Getting Started with TinkerCAD Circuits

TinkerCAD is a **web-based simulator** that allows you to design and test electronic circuits, including those with Arduino boards, directly in your web browser. It is particularly **beginner-friendly** and is considered an excellent tool for learning electronics and coding, especially if you do not have access to physical components.

Here's a breakdown of what TinkerCAD is and how it benefits beginners:

What is TinkerCAD Circuits?

TinkerCAD Circuits is a **fully functional Arduino + Circuit simulator** that operates within your web browser. Unlike many traditional circuit simulators that use complex "schematic views" with abstract symbols, TinkerCAD provides a **"breadboard view"**. This view uses **graphics that resemble the physical circuit parts** you would use in real life, making the learning process much easier for beginners to grasp. An added benefit is its ability to **simulate an Arduino and run code directly in the web browser**.

Key Features for Beginners

- **User-Friendly Interface**: TinkerCAD operates with a **drag-and-drop interface** for adding electrical components. Once you are in the circuit editor, you can connect different components to your Arduino, add code, and simulate how your circuit would behave if the code were uploaded to a physical Arduino board.
- **Simulation Capability**: You can test your circuit design at any time by clicking the **"Start Simulation" button** located on the right side of the toolbar. When the simulation runs, the power cord automatically plugs into the Arduino board, providing electricity.
- **Flexible Coding Environment**: TinkerCAD allows users to program in two ways:
    - **Blocks**: This method uses prebuilt code blocks that link together like puzzle pieces, similar to Scratch or Snap. It's an easier starting point but offers less flexibility for complex computations or controlling multiple motors/sensors.
    - **Text**: This option uses a C++ based programming language, allowing users to type their code to operate the circuit. This offers a wider range and flexibility for programming.
- **Component Editing**: You can easily edit the properties of components, such as changing the color of an LED or the resistance of a resistor, using an "inspector" that appears when you click on a component.
- **Wiring Made Easy**: TinkerCAD highlights how holes on a breadboard are connected, making it simpler to wire components correctly. It also allows you to change wire colours for better organisation.

Why Use TinkerCAD Circuits?

TinkerCAD Circuits proves useful in various educational and practical scenarios:

- **Active Learning**: It supports active learning formats in large classes, allowing students to follow along and engage more effectively.
- **Online Teaching**: It is an excellent resource for teaching electronics and Arduino classes entirely online, offering an engaging alternative when physical hardware is unavailable.
- **Remote Debugging**: It simplifies debugging circuits remotely, as students can rebuild their circuits in TinkerCAD and screen share for easier assistance.

- **Introductory Workshops**: It provides a way for participants in makerspace or library workshops to continue exploring and practicing what they have learned without needing to purchase their own Arduino at home.

Why Simulation Is Important

Simulation is a critical phase for validating a design. In Tinkercad, the simulation feature allows users to test their circuit designs virtually and observe how components behave according to the programmed code.

- Virtual Testing: The simulator lets you test your circuit design at any time, providing a virtual environment to see "what it would be like if we uploaded the code to our Arduino circuit".
- Real-world Behavior Observation: You can observe the circuit's behaviour, such as an LED blinking or changing brightness based on the code and component properties. For instance, starting the simulation on an Arduino Uno board will automatically plug in a power cord, providing electricity to the board.
- Interactive Testing: Users can interact with virtual components while the simulation is running, such as clicking virtual buttons to trigger actions or dragging a potentiometer's knob to adjust resistance and observe effects like LED brightness.
- Continuous Operation: The simulation runs continuously until the user explicitly presses "Stop Simulation".

