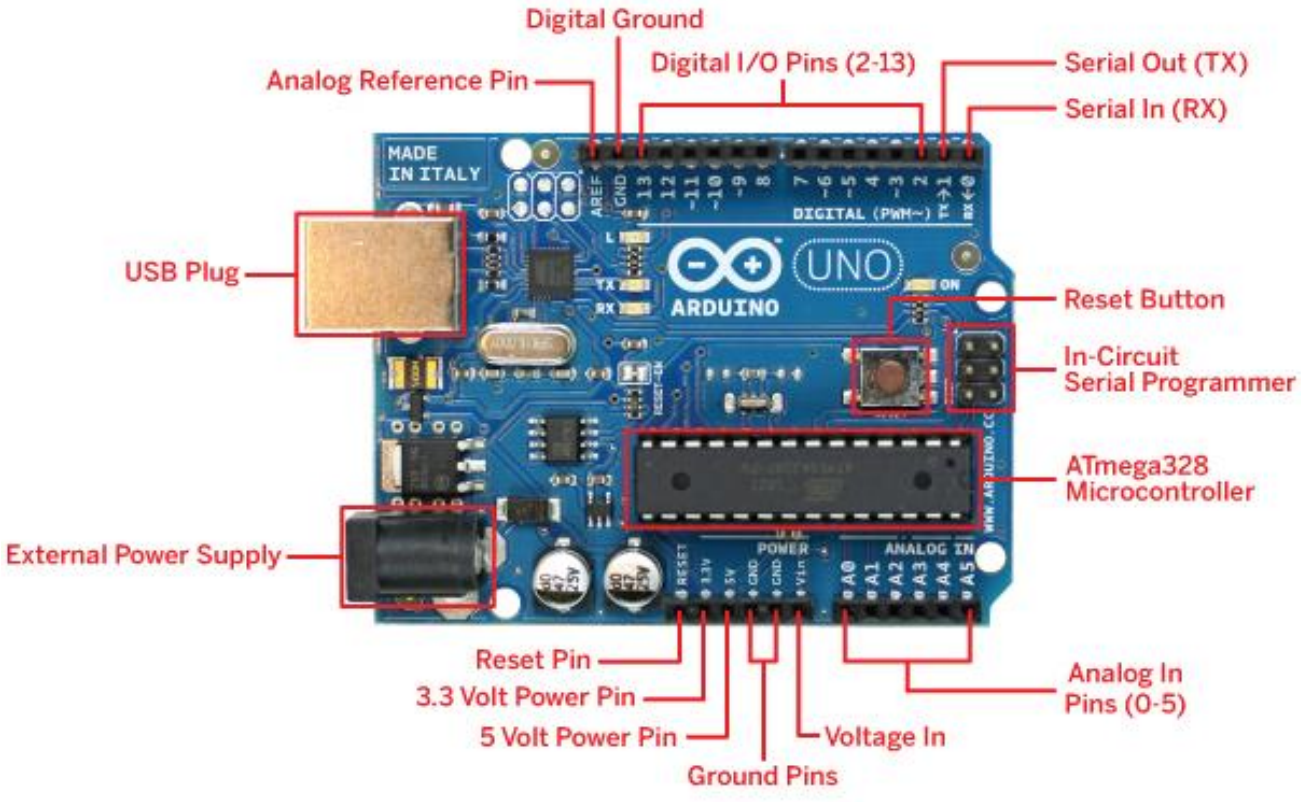
STEM Maker Team Training Project
Student Notes
Basic Knowledge of Useful Tools

Designing electronic circuits and writing programs for the microcontroller are essential in creating an electronic device prototype. It's a must for a maker to get familiar with the following useful tools including Arduino UNO R3, breadboard, breadboard diagram software and Arduino IDE.

