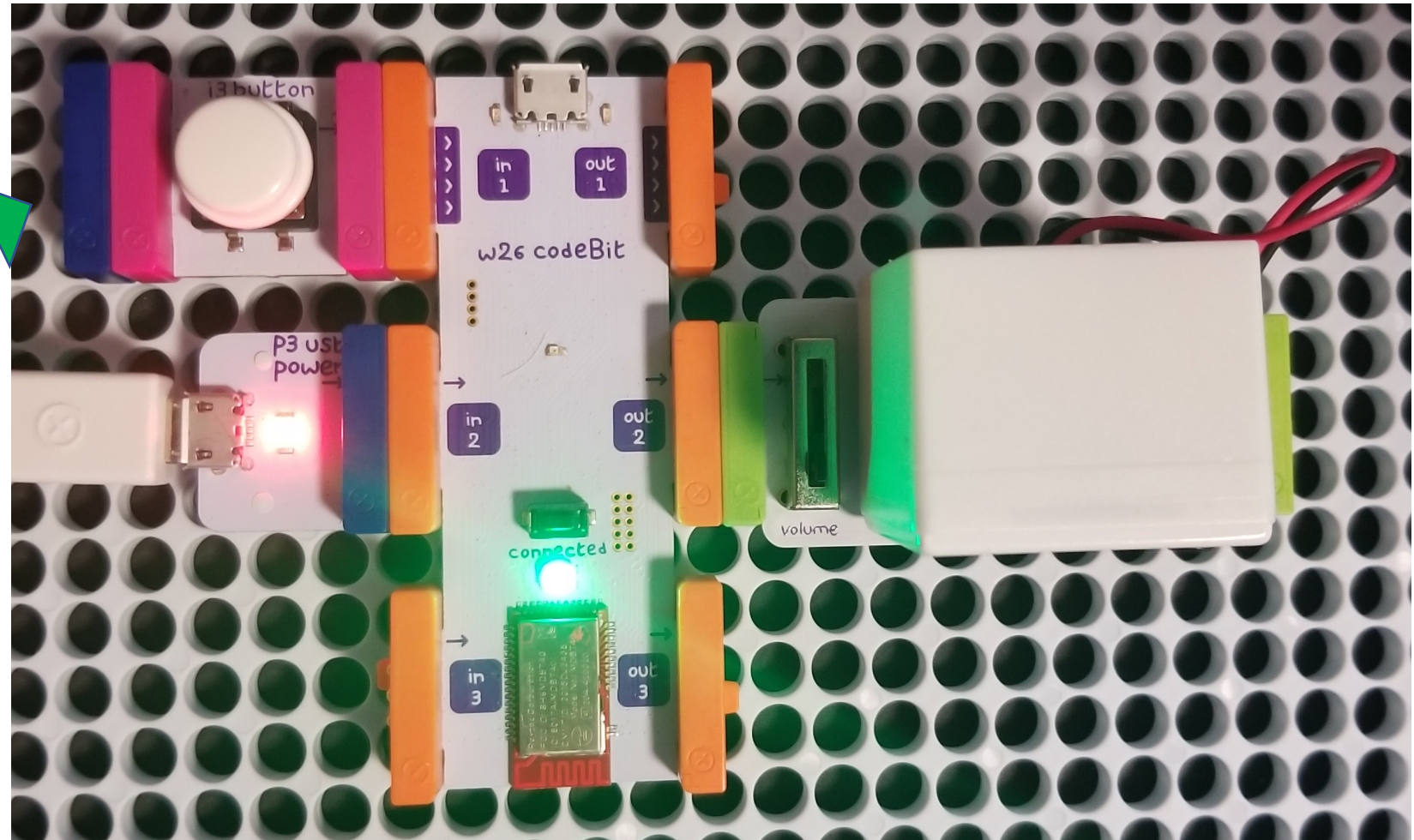
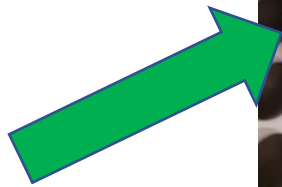
- Hardware
- Software

