

# **Physical Computing ('21-'22)**

**Innovations in Computing**

**Resources**

Name(s) \_\_\_\_\_ Period \_\_\_\_\_ Date \_\_\_\_\_

## Activity Guide - Computing Innovations



### Innovation Research

Choose one of the following topics, and research the latest innovations in computing hardware. The goal here is to find the **most recent** innovative computing devices within your chosen topic. Keep an eye out in particular for devices that don't *look* like what you might expect a computer to be.

#### Topics (check the one you've selected)

- ☐ **Wearable Technology** (eg. clothing, jewelry, or accessories with built-in computers)
- ☐ **Health and Safety** (eg. devices that treat disease, track your health, or protect users from danger)
- ☐ **Agriculture** (eg. technology to improve the effectiveness, sustainability, or efficiency of farming)
- ☐ **Manufacturing** (eg. advancements in rapid prototyping, industrial robotics, and the production of goods)
- ☐ **Art and Design** (eg. interactive art or public installations)
- ☐ **Smart Home** (eg. devices that allow you to interact with your thermostat, locks, or lights using computers)

#### Researching your Topic

With your chosen topic as guidance, go online to research recent innovative computing devices within that topic. Try to find a product that you think is both innovative (in that it's attempting to solve a new problem, or an old problem in a new way) and personally interesting. Visit Code Studio for some recommended sites to kick off your research, as well as more detailed descriptions of each of the topics. As you do your research, consider checking out some of the crowdfunding sites (such as Kickstarter or Indiegogo) to find products that haven't even been released yet!

Use the space below to record notes about interesting products you find, patterns that you're seeing, or problems within your chosen topic that people are trying to address.

#### Research Notes

# An Innovative Solution

Based on the research your group did on the last page, select **one** of the devices you found to focus on. Answer the following questions for your chosen device.

You may need to head back online to gather more details about your chosen device.

## What Problem Does it Solve?

*This is probably the main sales pitch of the product - why do the creators think this is useful?*

## What Is Innovative About It?

*What makes this device different or better than other solutions out there?*

## How Do You Interact With It?

*Focusing on the Input and Output elements of our model for a computer, how does this device take input from the user, and how does it display output? Try to be as specific as possible.*

## How Could You Improve It?

*What are some changes that could make this device better? Are there common complaints, or clear issues that you might be able to address?*

# **Physical Computing ('21-'22)**

**Input Unplugged**

**Resources**

# Activity Guide - Input and Events: Program



## Your Role: Program

### Version A: Asking for Input

#### Your Role

Your role is the **Program**. It's your job to ask each of the Inputs for their current value whenever the code says to. If the current value of a given Input matches your conditional, then **you perform** whatever behavior is in the conditional.

#### Other Roles

Each of the other members of your group is an Input. The Inputs will each draw a card every time the draw loop runs, which they will then show to you if asked.

#### Rules

- Read through the Program aloud
- When you reach an input block, ask the appropriate input for their current value
- The Input will show you their card. If it matches your condition, perform the associated action

Run the program to the right a few times, changing roles each time so that every group member gets a chance to be the Program. Once everyone is comfortable with the system, try coming up with some programs of your own.

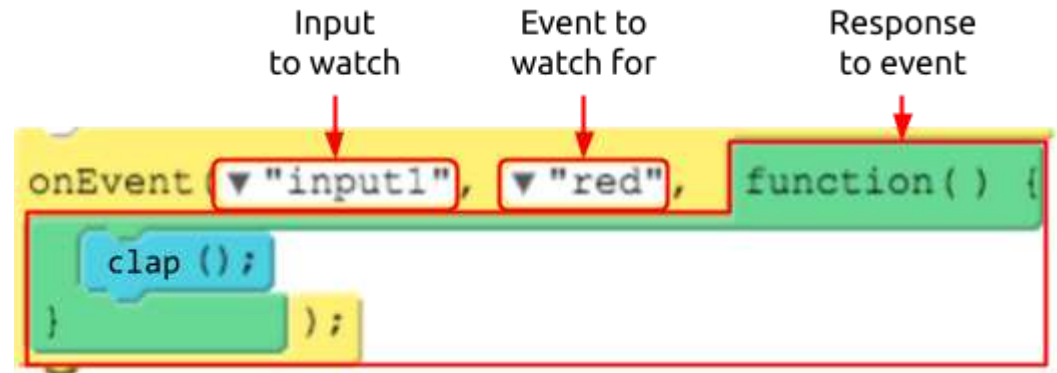
Program	Action
<code>function draw() {</code>	Start loop, each Input takes one card from the deck
<code>  if (inputOne() == "red") {</code> <code>    clap()</code> <code>  }</code>	Program asks Input One for input Input One shows their card to the Program. If the card is red, Program claps.
<code>  if (inputTwo() == "red") {</code> <code>    stand()</code> <code>  }</code>	Program asks Input Two for input Input Two shows their card to the Program. If the card is red, Program stands.
<code>  if (inputThree() == "red") {</code> <code>    sit()</code> <code>  }</code>	Program asks Input Three for input Input Three shows their card to the Program. If the card is red, Program sits.
<code>}</code>	Return to start