1. Arduino UNO R3

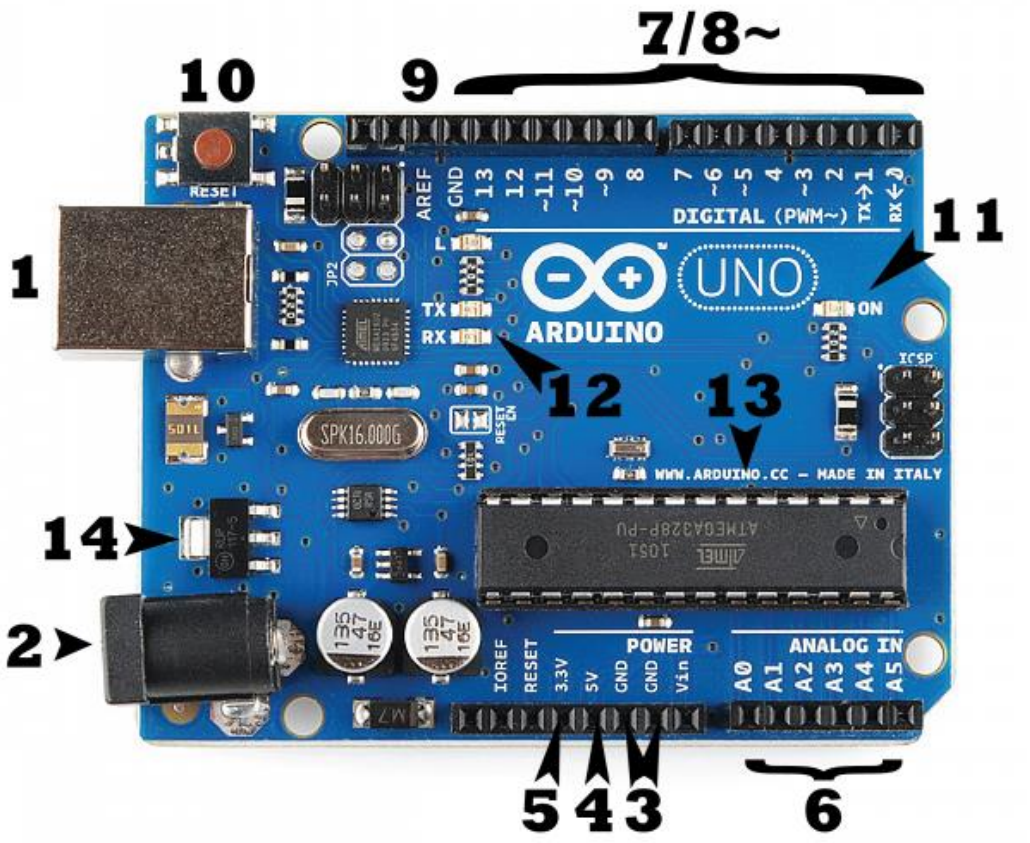
Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

There are many varieties of Arduino boards that can be used for different purposes. The one below is the most common type called Arduino Uno. Some boards look a bit different from the one below, but most Arduinos have the majority of these components in common:



(credit: https://www.studentcompanion.co.za/getting-started-with-flowcode-for-arduino/)

Different Components in Arduino:



(credit: https://learn.sparkfun.com/tutorials/what-is-an-arduino/all)

1. Power (USB / Barrel Jack)

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply that is terminated in a barrel jack. In the picture above the USB connection is labeled **(1)** and the barrel jack is labeled **(2)**.

The USB connection is also how you will load code onto your Arduino board.

NOTE: Do NOT use a power supply greater than 20 Volts as you will overpower (and thereby destroy) your Arduino. The recommended voltage for most Arduino models is between 6 and 12 Volts.

2. Pins (5V, 3.3V, GND, Analog, Digital, PWM, AREF)

The pins on your Arduino are the places where you connect wires to construct a circuit probably in conjunction with a breadboard and some wire. They usually have black plastic 'headers' that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

- **GND (3)**: Short for 'Ground'. There are several GND pins on the Arduino, any of which can be used to ground your circuit.
- **5V (4) & 3.3V (5)**: The 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run off of 5 or 3.3 volts.

