Let'sStartCoding:Education

Facilitator Guidebook: 8/9/16 Edition

Contents

First, thank you for your interest in using our tools and products for your organization!

Our Approach to Coding

We believe that most engineers didn't know they were engineering at first. They didn't start with datasheets and grand plans; they started by tearing apart a DVD player or a go-cart.

We think code should be the same way. By starting with something that works and tearing it down, learners can have fun at the same time that they learn.

It's OK to break the code. You can fix it.

Learning Goals for You

After following these documents and the recommended activities, you'll be able to:

1) Teach others to use Maker Board and Let's Start Coding resources
2) Help with common troubleshooting challenges
3) Guide a group discussion and pick projects that serve your learning goals

---

What You Won't Learn

The focus of these documents is *not* to teach:

1) How to engage or manage a classroom- you're the expert there!
2) How to write computer code
3) How to build specific projects and use components

2 and 3 are features of the product: it's built in! *Using* the products correctly will teach you these things.

Vocabulary

- **Hardware** refers to what you physically manipulate with the code. Hardware is made up of Maker Board, Carrier Board, and various passive components.
- **Software** is what we call the application you use to program Maker Board. It's the Arduino software.
- **Programs** are the lines of code you write to do something. We use this word interchangeably with **code.**
- **Physical Computing** is using inputs from the real world to interact with your computer *and/or* using code to act upon the real world.
    - Examples are using the temperature to turn on a fan or turning off a light when it's bright outside.

The World of Let's Start Coding

There are many parts interacting within one Let's Start Coding kit or project.

- Maker Board is **hardware.** It is a **microcontroller** designed by Let's Start Coding.
- Beginners use the Arduino **software** or Codebender software to write code for Maker Board. It is a free software distributed by a team of people.
    - Think of the Arduino software like Microsoft Word - it gives you a place to put your ideas and helps correct them.
- The sensors (sound, light, or temp sensor) and outputs (speaker, LEDs, motors, screens) you'll use are general **components**. There are thousands of these available, but we provide the ones easiest to use and learn with.
    - Sensors and outputs fall under **hardware**.
- We provide example **programs**, which are lines of code that make the hardware do something.
    - Code for Maker Board is written in **C++**.

- When we combine programs and hardware to do something cool, we call that a **project**.

- Let's Start Coding kits are a combination of all of these: hardware, programs, components, **and** learning materials about each.

Tools at Your Disposal

Every kit contains cards that explain each component included in your kit. Encourage your group to review them and keep them close at hand.

The **Learn pages** on our website (www.letsstartcoding.com/learn) contain a step-by-step program of 14 lessons (and two bug hunts) that will take you and your students through projects that are progressively more complicated and teach you the underlying code concepts along the way.

The **Code Index** on our website (www.letsstartcoding.com/welcome-to-coding) is a collection of the most common concepts you'll see in our code as well as general code knowledge. This is really helpful to learn deeply about the different parts of your favorite project.

The **Help section** of our website (www.letsstartcoding.com/help) covers the most common errors codes you'll find when first starting to code in the Arduino software. But more than that, it's a guide about *how to* troubleshoot. Reading it before you have troubles is really helpful.

The **Project pages** of our website (www.letsstartcoding.com/projects) are less structured than our Learn pages. They provide you with example code, hookup guides, and code walkthroughs for dozens of projects. If you're new to coding, we recommend starting with Learn and branching out to the projects once you have some basics under your belt.

You can always ask for help! You can email us directly at info@letsstartcoding.com . We'll answer your query as soon as we can. Please share as much detailed information about your challenge as you can so that we can help in detail.

Safety

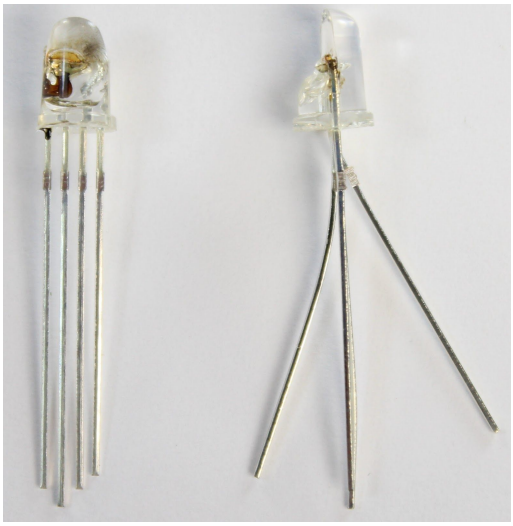
Maker Board and Carrier Board are designed to be safe and tough. They will not shock anyone or get hot when working properly. Students can grab it, squeeze it, and drop it and nothing will break.

Scraping the board with anything harder than a fingernail (like a pen or screwdriver) is a bad idea.

All of the hardware is designed to run on 5 volts at about 40 milliamps (40mA). For reference, a typical tablet charger is 5 volts at 1 amp (1000mA). Maker Board doesn't require very much power.