# Your Role: Program

## Version B: Input Events

In this version the Program *does not* continually ask for values and check them with conditionals. Instead, the Program assigns each Input an *Event* to watch for.

Once an Input knows what Event to watch for, the program no longer has to worry about it. It's now the Input's job to report back to the Program if they see the event they've been assigned. A red card being drawn is considered a "red" event, while a black card could be a "black" event.



### Rules

- Inputs draw cards off the top of the deck continually at a rate of roughly one card per second
- The Program reads the code aloud
- When the Program reads an `onEvent()` block, tell the appropriate Input what Event to watch for, and how to respond
- When an Input draws a card that matches an event that they're watching, they tell the Program how to respond.

Program	Action
	All Inputs begin drawing cards from the deck once a second
<code>onEvent("inputOne", "red", function() { clap(); })</code>	Input One starts looking for red cards. Whenever Input One draws a red card is drawn they tell Program to clap.
<code>onEvent("inputTwo", "black", function() { jump(); })</code>	Input Two starts looking for black cards. Whenever Input Two draws a black card they Program to jump.
<code>onEvent("inputThree", "red", function() { snap(); })</code>	Input Three starts looking for red cards. Whenever Input Three draws a red card, they tell Program to snap.

As before, run the program a few times, changing up the roles so everyone gets a chance to be the Program. Once you've got the hang of it you can try speeding up the pace of card drawing or writing programs of your own to test. If you write a new program, make sure that the Program is the one who still "runs" all of the code and performs the action.

## Your Role: Input 1

### Version A: Asking for Input

#### Your Role

Your role is the **Input 1**. It's your job to represent an input to the program by drawing a new card from the deck every time the draw loop runs, showing it to the Program if asked. The program will then decide whether to perform an action based on your input.

#### Other Roles

- **Program** - Runs the main program and asks for input when appropriate.
- **Inputs 2 and 3** - At the beginning of each draw loop, selects a card from the deck.

#### Rules

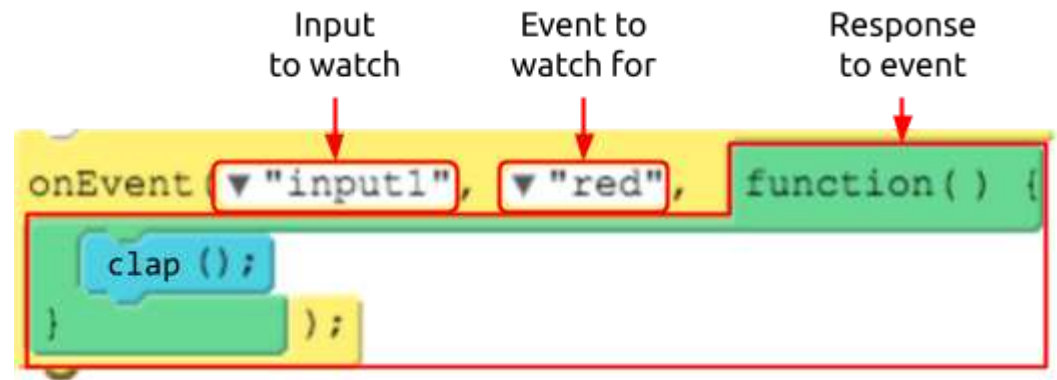
- Every time the draw loop is run, each Input draws a new card from the deck.
- If the Program asks you for input, show your card.

# Your Role: Input 1

## Model B: Input Events

In this version the Program *does not* continually ask for values and check them with conditionals. Instead, the Program assigns each Input an *Event* to watch for.

Once an Input knows what Event to watch for, the program no longer has to worry about it. It's now the Input's job to report back to the Program if they see the event they've been assigned. A red card being drawn is considered a "red" event, while a black card could be a "black" event.



