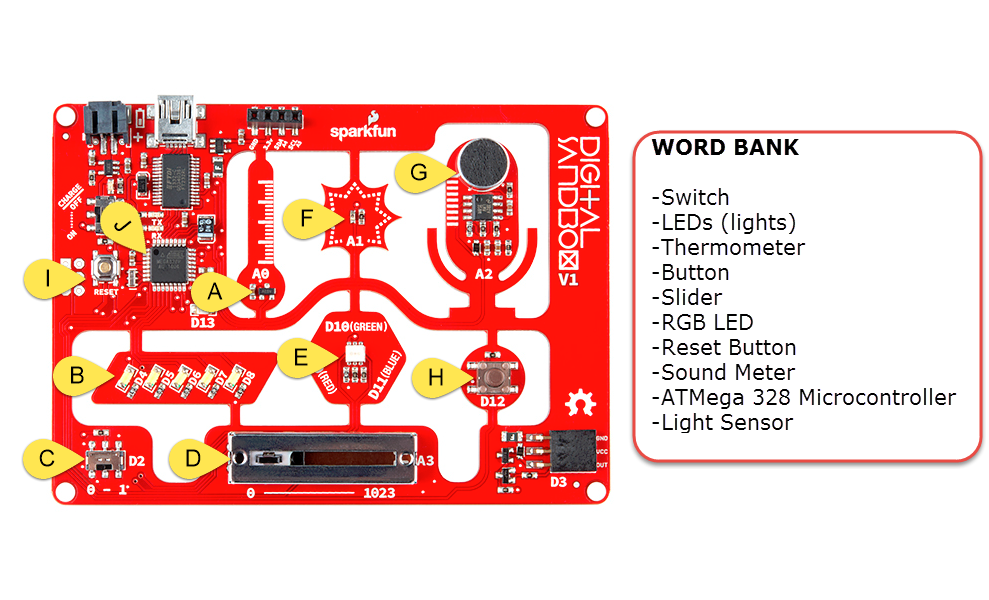
1 - Introduction to Physical Computing with

the Digital SandBox

# Getting Started

The Digital SandBox is an integrated microcontroller board with a variety of sensors, switches, buttons, and LEDs.



**1. Looking at the board and the word bank above - try to identify the name of each labelled component.**

|  |  |
| --- | --- |
| 1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| 1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| 1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| 1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| 1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

The ATMega328 is a microcontroller or the “brain” of this board. It can be programmed to blink LEDs, read sensors, and interact with your physical environment. We define an **INPUT** to be any type of sensor or external signal that the microcontroller reads or uses, and an **OUTPUT** to be a signal that the microcontroller generates.

**2. Label each of the components with an “I” for Input or a “O” for Output.**

**3. Notice that there are labels marked D4, D5, D6… and A0, A1, A2… What do you think is the difference between the “D” and the “A” components?**

# Programming Environment

There are many options to program an Arduino board. Most people either use the traditional Arduino IDE or [codebender](http://codebender.cc). This series of tutorials will walk through using codebender as an on-line programming environment for Arduino. It is like the google docs of the physical hardware realm. Codebender is great for sharing code, working on multiple machines, and working on chromebooks.

Open up a web-browser to [www.codebender.cc](http://www.codebender.cc). If you haven't created an account yet, do so. Codebender will also prompt you to install drivers and a browser-extension. Note: codebender works best in **Firefox** or **Chrome**.

## Getting Started

The setup will walk you all the way through connecting up a board to test to make sure that everything is working properly. On step 4 of 5, it will prompt you to: *Connect your device, select port and click on 'Run on Arduino'*. When you see this go ahead and connect the Digital Sandbox to your computer using the USB cable.



Select the **SparkFun Digital SandBox** and the appropriate serial port. On Windows machines, this will appear as a **COM#** and on OS X machines this will appear as something resembling **/dev/cu.usbserial-Axxxx**. After you've selected these, click 'Run on Arduino', and watch what happens! If all is correct, you should have a blinking LED!

If a valid port does not appear, you may need to manually install drivers. Codebender facilitates the driver installation at: <https://codebender.cc/static/walkthrough/page/3> Double check to make sure that you have administrative privledges on your machine.

## Writing our own code:

Let's get started and see how this program works. In Arduino, we call the programs "sketches." You'll see this name throughout. Click on the **Homepage** button to go to your homepage where you can create a new Sketch.