The most common permanent problem is caused by a short circuit- connecting a component incorrectly and directly to 5V (+) on the Carrier Board. Shorts allow power to flow unrestricted through the component. The component will get warm or hot and components (like LEDs) may make a 'pop' sound, a slight burning odor and casing may break.

If a component 'pops' or starts to 'smell funny' or a student reports that a component is uncomfortably warm to the touch, remove the **power source** from Maker Board. Don't remove the component- it could be too warm to hold.



These LEDs were plugged in to direct power (5V) for about 15 seconds. Notice the discoloration inside the left LED and where the leg meets the body of the LED. The casing

of the right LED has broken with a 'pop' noise. Most instances of providing too much current to an LED are not this dramatic- you may notice that an LED simply doesn't work anymore. These LEDs will not work again- toss them in the garbage.

Allow everything to cool down for 2 to 3 minutes. Examine the Carrier Board and component to see if there is a direct and incorrect connection between 5V(+) and Ground (GND). Review the specific component card to see how the component *should* be connected.

Most components will continue to work normally after a short circuit. Some will be broken and will not work again.

If you are using a carrier board on a metal surface, put duct tape or some insulator between the carrier board and the metal surface. You do not want to short the metal points of the carrier board on the table.

Tips for a Successful Session

1) Ensure you have an internet connection.

2) Ensure you have administrative privilege on the computers: you will be visiting our website and may download or install a browser plugin or an application on the computer.

3) Build *at least one* project with your own computer and make one code modification to it. By the time you've been through the entire process, you will identify both challenges and solutions.

4) Take notes of the places where you get confused or stuck. Takes notes of where your students get confused or stuck. Use those in subsequent classes.

#1 Tip for a Successful Session

Complete the following for yourself when you have time to think and take notes:

1) Install the Arduino software *or* set up a Codebender account from our Start Guides. www.letsstartcoding.com/start

2) Upload example code to your Maker Board*

3) Modify and upload that code to your Maker Board*

*These steps won't be possible without a Maker Board, but step 1 is still possible and valuable so that you know what your students should expect to see.

Getting Started: Do It!

Go to www.letsstartcoding.com/start and follow the instructions for downloading and installing the software and example code. Take notes of the places where you get stuck. Use those notes when you're setting up a session.

Troubleshooting

Lots of things can go 'wrong' when writing code and using software. Most of the time, there is a small, accidental error that can take 2 hours to find and 2 seconds to fix. In this section we'll help you spend less time troubleshooting by identifying the most common problems you'll face.

Installation

Most student problems arise from not following instructions entirely. Skipping a step or stopping too early will result in the student missing some of the software they need to write programs. Ensure that students have gone through each step of the installation process for their computer.

On Windows computers, the auto-updater *usually* keeps your software drivers up-to-date. A program to update drivers may open automatically when Maker Board is

plugged in and it may take 5-10 minutes to complete the update of the drivers. You'll see a green loading bar as Windows searches the web for the drivers and installs them.

If Maker Board is plugged in correctly to a Windows computer, but the **Tools → Port:** menu is not available (grayed out), the drivers may not be installed on your computer. You can find the manual driver installation links at:
http://www.letsstartcoding.com/start-win7-ftdi-install
http://www.letsstartcoding.com/start-win10-ftdi-install

The most certain way to ensure that all of your students can easily get started programming is to install the software on each computer in your class and achieve one example code upload to a Maker Board from *each computer*.

First Upload

For a successful first upload, your students must have followed many steps very closely. The first upload is often the most difficult. Once it's complete, you won't have to face these challenges on this computer again!

If you're having trouble, try these common steps:

1) Ensure that the green light on Maker Board is on. In most laptops, the Maker Board chips should be facing the ceiling. When using the cable, it's possible to get Maker Board plugged in upside-down.

2) If the light is on but pressing upload doesn't get the code onto the board, navigate first to **Tools → Port.** On Windows, you'll see **COM**.... and a number. There will likely be 2 or 3 numbers. Select the top one and try to upload. Doesn't work? Come back and select the second one and try to upload. Repeat for each available port.

   On Apple computers, the port will have a name like **/dev/cu....**

3) Open a new or different example program. Don't make any changes to the program. This will ensure that it's not an accidental code error that makes uploading impossible.

Example Programs

Make a habit of 'the double check', especially if you've already had at least one successful upload to your Maker Board. Here's the double-check checklist:

- Is Maker Board on?
- Is a COM port selected under **Tools→ Port:** ?
- Do your component's pins match the project card you're following?
    - It is very common for students to plug in a component backwards or into the port *next to* the one they should.
- Does the orientation of your component match what the component card says it should match?

Modified Example Code

A typing error is the most common error when modifying code. For example, replacing a **;** with a **:** will break the code. These syntax errors are tough to spot, but you can use the resources at http://www.letsstartcoding.com/syntax/ and the code cheat sheet at http://www.letsstartcoding.com/cheat-sheet/ to help find and fix these.