# Switch Controlled Sound

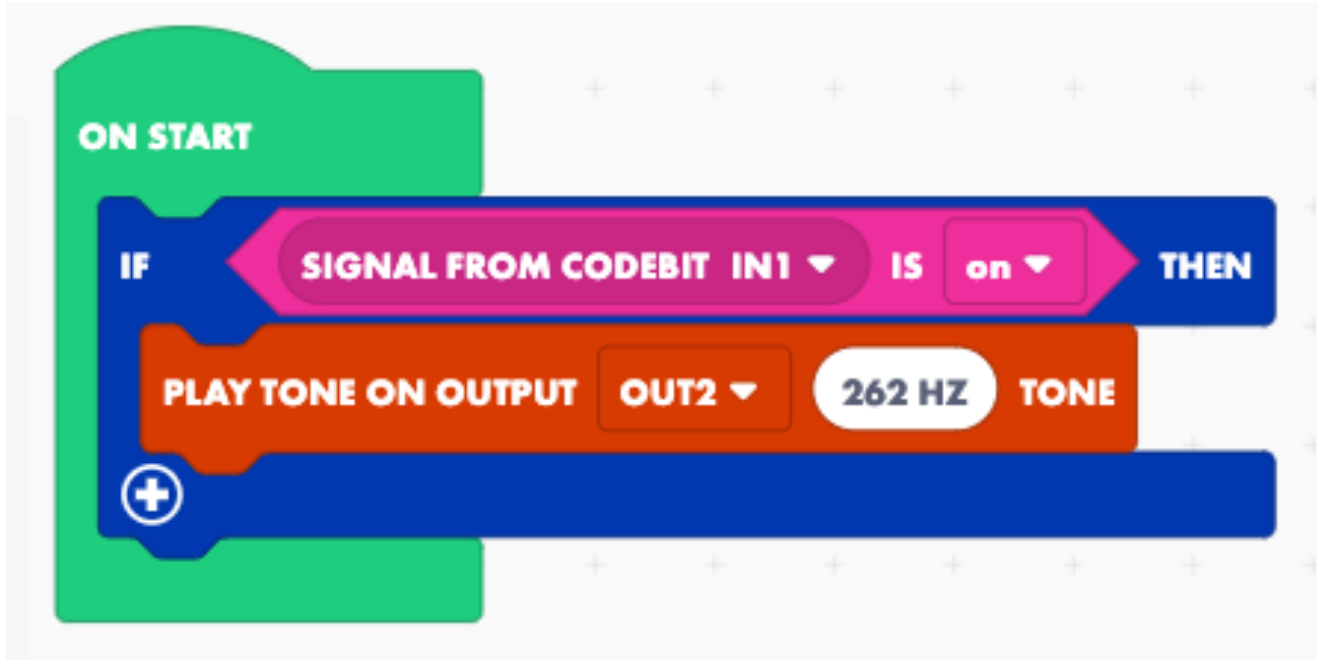


# Switch Controlled Sound

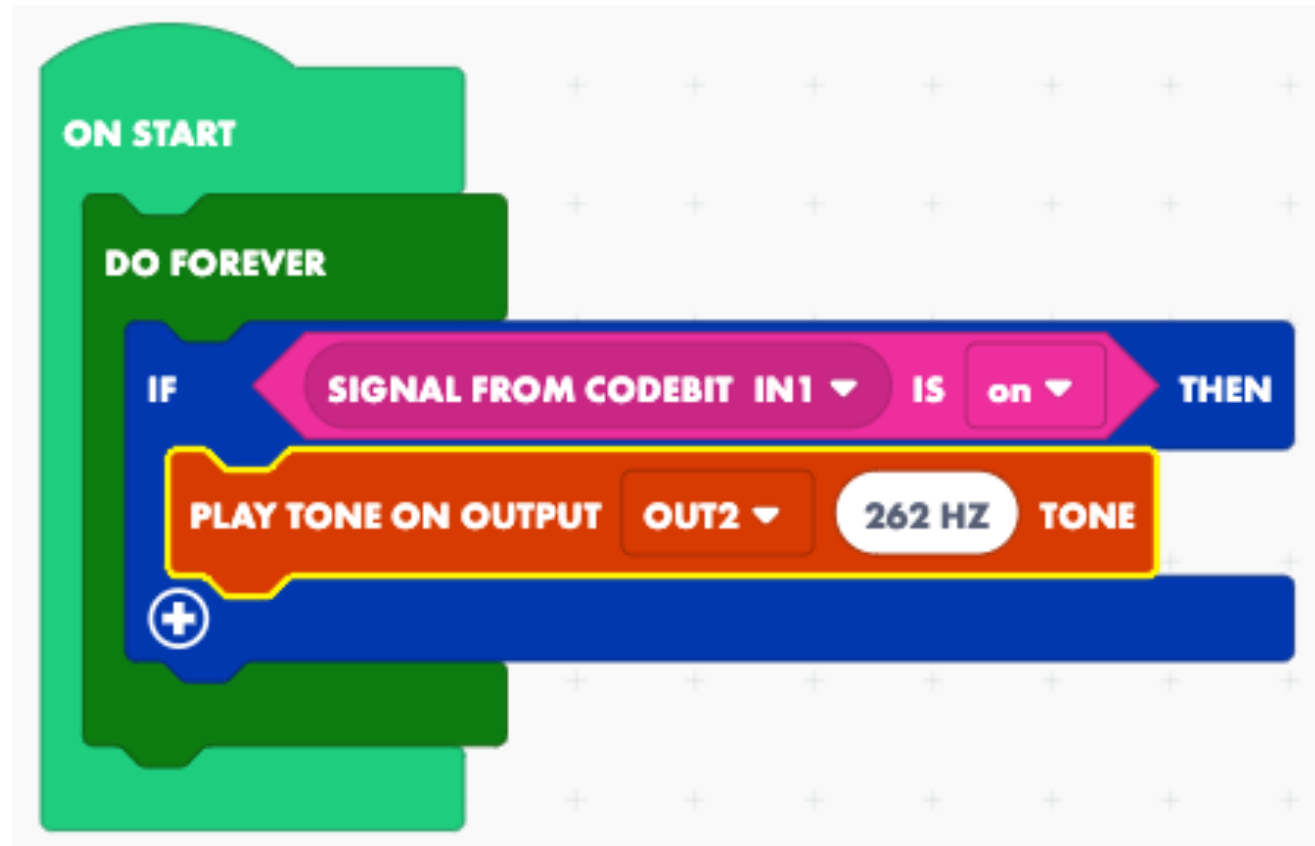
Addition of powerBit



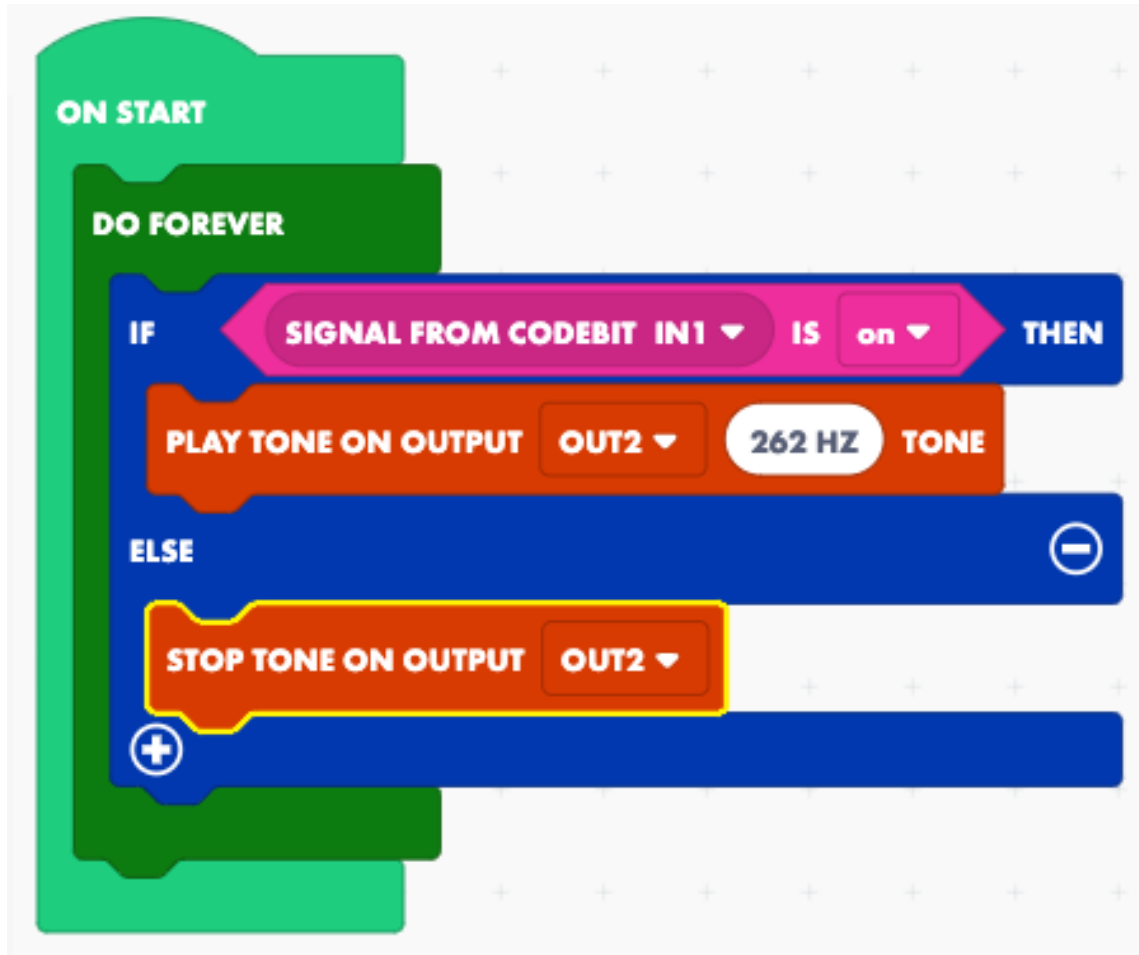
# Switch Controlled Sound



# Switch Controlled Sound



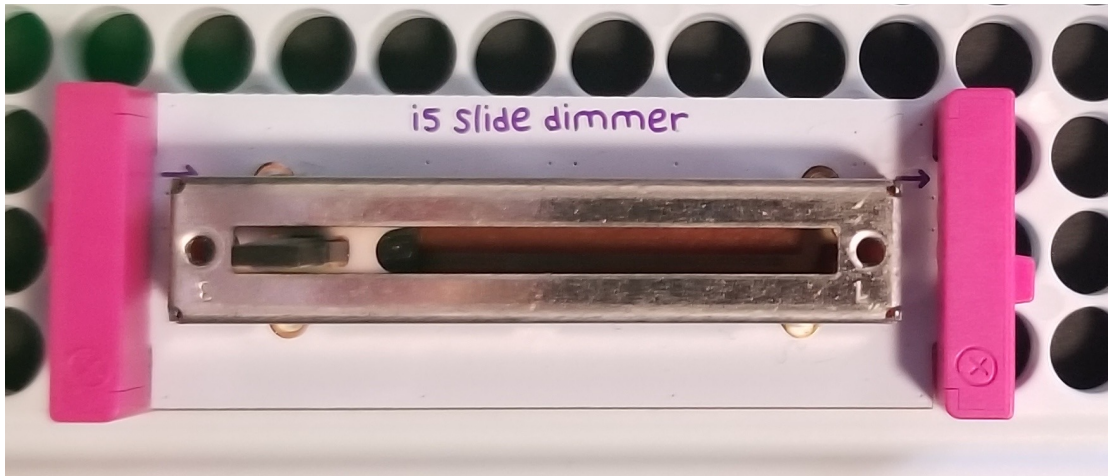
# Switch Controlled Sound



# Analog vs Digital



# Slide Switch



# Slide Switch Controlled Sound



- Hardware
- Software

