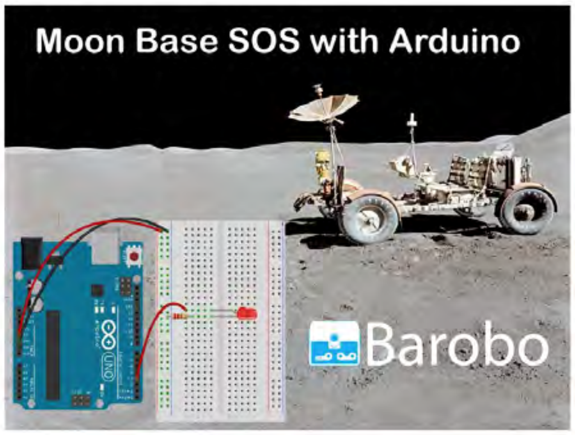
Moon Base SOS with Arduino: An Introduction to Physical Computing

Lesson Plan and Teacher's Notes for an Hour of Code™ Activity



https://www.roboblockly.com/curriculum/hourofcode/
info@barobo.com
http://www.barobo.com
Barobo, Inc.

Outline

1. Introduction

You're one of the pioneer astronauts at the first permanent moon base, but while you are on an expedition away from base, a malfunction has incapacitated your lunar rover. To make matters worse, the malfunction also took out the radio telecommunications equipment, and you can't communicate with either the base or Mission Control on Earth. Your air supply will be running low soon, but you're a topnotch engineer, and you realize that with a little ingenuity you have the necessary equipment to create a light-signaling device that will enable you to send an SOS signal to Mission Control.

In this series of activities students will learn how to create a light-signaling device, using a Barobo Arduino microcontroller board, an LED, some wires, and a breadboard to connect it all together. Students will use the drag-and-drop RoboBlockly interface at www.roboblockly.com (no registration required) to program the Arduino board. The activities are suitable for beginning students in grades 8 and above. (They may also be appropriate for students in lower grades, assuming they have suitable dexterity and care in connecting the wiring involved.) After the Hour of Code™ students may continue to explore the many Arduino, coding, and math-related activities available at www.roboblockly.com.

1.1. Learning Objectives

1. Learn the basic terminology and concepts of physical computing and controlling digital devices, including the concepts of code blocks, commands, arguments (input values to commands), run/execute, bugs, debugging, input/output, breadboards, and circuits.
2. Learn how Morse Code works and how to create messages using it.

1.2. Equipment and Software Needed for Each Student or Pair of Students

1. Computer (PC, Mac, or Chromeboook) with internet connection
2. Barobo Arduino microcontroller board and USB cable
3. Barobo RGB LED module
4. Breadboard
5. Single red LED
6. One 220Ω (Red-Red-Brown) Resistor
7. Three male-to-male jumper wires
8. Software to connect the Arduino board to the computer: either ChDuino for Windows and MacOS machines or Arduino Controller for Chromebooks (available for free, installation instructions given in Section 7)
9. Web-based RoboBlockly interface at www.roboblockly.com to program the Arduino board and run the activities (no registration needed)

All hardware except the computer is included in the Barobo Arduino Starter Kit, available at https://www.barobo.com/product-page/arduino-uno-starter-kit:

2. List of Activities (details in Section 8 below)

1. Blink 1: Making an LED Blink with Arduino
2. Blink 2: Multiple Blinking with Arduino
3. Blink 3: Multiple Color Blinking
4. Blink 4: Using a Breadboard
5. Sending an SOS Message with Morse Code

3. How to Use Barobo's Hour of Code™ Activities in your Classroom

It will take students about an hour to get through each set of activities. However, we believe that students should be able to learn at their own pace, and so we encourage you to give students additional time if needed to complete the activities or make it clear that they don't need to finish the entire set of activities. The final activity has open-ended suggestions for further work if there are students who finish sooner. Time should be budgeted for setting up and connecting the equipment (usually 3-5 minutes for an individual student or 5-10 minutes for a group).

4. Before Hour of Code™

4.1. Prepare your classroom

- Make sure you have a computer (PC, Mac, or Chromebook) for each student or pair of students. (Tablets are not supported for Arduino activities.)
- Make sure you have good Internet access in the classroom, as you will need to access the RoboBlockly activities at www.roboblockly.com to control the Arduino board and run the activities (no registration required).
- Make sure you have a modern browser installed on the computers or devices. Test RoboBlockly on students' computers.
- Make sure to have enough Arduino boards and materials for each student or pair of students, connect them to the computer using either the ChDuino software (Windows and MacOS machines) or Arduino Controller (Chromebooks), and test them beforehand. (Installation instructions given in Section 7 below.)

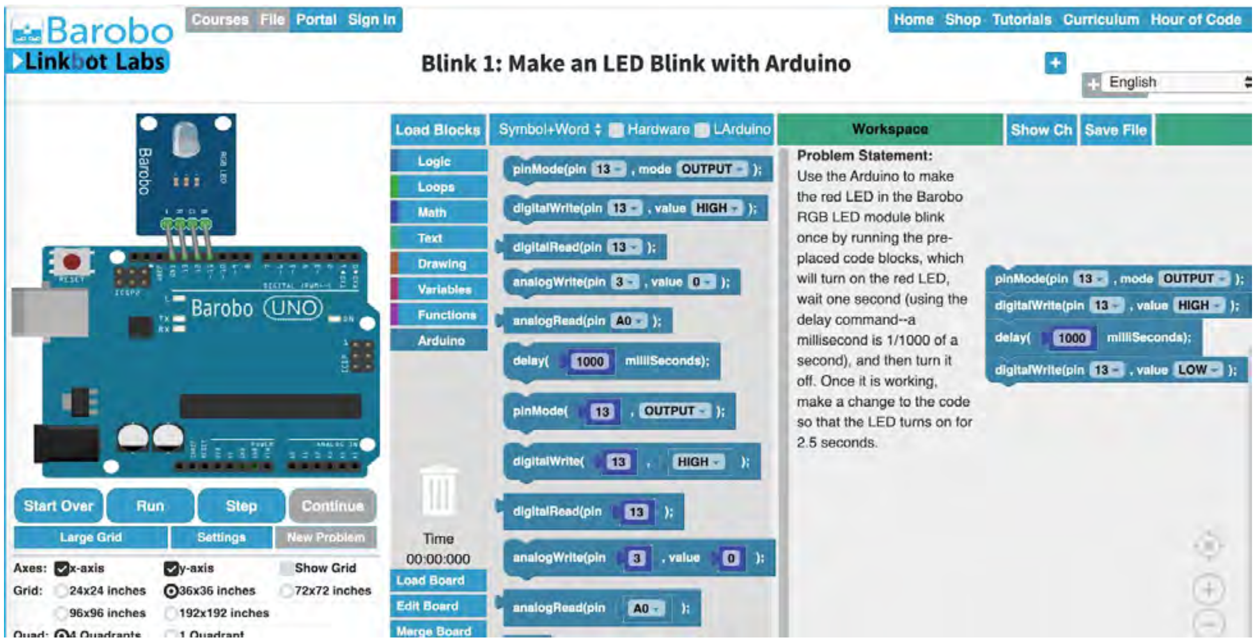
4.2. Prepare yourself

- See Section 7 below to review:
    1. How to connect an Arduino board to a computer via a USB connection.
    2. Connecting the RGB LED module to the Arduino board.
    3. Testing the connection using ChDuino (Windows, MacOS) or Arduino Controller (Chromebooks) software.
    4. Creating a circuit with a single LED and resistor using the Arduino and breadboard.
- Go through the Hour of Code activities yourself so that you are familiar with what your students will be experiencing.

- Each activity has an introduction that describes the activity and concepts involved, and a problem statement that instructs students on what they should do. (Full details in Section 8 below.) These are designed to be self-contained and self-explanatory, but some students may need clarifications or other assistance.

4.3. Prepare your students

- When using RoboBlockly in class, first demonstrate to students how to navigate and use the RoboBlockly website, as shown in Figure 4.1 below. (Your view of the website may be slightly different, depending on the activity loaded.) The basic idea is that users create programs to control the Arduino board by dragging code blocks (commands) from the middle Blocks section to the Workspace on the right. Code blocks can be clicked together to form a sequence of commands. Clicking the Run button below the grid on the left will then run ("execute") the commands in sequence.


Barobo
Linkbot Labs
Courses File Portal Sign In
Home Shop Tutorials Curriculum Hour of Code
Blink 1: Make an LED Blink with Arduino
English
Load Blocks
Symbol+Word
Hardware
LArduino
Workspace
Show Ch
Save File
Logic
Loops
Math
Text
Drawing
Variables
Functions
Arduino
Start Over
Run
Step
Continue
Large Grid
Settings
New Problem
Axes: x-axis y-axis Show Grid
Grid: 24x24 inches 36x36 inches 72x72 inches
96x96 inches 192x192 inches
Quad: 4 Quadrants 1 Quadrant
Time 00:00:000
Load Board
Edit Board
Merge Board
pinMode(pin 13, mode OUTPUT);
digitalWrite(pin 13, value HIGH);
digitalRead(pin 13);
analogWrite(pin 3, value 0);
analogRead(pin A0);
delay(1000 milliSeconds);
pinMode(13, OUTPUT);
digitalWrite(13, HIGH);
digitalRead(pin 13);
analogWrite(pin 3, value 0);
analogRead(pin A0);

Problem Statement:
Use the Arduino to make the red LED in the Barobo RGB LED module blink once by running the pre-placed code blocks, which will turn on the red LED, wait one second (using the delay command--a millisecond is 1/1000 of a second), and then turn it off. Once it is working, make a change to the code so that the LED turns on for 2.5 seconds.

```
pinMode(pin 13, mode OUTPUT);
digitalWrite(pin 13, value HIGH);
delay( 1000 milliSeconds);
digitalWrite(pin 13, value LOW);
```

Figure 4.1: Sample Screenshot of the RoboBlockly User Interface (www.roboblockly.com)

- Introduce dual programming for collaborative learning. This learning model is especially helpful for students who may need some extra assistance.
- Help students get excited about Hour of Code™ by inspiring students and discussing how computer science impacts every part of our lives. As a class, list things that use code in everyday life, or discuss different ways technology impacts our lives, etc.