### Rules

- When the program begins, start drawing cards from the deck at a rate of roughly one per second
- If the Program assigns you an Event, write down the details in your Events to Watch table
- If you draw a card that matches one of the Events in your table, tell the Program to do whatever is in the *Response* column

## Input 1 Events to Watch

Event	Response

As before, run the program a few times, changing up the roles so everyone gets a chance to be the Program. Once you've got the hang of it you can try speeding up the pace of card drawing or writing programs of your own to test. If you write a new program, make sure that the Program is the one who still "runs" all of the code and performs the action.



## Your Role: Input 2

### Version A: Asking for Input

#### Your Role

Your role is the **Input 2**. It's your job to represent an input to the program by drawing a new card from the deck every time the draw loop runs, showing it to the Program if asked. The program will then decide whether to perform an action based on your input.

#### Other Roles

- **Program** - Runs the main program and asks for input when appropriate.
- **Inputs 1 and 3** - At the beginning of each draw loop, selects a card from the deck.

#### Rules

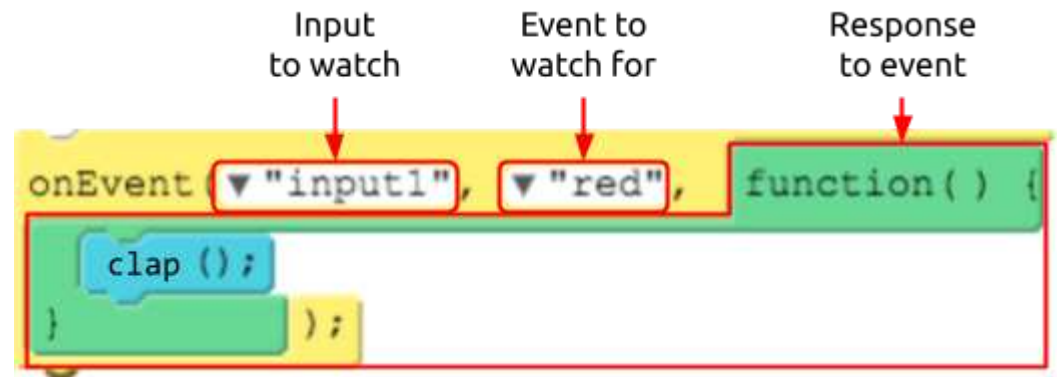
- Every time the draw loop is run, each Input draws a new card from the deck.
- If the Program asks you for input, show your card.

# Your Role: Input 2

## Model B: Input Events

In this version the Program *does not* continually ask for values and check them with conditionals. Instead, the Program assigns each Input an *Event* to watch for.

Once an Input knows what Event to watch for, the program no longer has to worry about it. It's now the Input's job to report back to the Program if they see the event they've been assigned. A red card being drawn is considered a "red" event, while a black card could be a "black" event.



### Rules

- When the program begins, start drawing cards from the deck at a rate of roughly one per second.
- If the Program assigns you an Event, write down the details in your Events to Watch table.
- If you draw a card that matches one of the Events in your table, tell the Program to do whatever is in the *Response* column.

## Input 2 Events to Watch

Event	Response

As before, run the program a few times, changing up the roles so everyone gets a chance to be the Program. Once you've got the hang of it you can try speeding up the pace of card drawing or writing programs of your own to test. If you write a new program, make sure that the Program is the one who still "runs" all of the code and performs the action.

## Your Role: Input 3

### Version A: Asking for Input

#### Your Role

Your role is the **Input 3**. It's your job to represent an input to the program by drawing a new card from the deck every time the draw loop runs, showing it to the Program if asked. The program will then decide whether to perform an action based on your input.

#### Other Roles

- **Program** - Runs the main program and asks for input when appropriate.
- **Inputs 1 and 2** - At the beginning of each draw loop, selects a card from the deck.

#### Rules

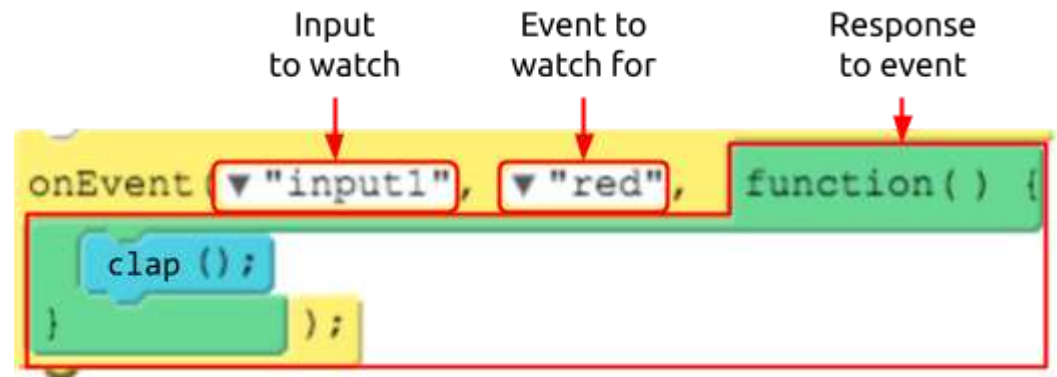
- Every time the draw loop is run, each Input draws a new card from the deck.
- If the Program asks you for input, show your card.