Have your students undo their changes one at a time by pressing CTRL + Z (Windows) or Command + Z (Mac) and pressing the compile or verify button after each upload. If they've started with an example program, they will eventually 'fix' the program or get it back to the original state. Then they can redo every change with CTRL + Shift + Z (Windows) or Command + Shift + Z (Mac) and fix the problem for their modified file.

Ask students to keep a list of their coding errors. It will help them find common mistakes so they know where to look first next time they have a problem.

Modified Example Hardware

If an example program worked, then the hardware was changed, most often the student needs to ensure the new hardware matches the code.

For example, moving a button from pin 2 to pin 3 requires that you update the void setup() code so that pin 3 is now designated as a button. Any time the button is used throughout the code, you should ensure that you're replacing 2 with 3.

If the student wants to add a new piece of hardware to a project, encourage them to find an example program that uses that piece of hardware and copy and paste the parts that they need from that program. Look at the examples at http://www.letsstartcoding.com/component-demos.

Brand New Programs

There are thousands of new projects (and errors) that can be created with code. The most common errors in the Arduino software are found at http://www.letsstartcoding.com/help.

Codebender gives much more clear error messages than Arduino. Read them closely and use them as clues to find your error.

You should focus on the *method of problem solving* rather than trying to learn the solution to every student's specific problem. Better yet, encourage the students to practice their problem solving skills by asking them these questions:

- Have you thought about the pieces that are needed to make your project? Both hardware and software. Sometimes you can draw or write out the idea and it will help you program it.
- Have you read about and practiced with the code concepts that you want to use in your project?
- Can you find an example project that is close to what you're trying to do? Read through it to find which parts you want to use. You can copy and paste the code into your own code.

Encouraging Independent Exploration

As with anything new, students will have many questions when they are learning to code- that's a good thing!

We encourage you to create an environment of experimentation, not question-and-answer. By helping students learn how to ask and then solve their own questions, you'll save yourself from needing to know every answer. Instead, you'll learn *with* your students and increase their independence.

Encourage your students to think of the projects more like puzzles than toys. They will experience a little confusion and frustration. Rather than give up or get upset, students should know that there is a solution, they just need to find it.

Ask Students Questions

Asking students questions that can be answered by reading code on their screen is a great way to help them realize that many of the puzzle pieces they need are right in front of them. You might ask questions like:
“Where in the code does it say which pin our LED is on?”
“What do you think will happen when we run this code?”
“If I delete this line, what do you think will happen?”
“Can anyone tell me what this number does?”

When students start to analyze code line-by-line and make mental predictions about what a program will do, they’re beginning to think like a computer, which will greatly enable their ability to troubleshoot and predict outcomes later.

Helping Articulate a Question

An inclination of many beginners who have a teacher available to them is to raise their hand and say “It doesn’t work and I don’t know why.” This is not a very well articulated challenge, so it’s very hard to solve. Before diving into the code, help students articulate their question by asking things like:

“Has this exact project worked in the past?”
“Did you start with an example project that worked?”
“If you get code errors, what clues do you have to solve the problem?”
“Can you pinpoint where the code errors are?”
“Have you double-checked all of your components?”

These questions will get the student started down a path of questions that may lead to a solution. For example, if a student started with a working example program and made changes that broke it, the question is now “Which of my changes broke the code?”. That question is much easier to answer.

Helping Translate an Idea into a Project

Well articulated questions are critical not only to solving problems, but also in designing solutions. For example, a beginner may say “I want a multicolored strobe light”. That’s a good project idea, but if a student cannot find a strobe light example, they may feel discouraged. When students are brainstorming ideas or modifying code, some good questions to ask may be:

“Does your idea contain multiple components? What are they? Do you have them?”
“What are the steps of your project? List out in your own words what the code will do.”

By asking these questions, students may realize that a strobe light is simply LEDs turning on, pausing, then turning off, pausing, then repeating. Because the project is now more clearly articulated, they can use the example of a blinking LED as a model for their strobe light.

Have You Tried a Different Example Program?

Because programming is made up of building blocks arranged in different ways, looking at more programs can lead to a breakthrough where a student was previously stuck.

Moving to another program may be best when feeling lost about what to do next. Ask a student to clearly articulate what they could not figure out and write it down. As they navigate other code resources, they may find pieces of the answer that then allow them to solve the earlier challenge.

Taking a break helps, too! Fifteen minutes of focusing attention elsewhere may allow students to come back with a fresh perspective.

Checklist for Student Questions

Getting students used to the process of problem solving may be helped with the use of a checklist they should complete before asking for your help. It won't take long for them to internalize this list, but it may be helpful to post it somewhere for your students to use as they learn.

Understanding the Intent

1) Did you read the description of what the program is supposed to do?
2) Can you state what the program should do in your own words?
3) If the program isn't doing what it's intended to do, can you identify any differences between what is happening and what is supposed to happen?