Why an Engineer Should Learn Through Simulation

For hardware design engineers, understanding and avoiding frequent errors is crucial for project success. Simulation plays a vital role in this learning and development process:

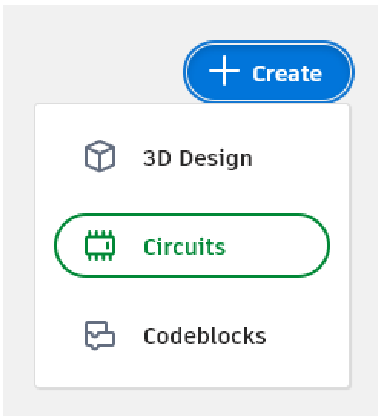
- Early Error Detection: While simulations can reveal theoretical flaws, they also help in identifying potential issues that might not be apparent in physical prototypes. For example, adjusting resistor values in Tinkercad's simulation can immediately show the impact on LED brightness, even flagging caution if resistance is too low.
- Cost and Risk Reduction: Simulation allows for extensive testing without the risk of damaging physical components or incurring the costs associated with prototyping failures. For example, Tinkercad warns against creating a short circuit, explaining that it can damage the Arduino and can be avoided by using a resistor.
- Iterative Design and Debugging: Engineers can quickly iterate on designs and code, making changes and instantly simulating them to see the effects. If a circuit isn't working, the debugger tool in the code editor can be used to check for code errors.
- Understanding Component Interactions: Simulation provides a safe space to experiment with various components (like resistors, LEDs, pushbuttons, and potentiometers) and understand their electrical properties and how they interact within a circuit without needing to physically wire them up first.
- Practice with Programming Logic: Tinkercad's code editor allows users to write programs using Blocks or C++ based Text code, applying concepts like void setup(), void loop(), pinMode, digitalWrite, and delay. This enables engineers to practice controlling digital I/O pins and implementing program flow, such as blinking LEDs or responding to button presses, without direct access to physical hardware.

- Exploring Edge Cases: Simulators allow engineers to design tests that account for all possible scenarios, including worst-case conditions, which is crucial as ignoring edge cases can lead to unreliable hardware.

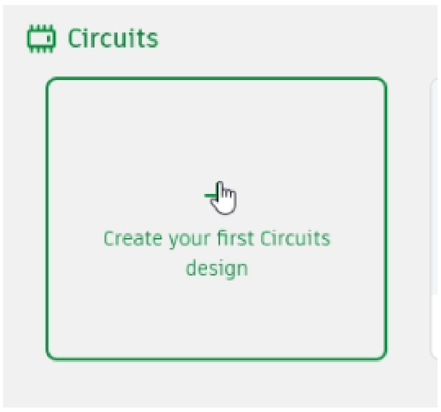
- Enhanced Reliability and Efficiency: By diligently using simulation, engineers can enhance the efficacy and dependability of their designs by fortifying requirements, simplifying designs, and rigorously testing.Tinkercad Circuits offers a highly valuable environment for learning electronics and coding, with its simulation capabilities being particularly important for both beginners and aspiring engineers.

To begin using TinkerCAD for circuits:

1. Log in to TinkerCAD (you may need to create an account at www.tinkercad.com).
2. Navigate to "Circuits" from the left menu.
3. Click "Create new Circuit"

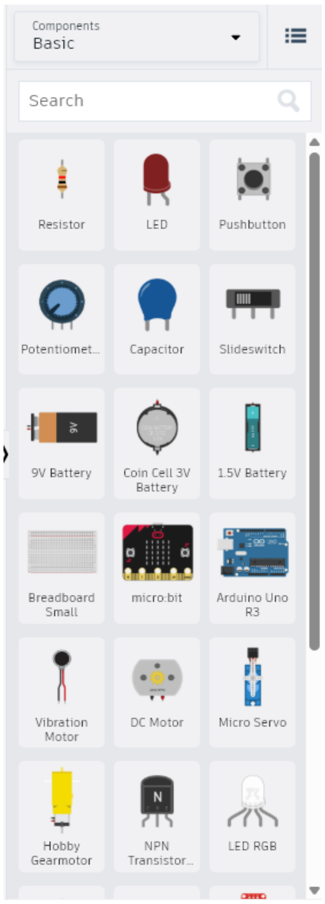


Or,



4. You will then be in the circuit editor, ready to place components and start designing.

Components Tab



The "components tab" in Tinkercad is referred to as the component selector, which is a scroll box located on the right side of the circuit editor screen.