## Your Role: Input 3

### Model B: Input Events

In this version the Program *does not* continually ask for values and check them with conditionals. Instead, the Program assigns each Input an *Event* to watch for.

Once an Input knows what Event to watch for, the program no longer has to worry about it. It's now the Input's job to report back to the Program if they see the event they've been assigned. A red card being drawn is considered a "red" event, while a black card could be a "black" event.



#### Rules

- When the program begins, start drawing cards from the deck at a rate of roughly one per second.
- If the Program assigns you an Event, write down the details in your Events to Watch table.
- If you draw a card that matches one of the Events in your table, tell the Program to do whatever is in the *Response* column.

### Input 3 Events to Watch

Event	Response

As before, run the program a few times, changing up the roles so everyone gets a chance to be the Program. Once you've got the hang of it you can try speeding up the pace of card drawing or writing programs of your own to test. If you write a new program, make sure that the Program is the one who still "runs" all of the code and performs the action.

# **Physical Computing ('21-'22)**

## **The Program Design Process**

### **Resources**

Name(s) \_\_\_\_\_ Period \_\_\_\_\_ Date \_\_\_\_\_



# Project Guide - Emoji Race

## Overview

Building a larger piece of software like a game can quickly get complex. Starting with a plan can help you stay organized and identify issues ahead of time. A lot of the work you do here will make it much easier to keep track of what you need to do once you begin writing your actual code.

## Program Goal and Design

Start by thinking about what your game actually does. How will the user interact with it? How does it communicate information to the player? What will make it fun, interesting, or relevant to the player?

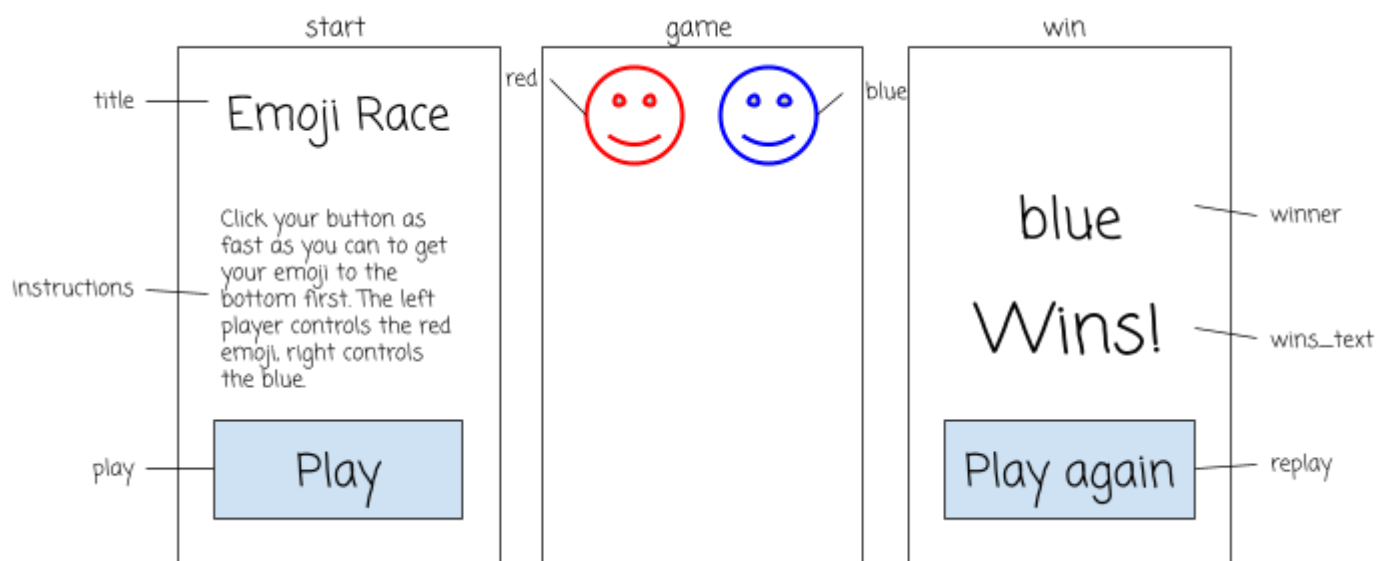
## Describe Your Program

In a couple of sentences describe the program you are going to build and how it will work.