Checking Your Hardware

1) Is Maker Board plugged in correctly to your computer? The green light on Maker Board should be on and the board should be connected to the USB port.
2) Are your components plugged into the right ports? To be certain, remove them and plug them back in, checking *very carefully* that the orientation matches the component card and the port number matches the project.
3) Have you tried a different component, like swapping one button for another or one LED for another color?

Checking Your Software

1) Does your code run on Maker Board *without* any components plugged in? Can you get to the 'Upload Successful!' message?
2) Have you read the errors in your code carefully, trying to use them as clues about where the problem exists in your code?
3) Can you identify precisely where the error is occuring? Does part of the program work?
4) Have you asked a neighbor for help?

Hey! Can I make.....?

Once a student understands that they can make code do something new, their imagination runs wild. They may ask things like:

- Can we make a quadcopter?
- Can I make a laser gun for my dog?
- Can I make a 10 foot spinning chandelier?
- Can I connect this to the internet?
- How?

Rather than trying to learn all of the answers to all of these questions, it's better to share the *method of problem solving* with your students. Asking them the right questions may lead them to their own answer. Here are some good prompts for your students:

- Think about the parts of the program. What have you built that is similar to your idea? Do you think you can modify your previous project to create your idea?
- Think about the hardware parts of your idea. Do you have access to them? Are they similar to what you have used today?
- Can you build a simple version of your idea with what you have?

These questions will often help students realize when a project is within their grasp and when it is far outside their grasp. Or they will help inspire the student to search and learn more about what they're trying to do!

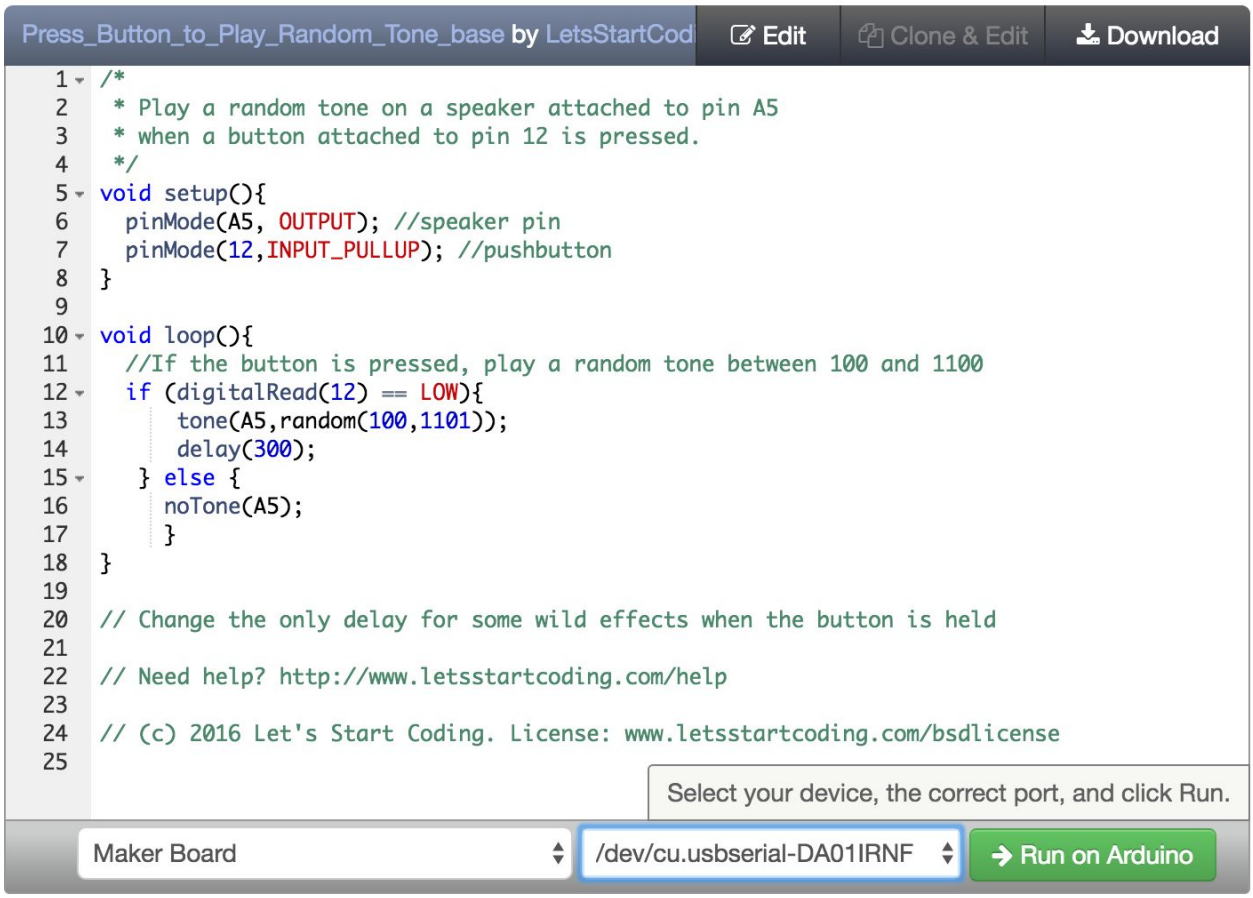
One of the intellectual challenges of programming is articulating exactly what you want the program to do. Getting students to develop the mental process of articulating a problem and solution clearly will help them write better code and think more clearly in general.