- **Analog (6)**: The area of pins under the 'Analog In' label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.
- **Digital (7)**: Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).
- **PWM (8)**: You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM), think of these pins as being able to simulate analog output (like fading an LED in and out).
- **AREF (9)**: Stands for Analog Reference. It is used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

3. Reset Button

Pushing **reset button (10)** will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn't repeat, but you want to test it multiple times.

4. Power LED Indicator

Just beneath and to the right of the word "UNO" on your circuit board, there's a tiny LED next to the word **'ON' (11)**. This LED should light up whenever you plug your Arduino into a power source.

5.TX RX LEDs

TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. **TX and RX indicator LEDs (12)** will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we're loading a new program onto the board).

6. Main IC

The black thing with all the metal legs is an IC, or **Integrated Circuit (13)**, as the brains of Arduino. The main IC on the Arduino is slightly different from board type to board type, but is usually from the ATmega line of IC's from the ATMEL company.

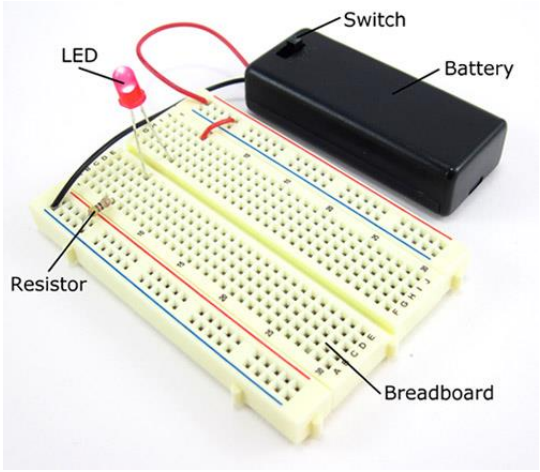
7. Voltage Regulator

The **voltage regulator (14)** controls the amount of voltage that is let into the Arduino board. It will turn away an extra voltage that might harm the circuit. Of course, it has its limits, so don't hook up your Arduino to anything greater than 20 volts.

(credit: https://learn.sparkfun.com/tutorials/what-is-an-arduino/all)

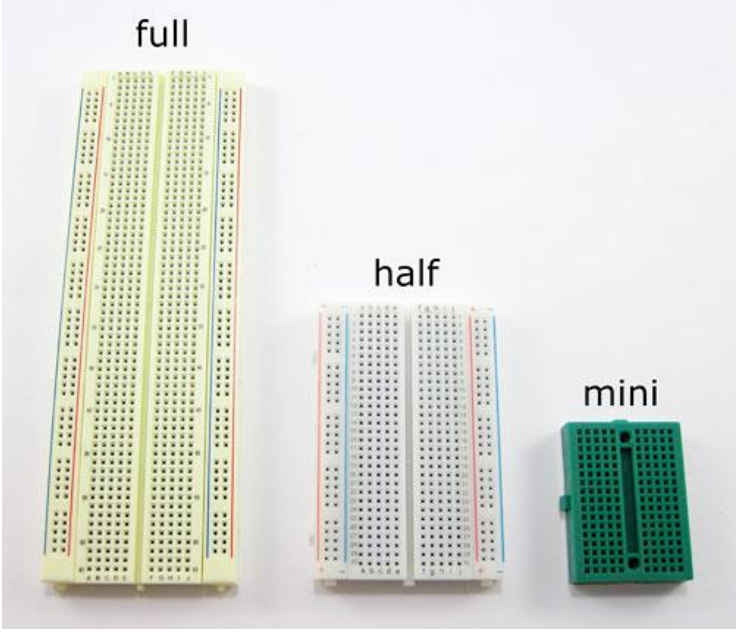
2. Breadboard

A breadboard is a rectangular plastic board with a bunch of tiny holes in it. These holes let you easily insert electronic components to **prototype** (meaning to build and test an early version of) an electronic circuit, like this one with a battery, switch, resistor, and an LED (light-emitting diode).