Here are some details regarding the components tab in Tinkercad:

Location and Access:

◦ It is a scroll box found on the right side of the Tinkercad circuit editor.

◦ You can select a component by clicking it once in the selector, and then click in the open space on the workplane to place it.

◦ To move components around, you can click on a spot that is not a hole on the component and drag it. Clicking on a hole will start placing a wire. You can press the Esc key to cancel wire placement.

Searching for Components:

◦ The component selector includes a search bar where you can type the name of a component to find it. For example, you can type "Arduino" to find the Arduino Uno R3 board or "breadboard" to find different sizes of breadboards.

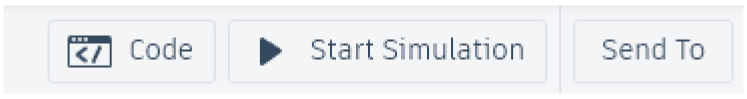
Categories of Components:
The component selector initially shows a "Basic" category. You can scroll down within this category to find various electrical components.
Available Components:

- Resistor.
- LED (Light Emitting Diode).
- Pushbutton.
- Potentiometer.
- Capacitor.
- Slide switch.
- 9V Battery.
- Coin Cell 3V Battery.
- 1.5V Battery.
- Breadboard Small and Breadboard (larger sizes).
- Arduino Uno R3.
- Vibration Motor.
- DC Motor.
- Micro Servo.
- You can also find "Starter Circuits" by selecting the "components" dropdown and then "Starters > All".

Component Properties:
Once a component is placed, you can click on it to see an inspector that allows you to edit its properties. For example, you can change the color of an LED or the resistance value of a resistor (e.g., to 220 Ohms, 10k Ohms, or 10 Ohms).
◦ Components can be rotated using the rotate tool in the upper left.
◦ They can be deleted using the delete button.
◦ The undo and redo buttons are also available for correcting actions.
This tab is crucial for building circuits in Tinkercad, allowing users to drag and drop various electrical components onto the workplane to design and simulate circuits.

Code Editor

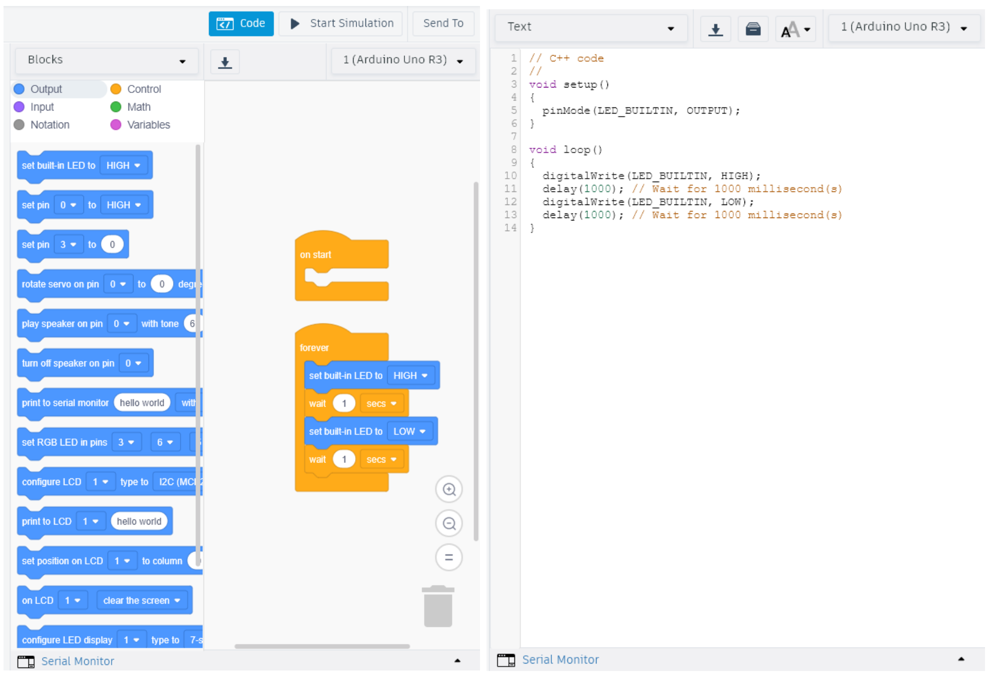
The code editor in Tinkercad allows users to write and manage the programs that control their circuits, particularly with microcontrollers like the Arduino Uno R3.