Two players are competing clicker race. Each player needs to click one of the Circuit Playground buttons as fast as possible to move their emoji to the bottom of the screen. Whoever gets to the bottom first. The buzzer plays a high note if the red player wins and a low note if the blue player wins.

## Draw Your Screen(s)

Draw a quick sketch of the screen(s) you'll need. What design elements will you use? What should their IDs be?



# Circuit Playground

Which components of the Circuit Playground does this program use? Make sure that you are using at least one input (eg buttons and sensors) and one output (eg LED or buzzer).

Board Component	What it is Used For
buttonL	Player one clicks buttonL to move the red player down
buttonR	Player two clicks buttonR to move the blue player down
buzzer	Plays when the game is over

## Events and Functions

Using the description of your program above, figure out what events you'll need to respond to and which functions

### Events

In the table below list information about all of the events that your program will use, and what will happen when they are triggered

Name or ID	Event Type (eg "click")	Description (What happens when this event occurs?)
buttonL	press	Move the red emoji down the screen by 10 pixels and check to see if they have reached the bottom
buttonR	press	Move the blue emoji down the screen by 10 pixels and check to see if they have reached the bottom
"play"	click	Reset players to the top of the screen and change to the "game" screen
"replay"	click	Reset players to the top of the screen and change to the "game" screen

## Functions

Your events shouldn't have a lot of complex code. Instead, break your program up into the major steps you'll need for it to work. The different behaviors you described in your events should help you decide what these steps should be.

Function name	Parameters (Inputs to the function)	How It Changes During the Program (What's the starting value, when will it change?)
startGameO		Resets the players to the top of the screen and changes to the "game" screen
movePlayerO	player	Moves the specified player 10 pixels down the screen
checkWinO	player	Checks to see if the specified player has reached the bottom.. If so, set the text of "winner" to the player, switch to the "win" screen, and buzz

## Additional Notes

Use this area to take any extra notes that you might need to complete the program. This could include any variables that you might need, hardware setup for the board, or resources that you'll need to find (like images, sounds, etc), or ideas for more features that you want to explore.

In the movePlayerO and checkWinO functions, I'm going to use a variable player\_y to store the player's current y position so that I can add 10 to it and reset.

I still need to figure out how to keep the players from moving their emojis when they shouldn't (like when the "start" or "win" screens are showing).

It would be cool if the emojis started out frowning and then turned into meh and eventually smiling as they move down the screen.



# **Physical Computing ('21-'22)**

## **Project - Make a Game**

### **Resources**

Name(s) \_\_\_\_\_ Period \_\_\_\_\_ Date \_\_\_\_\_

# Computer Science Practices Reflection



How I've grown	Practice	How I want to grow
	Problem Solving	
	Persistence	
	Creativity	
	Collaboration	
	Communication	

Name(s) \_\_\_\_\_ Period \_\_\_\_\_ Date \_\_\_\_\_

# Project Guide - Make a Game



## Overview

Building a larger piece of software like a game can quickly get complex. Starting with a plan can help you stay organized and identify issues ahead of time. A lot of the work you do here will make it much easier to keep track of what you need to do once you begin writing your actual code.

## Program Goal and Design

Start by thinking about what your game actually does. How will the user interact with it? How does it communicate information to the player? What will make it fun, interesting, or relevant to the player?

## Describe Your Program

In a couple of sentences describe the program you are going to build and how it will work.

## Draw Your Screen(s)

Draw a quick sketch of the screen(s) you'll need. What design elements will you use? What should their IDs be?

# Circuit Playground

Which components of the Circuit Playground does this program use? Make sure that you are using at least one input (eg buttons and sensors) and one output (eg LED or buzzer).

Board Component	What it is Used For

## Events and Functions

Using the description of your program above, figure out what events you'll need to respond to and which functions

### Events

In the table below list information about all of the events that your program will use, and what will happen when they are triggered.

Name or ID	Event Type (eg "click")	Description (What happens when this event occurs?)

## Functions

Your events shouldn't have a lot of complex code. Instead, break your program up into the major steps you'll need for it to work. The different behaviors you described in your events should help you decide what these steps should be.

Function name	Parameters (Inputs to the function)	What does it do?

## Additional Notes

Use this area to take any extra notes that you might need to complete the program. This could include any variables that you might need, hardware setup for the board, or resources that you'll need to find (like images, sounds, etc), or ideas for more features that you want to explore.