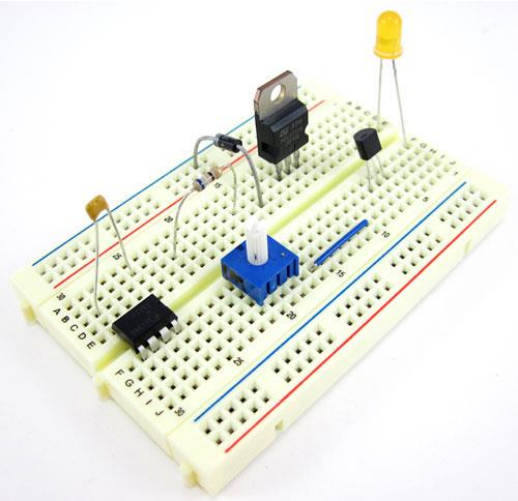


The connections are not permanent, so it is easy to *remove* a component if you make a mistake, or just start over and do a new project. This makes breadboards great for beginners who are new to electronics. You can use breadboards to make all sorts of fun electronics projects

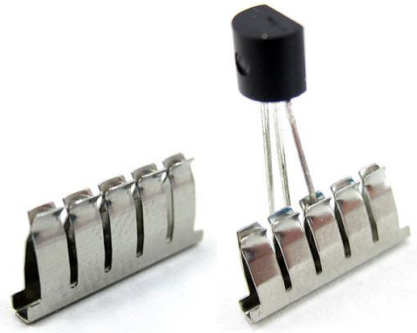
Modern breadboards are made from plastic, and come in all shapes, sizes, and even different colors. While larger and smaller sizes are available, the most common sizes you will probably see are "full-size," "half-size," and "mini" breadboards. Most breadboards also come with tabs and notches on the sides that allow you to snap multiple boards together. However, a single half-sized breadboard is sufficient for many beginner-level projects.



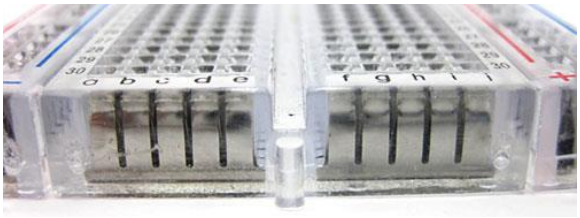
Technically, these breadboards are called **solderless** breadboards because they do not require soldering to make connections. Breadboards are designed so you can push these leads into the holes. They will be held in place snugly enough that they will not fall out (even if you turn the breadboard upside-down), but lightly enough that you can easily pull on them to remove them.



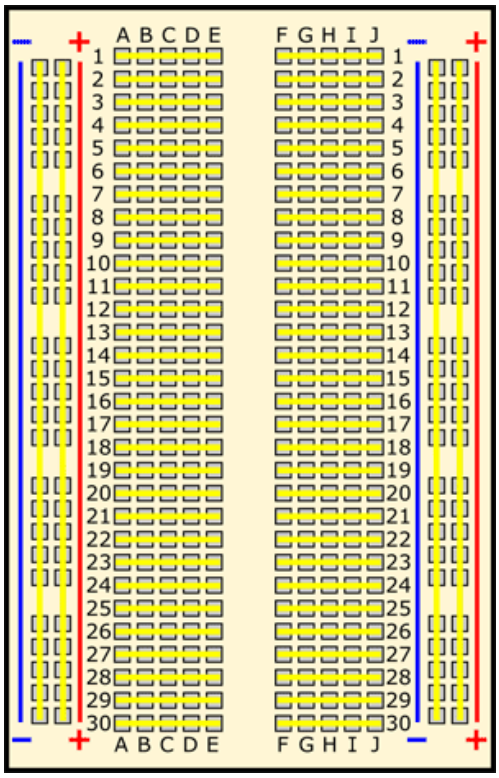
The leads can fit into the breadboard because the *inside* of a breadboard is made up of rows of tiny metal clips. This is what the clips look like when they are removed from a breadboard. When you press a component's lead into a breadboard hole, one of these clips grabs onto it.



Some breadboards are actually made of transparent plastic, so you can see the clips inside.