5. Getting Started in Class (10 minutes for intro, 40-45 minutes for activities)

1. Guide each student (or pair of students) in connecting the Arduino to the computer via a USB cable, using the ChDuino or Arduino Controller software to establish the connection, and plugging the LED module into the Arduino (5-10 minutes).
2. Have each student test the connection using the ChDuino or Arduino Controller software to manually turn on and off the red LED in the module (5 minutes).
3. Going through the first activity as a group is a good way to start. The RoboBlockly website contains far more options than are used in the Hour of Code activities, so the most important features to cover are:
    a. Terminology: program, code, code blocks, commands, arguments (input values to commands), run/execute, debugging
    b. The basic layout: the grid, the middle code block section, the Workspace
    c. Important buttons: Run, Reset (label that appears on the Run button after Run is clicked), Step, Start Over, the Show Robot checkbox (in the “Robot 1” section at the lower left).
    d. Basic actions: running pre-placed code blocks, changing the argument values (input values) for commands, dragging and dropping code blocks from the center block section to the Workspace, unhooking code blocks from other blocks in the Workspace and dragging them back to the center block section to delete.
4. Point the students to the rest of the activities at https://roboblockly.com/curriculum/hourofcode/arduino/ (35-40 minutes). (Students may require additional assistance in wiring the Arduino, LED, resistor, and breadboard in Activity 4.)
5. Circulate around to offer encouragement and give assistance as needed.

6. Wrapping Up in Class (5 minutes)

- If you and/or your students use social media, we encourage you to share your Hour of Code™ experience (as appropriate). For example, “I’ve saved the day by creating a light-signalling Arduino device for the Moon Base SOS activity!” #HourOfCode #robotics4learning.” Use the hashtag #HourOfCode (with capital letters H, O, C).
- After Hour of Code™, encourage your students to continue learning on RoboBlockly by exploring the Playground, Robotics, Computing, or grade-level Math activities.
- ***All student activities on RoboBlockly are free - students do not have to create accounts!***

7. Connecting an Arduino Board to the Computer

7.1. Introduction

This section is an introduction to how to connect an Arduino microcontroller board to a computer (Windows, MacOS, or Chromebook) and test the connection with a digital output device (an LED in this example). “Digital” means that there are only two possible states, on and off, which electrically means there is some level of voltage (on) or no voltage at all (off). For the Arduino, and most similar microcontrollers, the ”on” output voltage is 5 volts and the ”off” is 0 volts. The on state is also called the High state and is often represented in computing as a 1. The off state is also called Low or ground and is usually represented by a 0. For digital output devices, the microcontroller can be told to write a 1 (High) to the pin the device is connected to, thus turning the device on. If the microcontroller is told to write a 0 (Low) to the device’s pin then the device will be turned off.

7.2. Required Components and Materials

We will cover two different ways to connect an LED device to the Arduino:

1. Using an LED module (with three LEDs in one bulb) that plugs in directly to the Arduino.
2. Using a “breadboard” that allows you to wire up a circuit with an individual LED, a resistor, and the Arduino.

The first approach simply requires an RGB LED module and the Arduino board, while the second requires the Arduino plus a breadboard, a 220Ω (Red-Red-Brown) resistor, and an individual LED component. All these components are available in the Barobo Arduino Starter Kit (https://www.barobo.com/arduino-and-raspberry-pi-kits, $34.99, classroom bundles available). You will also need a computer (Windows, Mac, or Chromebook machine) to connect to the Arduino board using the USB cable that comes in the Starter Kit.

7.3. Connecting the RGB LED Module to the Arduino

The RGB (“red green blue”) LED module has four pins coming out of it, as shown in Figure 7.1 below. They are labeled “B”, “G”, “R”, and “-“, standing for blue, green, red, and ground.

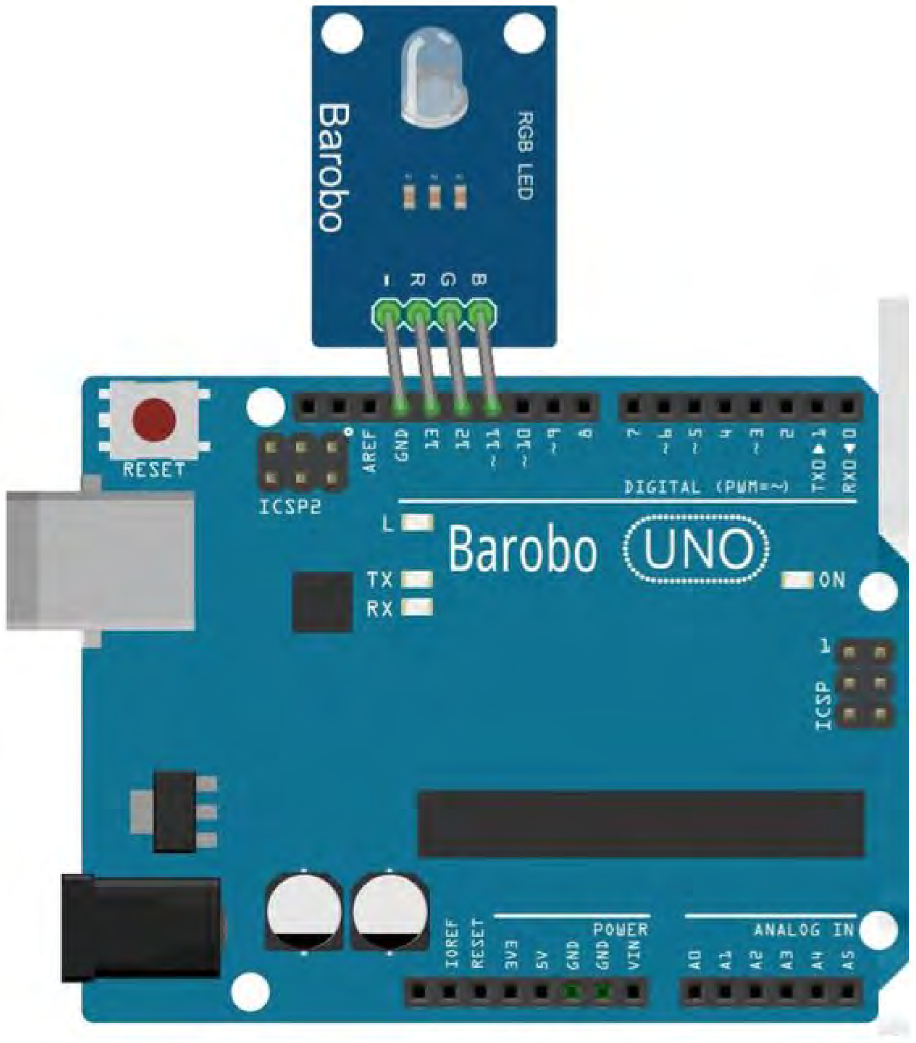


Figure 7.1: Connecting the RGB LED Module to the Arduino

One side of the Arduino board has a row of sockets labeled 0 through 13 and GND. (Sometimes on diagrams these are labeled D0 through D13, the "D" standing for digital. In addition, even though they are sockets, they are usually referred to as pins.) Plug in the LED module so that its "-" pin goes in the GND socket, and the R, G, and B pins naturally go in to the 13, 12, and 11 sockets, respectively, as shown in Figure 7.1.

7.4. Wiring an Individual LED to the Arduino Using a Breadboard

Plugging an LED module directly into the Arduino board is very simple, but it doesn't allow for other types of devices and circuits. That's where a "breadboard" comes in, shown in Figure 7.2.

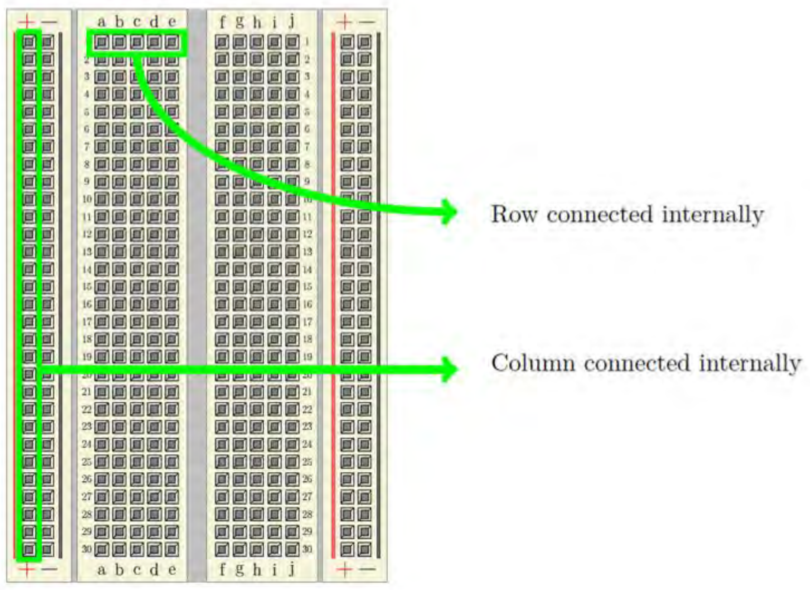


Figure 7.2: A Breadboard

A breadboard allows us to create various circuits and connections to the Arduino by plugging in wires and various devices, such as LEDs, switches, resistors, capacitors, transistors, small motors, and sensors. (For the “blink LED” activities we will only need an LED and a resistor.)

When looking at the breadboard with the shorter edge as the horizontal, notice that the two left-most and two right-most columns are marked with a ’+’ and a ’-’. These are the power strips, and each column is connected internally all the way down the board. Thus, if one wire is plugged into the far upper socket of a column and another wire is plugged into the far lower socket of that column, then those two wires are connected at the same voltage. The strips marked ’+’ are typically where the positive power lead is connected and the strips marked ’-’ are where the ground lead is connected.

If you look in between the two sets of power strips you will see that there are two 5 x 30 socket grids with a trench in between them (running vertically down the middle). The five sockets in each row are internally connected to each other so that if two wires are plugged into any of these 5 sockets then those wires are connected at the same voltage. The rows of one grid are not connected to the rows of the other grid, the trench separates them. Thus for any row, pins a, b, c, d, and e are connected internally and pins f, g, h, i, and j are also connected internally but separately from a-e.

The connections we need for the “blink LED” activity are shown in Figure 7.3. We plug one end of a wire into one of the columns marked ’+’ and plug the other end into one of the Arduino pins marked ’5V’ (the long red wire shown), and do the same with another wire to connect the ’-’

column and the pin marked ’GND’ just under the 5V pin (the long black wire shown; the colors of the wires don’t matter, but black is often used for a ground wire).

IMPORTANT: Make sure that the breadboard is oriented so that the leftmost column (as seen in Figure 7.3) is a “+” column, as indicated by a “+” sign at the top of the column. If you notice that the leftmost column is labeled “-” at the top, simply rotate the breadboard by 180 degrees, which will orient it so that a “+” column is the leftmost column.