Additional Resources

Navigating Codebender from Our Website

To set up you computer using Codebender on our website, open a Chrome browser and navigate to www.letsstartcoding.com/start. From there, follow the steps for Quick Start with Codebender.

We have embedded all of our example projects on our site with the coding technology from Codebender. Students can upload and modify code directly from the project pages. Without a Codebender account, they cannot save their code directly. They *can* copy and paste it somewhere else, like a Word document, and then copy/paste it back into a web browser the next time they code.



This image is a screenshot of what the Codebender code window looks like on our site.

Starting on top, the buttons from left to right:

- The ‘Edit’ button allows you to click in the code window and modify the code. Once clicked, ‘Edit’ becomes ‘Restore’. When you click ‘Restore’, the code is set back to its original format.

- ‘Clone & Edit’ is only available if you have a Codebender account (requires email address) *and* you are logged in to that account. The account does not have to be open in another tab, just logged in. The button will open a new tab and paste the example code into a new program within your Codebender account. Here you can modify, edit, upload, and save the code to your Codebender account.

- ‘Download’ will download the example code *without any of your modifications* to your computer. It will download as an .INO file, which is the file type for the Arduino desktop software. If you do not have the Arduino desktop software, you cannot open this file.

From left to right, the buttons on the bottom row:

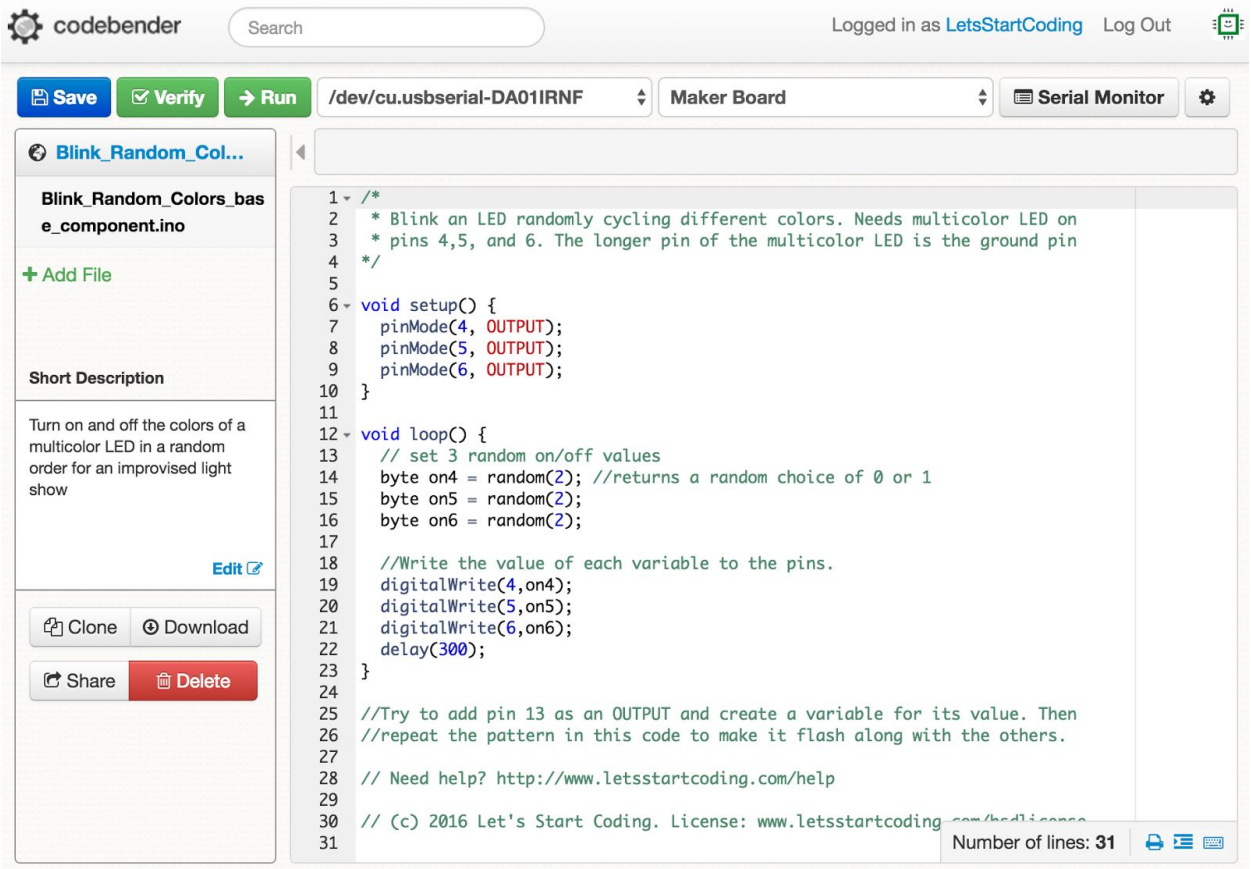
- The menu that says Maker Board in the screenshot is the **board** drop-down menu. If you click it, you will see the list of available boards that work with Codebender. Select Maker Board.