Access and Types
You can open the code editor by clicking the "Code" button located in the upper right of the circuit editor.
Tinkercad offers two primary ways to program:

- **Blocks:** This method uses prebuilt code blocks that link together like puzzle pieces, similar to Scratch or Snap. It's a good starting point for beginners but offers less flexibility for complex computations or controlling multiple components.

- **Text:** This option uses a C++ based programming language, allowing users to type their code. It provides a wider range and flexibility for programming intricate circuit operations. When switching to Text, a pop-up message will appear, prompting you to continue.



```
// C++ code
//
void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

Arduino Programming Syntax: Arduino's programming syntax is very similar to C++, including common elements like comparison (IF/THEN statements), mathematical computations, loops (FOR, DO, DO/WHILE), and ending line statements (;).

- **void setup():** Any code typed within this function executes once at the beginning of the program.
- **void loop():** This function contains code that executes an infinite number of times, continuously looping after the setup() function completes.

Pin Control

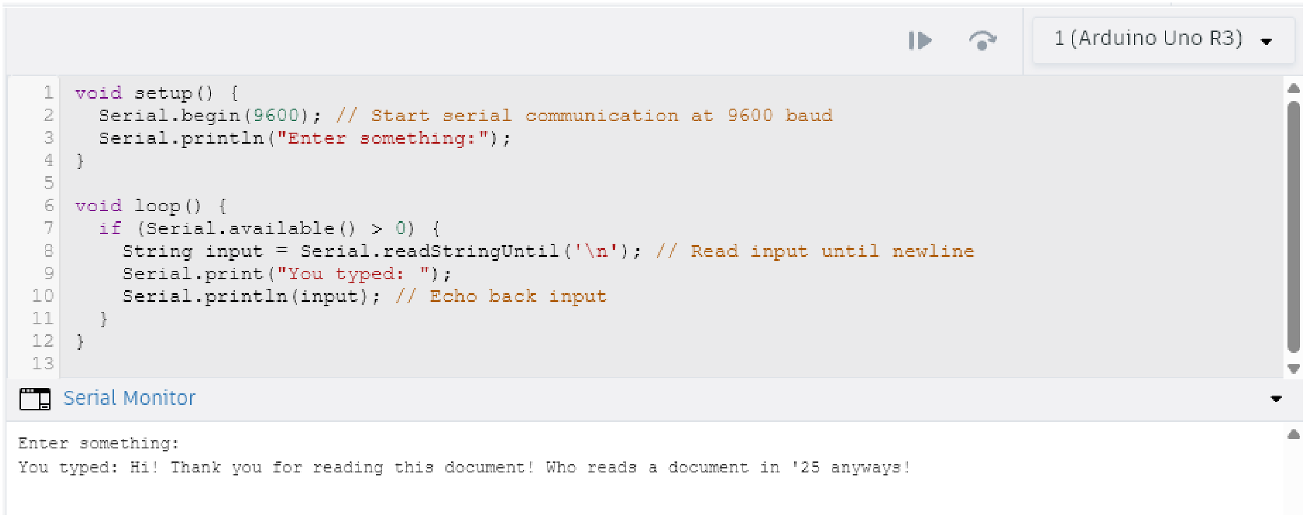
Digital I/O Pins: These pins (0-13 on the Arduino Uno R3) operate in an ON/OFF state, meaning the controlled component is either fully on or fully off.

- pinMode(pin, OUTPUT): This command declares a specific pin as an output, indicating it will be used to control something.
- digitalWrite(pin, HIGH): This sends a signal to a pin to turn on the flow of electricity, internally connecting the pin to 3.3V.
- digitalWrite(pin, LOW): This turns off the power to a pin, internally connecting it to GND.
- delay(milliseconds): This function pauses the program's execution for a specified number of milliseconds.
- Variables: Variables, such as integers (int), can be declared to represent pin numbers, making the code more adaptable. For example, int redled = 7; allows you to refer to pin 7 by the name redled

throughout the code. Variables can be declared in various scopes, such as before void setup() for global use, or within void setup() or void loop() for local use.

Serial Monitor

Located at the bottom left of the programming screen, the Serial Monitor is used for outputting information.