The inside of the breadboard is made up of sets of five metal clips. This means that each set of five holes forming a half-row (columns A–E or columns F–J) is electrically connected. For example, that means hole A1 is electrically connected to holes B1, C1, D1, and E1. It is *not* connected to hole A2, because that hole is in a different row, with a separate set of metal clips. It is also *not* connected to holes F1, G1, H1, I1, or J1, because they are on the other "half" of the breadboard—the clips are not connected across the gap in the middle. Unlike all the main breadboard rows, which are connected in sets of five holes, the buses typically run the entire length of the breadboard (but there are some exceptions). This image shows which holes are electrically connected in a typical half-sized breadboard, highlighted in yellow lines.



Buses on opposite sides of the breadboard are *not* connected to each other.

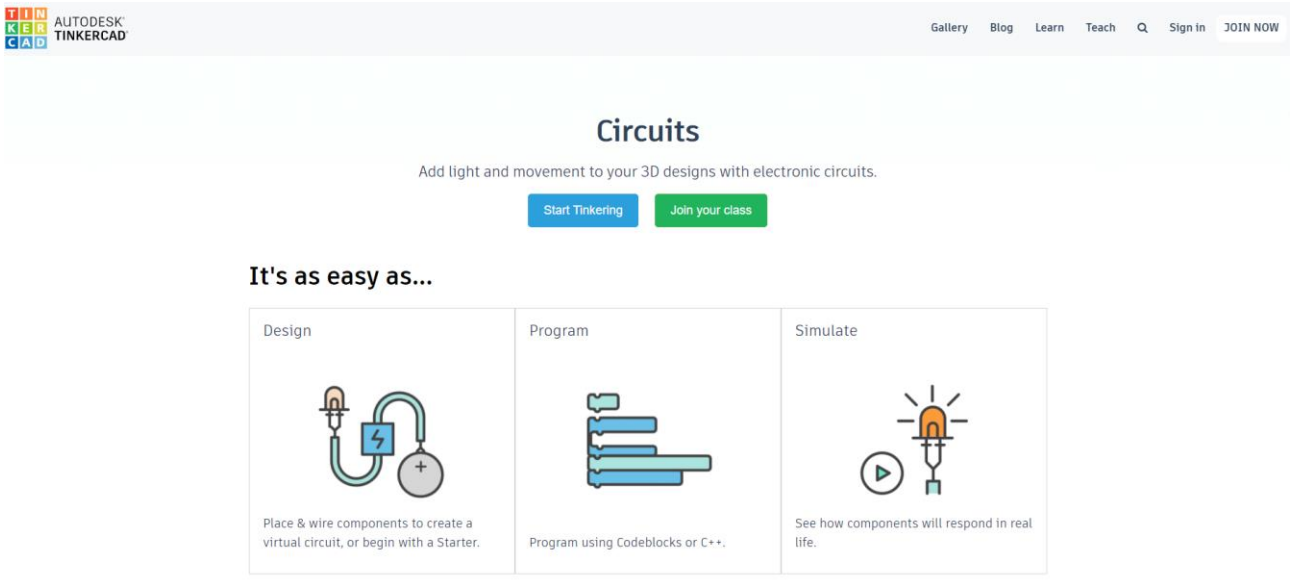
(credit: https://www.sciencebuddies.org/science-fair-projects/references/how-to-use-a-breadboard#introduction)

3. Breadboard Diagram Software

When creating a project or circuitry you usually need to draw out a schematic of the circuit before you start. There are two common software that can help you to do the job:

(1) Tinkercad

Tinkercad is a free, online 3D modeling and simple circuit designing program that runs in a web browser, known for its simplicity and ease of use.