Whenever you want to return to your homepage to start a new sketch, simply click on the **Codebender icon** in the upper left to return to this page.

Click **Create Sketch** button to start a new program and give it a name. Generally, program names should not have special characters, spaces, or start with a number. This rule holds true for almost all naming conventions in programming. Let's call this sketch **myBlink**. You should have some code that looks like this. It is what we call the **BareMinimum** sketch.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | void setup()  {    }  void loop()  {    } |

Look at the code. What patterns, clues, or structure do you notice about the code? Write 2 - 3 of your observations below:

|  |
| --- |
|  |

Notice that there is a structure to the code? There are two parts - one called **setup**() and one called **loop**(). This structure is what are called functions. Your program must be organized so that the Arduino can understand it.

Every function has a ***data type***, a ***name***, and one or more lines of code that go with it. setup() and loop() have an empty datatype that is called void. The code that goes with the function is encased by two curly braces { and }. Can you find these keys on your keyboard?

# Do this

Add these few lines so that your code looks like this. Remember that when writing code, you need to pay attention to all of the details. Make sure that the text is typed exactly it looks below - for example, the M in pinMode must be capitalized, all of the letters of OUTPUT must be in caps, and each line needs to have a semi-colon at the end.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | void setup()  {  **pinMode(4, OUTPUT);**  }  void loop()  {  **digitalWrite(4, HIGH);**  **delay(500);**  **digitalWrite(4, LOW);**  **delay(500);**  } |

After you’ve double checked that your code matches the example above, upload this code to your Digital Sandbox by clicking on →**Run on Arduino** or (CTRL+U).

Wait a few seconds for your computer to compile the code and then upload it to the Digital SandBox. You’ll notice some blinking red (TX) and green (RX) lights as the program is sent over. What’s happening? Do you see a different blinking LED?

# Hack (Change something)

We use the term “hack” to mean alter or change something. So, play around with this. How would you get two LEDs to blink at the same time? Can you figure out what the four lines of code in the loop() part of the code are doing? Play around and see if you can get two LEDs to turn on at the same time!

## void setup()

The setup() function runs once at the beginning when you first power up the board. The Arduino (Digital Sandbox) is a generic microcontroller board. The sketch doesn’t know what pins are connected to what. To configure or setup these pins to control LEDs, we use a command called pinMode().

|  |
| --- |
| **pinMode(4, OUTPUT);** |

You'll need to add a separate pinMode command for any LED pin that we use. For now, we're just going to use the first one (the one labeled D4). If you wanted to control another LED, you'd simply add another line of code here that says pinMode(5, OUTPUT); for example.

## void loop()

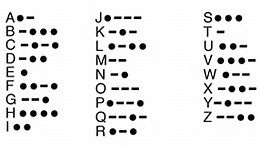
Now, for the blinking part. The second part of the code is called the loop(). This part of the code will repeat over and over again. To blink, we need an instruction that turns the LED on, an instruction that waits, an instruction that turns the LED off, and finally an instruction that waits again before repeating. That’s exactly what’s going on in our code. There are the two new commands (also functions): **digitalWrite**() and **delay**().

|  |
| --- |
| **digitalWrite([pin], [state]);**  **delay([time\_ms]);** |

**digitalWrite**() uses two arguments, a pin and a state. To turn on the LED on pin 4, we use the command: **digitalWrite**(4, HIGH); To turn the LED off, we use the command: **digitalWrite**(4, LOW);

**delay()** causes the program to pause for a certain number of milliseconds before continuing to the next line of code. To create a 1 second delay, you would type delay(1000);

# Things to try...

There are a lot of things we can do with this board, but for now we’re only going to work with one LED. Try out these activities using just one LED on your board.

* **Think Fast!** How fast of a blink can you see? Change both delay times to 1 millisecond. Can you still see it blinking? Change the number until you can just barely see it blinking. Now, slide the digital sandbox back and forth quickly while watching the LED. What do you see? What do you think is going on?

|  |  |
| --- | --- |
| * **Dot-dot-dot-dash-dash-dash…** In the old days, people used dots and dashes to communicate to each other. A dot was a short blip and a dash was a longer blip. This is what is called Morse code. Compose a short message (maybe a single word) and write out the code needed to reproduce it. | Hint: to create an “A”, the code snippet might look like:  digitalWrite(4, HIGH);  delay(200);  digitalWrite(4, LOW);  delay(200);  digitalWrite(4, HIGH);  delay(600);  digitalWrite(4, LOW);  delay(600); |