## Unit 6 Lesson 9 Project Rubric

Key Concept	Extensive Evidence
Developing your program	<ul style="list-style-type: none"><li><input type="checkbox"/> Your program description and screen sketches show what the program does and how it will work</li><li><input type="checkbox"/> Your program description includes all relevant screen elements and their ID's</li><li><input type="checkbox"/> It is clear to the reader how the user would interact with your program.</li></ul>
Using board inputs and outputs to send information	<ul style="list-style-type: none"><li><input type="checkbox"/> Your program uses multiple board inputs and outputs</li><li><input type="checkbox"/> The purpose of each input and output is described in your program guide</li><li><input type="checkbox"/> Your program is able to send information to and from the board as described in your project guide</li></ul>
Events and user input	<ul style="list-style-type: none"><li><input type="checkbox"/> Your program uses multiple events</li><li><input type="checkbox"/> At least two events are triggered by a board input</li><li><input type="checkbox"/> Events, and what they trigger, are accurately described in your project guide</li></ul>
Use getters and setters to manipulate properties	<ul style="list-style-type: none"><li><input type="checkbox"/> You used getters and setters in all appropriate places in the program</li><li><input type="checkbox"/> You use setters and getters for various elements and property types</li></ul>

## Unit 6 Lesson 9 Project Rubric

Key Concept	Extensive Evidence	Convincing Evidence	Limited Evidence	No Evidence
Developing your program	Your program description and screen sketches show what the program does and how it will work, including all relevant screen elements and their ids. It is clear to the reader how the user would interact with your program.	Your program description and sketches generally show how it will work. Some screen elements and their ids are included. The reader may have a few minor questions about how a user would interact with your program.	Your program is described, but it may not be clear how it will work. There may be key elements or screens missing. The reader can understand the general purpose of your program, but they may not understand how it will work.	There is no description of your program, or the description and screen sketches are so vague that the reader cannot understand what your program does or how it will work.
Using board inputs and outputs to send information	Your program uses multiple board inputs and outputs, and the purpose of each is described in the program guide. Your program is able to send information to and from the board as described in the project guide.	Your program uses at least one board input and one board output, and the purpose of each is described in the program guide. Your program is generally able to send information to and from the board, but may have minor errors.	Your program uses at least one board input or output as described in the project guide. Other inputs or outputs may be included, but do not work as described.	Your program does not use the board as described in the program guide.
Events and user input	Your program uses multiple events, at least two of which are triggered by a board input. The events and what they trigger are accurately described in your project guide.	Your program uses at least one event that is triggered by a board input. The event and trigger is accurately described in your project guide.	Your program uses at least one event, but the behavior may not match what is described in your project guide.	Your program does not include any events, or the events do not work.
Use getters and setters to manipulate properties	You used getters and setters in all appropriate places in the program, for various elements and property types.	Your program makes use of getters and setters where appropriate and they generally work as designed.	Your program may use a getter or setter, but use is minimal or there are major errors that prevent them from running correctly.	Your program does not use getters and setters.

# **Physical Computing ('21-'22)**

## **Circuits and Physical Prototypes**

### **Resources**



Name(s) \_\_\_\_\_ Period \_\_\_\_\_ Date \_\_\_\_\_



# Project Guide - Smart Bike Prototype

## Overview

Designing a computing device that combines hardware and software requires a good deal of preparation. Starting with a clear plan can help you stay organized and identify issues ahead of time. A lot of the work you do here will make it much easier to keep track of what you need to do once you begin creating your device, both the physical and software components.