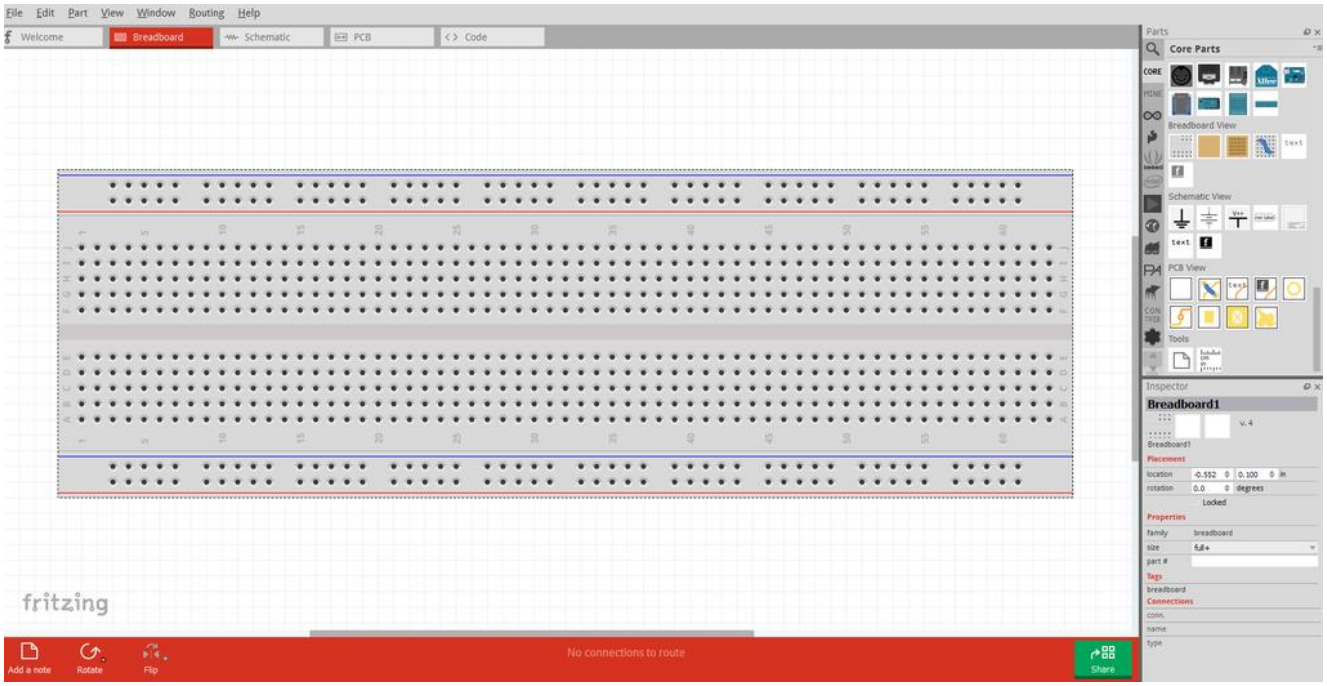


Got to https://www.tinkercad.com/learn/circuits/learning to learn more about how to use Tinkercad in designing simple circuits.

(2) Fritzing

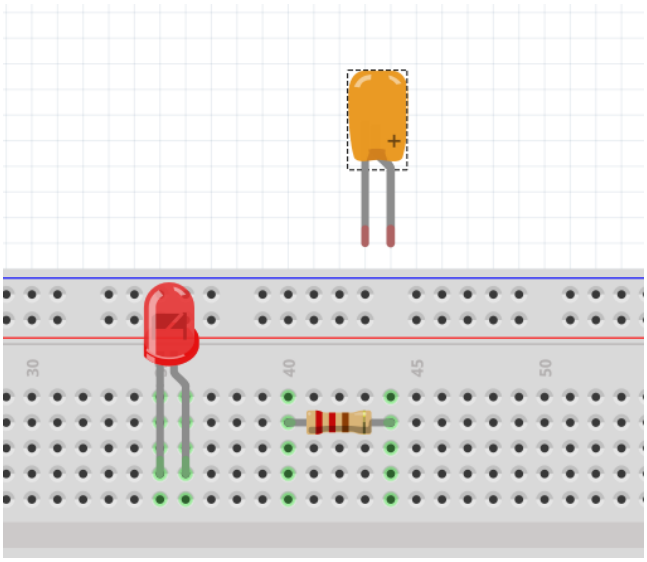
Fritzing is an open source hardware initiative that allows you to create and plan circuits out before creating them. The program comes with a bunch of pre-loaded circuit boards from different companies such as Arduino and Sparkfun. You can use these boards to plan out your project pretty accurately because they are made to be the actual size of the physical board.

