

Lesson 6: Mini-Project - Field Collector App

90 minutes

Overview

In this mini-project, students will use the micro:bit to collect data, then use App Lab to analyze the data that was collected. This is similar to citizen science fieldwork or survey apps where a person collects data on a device "in the field", then returns to a central location to view and analyze the data (such as cataloguing endangered species in the wild). Students will use variables and events to collect data from the micro:bit, then use if-statements to make decisions or recommendations based on the data they collect.

Question of the Day: How can I use events and variables to create an app that collects and analyzes data?

Standards

Full Course Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming
- ▶ **CS** - Computing Systems

Agenda

Warm Up (5 minutes)

Journal

Activity (80 minutes)

Project Ideation

Creating the Project

Code Studio

(Optional) Gallery Walk

Wrap Up (5 minutes)

Reflection

Objectives

Students will be able to:

- Create a project that incorporates inputs from the micro:bit
- Create a project that incorporates outputs from the app screen
- Create a project that incorporates variables and if-statements

Preparation

- Review the Project Guide and Rubric to understand the scope of the project.
- Review the exemplar projects as examples of what students can create.
- (Optional) You may decide to show one of these projects to students to help spark ideas.
- Check the "**Teacher's Lounge**" forum for verified teachers to find additional strategies or resources shared by fellow teachers

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the teachers

- **Mini-Project: Field Collector App** - Slides [▼ Make a Copy](#)

For the students

- **Problem Solving Process with Programming** - Resource [▼ Make a Copy](#)

- **Project Guide - Field Collector App -**
Activity Guide [▼ Make a Copy](#)
- **Rubric - Field Collector App -** Rubric
[▼ Make a Copy](#)

Teaching Guide

Warm Up (5 minutes)

Journal

 **Prompt:** *Think about a time when you've needed to collect data to help solve a problem or answer a question, or when data is being collected about you. How was that data collected? What happened to that data?*

Circulate: Hopefully students may think of ways their phones or other devices collect data about them, such as completing a survey or checking a pedometer or when a car keeps track of how many miles someone has driven. Students may also think of situations where data is collected about them, such as when they are captured on video or when they use a credit card.

Share Out: Have students share their responses with a partner, then share out as a class

Discussion Goal: As students share answers, highlight that there is usually a device that does the data collecting, and then there is a separate service that analyzes or makes decisions about that data. For example, a watch may collect heart rate data, and then an app on your phone tells you how many calories you burned. Or you may answer a survey question, and then later you see the data from all the people who answered the same question.

Remarks

Today, we're going to complete a project where we design an app that uses the micro:bit to collect data, and then uses an app to make a decision or recommendation based on that data.

(Optional) Example Apps: Demonstrate for students one or two of the exemplar apps as an example of what students can create. This can help spark ideas, and give students a model of what to work towards when making their own devices. Since students begin this project by sketching their ideas on paper, we do not recommend passing out micro:bits and having students experiment yet - instead, it can be better to demonstrate as a full-class and discuss the apps together.

Teaching Tip

Example Projects & Circuit Playgrounds: Each of the exemplar projects was originally designed and created using a Circuit Playground, which was the device in our original curriculum. You may decide to show students the project guide and the design of the app, but the app itself will not work with a micro:bit. Even though these examples use a circuit playground, the types of projects you can make are equivalent between the two.

If you or your students create projects you would consider "examples" for this project, please share them with us by emailing support@code.org!

Extended Examples: You may decide to extend these examples to issues that are relevant to your local community - for example, instead of surveying birds, it may be more appropriate to relate this app to surveying marine life or desert fauna. Consider if there are any local initiatives in their "data gathering / survey" phase that could be an example for students as well

Real-World Connection: Gathering survey or location data to make a recommendation or create a narrative is a common practice amongst geographers who use Geographic Information Mapping (GIS) software. A student competition is held every year with [past winners available here](#) - you may decide to show students some of these projects as examples of how the prototype they will build could lead to a more realistic application

 **Question of the Day:** How can I use events and variables to create an app that collects and analyzes data?

Activity (80 minutes)

Teaching Tip

Facilitating Mini-Projects: Mini-Projects act as checkpoints in the curricula and cover the subset of skills students have seen so far in the unit. They are designed for 1-2 days of implementation as a way to check-in with how well students understand the course content so far. You may decide to extend these projects as a way to support or challenge students, which could allow you to revisit difficult concepts or support students who may have missed lessons and are trying to catch up. However, we recommend deciding this ahead of time and being firm with students about how much time they have for each project - otherwise, it's easy for projects to drag-out to multiple days and for student's work to spiral beyond the scope of this project.