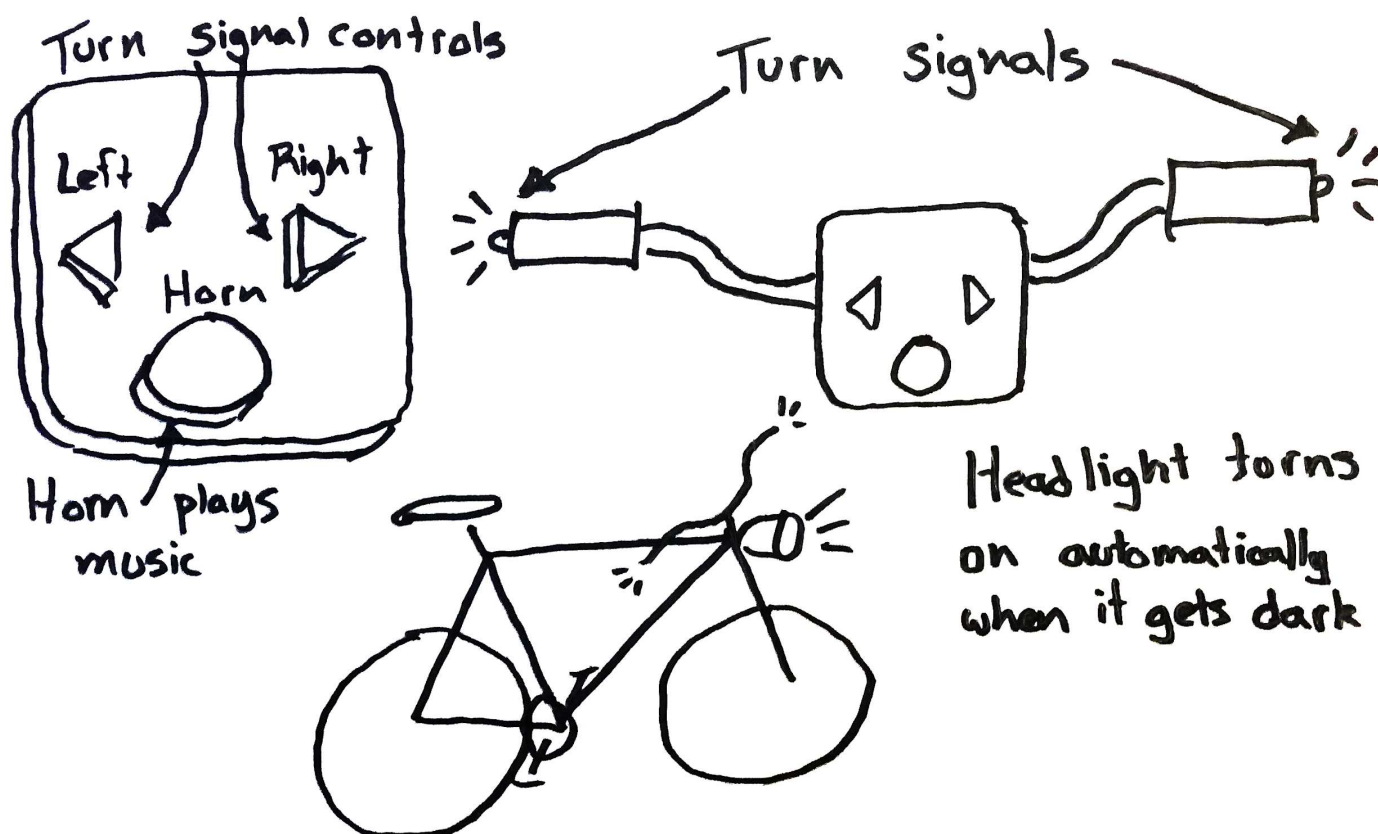
## Device Goal and Design

Start by thinking about what problem your device is going to solve. How will the user interact with it? How does it communicate information back to the user? What shape will it take?

## Sketch and Describe Your Device

Describe your device and roughly sketch out the main elements. Don't worry about making it pretty.

The smart bike controller is a computer that can be attached to the handlebars of any bike to turn a regular bike into a "smart" bike. It has controls to activate turn signals on the ends of the handlebars, a horn that can play a customizable tone or song, and automatic headlights that turn on when it gets dark. We might also include a speedometer and a safety sensor that detects when a car is too close.



## Inputs

What inputs will your prototype need to function? What will they be used for?

Input type	What it is used For
Buttons (at least 3)	Activating turn signals and horn
Light sensor	Turning on headlight automatically

## Outputs

What outputs will your prototype need? What will they communicate to the user?

Output Type	What does it communicate?
Handlebar LEDs	Blinking when the rider intends to turn. May also be used as an alert when cars get too close.
Buzzer	Acts as a horn. Could also be used as an alert when cars get too close.
Headlight	Bright LEDs to improve vision and visibility in the dark.

## Processing

How will you use the inputs to decide what the outputs should be? Break the program up into the major steps you'll need for it to work. The different behaviors you described in your events should help you decide what these steps should be.

Function name	Parameters (Inputs to the function)	How It Changes During the Program (What's the starting value, when will it change?)

## Additional Notes

Use this area to take any extra notes that you might need to complete the program. This could include any variables that you might need, hardware setup for the board, or resources that you'll need to find (like images, sounds, etc), or ideas for more features that you want to explore.