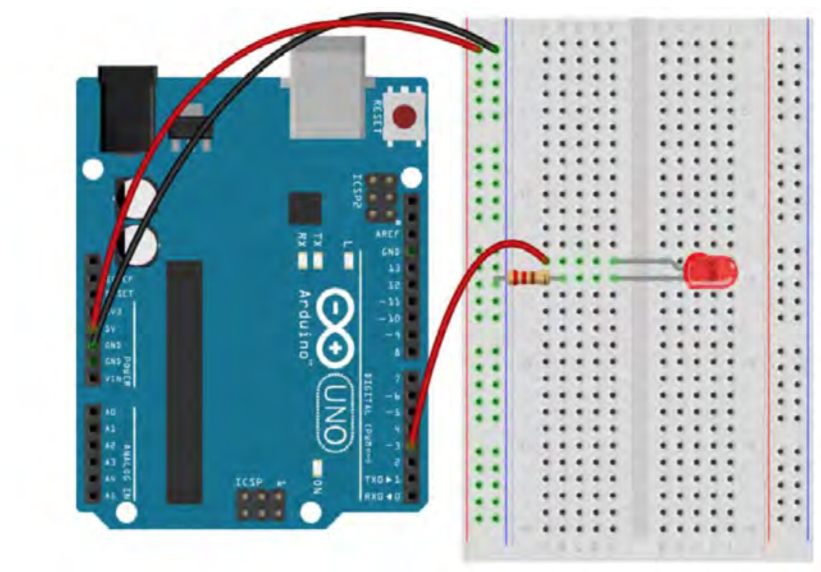


Figure 7.3: Wiring Setup to Connect LED to Arduino Using a Breadboard

The next step is to plug in the LED. A single LED device looks like the red one shown in Figure 7.4.

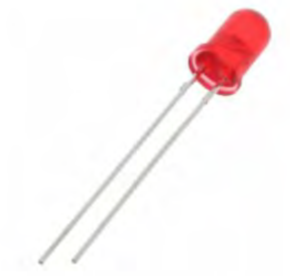


Figure 7.4: An Individual LED Device

Note that it has two wire leads coming out of it, and that one is longer than the other. The longer one is the positive lead, and the shorter one is negative. This is important because the LED, being a “diode” (LED = light emitting diode), will only pass current in one direction, and so we need to make sure it is plugged in to the breadboard correctly. It is common practice to bend the longer lead slightly and put a kink in it, so that it is easy to tell which is the positive lead when it is plugged in.

Plug the LED into the breadboard as shown in Figure 7.3 above (note the position of the kinked lead). Make sure the two LED leads do not share the same row because then the two leads would be connected at the same voltage and the circuit will not work. Plug one end of one Red-Red-Brown resistor (220 Ohm) into the same row as the negative (shorter) lead of the LED and plug the other end into the same ‘-’ column that the long black ground wire is plugged into at the top of the column. Lastly, place a wire into the row with the longer end of the LED, the one that should be bent. Plug the other end of the wire into digital pin ‘3’ on the Arduino. (This is the shorter red wire shown in Figure 7.3.) That’s it!

7.5. Installing Software to Control the Arduino from a Computer

Once the LED is connected to the Arduino (whether using the LED module or the individual LED and breadboard), the next step is to connect the Arduino to the computer and test the setup. There are two parts to making the connection: a hardware connection via a USB cable (supplied in the Arduino Starter Kit) and a software connection via “ChDuino” software (for Windows and MacOS machines) or “Arduino Controller” software (for Chromebooks).

We will install the software first and then make the hardware connection in the next section. There is one way to install the software for Windows and MacOS machines and a different way to do it for Chromebooks, so we will cover each in turn.

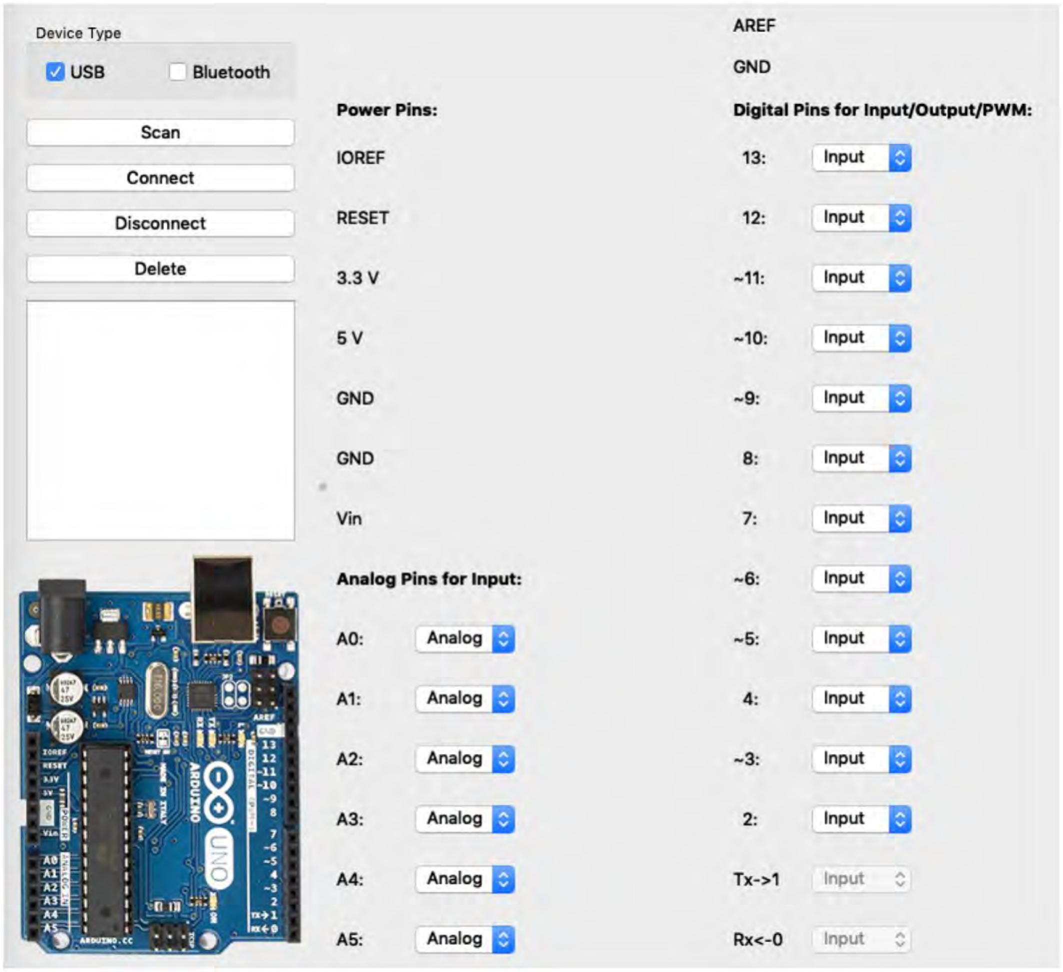
(a) Downloading and Installing ChDuino for Windows and MacOS

The Windows and MacOS versions of ChDuino are available as a free download from Barobo as part of its “Linkbot Labs” software. To download the latest version of Linkbot Labs, go to https://www.barobo.com/downloads and follow the instructions for the type of computer you’re using (Windows or MacOS). After you have downloaded and installed the software, you should see ChDuino as one of your applications. Its icon looks like this:

Figure 7.5: The ChDuino Application Icon

(The “Ch” indicates that the software uses the C programming language, via the Ch interpreter, to program the Arduino. But we won’t need to use Ch for a simple “blink LED” activity.)

When you open ChDuino, you should see a graphical user interface like that shown in Figure 7.6. It gives a visual representation of the various pins on the Arduino board, and allows us to control the input and output pins on the board using the interface. You can view the analog voltage values read by the A0-A5 ports (on the left side), can view digital input values, and can control the digital output values (as we will see). For the next step, continue with Section 7.6 below.



Device Type

[✓ USB] Bluetooth

Power Pins:

IOREF

RESET

3.3 V

5 V

GND

GND

Vin

Analog Pins for Input:

A0: Analog

A1: Analog

A2: Analog

A3: Analog

A4: Analog

A5: Analog

Digital Pins for Input/Output/PWM:

13: Input

12: Input

~11: Input

~10: Input

~9: Input

8: Input

7: Input

~6: Input

~5: Input

4: Input

~3: Input

2: Input

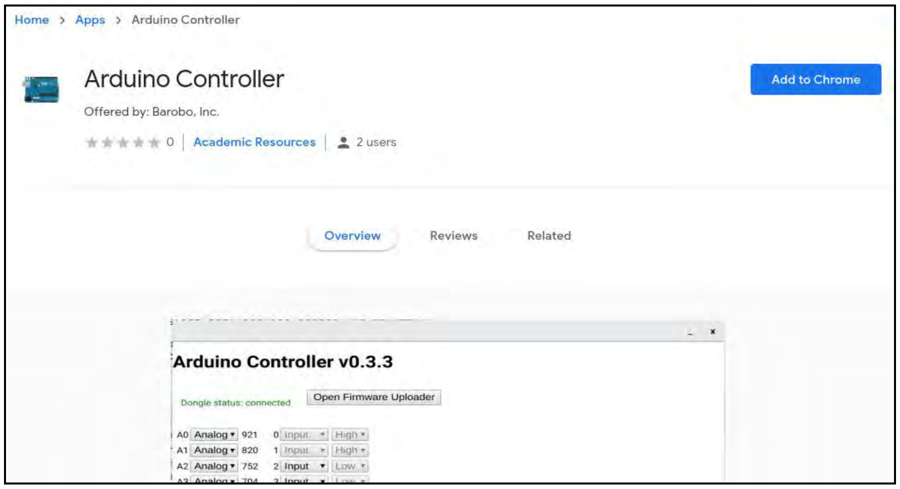
Tx->1 Input

Rx<-0 Input

Figure 7.6: The ChDuino Interface for Windows and MacOS

(b) Downloading and Installing Arduino Controller for Chromebooks

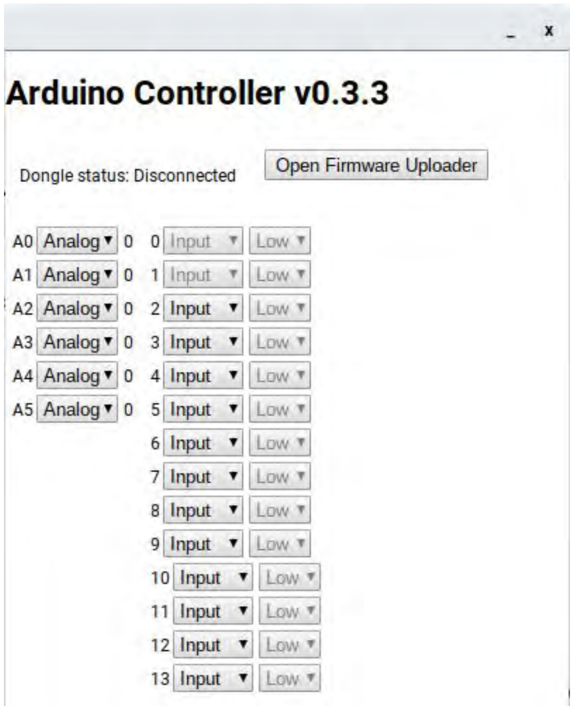
The Arduino Controller software for Chromebooks is available as a free download from Barobo. To download the latest version, go to https://www.barobo.com/downloads, scroll down to the Chromebook section, and click the button labeled "Arduino Controller." A new tab or window should open for the Arduino Controller page at the Chrome Web Store, as shown in Figure 7.7 below.



```
A0 Analog ▾ 921   0 Input ▾ High ▾
A1 Analog ▾ 820   1 Input ▾ High ▾
A2 Analog ▾ 752   2 Input ▾ Low ▾
```

Figure 7.7: The Arduino Controller Page at the Chrome Web Store (for Chromebook users)

Click the “Add to Chrome” button on the right, and then click the “Add app” button in the popup confirmation window that appears. (If you don’t see the “Add to Chrome” button, but instead it is labeled “Launch app,” it means that you already have Arduino Controller installed as one of your Chrome extensions.) After a few seconds the Arduino Controller page will reappear with the “Add to Chrome” button changed to “Launch app.” Click the “Launch app” button. There will be a brief “Loading” message, and then you should see the window shown in Figure 7.8 below.