Project Ideation

 **Distribute:** (Optional) pass out copies of the project guide. Students will use this to plan their project, and includes items that can be used for assessment at the end of the project.

Teaching Tip

Facilitating Group Projects: If students are working in pairs or small teams to complete projects, consider showing these two videos to the class:

- [How Teamwork Works](#)
- [Dealing with Disagreements](#)

Depending on your goals with this project, consider having teams complete a [Student Guide to Team Planning](#), which reinforces the message in the video

 **Slides:** Present an overview of the project and the steps students will take to complete it. There are multiple slides available to help explain the project.

(Optional) Class Brainstorm: Before diving into individual projects, consider leading a class brainstorm of what some example projects could look like that fit this criteria. You may also decide to show one or two of the exemplar apps to help inspire the brainstorm. Using these ideas, students may find it easier to latch onto an initial idea for their project

Teaching Tip

Scoping Student Projects: Students may ideate projects that are beyond the skills they currently have or that would take longer than the allotted time to implement. Rather than asking students to choose a different project, consider asking students to imagine a more scaled-down version of their initial idea. As an analogy, if students initial idea is the "Run" step, imagine a less intense version that represents what the "Walk" step would look like. If necessary, you can keep going back further to a "Crawl" step as well.

Digging Deeper: This is sometimes referred to as the Minimal Viable Product - you can learn more about this process and adapt it into your project strategies by reading this article: [Making Sense of MVP](#) by Henrik Kniberg

Creating the Project

Remarks

Many of you are ready to start creating your programs. One thing that could make this challenging is the *blank screen* effect: unlike previous levels, you won't have any starter code or direction on what to create. This means you might not be sure what exactly you're supposed to do, or you might run into bugs you need to fix which can be frustrating. Luckily, we can also use the problem-solving process to help with these types of projects as well!

Distribute: Hand out a copy of the [Problem Solving Process with Programming](#) to pairs of students. Encourage students to look over the guide.

 **Display:** Show the slide with the problem solving process graphic.

Remarks

If you feel stuck or you're not sure what to do next, remember you can always follow the steps of the problem-solving process to **define** your next step, **prepare** for what you want to code, **try** it out, then **reflect** on whether or not it solved your problem.

 **Transition:** Send students to Code Studio.

Code Studio



Troubleshooting Devices: If student devices aren't able to connect, they can click the Help and Tips tab and then access the Maker Setup page (click the image to see more). This will let them re-connect their board to App Lab.

If students continue to have issues, refer to the Troubleshooting section of our [Maker Support Article](#)

Circulate: Encourage students to use the steps in the Problem-Solving Process for Programming when they get stuck or are unsure of what to do next.

Teaching Tip

Using the Problem Solving Process: Not all bullets in the Problem Solving Process will be applicable to every problem a student has. Instead, encourage them to pick one or two from each category to try each time they are stuck.

This document is also available in the Help and Tips for each level on Code Studio.

Debugging Strategies: As students design and implement their own project ideas, they may find themselves with new bugs that they need to untangle and you may find yourself looking at completely unfamiliar code as students look for help troubleshooting their errors. To help smooth out the debugging experience, consider the following

strategies:

- Review the [Teacher Guide to Debugging](#) for some common questions and strategies to help support students in debugging their code
- Have students follow the steps in the [Student Guide to Debugging](#) and use the [Bug Report Quarter-Sheets](#) as an initial step in the debugging process. This helps students prepare and communicate their issue before asking for help.
- If students haven't seen it yet, consider showing the [Debugging Video](#) to the class to reinforce debugging best practices.

(Optional) Gallery Walk

Allow students to walk around the room and interact with each other's devices. Celebrate all of the different ideas that students were able to implement with the same basic code.

Teaching Tip

You may choose to formalize this process by having each student write down one positive quality of each project, or having students "draw names" to comment on one particular classmate's work.

Wrap Up (5 minutes)

Reflection

 **Prompt:** *What was your favorite part of the app you created? What is something you wish you had time to improve?*

Discussion Goal: You may decide to have students share these with a partner, or you may decide to have students keep their responses private. Either way, the goal of the prompt is to have students reflect on the apps they've created and celebrate what went well and acknowledge what could be improved for next time.



This work is available under a [Creative Commons License \(CC BY-NC-SA 4.0\)](#).

If you are interested in licensing Code.org materials for commercial purposes [contact us](#).