# **Physical Computing ('21-'22)**

**Project - Prototype an Innovation**

**Resources**

Name(s) \_\_\_\_\_ Period \_\_\_\_\_ Date \_\_\_\_\_

# Computer Science Practices Reflection



How I've grown	Practice	How I want to grow
	Problem Solving	
	Persistence	
	Creativity	
	Collaboration	
	Communication	

# Unit 6 Chapter 2 Project Student Rubric

Key Concept	Extensive Evidence	Convincing Evidence	Limited Evidence	No Evidence
Device Description	Your device description shows what the device does and how it will work. It is clear to the reader how the user would interact with the device.	Your device description generally shows how it will work. The reader may have a few minor questions about how a user would interact with the device.	Your device is described, but it may not be clear how it will work. The reader can understand the general purpose of the device, but may not understand how it will work.	There is no description of your device, or the description is so vague that the reader cannot understand what the device does or how it will work.
Functions	At least two functions with parameters are used. At least one of these functions is called multiple times in the program with different parameters.	At least two functions are used in your program. At least one of these functions uses parameters.	At least one function is used in your program.	There are no functions in your program, or the functions do not appear to serve any purpose.
The Program	Your program uses multiple board inputs and outputs. Your program is able to send information to and from the board as described in the project guide.	Your program uses at least one board input and one board output. Your program can send information to and from the board, but may have minor errors.	Your program uses at least one board input or output. Other inputs or outputs may be included, but do not work as described.	Your program does not use the board as described in the program guide.
Physical Prototype	Your physical prototype has been created as described in the project guide. It uses physical connections to function.	Your physical prototype has been mostly created as described in the project guide. It uses physical connections to function, but there may be minor bugs or missing pieces.	Your physical prototype has some characteristics as described in the project guide, but there may be major problems that prevent it from functioning properly.	Your physical prototype was not created, or it has no relationship to what was described in the project guide, or it has major problems that make it impossible to use.
Arrays and Loops	At least one array is used in your program, and a for loop iterates over the array accessing each element.	At least one array is used in your program, and multiple elements of the array are accessed.	An array is used in your program and at least one element is accessed.	There are no arrays in your program, or the elements are never accessed.

Name(s) \_\_\_\_\_ Period \_\_\_\_\_ Date \_\_\_\_\_

# Project Guide - Innovation Prototype



## Overview

Designing a computing device that combines hardware and software requires a good deal of preparation. Starting with a clear plan can help you stay organized and identify issues ahead of time. A lot of the work you do here will make it much easier to keep track of what you need to do once you begin creating your device, both the physical and software components.

## Device Goal and Design

Start by thinking about what problem your device is going to solve. How will the user interact with it? How does it communicate information back to the user? What shape will it take?

## Sketch and Describe Your Device

Describe your device and roughly sketch out the main elements. Don't worry about making it pretty.

## Inputs

What inputs will your prototype need to function? What will they be used for?

Input type	What it is used For

## Outputs

What outputs will your prototype need? What will they communicate to the user?

Output Type	What does it communicate?



## Processing

How will your program turn input into output? You won't be able to do everything at once. Instead, break your program up into the major steps you'll need for it to work. The different behaviors you described in your events should help you decide what these steps should be.

Function name	Parameters (Inputs to the function)	What does it do?

## Develop Your Prototype

Once your teacher has approved your design, use the materials you are given to develop your prototype.

## Test Your Prototype

Test your prototype with multiple users and, if time allows, incorporate feedback from your testing.

## Reflect

What part of your project are you most proud of? \_\_\_\_\_

Why? \_\_\_\_\_

If you had more time, what improvement would you make to your innovation?

---

---

## Unit 6 Chapter 2 Project - Student Checklist

Key Concept	Extensive Evidence
Device Description	<ul style="list-style-type: none"><li><input type="checkbox"/> Your device description shows what the device does and how it will work</li><li><input type="checkbox"/> Your device description includes the device main elements.</li><li><input type="checkbox"/> You've made it clear to the reader how the user would interact with the device.</li></ul>
Functions	<ul style="list-style-type: none"><li><input type="checkbox"/> At least two functions with parameters are used as described in your project guide</li><li><input type="checkbox"/> At least one of these functions is called multiple times in the program with different parameters.</li></ul>
The Program	<ul style="list-style-type: none"><li><input type="checkbox"/> Your program uses multiple board inputs and outputs</li><li><input type="checkbox"/> Your input and output purposes are described in the program guide</li><li><input type="checkbox"/> Your program is able to send information to and from the board as described in the project guide.</li></ul>
Physical Prototype	<ul style="list-style-type: none"><li><input type="checkbox"/> Your physical prototype has been created as described in the project guide</li><li><input type="checkbox"/> Physical connections to the board allow for the described functionality.</li></ul>
Arrays and Loops	<ul style="list-style-type: none"><li><input type="checkbox"/> At least one array is used in your program</li><li><input type="checkbox"/> A for loop iterates over the array and accesses each element.</li></ul>