Dongle status: Disconnected

```
A0 Analog ▾ 0   0 Input ▾ Low ▾
A1 Analog ▾ 0   1 Input ▾ Low ▾
A2 Analog ▾ 0   2 Input ▾ Low ▾
A3 Analog ▾ 0   3 Input ▾ Low ▾
A4 Analog ▾ 0   4 Input ▾ Low ▾
A5 Analog ▾ 0   5 Input ▾ Low ▾
                6 Input ▾ Low ▾
                7 Input ▾ Low ▾
                8 Input ▾ Low ▾
                9 Input ▾ Low ▾
               10 Input ▾ Low ▾
               11 Input ▾ Low ▾
               12 Input ▾ Low ▾
               13 Input ▾ Low ▾
```

Figure 7.8: The Arduino Controller Interface for Chromebooks

The Arduino Controller window gives a visual representation of the various pins on the Arduino board, and allows us to control the input and output pins on the board using the interface. You can view the analog voltage values read by the A0-A5 ports (on the left side), can view digital input values, and can control the digital output values (as we will see).

If you lose sight of the Arduino Controller window, click its icon (Figure 7.9) in the Chromebook "shelf" (the row of apps along the bottom or side of your screen). The icon is a small image of the Arduino board. If you don't see the icon, then you will need to relaunch the controller by going back to its page in the Chrome Web Store and clicking the "Launch app" button. (Tip: If you're having trouble finding it again in the Chrome Web Store, go to www.barobo.com/downloads and click the "Arduino Controller" button in the Chromebook section. It will take you to the proper page in the Chrome Web Store. It's also good to bookmark it, of course.)

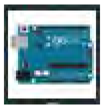


Figure 7.9: The Arduino Controller Icon

7.6. Connecting the Arduino to the Computer

The next step is to physically connect the Arduino board to the computer. It's a basic USB connection, using the USB cable supplied in the Arduino Starter Kit. As shown in Figure 7.10, one end plugs into the large silver socket on the Arduino, and the other end plugs into a regular USB port on the computer.

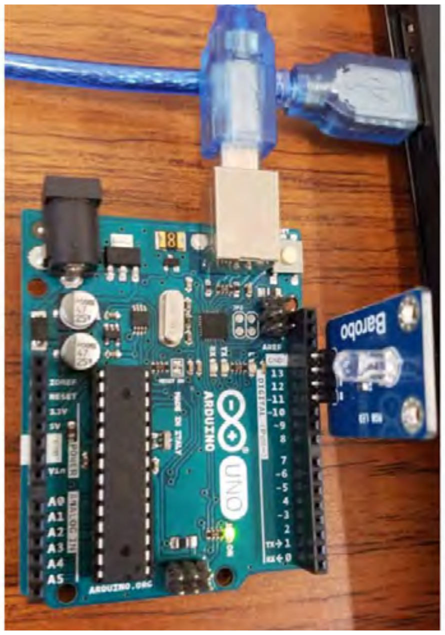


Figure 7.10: Connecting the Arduino to the Computer via USB cable
(only the two ends of the blue USB cable are shown—it's all one cable)

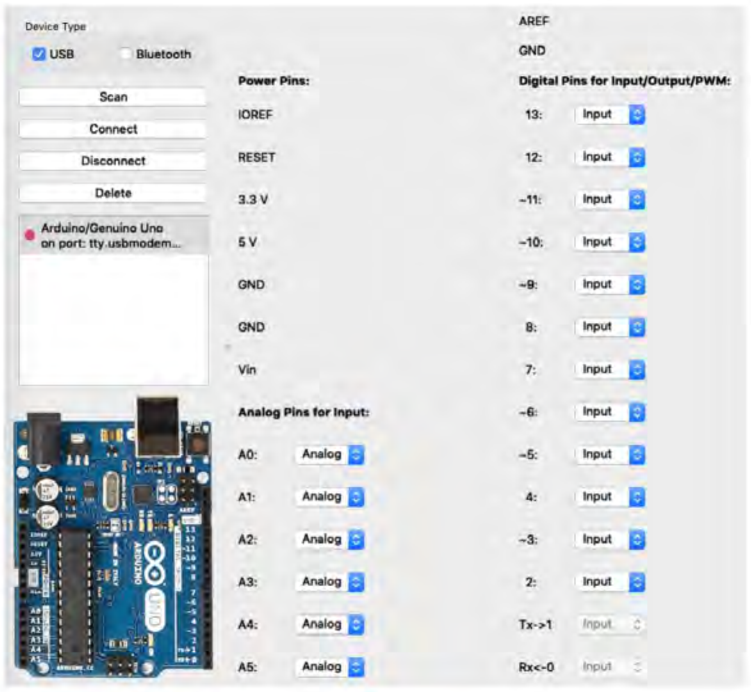
Once the Arduino is connected to the computer via the USB cable, you should see a light on the board turn on, indicating it is getting power from the computer (assuming the computer is on).

In order to complete the connection, we need to get the Arduino software (ChDuino for Windows and MacOS machines and Arduino Controller for Chromebooks) to recognize the Arduino board. As usual, the process is slightly different for Windows/MacOS vs. Chromebooks, so we'll cover each in turn.

(a) Completing the connection for Windows and MacOS machines

If the ChDuino program is not open, you can open it now. Make sure the "USB" checkbox at the top left of the ChDuino interface (see Figure 7.6) is checked, and then click the "Scan" button at the top left, and it will scan for an Arduino board that is physically connected. Yours should appear in the message area in the middle left, as shown in Figure 7.11 below, with a red dot next to it. The red dot indicates that the Arduino is physically connected but not yet connected via the software. The COM port where the Arduino is connected may also be listed. (It's possible to have multiple Arduinos connected to the same computer.)

To complete the connection click the "Connect" button at the top left. The red dot should turn to green (Figure 7.12). In addition, fluctuating numbers should appear in the Analog pin section of the ChDuino interface, and you should see some blinking lights on the Arduino. This all indicates that the Arduino is properly connected and ready for use.



Device Type: [✓ USB] Bluetooth

● Arduino/Genuino Uno on port: tty.usbmodem...

A0:	Analog
A1:	Analog
A2:	Analog
A3:	Analog
A4:	Analog
A5:	Analog

13:	Input
12:	Input
~11:	Input
~10:	Input
~9:	Input
8:	Input
7:	Input
~6:	Input
~5:	Input
4:	Input
~3:	Input
2:	Input
Tx->1	Input
Rx<-0	Input

Figure 7.11: ChDuino Interface with Arduino Listed (red dot) (Windows and MacOS)

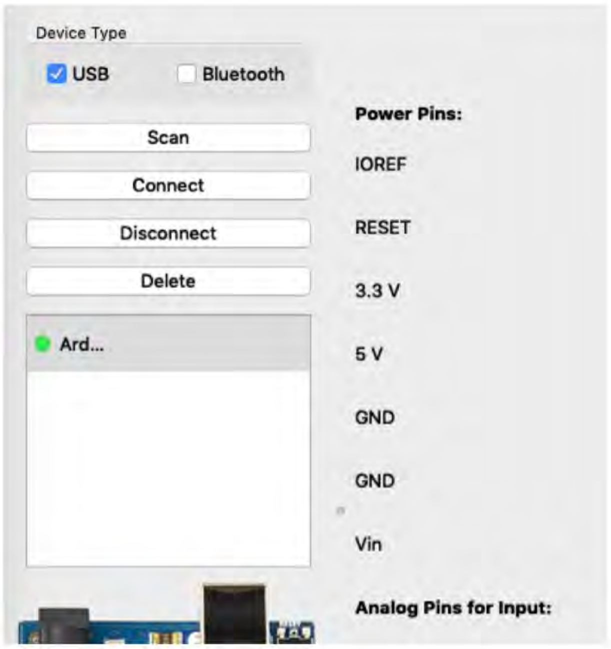
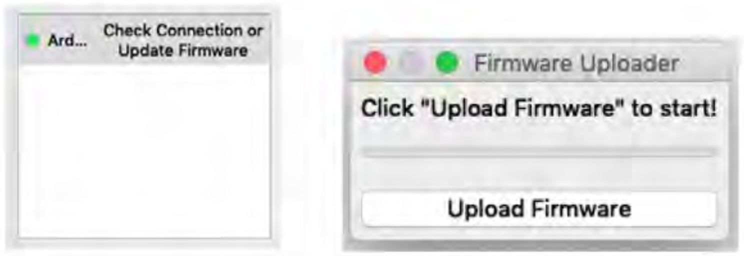


Figure 7.12: Green Dot Showing Arduino Connection is Complete (Windows and MacOS)

Sometimes the connection will be lost, as indicated by the absence of the blinking lights on the Arduino, the static nature of the analog pin values, and/or the green dot turning to red. If that's the case, simply click the "Connect" button to reconnect.

If, when you click the "Connect" button, the red dot turns to green but there's also a message along the lines of "Check connection or update firmware," click the message. A small window should open with a button that allows you to update the firmware on the Arduino board (Figure 7.13). It usually only takes a few seconds, and then you're ready to go. (Firmware is the computer code stored on the Arduino that runs the board. When new features are added, or software bugs fixed, a new version of the firmware must be uploaded and stored in the Arduino board.)

For the next step, continue with Section 7.7 below.

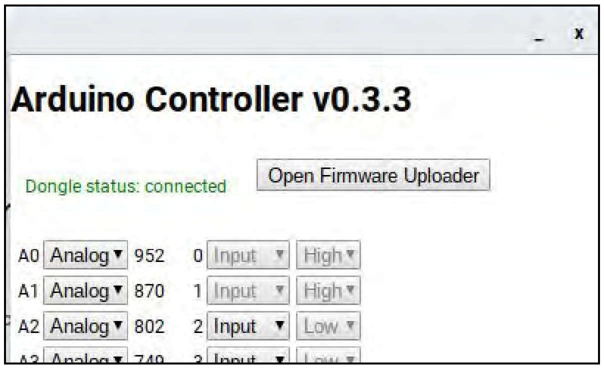


Click "Upload Firmware" to start!

