# ElectricVLab

*Release 1.4.2*

**Q1Tek LLC**

**May 01, 2022**

# CONTENTS


**1 Introduction** **1**
- 1.1 Mouse Action Terminology . . . 2

**2 Tutorial** **3**
- 2.1 User Interface Overview . . . 3
- 2.2 Let's Build a Circuit . . . 4
- 2.3 Let's Simulate . . . 6
- 2.4 Let's Load a Sample Circuit . . . 8

**3 Feature and Usage Details** **11**
- 3.1 Components . . . 11
- 3.2 File Menu . . . 15
- 3.3 Options Menu . . . 17
- 3.4 Help Menu . . . 17
- 3.5 Build Tools . . . 18
- 3.6 Simulation Mode Interaction . . . 21
- 3.7 View Controls . . . 22
- 3.8 Load Circuit Dialog . . . 22
- 3.9 Save Circuit Dialog . . . 23
- 3.10 Parameter Editor Dialog . . . 23
- 3.11 Application Preferences Dialog . . . 23
- 3.12 Circuit Settings Dialog . . . 24
- 3.13 Circuit Information Dialog . . . 24
- 3.14 Oscilloscope Dialog . . . 25
- 3.15 Curve Editor Dialog . . . 26
- 3.16 Spectrum Analyzer Dialog . . . 27
- 3.17 Component Labels . . . 27
- 3.18 3D View and 2D View . . . 27
- 3.19 Ground Point . . . 28
- 3.20 Momentary Switch . . . 28

**4 Common Usage Topics** **29**
- 4.1 Zooming, Moving, and Rotating the View . . . 29
- 4.2 Rotating Components . . . 29
- 4.3 Bringing Up the Parameter Editor Dialog for a Component . . . 30
- 4.4 Function Generator . . . 30

**5 Interfacing ElectricVLab with Arduino Board** **31**
- 5.1 Copying the ElectricVLab_Arduino Library to Appropriate Location . . . 32
- 5.2 Virtual Inputs to Arduino Board from ElectricVLab . . . 34

5.3 Using Virtual Instruments of ElectricVLab to View Arduino Board Signals . . . . . 36

**Bibliography** **39**

**Index** **41**

CHAPTER

ONE

# INTRODUCTION

Welcome to *ElectricVLab : 3D Virtual Lab for Electricity and Electronics*! This software allows you to build, modify, and simulate electrical/electronic circuits, all in an interactive 3D environment. It provides a wide range of *circuit components* to build your virtual circuits, including power sources, switches, and a variety of analog and digital electronic devices.

In ElectricVLab, you build virtual circuits by placing components on a circuit board and connecting them together with wires. Building circuits in ElectricVLab is almost like building the circuits for real, but without the cost of buying the components or the labor of soldering. Components can be added or deleted, the circuit layout of wire connections can be changed, and switches can be turned on and off, all with just a few clicks of the mouse. Also, the parameters of most of the components can be modified with just a few keystrokes. And of course you can save your circuit to disk at any time and load it back in later to continue your work on it.

Once the circuit is built, you can "run" (i.e., *simulate*) the circuit and observe its behavior. The software employs 3D graphics, animations, and visual effects to make the experience enriching and fun. One can view the voltage at any component terminal or wire junction just by pointing the mouse at it. Animations played on the wires indicate the extent of current flowing through them. The precise amount of current flowing through a wire can be seen by just moving the mouse over the wire. There are also several virtual instruments you can use, such as voltmeters, ammeters, oscilloscopes, and spectrum analyzers which display real time circuit information.

ElectricVLab was designed with the intention of being useful for people of all all ages, from budding young enthusiasts to skilled students and professionals of electricity and electronics.

- For those interested in learning about electricity and electronics, ElectricVLab is a very good platform for experimentation and *learning by doing*. You can keep building a variety of circuits involving different components and observe their operation. ElectricVLab also comes with a collection of prebuilt circuits you can look at, modify, and save as your own.
- For those young at heart, ElectricVLab provides a number of fanciful components including those that produce a variety of visual effects such as fireworks and water fountains. The rate or intensity of the effects vary depending on the voltages, creating lively visualizations of circuit operations.
- For serious electronics hobbyists or professionals who build real electronic circuits, ElectricVLab is a great experimentation and design platform. One can first build the virtual circuit in ElectricVLab, choose the components, connections, and parameters while interactively viewing its operation. Once the design of the circuit is perfected using ElectricVLab, one can then choose to build the actual circuit using real components.
- For people using Arduino boards, ElectricVLab provides some special features. By interfacing the physical Arduino boards with ElectricVLab, one can build interesting systems wherein the virtual circuitry of ElectricVLab complements the functioning of the physical Arduino board.