- The menu that says /dev/cu..... in the picture above is the **port** drop-down menu. Maker Board must be plugged in for this port to be active.
    - On Windows, clicking this menu should reveal COM and a number (like COM3). Select the top one and continue. If you have trouble uploading code to Maker Board, click this menu and select the next COM. Continue until you get an upload.

    - On Mac, this menu may reveal some ports including the word ‘bluetooth’. Don’t click those. Select a port that says ‘usbserial’ in the name. If this port fails to upload code to Maker Board, come back to this menu and select a different port that contains ‘usbserial’.

- ‘Run on Arduino’ is the button that sends your code to Maker Board (must be plugged in). Before the code runs on Maker Board, it is checked for errors. Hit this button often as it will help you find errors as you code.

Navigating Your Codebender Account



Creating a free Codebender account requires an email address, but it allows you to save your code in your account. The screenshot above is a program opened from our Codebender homepage. This is also what you will see if you have a Codebender account and click 'Clone and Edit' on our website (see 'Navigating Codebender from Our Website').

First, we'll look at the buttons and menus across the top of the screen:

- 'Save' will save the current program in your account.

- 'Verify' checks the code for validity. Think of it like a spell check for code. It checks for coding errors like syntax problems. Pressing this button will trigger a sequence of checks on the code that may take up to a minute. When it is complete, you will either see a menu with errors pop up from the bottom of the window or you will see text that says "Verification Successful!"

- ‘Run’ attempts to upload your code to your Maker Board. The verification process occurs when you press ‘Run’, so you do not have to do them both for each upload.

- The menu that says /dev/cu..... in the picture above is the **port** drop-down menu. Maker Board must be plugged in for this port to be active.
    - On Windows, clicking this menu should reveal COM and a number (like COM3). Select the top one and continue. If you have trouble uploading code to Maker Board, click this menu and select the next COM. Continue until you get an upload.

    - On Mac, this menu may reveal some ports including the word ‘bluetooth’. Don’t click those. Select a port that says ‘usbserial’ in the name. If this port fails to upload code to Maker Board, come back to this menu and select a different port that contains ‘usbserial’.

- The menu that says Maker Board in the screenshot is the **board** drop-down menu. If you click it, you will see the list of available boards that work with Codebender. Select Maker Board.

- Serial Monitor is a window of communication between the Maker Board and your computer. You can use it to view data from the Maker Board and send messages to the Maker Board. Some projects use it to send you information, like the value from a sensor reading. Using it is the best way to understand it. You can see a project using serial at http://www.letsstartcoding.com/drag-race-reaction-timer .

- The gear icon toggles on and off advanced settings. These settings are for people who are experimenting with blank microchips or doing advanced reprogramming of a chip. You will not need these settings for normal Maker Board use.

Now, let’s look at the options down the side of a Codebender programming window.

- The larger font name (Blink_Random_Col... in the above picture) is the editable name of the program. Clicking it once will turn the name into a text box where you can enter a new name and click a small checkmark to save that new name.
- The smaller font name is the full name of the program.

- The ‘+ Add File’ button allows you to add more program files into this program. It is for advanced users and you will not need it for normal Maker Board use.

- The Short Description is a text explaining what the program should do. You can click 'Edit' and update the description. We recommend that you update this description if you change and save your code. It's much easier to recognize what it does when you have a few lines of context.

- 'Clone' creates a copy of the current program *without any unsaved changes.* It will appear to refresh your browser tab, but the program name will have 'copy' appended to the end of it. If you plan to modify a program and save it, cloning it before making changes is a good practice.

- 'Download' creates a file from the current program *without any unsaved changes.* There are two download options: .ZIP and .hex.
    - The .ZIP file will contain a .INO file. This file type can be opened by the Arduino desktop software for editing on your computer without the internet. You will need the Arduino desktop software to open this file.

    - The .hex file format is for advanced users. You do not need it for typical Maker Board use.

- 'Share' allows you to send your program to someone else with a web address/URL. When the URL is opened, you will see the program and, if the receiver is logged in to *their* Codebender account, the receiver can upload, edit, and save the program to their own account. Changes that the receiving person makes are *not* reflected in the sending person's code.

- 'Delete' deletes the code file. This is a permanent action and it cannot be undone.