Figure 7.13: Update Firmware Message and Popup Window (Windows and MacOS)

(b) Completing the connection for Chromebooks

Completing the connection for Chromebooks is simple. Make sure that the Arduino Controller is running and its window is visible (relaunch it and/or click its icon in the shelf to bring it forward, if necessary). Once the Arduino Controller is running, as soon as you plug in the Arduino board's USB connection the Controller software recognizes it and makes the connection. The message near the top left of the Arduino Controller window will change from "Dongle status: Disconnected" to "Dongle status: connected" in green, as shown in Figure 7.14. (If you physically disconnect the USB cable, the message will be "Dongle status: error" in red. Plugging it back in should reset things and give the "Dongle status: connected" message after a few seconds.)


Arduino Controller v0.3.3

Dongle status: connected

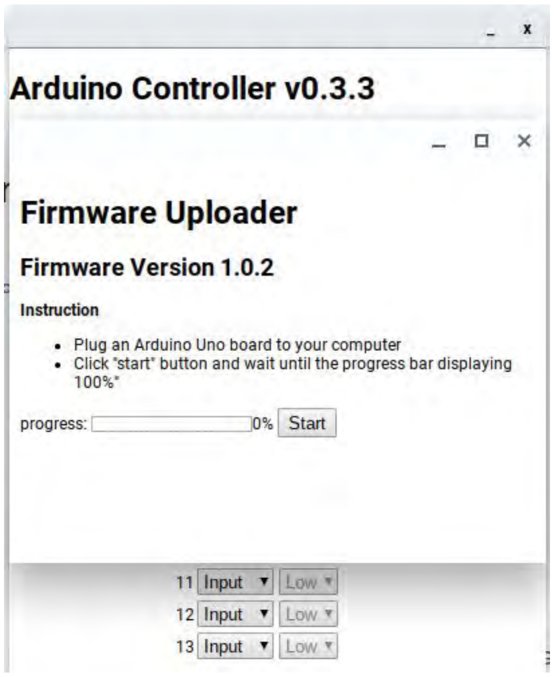
```
A0 Analog 952   0 Input High
A1 Analog 870   1 Input High
A2 Analog 802   2 Input Low
```

Figure 7.14: "Connected" Message Showing Arduino Connection is Complete (Chromebooks)

In addition, fluctuating numbers should appear in the Analog pin section of the Arduino Controller window, and you should see some blinking lights on the Arduino. This all indicates that the Arduino is properly connected and ready for use.

Note: If you physically plug in the Arduino's USB cable to the Chromebook before launching the Arduino Controller software, everything will still work. When the Arduino Controller is launched, it should recognize that an Arduino board is plugged in and make the connection, yielding the "Dongle status: connected" message.

You probably have also noticed that there is an "Open Firmware Uploader" button at the top right of the Arduino Controller window. Firmware is the computer code stored on the Arduino that runs the board. When new features are added, or software bugs fixed, a new version of the firmware must be uploaded and stored in the Arduino board. At times you may get a message to the effect of "firmware update needed." If so, click the "Open Firmware Uploader" button. A popup window like that shown in Figure 7.15 should appear. Click "Start" and wait until the progress bar reaches 100%, then close the Firmware Uploader window.


Arduino Controller v0.3.3

Firmware Uploader

Firmware Version 1.0.2

Instruction

- Plug an Arduino Uno board to your computer
- Click "start" button and wait until the progress bar displaying 100%"

progress: 0% Start

Figure 7.15: Firmware Uploader Window for Chromebooks

If you're not sure if you have the latest version of the firmware installed, it's fine to go through the upload firmware process. If the current firmware on board is the same as the firmware being uploaded, the "new" firmware will replace the current firmware, but there will of course be no difference in the end result, and everything should run fine.

7.7. Blinking the LED

At this point you should have the LED connected to the Arduino board (using either the LED module or the breadboard plus single LED), and the Arduino board connected to the computer via the USB cable and the ChDuino interface (for Windows and MacOS machines) or the Arduino Controller (for Chromebooks).

The final step is to send a High signal to the LED to turn it on (or Low to turn it off). The red LED is connected either to the Arduino's digital pin 13 (for the LED module setup) or digital pin 3 (for the breadboard setup). Because we want to output a signal from the Arduino to the LED, we need to set the appropriate pin (13 or 3) to "Output" mode using the pulldown menu next to the pin number in ChDuino or Arduino Controller. For the ChDuino interface, if pin 13 is set to "Input", it looks like this:

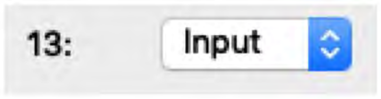


Figure 7.16: The Input/Output Pulldown Menu for Digital Pin 13 (ChDuino for Windows/MacOS)

Click on the pulldown menu and change it to “Output” and it will look like this:

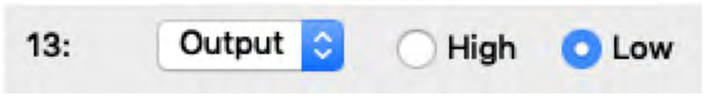


Figure 7.17: The Output High/Low Interface for Pin 13 (ChDuino for Windows/MacOS)

For the Arduino Controller interface, changing pin 13 to Output looks like slightly different:

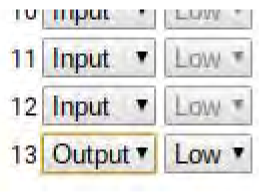


Figure 7.18: The Output High/Low Interface for Pin 13 (Arduino Controller for Chromebooks)

To output a “turn on” signal to the LED (assuming it’s connected to pin 13), simply click the “High” radio button for pin 13 in the ChDuino interface or use the pulldown menu for pin 13 in the Arduino Controller interface to select “High.” A 5 volt signal will be sent through pin 13 to the LED, thus turning it on. To turn it off, simply click the “Low” button (ChDuino) or select “Low” (Arduino Controller), which sends a 0 volt signal to the pin and out to the LED.

If you are using the LED module, you can also try turning on/off the blue light by outputting a High or Low signal on pin 11 and turning on/off the green light by doing the same for pin 12. It’s fine to turn on multiple lights at the same time. For example, turning on the red and blue LEDs at the same time gives a purple effect.

7.8. Troubleshooting

So that’s how you set up an Arduino board with an LED. If your LED does not turn on and off when you click “High” and “Low” in ChDuino or Arduino Controller, here are some things to check:

- Is the Arduino board connected properly and running? (Check for green power light, plus blinking lights on the board.)

- Is ChDuino (Windows/MacOS) or Arduino Controller (Chromebook) running and connected to the Arduino? (Check for green dot in ChDuino or “connected” message in Arduino Controller, and check that analog values are shown as fluctuating.)
- Is the LED connected to the proper pin, and is that the pin you are using in ChDuino or Arduino Controller to turn it from High (on) to Low (off)?
- Is the pin set to “Output”?
- If you are using the breadboard, does your wiring match that shown in the wiring diagram in Figure 7.3? (It’s easy to have one of the wires or other connections be in the wrong place, or not completely plugged in.)

7.9. Explore the “Moon Base SOS” (Hour of Code) and Other Arduino Learning Activities

You are now all set to go to https://roboblockly.com/curriculum/hourofcode/arduino/ and explore the “Moon Base SOS” activities for Hour of Code, or the other Arduino learning activities at https://roboblockly.com/curriculum/arduino/.

8. Moon Base SOS Activities

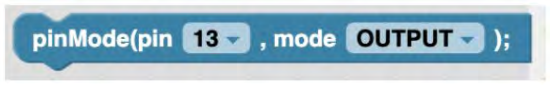
This set of activities introduces students to the terminology and concepts of physical computing and controlling digital devices, including the concepts of code blocks, commands, arguments (input values to commands), run/execute, bugs, debugging, input/output, breadboards, circuits, and Morse Code.

The following sections reproduce the online activity descriptions and possible solutions for the Moon Base SOS activities at www.roboblockly.com.

Activity 1. *Blink 1: Making an LED Blink with Arduino*

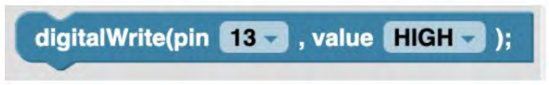
Initial prompt: In this activity you will gain a better understanding of how to use the Arduino microcontroller to manipulate a digital output device.

Lesson description: In this activity, you will be using the Arduino microcontroller to manipulate a digital output device, in this case an LED. Each digital output device has two possible states, HIGH and LOW, usually corresponding to whether the device is on (HIGH) or off (LOW). The HIGH and LOW states are represented by voltages. The Arduino uses 5 volts to represent HIGH and 0 volts to represent LOW. So to turn on an LED, we instruct the Arduino to send a "HIGH" (or 5V) signal to it. To turn it off, the Arduino sends a "LOW" (or 0V) signal to the LED.
To have your Arduino output voltage (i.e., send the voltage signals) to an LED, you need to know which Arduino pin the LED is connected to, and then set up that pin for output. We do this using the function **pinMode(PIN#, MODE)**, where MODE is either OUTPUT or INPUT and PIN# is the number of the pin being used (13, in this example):



```
pinMode(pin 13 , mode OUTPUT );
```

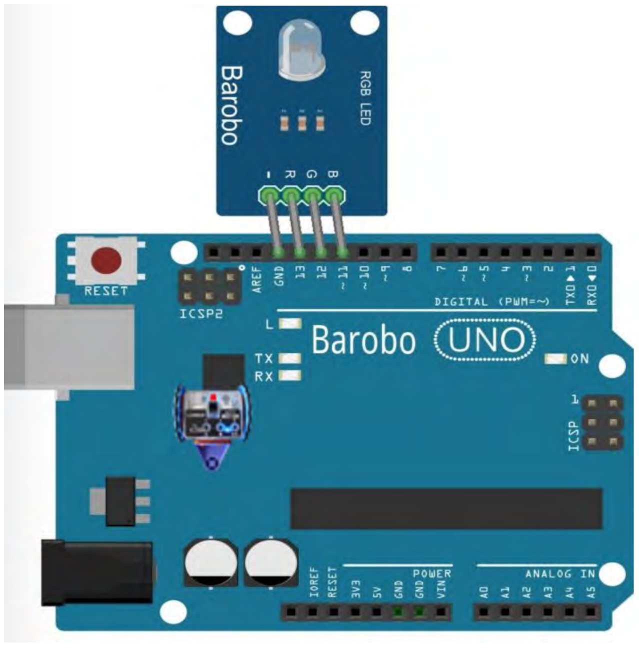
Once the pinMode is set (and the LED connected properly to that pin), we can instruct the Arduino to turn on the LED by sending a HIGH (5V) signal to that pin, or turn it off by sending a LOW (0V) signal to the pin. To do this, we "write" the signal to the pin by using the function **digitalWrite(PIN#, STATE)**, where STATE is either HIGH or LOW:



```
digitalWrite(pin 13 , value HIGH );
```

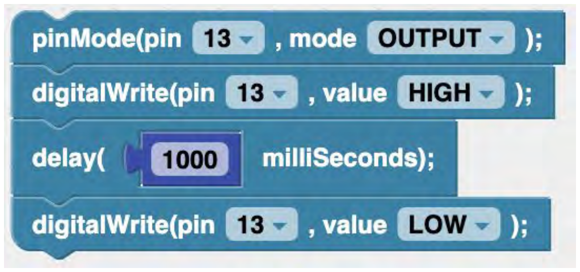
The pre-placed code blocks in the Workspace show how to put it all together, and the picture below shows how the Barobo RGB LED module should be connected to the Arduino for this activity. In particular, connect the LED module to pins 11, 12, 13, and GND on the Arduino as shown, so that the pin labeled "R" on the RGB LED module is plugged into pin 13 on the

Arduino (and the "G" pin is in pin 12, the "B" pin in pin 11, and the "-" pin in the GND, or ground, pin).