This manual will acquaint you with the features of ElectricVLab and how to use them. The *Tutorial* chapter provides a quick walk through of some core aspects of the software by building a simple circuit step-by-step. The chapter titled *Feature and Usage Details* delves into the features and usage of the software in more detail. Helpful information for common usage is provided in the chapter titled *Common Usage Topics*. The chapter titled *Interfacing ElectricVLab*

*with Arduino Board* discusses how people working with Arduino boards can benefit by interfacing Arduino boards with ElectricVLab.

## 1.1 Mouse Action Terminology

Before proceeding, some clarification regarding the terminology pertaining to mouse actions is in order.

Whenever we say *left-click*, it refers to pressing the left mouse button and immediately releasing it. Similarly, *right-click* refers to pressing and immediately releasing the right mouse button. Just saying *click* refers to *left-click*.

Whenever we say *drag* an entity with the mouse, it refers to the following sequence of actions:

- Press the left mouse button when the mouse is pointing at the entity.
- While still holding down the left mouse button, move the mouse to the location you want and then release the mouse button.

In the above sequence, the first action of pressing the left mouse button amounts to "grabbing" the entity. At that time, the entity gets highlighted and then on it moves with the mouse. Releasing the mouse button "releases" the held entity.

*Mousing over* means moving the mouse over something without pressing any mouse button.

CHAPTER

TWO

# TUTORIAL

Go ahead and start ElectricVLab. The user interface window should look as shown in Fig. 2.1.