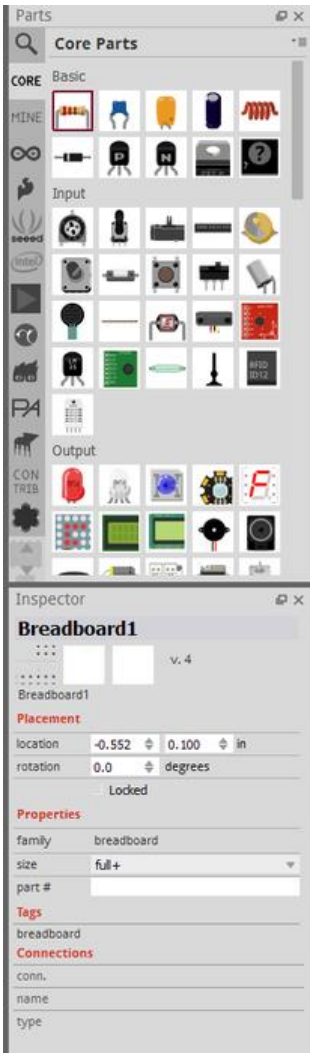
Step 1: Creating a New Circuit



When you start the Fritzing program you will be prompted with a welcome screen. This screen contains news, your recent files, and some tips of the day. From here you can select the Breadboard window to start a new circuit. There will be a blank breadboard already in place. You can either build off of this board or you can delete it. This is the main work area.

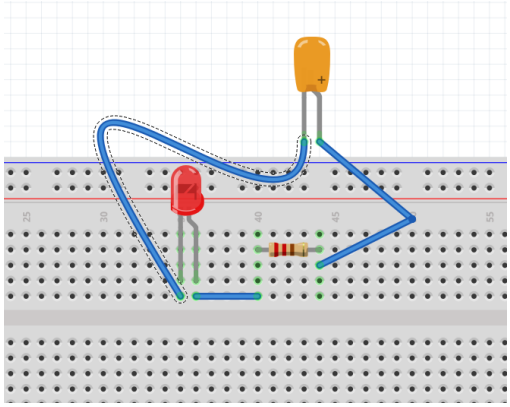
Step 2: Adding Components





Adding new components is pretty simple. There will be a window on the right side of the screen with parts. Simply go click on one of them and drag into the window you are working in. When you click on the component the program will show information about the size, resistance or the purpose of the part. You can also rotate the part by using Crtl+R. If you need to find a specific part but not sure where to look there is a handy search bar that will find what you are looking for. There are other sections of the parts window that will lead to specific company's products. You can also upload any boards or parts you have created or downloaded. If you go to the Mine section of the parts window and right click you can import any parts.