Blocks used: pinMode(), digitalWrite(), delay()

Pre-placed blocks:

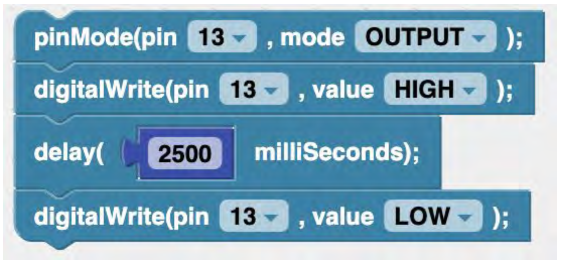


```
pinMode(pin 13, mode OUTPUT);
digitalWrite(pin 13, value HIGH);
delay( 1000 milliSeconds);
digitalWrite(pin 13, value LOW);
```

Problem statement: Use the Arduino to make the red LED in the Barobo RGB LED module blink once by running the pre-placed code blocks, which will turn on the red LED, wait one second (using the delay command--a millisecond is 1/1000 of a second), and then turn it off. Once it is working, make a change to the code so that the LED turns on for 2.5 seconds.

Hint: Make sure that the pin labeled "R" on the RGB LED module is plugged into socket 13 on the Arduino. For the code, the **pinMode** for pin 13 should be set to output. Then the **digitalWrite** function is used to turn on the red LED by setting pin 13 to a HIGH state, the **delay** function is used to wait one second, and finally the **digitalWrite** function is used again to turn off the LED by setting pin 13 to a LOW state. To increase the blink time, change the value used in the **delay** function.

Possible solution:



```
pinMode(pin 13 , mode OUTPUT );
digitalWrite(pin 13 , value HIGH );
delay( 2500 milliSeconds);
digitalWrite(pin 13 , value LOW );
```

Activity 2. *Blink 2: Multiple Blinking with Arduino*

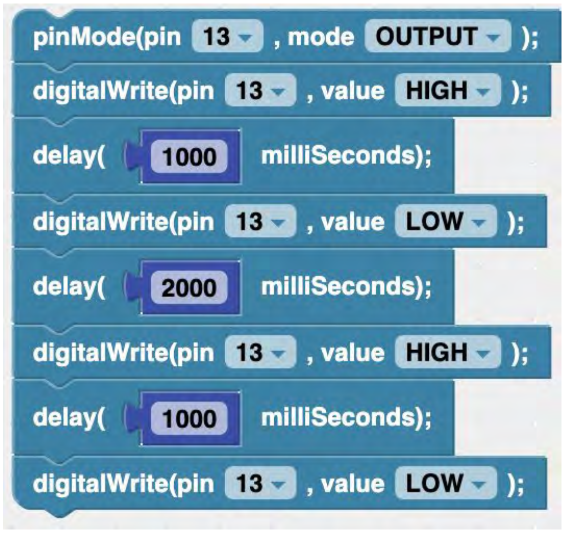
Initial prompt: This activity will give you more practice in controlling a digital output device using the Arduino microcontroller.

Lesson description: In this activity, you will expand on the previous "Blink 1" activity by using the Arduino to blink an LED multiple times. As before, the picture below shows how the Barobo RGB LED module should be connected to the Arduino.

Once you have the pre-placed code that is shown in the Workspace working, and you have tried changing both the delay time and the lengths of the blinks, the next step is to add more code blocks so that the LED will blink three times. To do so, use the computer's mouse (or trackpad) to drag and drop more **delay** and **digitalWrite** code blocks from the "Arduino" block section into the Workspace and connect them at the bottom of the existing blocks (they should click together).

Blocks used: pinMode(), digitalWrite(), delay()

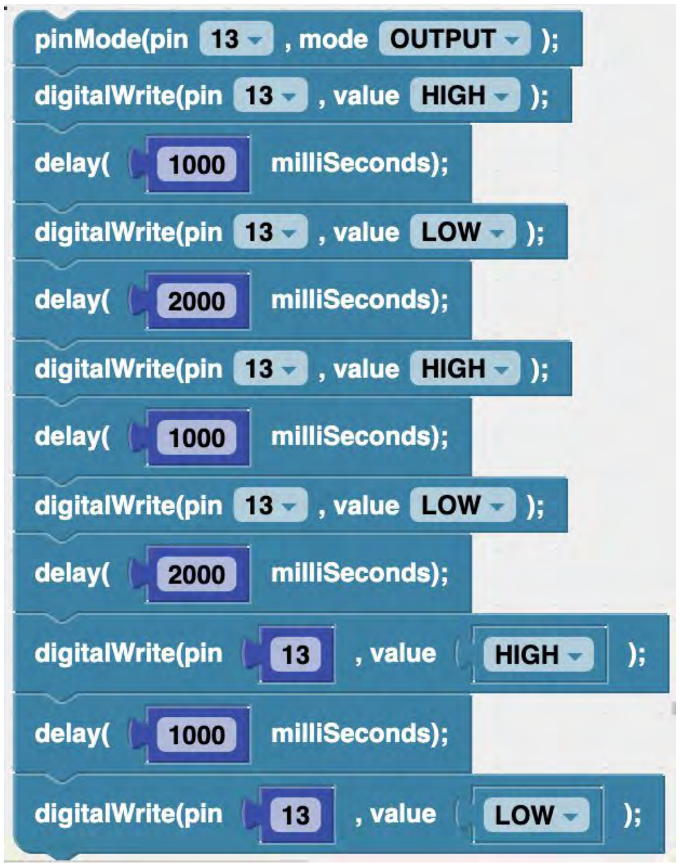
Pre-placed blocks:


```
pinMode(pin 13 , mode OUTPUT );
digitalWrite(pin 13 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 13 , value LOW );
delay( 2000 milliSeconds);
digitalWrite(pin 13 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 13 , value LOW );
```

Problem statement: The pre-placed blocks make the LED blink two times, with a two second delay between the blinks. Add code blocks so that the LED blinks three times for one second each, with two seconds between the blinks. Can you do other variations?

Hint: The code blocks to create the third blink will be the same types as those used for the first two blinks. But don't forget to put a two-second delay between the second and third blink (use the delay block).

Possible solution:



```
pinMode(pin 13 , mode OUTPUT );
digitalWrite(pin 13 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 13 , value LOW );
delay( 2000 milliSeconds);
digitalWrite(pin 13 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 13 , value LOW );
delay( 2000 milliSeconds);
digitalWrite(pin 13 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 13 , value LOW );
```

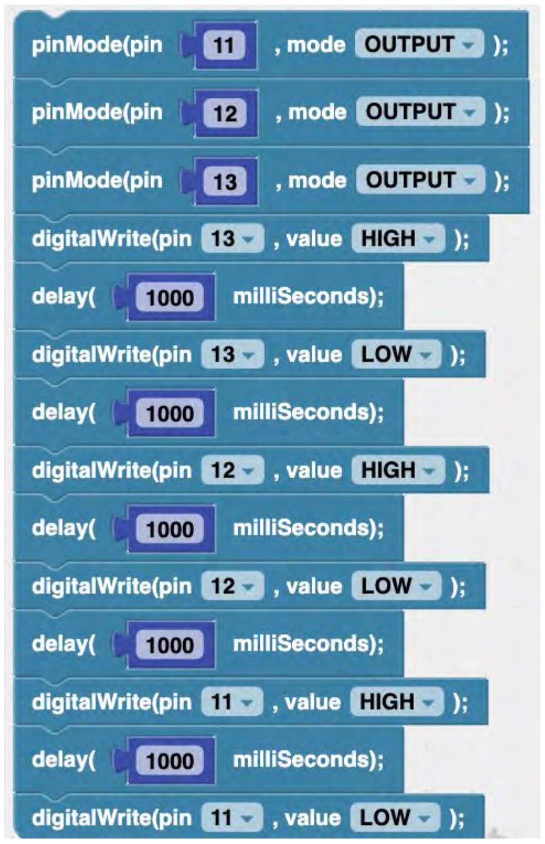
Activity 3. *Blink 3: Multiple Color Blinking*

Initial prompt: This activity will expand your capabilities in controlling a digital output device using the Arduino microcontroller.

Lesson description: In this activity, you will expand on the previous "Blink 1" and "Blink 2" activities by using the Arduino to control all three LEDs (red, green, and blue) in the LED module. The concept is simple: The pin for the red LED is plugged in to pin position 13 on the Arduino, and in the previous activities we controlled it by using the **pinMode** and **digitalWrite** commands with pin 13 specified. As seen in the picture below, the green LED corresponds to pin 12 and the blue LED corresponds to pin 11. So we use the same commands to turn on and off the LEDs, but with different pin numbers.