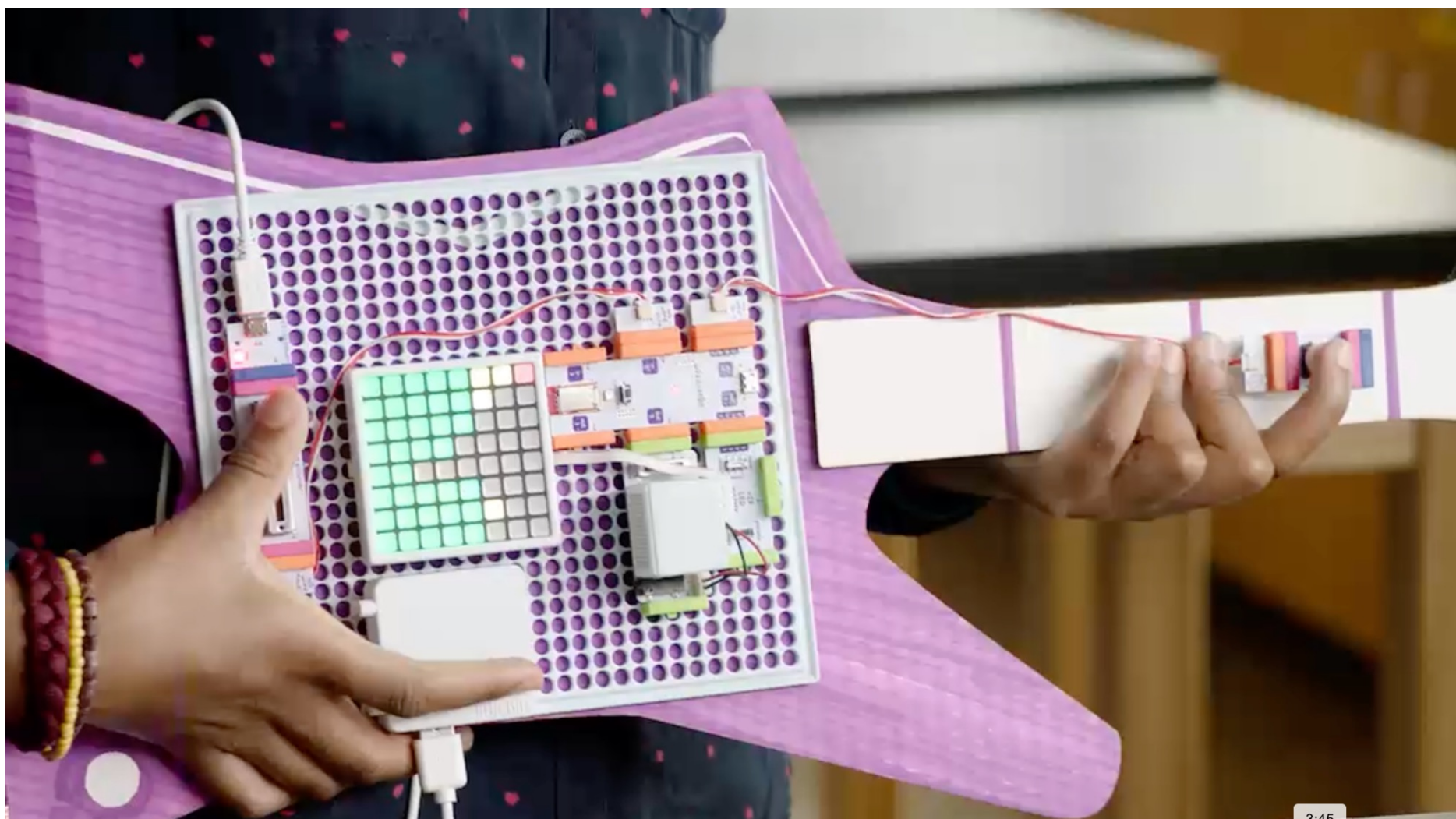
# Slide Switch Controlled Sound



# Slide Switch Controlled Sound



# Rockstar Guitar

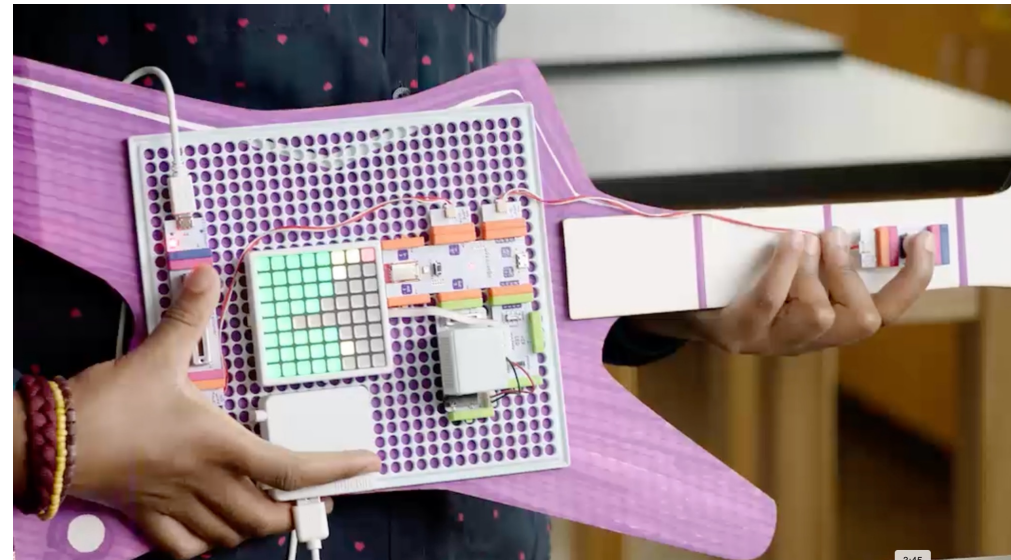


# Rockstar Guitar

- Use push button switch to turn sound on and off
- Use slide switch to control pitch of sound

## Extra Credit

- Use Round LED matrix to display graphic



# Slide Switch Controlled Sound

Use push button switch to turn sound on and off  