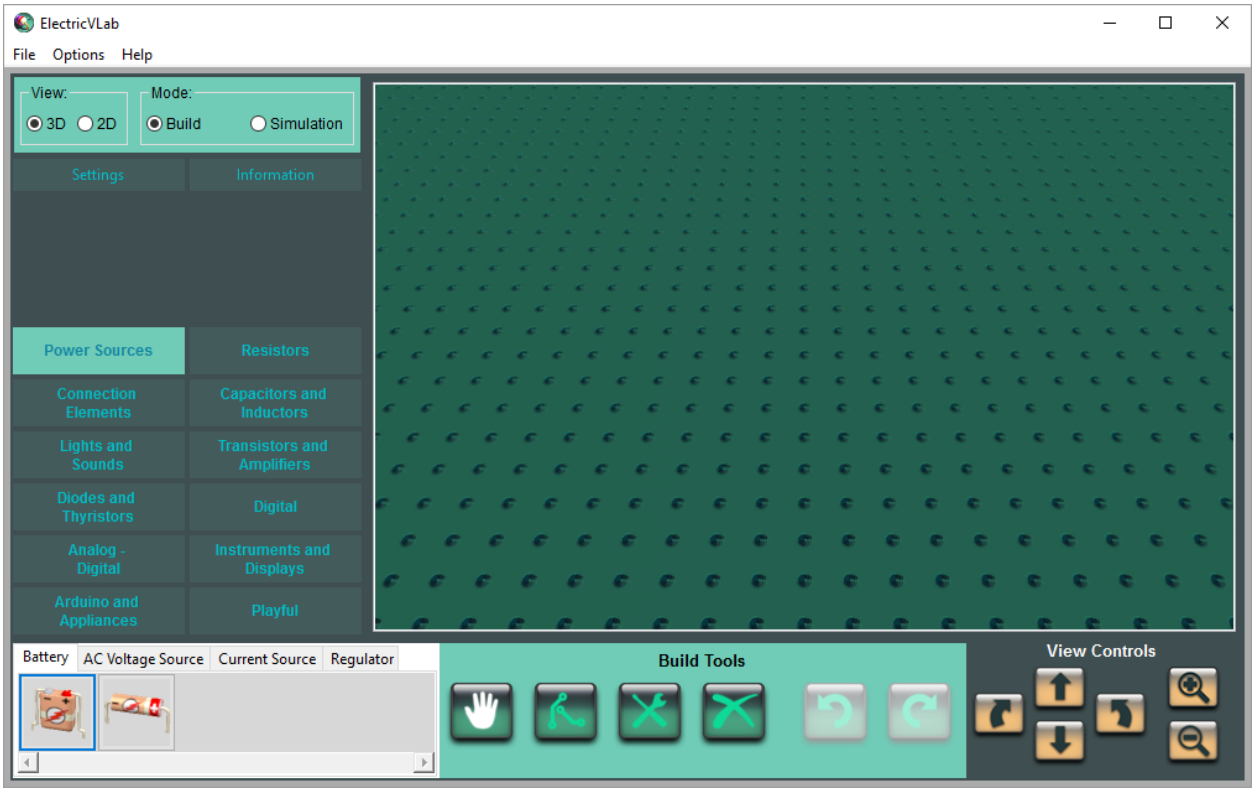



Fig. 2.1: User Interface

## 2.1 User Interface Overview

Near the upper-left corner of the window is the **Mode** panel containing two radio buttons which allow you to switch between two modes, *Build* and *Simulation*. You can switch between the two modes at any time. *Build* mode is for building and editing your circuit by adding or deleting components, configuring them, and connecting them with wires. *Simulation* mode is for seeing your circuit in operation. In *Simulation* mode, current can be seen flowing through the wires, and instruments such as voltmeters and oscilloscopes display the signal values. Also, in *Simulation* mode, visual effects like light bulbs lighting up, fireworks exploding, etc. add visual cues to the circuit's operation. Parameters of the components can be modified both in *Build* mode and in *Simulation* mode.

Under the **Mode** panel are two buttons labeled **Settings** and **Information**. Pressing the **Settings** button brings up the *Circuit Settings Dialog*. Pressing the **Information** button brings up the *Circuit Information Dialog*.

Below the **Settings** and **Information** buttons is the **Component Category** panel which shows the available component categories. As you can see, the categories are **Power Sources**, **Resistors**, **Connection Elements**, **Capacitors and Inductors** etc. Below the **Component Category** panel is the **Component Catalog** which displays the components belonging to the category that is currently selected in the **Component Category** panel. The **Component Catalog** contains multiple tabs corresponding to the different subcategories of the currently selected category. In Fig. 2.1, the tabs displayed correspond to the **Battery**, **AC Voltage Source**, and **Current Source** subcategories of the currently chosen category, namely, **Power Sources**. The icons in the **Component Catalog** represent the components belonging to the currently selected subcategory.

To the right of the **Component Category** panel is the 3D view of the **Circuit Board** on which you build your circuits.

Directly below the **Circuit Board** is a panel which contains mode-specific buttons for building and interacting with your circuit. When in *Build* mode, this panel contains the **Build Tools** and when in *Simulation* mode, it contains the **Simulation Controls**.

In the bottom-right corner of the window is the **View Controls** panel. It contains buttons for rotating, tilting, and zooming the 3D view of the **Circuit Board**. There is no button to move (scroll) the view. Instead, to move the view, right click anywhere on the **Circuit Board** and drag the mouse in the direction you wish the view to move. For more details about the 3D view adjustments, see the section on *Zooming, Moving, and Rotating the View*.

Near the top-left corner of the window is a **View** selection panel with two radio buttons labelled **3D** and **2D**. In the 3D view, components get displayed as 3D models while in the 2D view they get replaced by their corresponding circuit symbols. The 2D view essentially displays the circuit diagram (schematic) using electronic circuit symbols. The 3D view shows how the circuit can look when built with actual components. You can switch between these two views at any time.

## 2.2 Let's Build a Circuit

We are now ready to start building a circuit. Let's walk through the steps of building the simple circuit shown in Fig. 2.2.