```
void setup() {
  Serial.begin(9600); // Start serial communication at 9600 baud
  Serial.println("Enter something:");
}

void loop() {
  if (Serial.available() > 0) {
    String input = Serial.readStringUntil('\n'); // Read input until newline
    Serial.print("You typed: ");
    Serial.println(input); // Echo back input
  }
}
```

```
Enter something:
You typed: Hi! Thank you for reading this document! Who reads a document in '25 anyways!
```

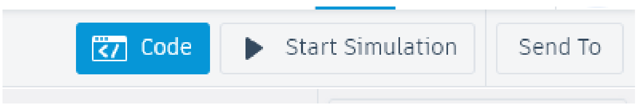
- **Serial.begin(baudRate):** This sets the baud rate for data transfer between the PC and the Arduino board; 9600 is a common value for English character data.

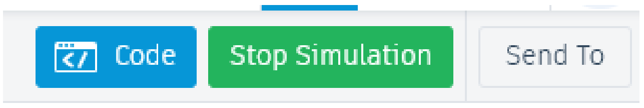
- **Serial.print("TEXT") or Serial.print(variable):** These commands print information to the Serial Monitor, keeping the cursor on the same line.

- **Serial.println("TEXT") or Serial.println(variable):** These commands print information and then move the cursor to the next line.

Simulation Tab

The simulation feature in Tinkercad allows users to test their circuit designs virtually, observing how components behave according to the programmed code.





Starting the Simulation: The simulation is initiated by clicking the "Start Simulation" button, typically found on the top right of the toolbar.

Purpose and Functionality:

- The simulator allows you to test your circuit design at any time.

- It provides a virtual environment to see "what it would be like if we uploaded the code to our Arduino circuit".
- Upon starting, a power cord automatically plugs into the Arduino board, providing electricity.
- You can observe the circuit's behavior, such as an LED blinking or changing brightness based on the code and component properties.

Interaction During Simulation: Users can interact with virtual components while the simulation is running. For example, you can click on virtual buttons (like in a piano circuit) to trigger actions, or click and drag the knob of a potentiometer to adjust its variable resistance and observe the effect on the circuit, such as LED brightness.

Testing and Debugging:

- The simulator helps identify potential issues that might not be apparent in physical prototypes, such as the need for appropriate resistance with LEDs; for instance, reducing resistance too much can make an LED brighter, but also trigger a caution indicator.
- If your circuit isn't working as expected, you can use the debugger tool in the code editor to check for errors in your code.
- The simulation runs continuously until the user explicitly presses "Stop Simulation".

Pre-made Designs: Tinkercad also offers "Starter Circuits" that can be found by selecting the "components" dropdown and then "Starters > All". These can be used to explore pre-built designs and their simulations.

Here’s a small tutorial on LED Control in TinkerCAD using blockcode-