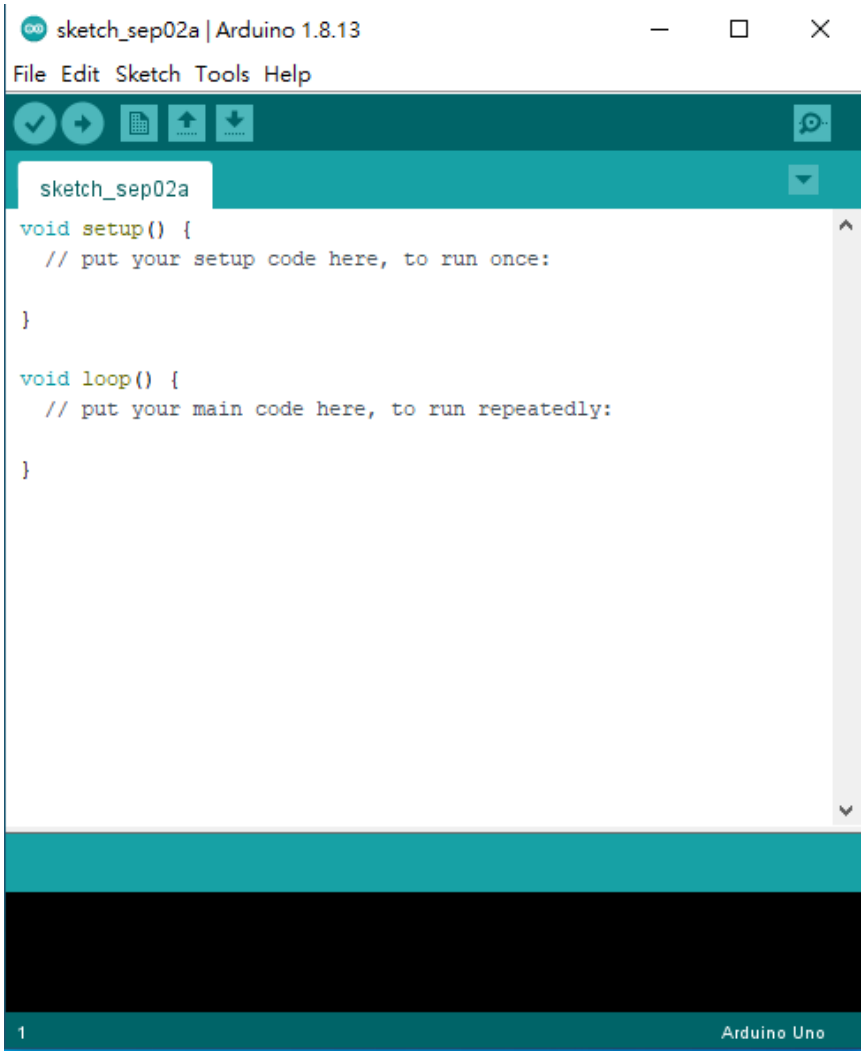
Step 3: Wiring Components



Once you have the components you will want to wire them together. By clicking on one lead of the component you can drag and start creating a wire. Drag the wire to the other pin you want to connect together and it will click to it.
If you are creating a large circuit with a lot of wires you can create bends in wires to change the path they take. Just by clicking on the segment of the wire you want to bend will create a bend point. You can manipulate this as much as you want. Another tip for working with a lot of wires is setting the wires to be curvy. If you go into the Edit drop down window and go to preferences. A window will open and if you click on the breadboard view you can have a curvy wire setting activated. This will allow you to bend wires in a unique fashion. You can also create bend points still by holding down Ctrl and clicking.
(credit: https://www.instructables.com/id/Fritzing-an-Introduction/)

4. Arduino IDE

The Arduino IDE is incredibly minimalistic, yet it provides a near-complete environment for most Arduino-based projects. The top menu bar has the standard options, including “File” (new, load save, etc.), “Edit” (font, copy, paste, etc.), “Sketch” (for compiling and programming), “Tools” (useful options for testing projects), and “Help”. The middle section of the IDE is a simple text editor that where you can enter the program code. The bottom section of the IDE is dedicated to an output window that is used to see the status of the compilation, how much memory has been used, any errors that were found in the program, and various other useful messages.



The Arduino IDE in its default state

Projects made using the Arduino are called sketches, and such sketches are usually written in a cut-down version of C++ (a number of C++ features are not included). Because programming a microcontroller is somewhat different from programming a computer, there are a number of device-specific libraries (e.g., changing pin modes, output data on pins, reading analog values, and timers). This sometimes confuses users who think Arduino is programmed in an “Arduino language.” However, the Arduino is, in fact, programmed in C++. It just uses unique libraries for the device.

The 6 Buttons

While more advanced projects will take advantage of the built-in tools in the IDE, most projects will rely on the six buttons found below the menu bar.

1. The check mark is used to verify your code. Click this once you have written your code.
2. The arrow uploads your code to the Arduino to run.
3. The dotted paper will create a new file.
4. The upward arrow is used to open an existing Arduino project.
5. The downward arrow is used to save the current file.
6. The far right button is a serial monitor, which is useful for sending data from the Arduino to the PC for debugging purposes.

(credit: https://www.digikey.com/en/maker/blogs/2018/introduction-to-the-arduino-ide)