First, make sure ElectricVLab is in *Build* mode by selecting the **Build** radio button in the **Mode** panel. Now, let's place a battery on the **Circuit Board**. For that, click on the **Power Sources** button in the **Component Category** panel and then click on the **Battery** tab in the **Component Catalog**. Icons for different battery types are displayed. To see their names, *mouse over* the icons. Click on the **Battery (Box shaped)** icon. The battery component should now appear on the **Circuit Board**. As you move the cursor over the **Circuit Board**, the battery component will move with it. When it is positioned where you want it, click the left mouse button and the battery gets placed on the **Circuit Board**. If you wish to move it, you can do so by *grabbing* it using left-click, moving the mouse to a new location and clicking again to release the battery. Now, let's add a resistor. Click on the **Resistors** button in the **Component Category** panel. Under the **Ordinary** tab of the **Component Catalog**, pick the component named **Resistor**. Move the resistor so it is positioned as shown in Fig. 2.2 and click to place it on the board. Next, let us add a fireworks component. Click on the **Playful** button in the **Component Category** panel. Unlike the other mainstream electronic *components* available in ElectricVLab, some of the components under the **Playful** category are semi-imaginary. They do however help to illustrate some aspects of circuit functionality in a playful way. In the **Component Catalog**, under the **Firework** tab, you would see a number of firework components. Those components produce different kinds of firework effects when they are electrically powered. Click on the icon corresponding to **Fireworks Chrysanthemum**. Place the component on the **Circuit Board** as shown in Fig. 2.2.

Now that the battery, resistor, and fireworks component have been placed, let's connect them with wires to form the circuit. To place wires, we need to use the **Wire Tool**. But, before going into the usage of the **Wire Tool**, a brief introduction to the **Build Tools** is in order.