In the bottom right of the programming window, there are a number of small icons:

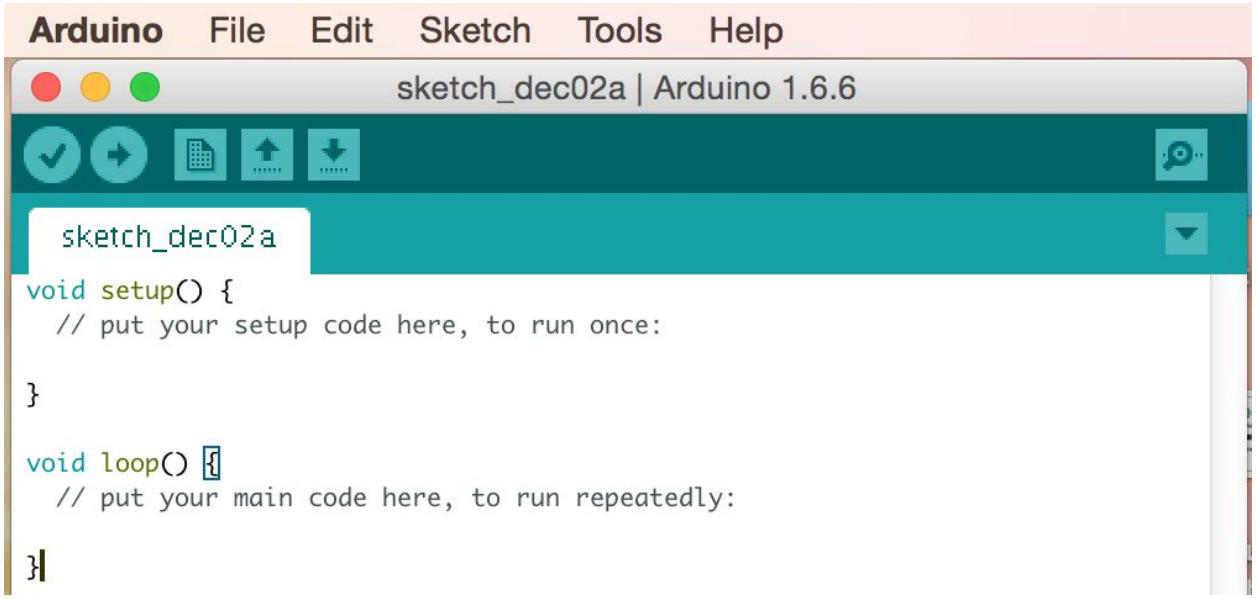
- The printer icon prints the current program, including unsaved changes.

- The icon with an arrow indenting two lines is to reformat the code. Pressing this does not change the functionality of the code, it only makes styling changes. For example, it will move braces ({}) to their own line. We don't recommend relying on this feature, as it sometimes results in errors or strange styling.

- The keyboard icon will show a window with the keyboard shortcuts available in Codebender.

- You will also notice small triangles between the line numbers and the words of the program. Those are clickable and allow you to ‘fold’ away code. They automatically appear when you type an opening brace ({). When you click it, the code between that opening brace and its pair closing brace (}) is hidden. To reveal it again, either click the arrow again or click between the two braces holding the hidden code.
    - Folding away code does not change the functionality of the code. It is helpful when the screen feels too crowded and you want to focus only on one part of a program at a time.

Navigating the Arduino Software



This image is a screenshot from an Apple computer, but all of the commands will be available on a Windows computer.

From left to right, the buttons in the Arduino software and their function:

- The check box is the “Verify” button. Think of it like spell check for the code. It checks for coding errors like syntax problems. Pressing this button will trigger a sequence of checks on the code that may take up to a minute. When it is complete, you will either see an orange error bar in the bottom of the window or you will see text that says “Done Compiling.”

- The right-facing arrow is the “Upload” button. Pressing this will start the process of loading the code onto Maker Board via the USB port. Part of the upload process is verifying the code, so you don’t need to press both. If there is a verification problem with the code, you’ll see the error in the orange bar at the bottom of the window. If there is an upload problem, you will see “Problem uploading to board” in the orange bar.

- The button with a sheet of paper on it opens a new window of the Arduino software. You can also use CTRL + N / Command + N.

- The button with the upward-facing arrow is for opening an existing program in a new window. You can also use CTRL + O / Command + O.

- The button with the downward-facing arrow is for saving your program. You can also use CTRL + S/ Command + S. Keep in mind that programs in the examples section of the Arduino software cannot be saved over. You should use the **File → Save As...** menu to save the file with a new name.