Blocks used: pinMode(), digitalWrite(), delay()

Pre-placed blocks:

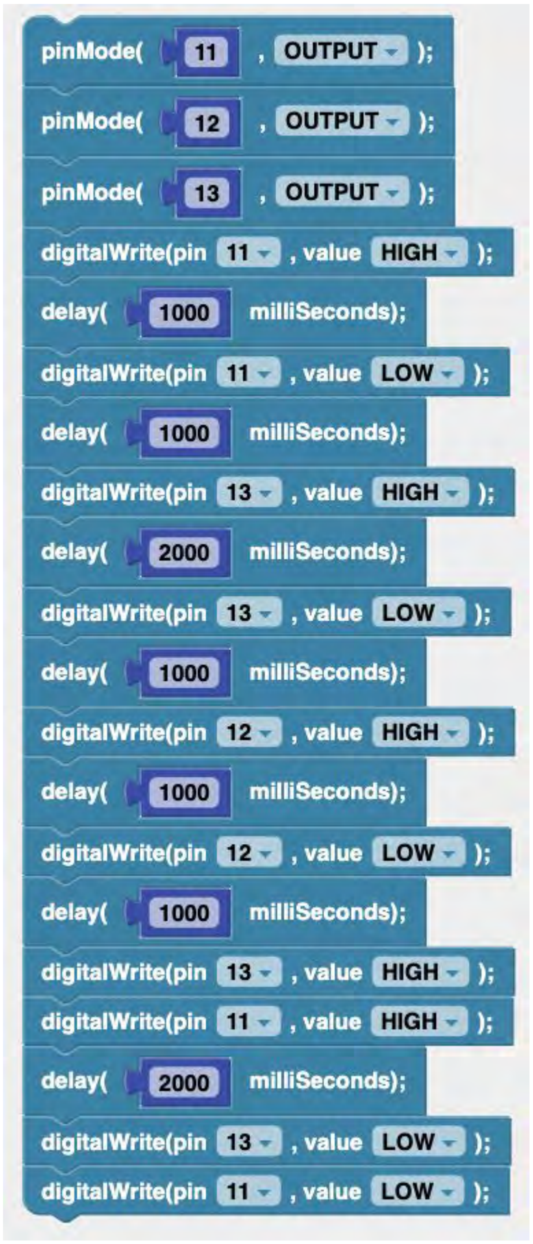


```
pinMode(pin 11 , mode OUTPUT );
pinMode(pin 12 , mode OUTPUT );
pinMode(pin 13 , mode OUTPUT );
digitalWrite(pin 13 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 13 , value LOW );
delay( 1000 milliSeconds);
digitalWrite(pin 12 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 12 , value LOW );
delay( 1000 milliSeconds);
digitalWrite(pin 11 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 11 , value LOW );
```

Problem statement: Run the pre-placed blocks, which will blink each LED color for one second (red, then green, then blue), with a one second delay between them. Then change the code and add blocks so that it blinks blue for 1 second, red for 2 seconds, green for 1 second, and both red and blue together for 2 seconds, with 1 second between each blink.

Hint: Make sure the pin modes for all three pins (11, 12, and 13) are set to OUTPUT. (This only has to be done once for each, at the beginning of the code.)

Possible solution:

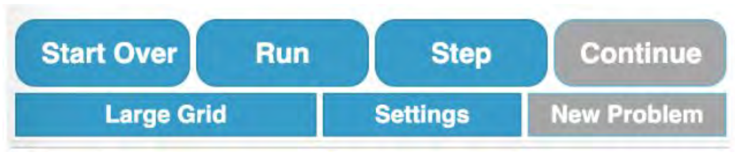


```
pinMode( 11 , OUTPUT );
pinMode( 12 , OUTPUT );
pinMode( 13 , OUTPUT );
digitalWrite(pin 11 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 11 , value LOW );
delay( 1000 milliSeconds);
digitalWrite(pin 13 , value HIGH );
delay( 2000 milliSeconds);
digitalWrite(pin 13 , value LOW );
delay( 1000 milliSeconds);
digitalWrite(pin 12 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 12 , value LOW );
delay( 1000 milliSeconds);
digitalWrite(pin 13 , value HIGH );
digitalWrite(pin 11 , value HIGH );
delay( 2000 milliSeconds);
digitalWrite(pin 13 , value LOW );
digitalWrite(pin 11 , value LOW );
```

Activity 4. *Blink 4: Using a Breadboard*

Initial prompt: In this activity you will gain a better understanding of how to use the Arduino microcontroller to manipulate a digital output device, connected via a breadboard.

Lesson description: In previous "Blink" activities, we plugged an RGB LED module directly into the Arduino. In order to use multiple devices and a variety of circuits with the Arduino, we need to use a connection device known as a "breadboard." We can plug various devices into the breadboard, and then use wires to create a circuit and connect it to the Arduino, as shown in the picture below for a single LED device connected to pin 3 of the Arduino via the wires (with a 220 Ohm Red-Red-Brown resistor as part of the circuit). Use the "Large Grid" button (below the "Start Over" and "Run" buttons) to enlarge the picture:

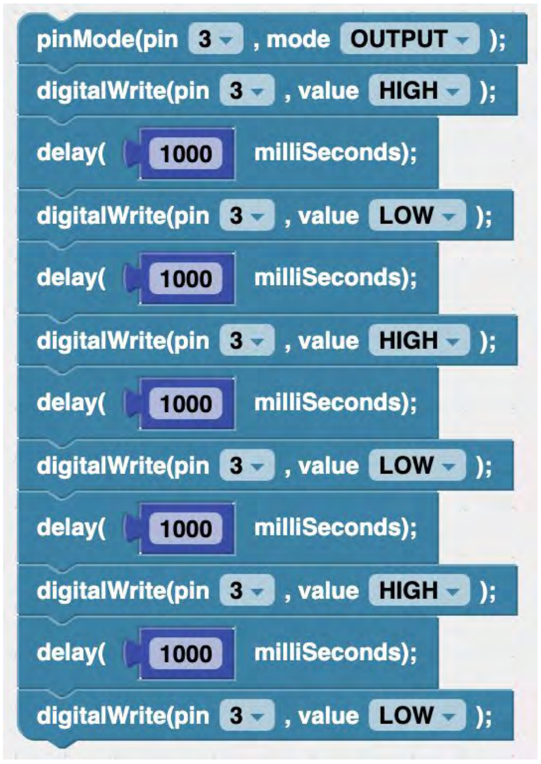


The individual LED device has two legs, a shorter one and a longer one:

The shorter leg is connected to ground through the resistor, while the longer one (shown with a kink in it in the breadboard picture) is connected to a pin with power (pin 3, in this case).

Blocks used: pinMode(), digitalWrite(), delay()

Pre-placed blocks:

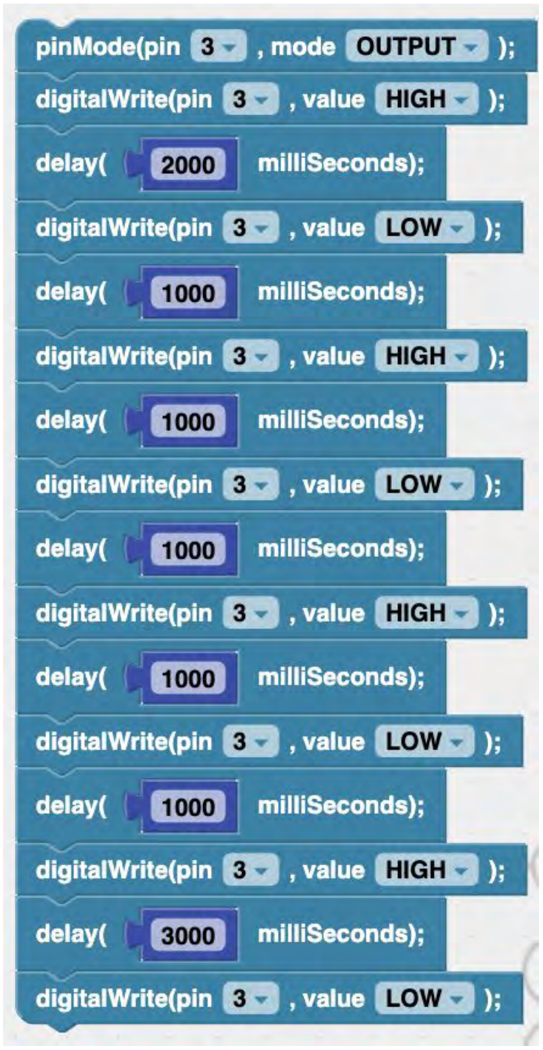


```
pinMode(pin 3, mode OUTPUT);
digitalWrite(pin 3, value HIGH);
delay( 1000 milliSeconds);
digitalWrite(pin 3, value LOW);
delay( 1000 milliSeconds);
digitalWrite(pin 3, value HIGH);
delay( 1000 milliSeconds);
digitalWrite(pin 3, value LOW);
delay( 1000 milliSeconds);
digitalWrite(pin 3, value HIGH);
delay( 1000 milliSeconds);
digitalWrite(pin 3, value LOW);
```

Problem statement: Run the code blocks provided to turn on the red LED for one second and then turn it off, for three times total. Once it is working, change the code so that it blinks 4 times total, for 2 seconds the first blink, then 1 second each for the second and third blinks, and finally 3 seconds for the fourth blink, with 1 second between each blink.

Hint: Check your wiring to make sure it's connected to pin 3, and that the **pinMode** for pin 3 is set to output. Then the **digitalWrite** command is used to turn on the red LED by setting pin 3 to a HIGH state, the **delay** command is used to wait one second, and finally the **digitalWrite** command is used again to turn off the LED by setting pin 3 to a LOW state. To increase the blink time, change the value used in the **delay** command.

Possible solution:

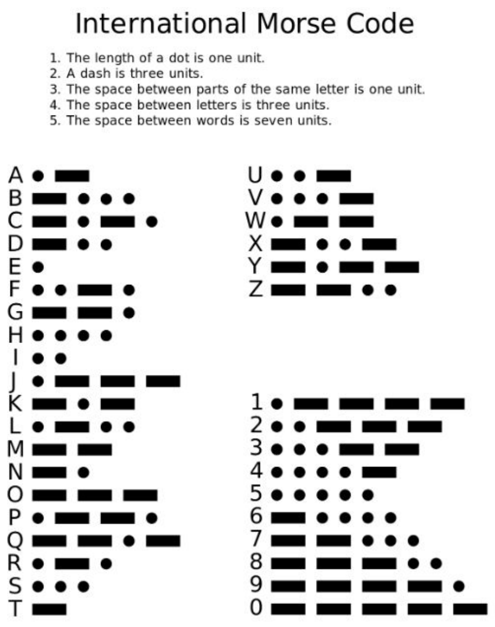


```
pinMode(pin 3 , mode OUTPUT );
digitalWrite(pin 3 , value HIGH );
delay( 2000 milliSeconds);
digitalWrite(pin 3 , value LOW );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value LOW );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value LOW );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value HIGH );
delay( 3000 milliSeconds);
digitalWrite(pin 3 , value LOW );
```

Activity 5. *Sending an SOS Message with Morse Code*

Initial prompt: In this problem, we are going to use Morse code to send an SOS message via an LED connected to the Arduino.

Lesson description: Now that we have an LED device wired to our Arduino microcontroller using a breadboard (or directly via an RGB LED module), we can send a message using light signals and Morse code. (Of course our LED is low intensity, but we will pretend it represents a high-intensity laser device that can be seen over great distances.) Morse code is a coding scheme used in telecommunication that encodes text characters as standardized sequences of two different signal durations called dots and dashes. The letter A, for example, is a dot followed by a dash.