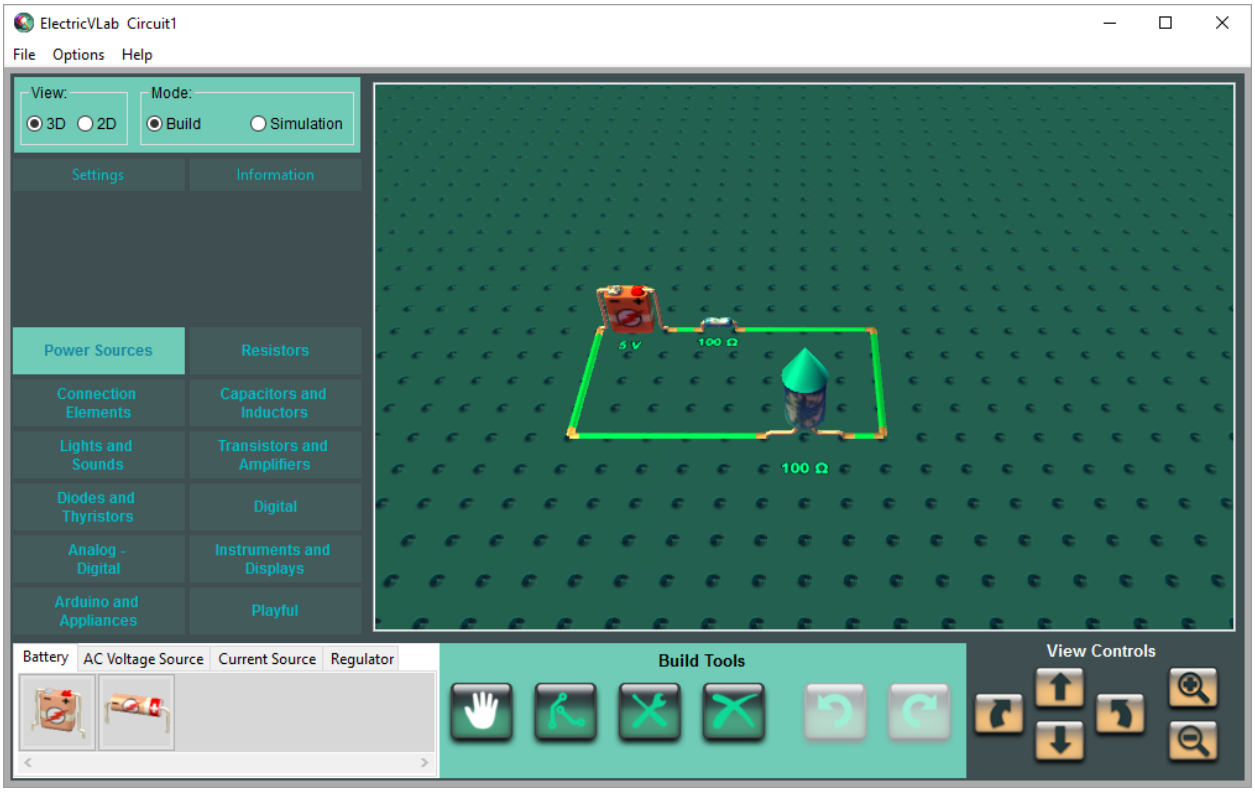



Fig. 2.2: Simple Fireworks Circuit

The **Build Tools** are for building and modifying circuits. You can see the names of the tools by mousing over the buttons on the **Build Tools** panel. They are, from left to right, the **Object Tool**, the **Wire Tool**, the **Parameter Tool**, and the **Delete Tool**. The *Object Tool* is for performing operations such as placing, moving, rotating, duplicating, and deleting components. The *Wire Tool* is for adding and deleting wires. The *Parameter Tool* is used to invoke the *Parameter Editor Dialog* for setting the values of the modifiable parameters of components. The *Delete Tool* is used to delete components as well as wires. A tool is activated simply by clicking on the corresponding button and its active state is indicated by the button getting highlighted. Activating a tool automatically deactivates the previously active tool because in *Build* mode, only one tool can be active at any time. When you are done using a tool, you can deactivate it by pressing the *Esc* key, which in turn automatically activates the **Object Tool** since that is the default build tool.

The last two buttons on the **Build Tools** panel, while not exactly tools per se, are the **Undo** and **Redo** buttons. After performing an operation, you can undo it by pressing the **Undo** button. You aren't limited to just one undo. You can undo as many operations as you want. You can also undo the undo operations by pressing the **Redo** button.

Now, let us get back to the task of connecting the three components we have placed on the **Circuit Board** with wires. Activate the **Wire tool** by clicking on its button in the **Build Tools** panel. Wires can be created between any two holes on the **Circuit Board**. To create a wire, press the left mouse button over the hole where you want the wire to begin. Holding down the left mouse button, move the mouse to the hole at which you want to end the wire. When the mouse is at the desired end location, release the left mouse button. The result is a straight piece of wire with exposed copper ends connecting the start and end holes. Now, go ahead and connect the three components by repeatedly creating wires to form the circuit shown in Fig. 2.2. While placing some of the wires, for better visibility, you may find it helpful to slightly rotate the view using the clockwise/anticlockwise rotation buttons on the **View Controls panel**. And don't worry if you make a mistake at any stage while building the circuit. You can always go back to a point prior to the mistake by repeatedly clicking the **Undo** button. Also, if you need to delete any wire or component during the process of building the circuit, you can use the *Delete Tool*.

Congratulations! You finished creating your first circuit with ElectricVLab. Let's save it now. Click on the **File** menu and pick **Save As**. The *Save Circuit Dialog* that comes up lets you select the folder and file to save the circuit into.

## 2.3 Let's Simulate

Now that you've finished building your first circuit, it is time to simulate it. Click on the **Simulation** mode radio button in the **Mode** panel. When you do so, the **Simulation Controls** panel will replace the **Build Tools** panel as seen in Fig. 2.3.