Use slide switch to control pitch of sound

The image shows a Scratch script for a slide switch controlled sound project. The script is contained within a 'DO FOREVER' loop. It starts with an 'ON START' block. The main logic consists of four 'IF' blocks, each checking the value of a signal from 'IN1' (rounded and converted to a number between 1 and 4) against a specific threshold. If the signal value is greater than or equal to the threshold, a tone is played on 'OUT2' with a specific frequency. The frequencies are 262 Hz, 294 Hz, 330 Hz, and 349 Hz, corresponding to the thresholds 1, 2, 3, and 4 respectively. The 'IF' blocks are connected by plus signs, indicating they are executed in sequence. The 'DO FOREVER' loop is highlighted in green, and the 'ON START' block is highlighted in blue.

```
ON START
DO FOREVER
  IF (ROUND (SIGNAL FROM IN1) CONVERTED TO NUMBER BETWEEN 1 AND 4) >= 1 THEN
    PLAY TONE ON OUTPUT OUT2 262 HZ TONE
  +
  IF (ROUND (SIGNAL FROM IN1) CONVERTED TO NUMBER BETWEEN 1 AND 4) >= 2 THEN
    PLAY TONE ON OUTPUT OUT2 294 TONE
  +
  IF (ROUND (SIGNAL FROM IN1) CONVERTED TO NUMBER BETWEEN 1 AND 4) >= 3 THEN
    PLAY TONE ON OUTPUT OUT2 330 TONE
  +
  IF (ROUND (SIGNAL FROM IN1) CONVERTED TO NUMBER BETWEEN 1 AND 4) >= 4 THEN
    PLAY TONE ON OUTPUT OUT2 349 TONE
  +
```