International Morse Code
1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.
A B C D E F G H I J K L M N O P Q R S T
U V W X Y Z
1 2 3 4 5 6 7 8 9 0

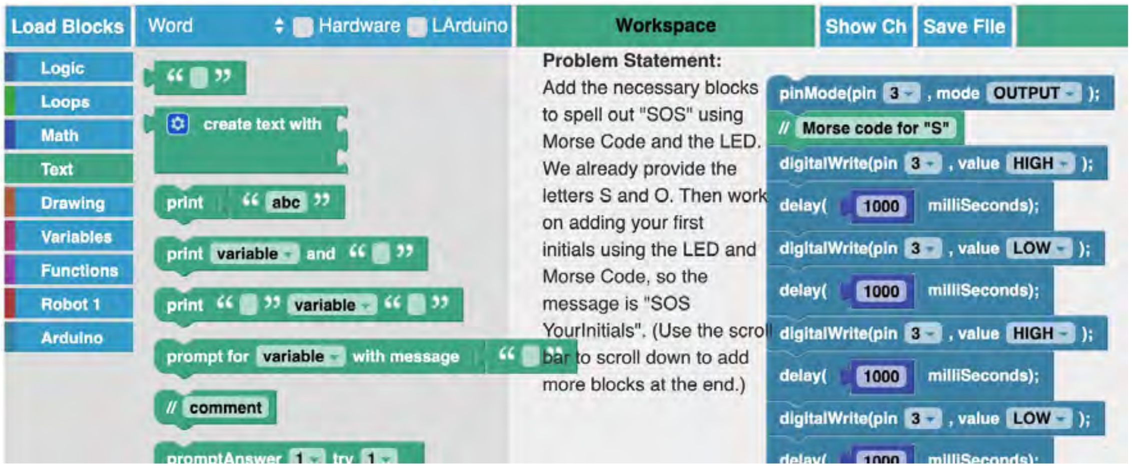
The Morse code rules we will use for signalling are:

- a dot lasts for one second
- a dash last for three seconds
- the space between dots and dashes that are part of the same letter is one second
- the space between different letters is three seconds
- the space between different words is seven seconds

For an SOS ("help") message, an S is three dots, an O is three dashes, and the second S is three dots again.

We have also introduced a new type of code block in the pre-placed blocks: a "comment" block:

We add comments to our code to help the reader understand what the code is doing, e.g., the "Morse code for "S" " comment in the pre-placed blocks. When the code is run, the computer ignores the comments (i.e., does not execute them). You can find the comment block by clicking the "Text" button in the middle block section (just below the "Math" button and above the "Drawing" button):

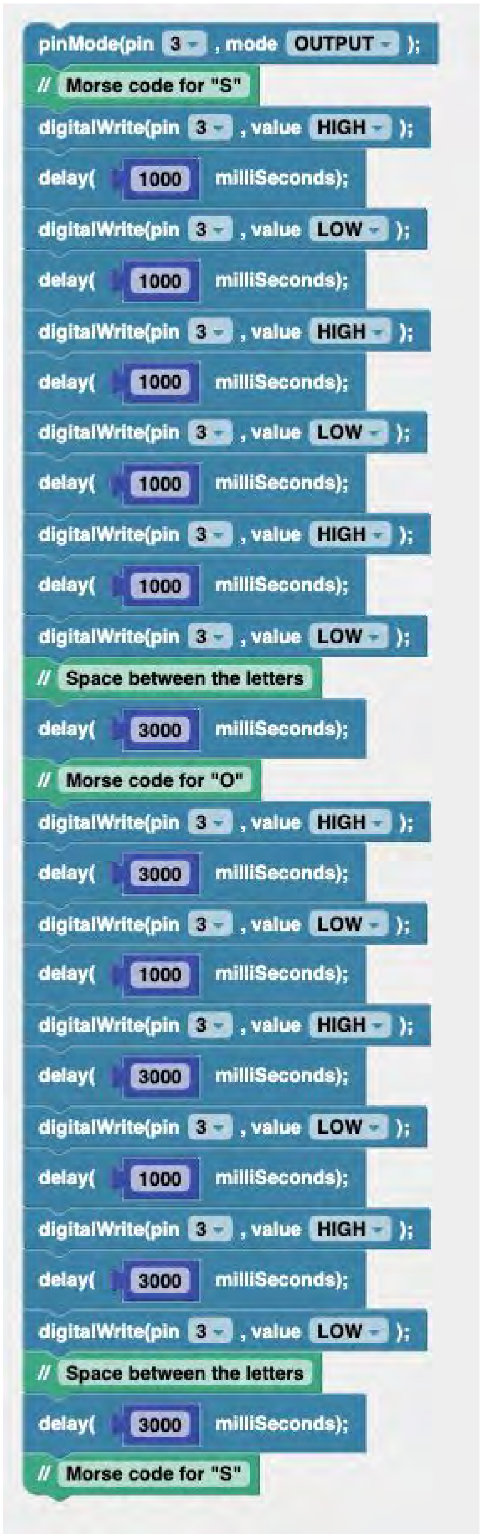


Problem Statement:
Add the necessary blocks to spell out "SOS" using Morse Code and the LED. We already provide the letters S and O. Then work on adding your first initials using the LED and Morse Code, so the message is "SOS YourInitials". (Use the scroll bar to scroll down to add more blocks at the end.)

```
pinMode(pin 3 , mode OUTPUT );
// Morse code for "S"
digitalWrite(pin 3 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value LOW );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value LOW );
delay( 1000 milliSeconds);
```

Blocks used: pinMode(), digitalWrite(), delay()

Pre-placed blocks:

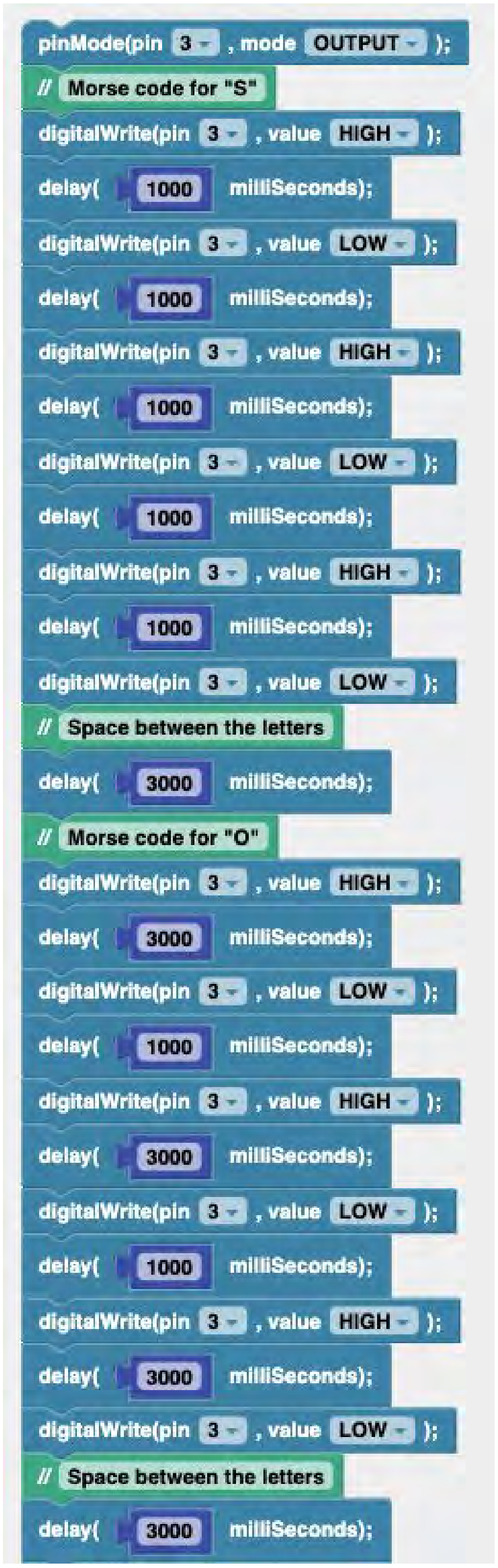


```
pinMode(pin 3 , mode OUTPUT );
// Morse code for "S"
digitalWrite(pin 3 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value LOW );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value LOW );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value LOW );
// Space between the letters
delay( 3000 milliSeconds);
// Morse code for "O"
digitalWrite(pin 3 , value HIGH );
delay( 3000 milliSeconds);
digitalWrite(pin 3 , value LOW );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value HIGH );
delay( 3000 milliSeconds);
digitalWrite(pin 3 , value LOW );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value HIGH );
delay( 3000 milliSeconds);
digitalWrite(pin 3 , value LOW );
// Space between the letters
delay( 3000 milliSeconds);
// Morse code for "S"
```

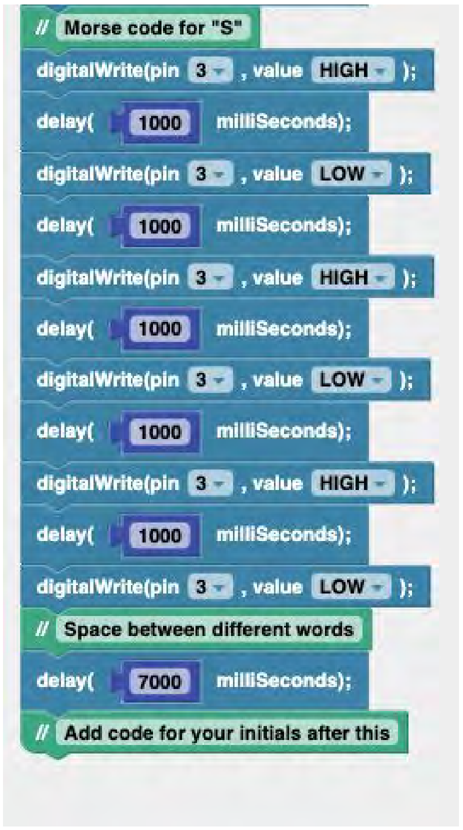
Problem statement: Add the necessary blocks to spell out "SOS" using Morse Code and the LED. We already provide the letters S and O. Then work on adding your initials using the LED and Morse Code, so the message is "SOS YourInitials". (Use the scroll bar to scroll down to add more blocks at the end.)

Hint: Make sure that you have the right pin values for your wiring setup (probably pin 13 for the breadboard set up or pin 3 for the red LED in the RGB LED module).

Possible solution:



```
pinMode(pin 3 , mode OUTPUT );
// Morse code for "S"
digitalWrite(pin 3 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value LOW );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value LOW );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value LOW );
// Space between the letters
delay( 3000 milliSeconds);
// Morse code for "O"
digitalWrite(pin 3 , value HIGH );
delay( 3000 milliSeconds);
digitalWrite(pin 3 , value LOW );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value HIGH );
delay( 3000 milliSeconds);
digitalWrite(pin 3 , value LOW );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value HIGH );
delay( 3000 milliSeconds);
digitalWrite(pin 3 , value LOW );
// Space between the letters
delay( 3000 milliSeconds);
```



```
// Morse code for "S"
digitalWrite(pin 3 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value LOW );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value LOW );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value HIGH );
delay( 1000 milliSeconds);
digitalWrite(pin 3 , value LOW );
// Space between different words
delay( 7000 milliSeconds);
// Add code for your initials after this
```