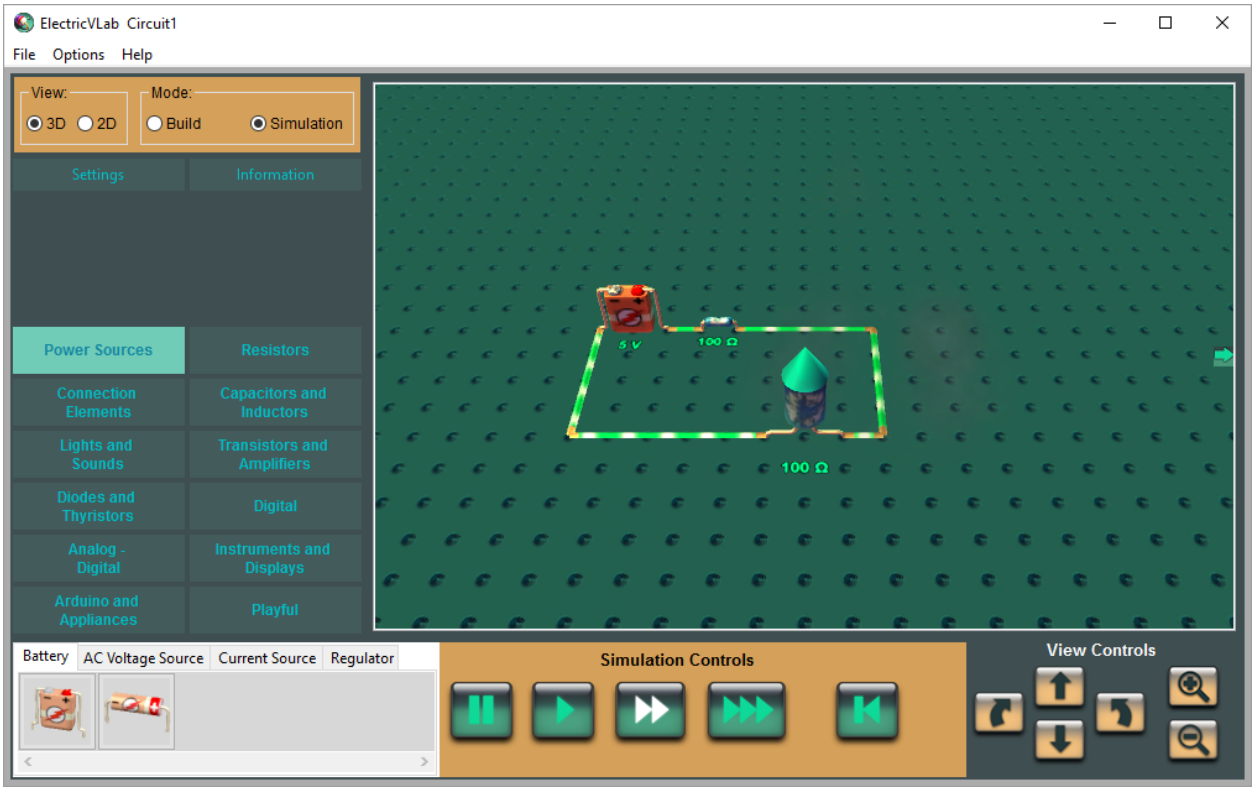



Fig. 2.3: Simple Fireworks Circuit in Simulation Mode

The **Simulation Controls** panel contains the following buttons:

- **Pause** button to pause the simulation.
- Three simulation speed buttons.
- **Reset** button to restart the simulation from the beginning.

You will see that rockets shoot out of the fireworks component once in a while. You will also see the animations depicting the current flow through the wires. The speed of the current flow animation indicates the amount of current flowing. You can see the actual value of the current by mousing over a wire. The amount of current in this circuit is governed by *Ohm's Law* which states that the amount of current is equal to the net voltage divided by the net resistance. The resistance of the fireworks component is 100 ohms. As you can check by clicking on them, the battery voltage is currently at its default value of 5 volts and the resistor is currently at its default value of 100 ohms. So, per Ohm's law, the expected current would be:

5/(Resistance of resistor + Resistance of fireworks component) = 5/(100 + 100) = 0.025A = 25mA

By mousing over a wire, you can see that the current value indeed gets displayed to be 25mA (milli Amperes) as expected.

Based on Ohm's law, we would expect that decreasing the resistance value should increase the amount of current and vice versa. Let's try that. *Right-click* the resistor component on the **Circuit Board** to bring up the context menu for the resistor. From the context menu, select the **Parameters** option. This will bring up the *Parameter Editor Dialog* in which you can alter the resistance value of the resistor. As shown in the **Parameter Editor Dialog**, its resistance is currently at its default value of 100 ohms. Click on the field displaying the resistance value and change the number to 10. Close the dialog by clicking the **OK** button. You will notice that the animations indicating the current flow through the wires are now moving at a faster rate indicating a greater amount of current. By mousing over a wire, you can see that the amount of current is 45.45mA now instead of the earlier value of 25mA. As the value of the current increases, you will notice that rockets shoot out of the fireworks component at a greater rate. Click on the battery component in the circuit. This brings up the *Parameter Editor Dialog* for it. Change the voltage to 20 volts instead of the default value of 5 volts and close the dialog. You can now see that the amount of current flowing in the circuit has increased even more. With that, the rockets get fired at a still faster rate.

To see the fireworks exploding, you may need to zoom out. One way to zoom out is to click on the zoom out button in the **View Controls** panel. It is the button with the magnifying glass icon containing the minus sign. Another way to change the zoom level is by scrolling the mouse wheel. The section on *Zooming, Moving, and Rotating the View* discusses different ways of changing the 3D view. When you zoom out far enough to see the explosions, the screen will look something like what is shown in Fig. 2.4. In case you want to turn off the explosion sounds, you can achieve that by disabling the audio in *Application Preferences Dialog*. That dialog also lets you set other preferences like changing the color of the **Circuit Board**.