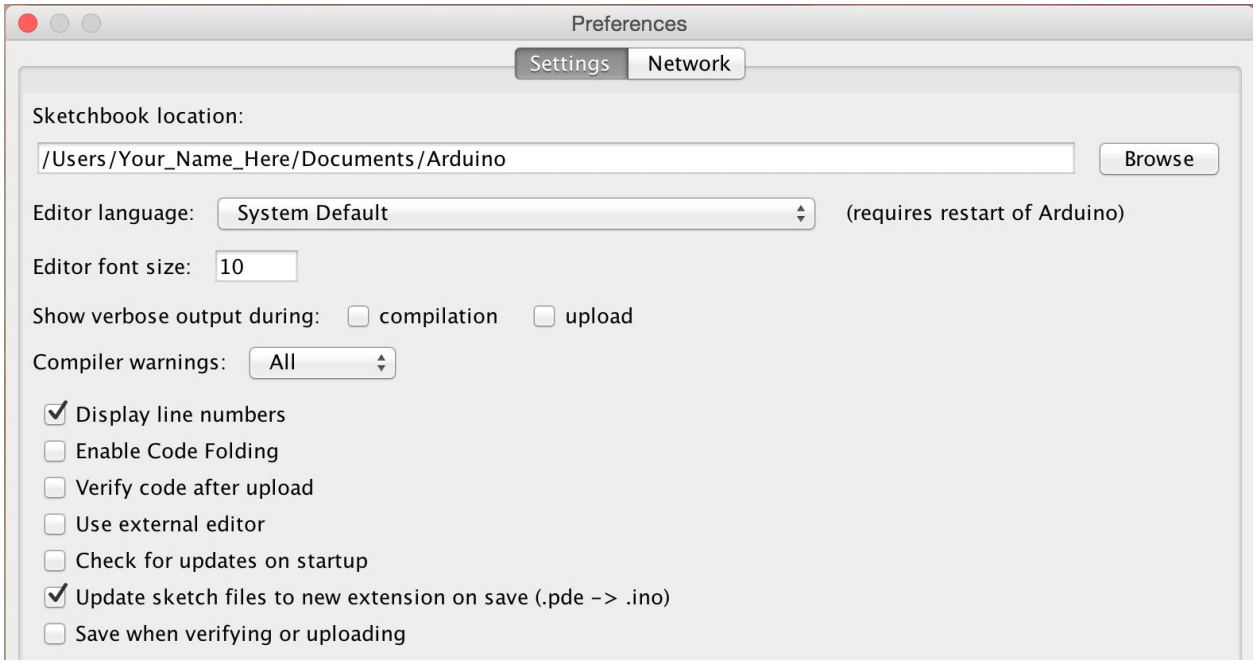
- The magnifying glass on the right is called the Serial Monitor. It is a window of communication between the Maker Board and your computer. You can use it to view data from the Maker Board and send messages to the Maker Board. Using it is the best way to understand it. You can see a project using serial at http://www.letsstartcoding.com/drag-race-reaction-timer .

Customizing Settings in the Arduino Software

You can modify the Arduino software settings in the **Arduino → Preferences** menu on Mac and **File → Preferences** menu on Windows.

The default settings in the Arduino software aren't ideal for beginners. We recommend turning on line numbers so that it's easier to code along with others and communicate about challenges. We also recommend turning *off* the autosave feature so that you can experiment without saving over your previous work. You may also want to increase the editor font size to make syntax details more clear.

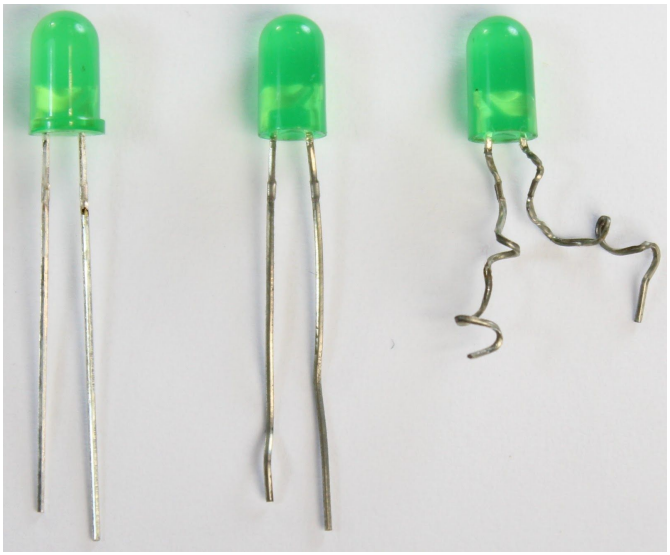
We recommend that your Arduino settings look like this:



Component Handling

Students and session leaders are sometimes nervous about mishandling the components that come with kits from Let's Start Coding. They do not want to break or damage anything. Most components are inexpensive to replace (under $5) and can take lots of abuse. Here are some guidelines for different types of components.

- Treat Maker Board like a USB drive. If you drop it from your pocket or squeeze it between your fingers, it will be OK. But you should never scrape Maker Board or get it wet.
- Components with legs are durable, too. Some components have legs that are meant to bend (like LEDs) and some have legs that aren't designed to bend (like the speaker). Students should be careful when they bend the legs, but they should realize that it is very normal for the legs to be bent.



The LED on the left is brand new and unused. No LED will look like this for long! The LED in the middle is typical- it's been bent a bit, but will function in this condition for hundreds or thousands more hours. The LED on the right has been abused on purpose. LEDs should not look like this, but this LED does still function when it's been unraveled.

Thanks very much for reading and for your interest in Let's Start Coding. If you have additional questions or would like to purchase component supplies for your organization, you can contact us directly at **info@letsstartcoding.com**.