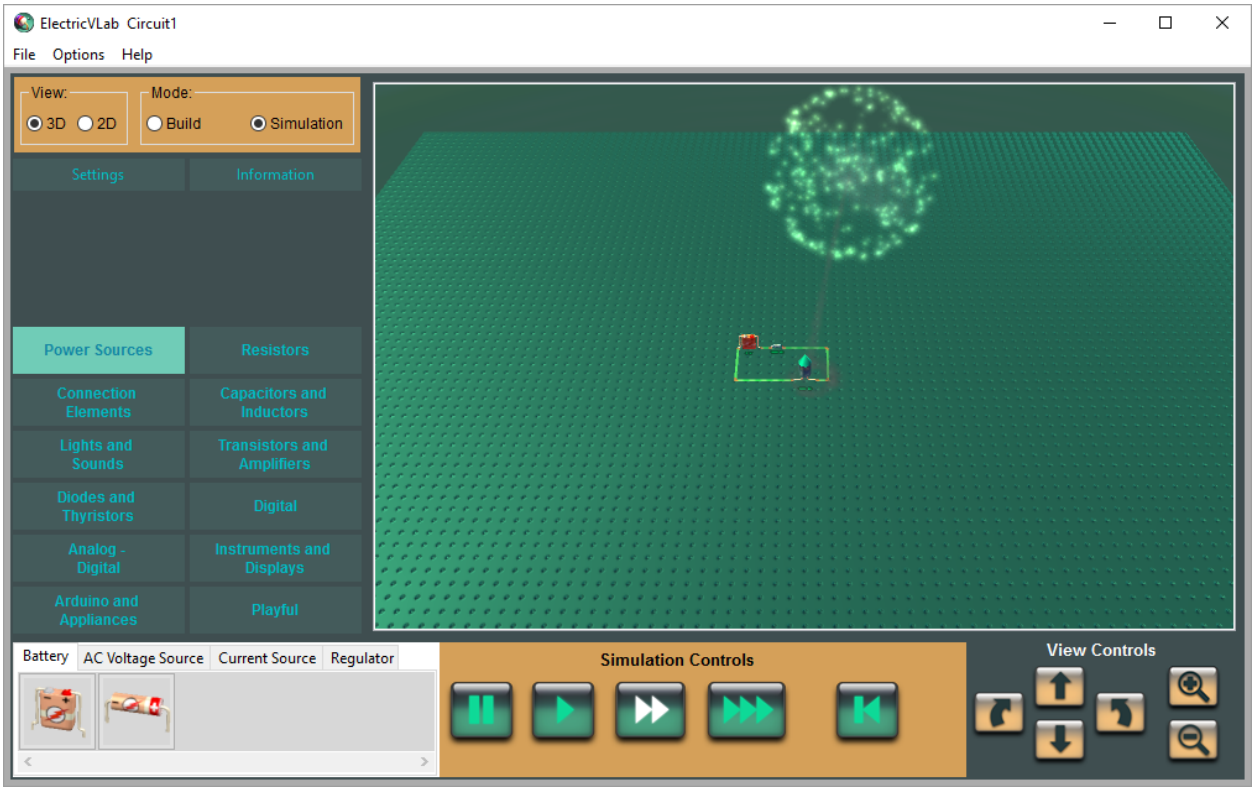



Fig. 2.4: Simple Fireworks Circuit in Simulation Mode (Zoomed out)

Now, zoom back in to get a closer view of the circuit so that we can do another update to the circuit. As mentioned earlier, the amount of current flow can be seen by mousing over the wires. But, to avoid having to mouse over every time we want to know the amount of current, let us add to the circuit an *Ammeter*, which is an instrument that displays the current value automatically. Return to *Build* mode by clicking on the **Build** radio button. Next, click on the

**Instruments and Displays** button in the **Component Category** panel. Select the **Ammeter** component under the **Instrument** tab in the **Component Catalog** and place it in the circuit as shown in Fig. 2.5.

Save the updated circuit by choosing **Save** under the **File** menu.

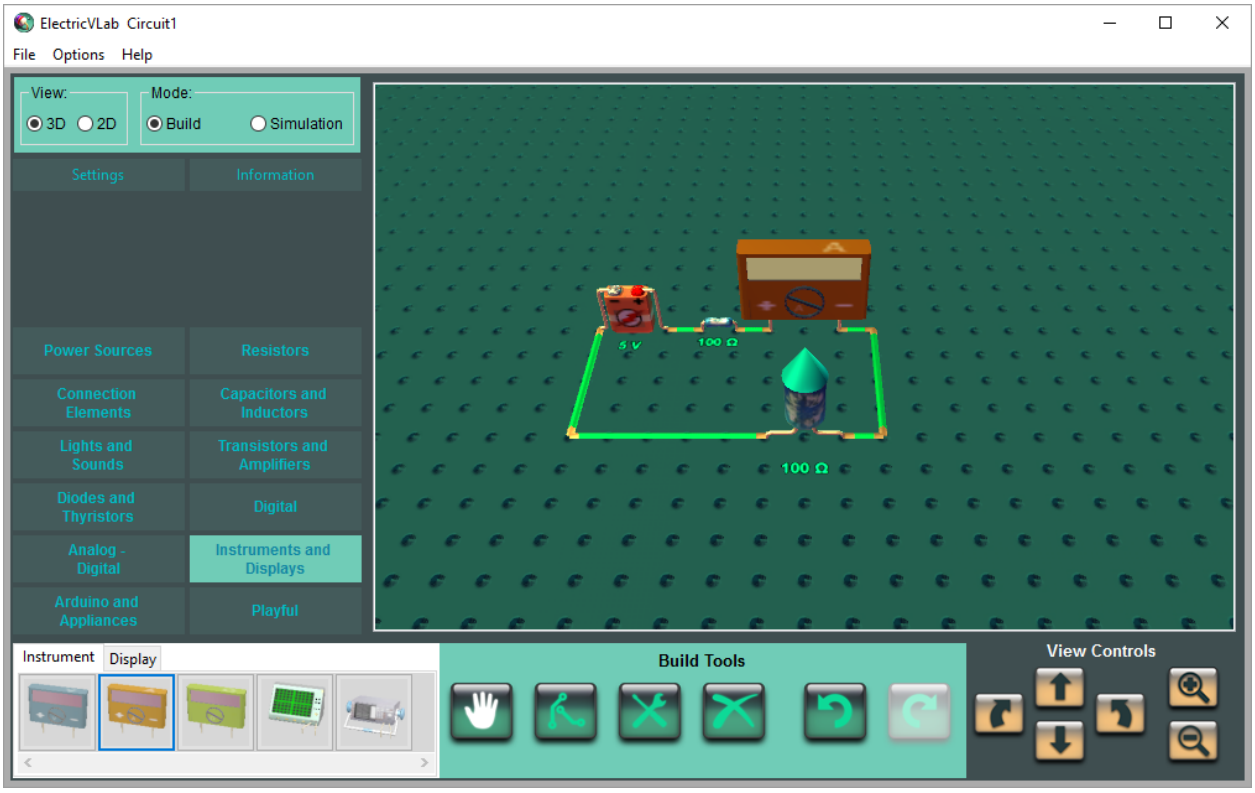



Fig. 2.5: Fireworks Circuit with Ammeter

Return to *Simulation* mode by clicking the **Simulation** radio button. Notice that the ammeter displays the amount of current flow. Now, change the voltage value of the battery by right-clicking on it and choosing the **Parameters** option from the context menu to bring up the *Parameter Editor Dialog*. After changing the voltage value, close the **Parameter Editor Dialog**. The ammeter would display the new value of the current.

Save the circuit again by choosing **Save** under the **File** menu.

## 2.4 Let's Load a Sample Circuit

So far, we walked through an example of building your own circuit. ElectricVLab also comes with several prebuilt circuits which you can play with. They are referred to as *Sample Circuits*. They get installed as part of ElectricVLab's installation. Let's load one of those sample circuits now.

From the **File** menu, choose **Load Sample Circuit**. That brings up the *Load Circuit Dialog*. In that dialog, go to the **L0_Tutorial** folder and load the **AC Voltage Divider** circuit. The screen will now look as shown in Fig. 2.6.

In the **AC Voltage Divider** circuit, a **Function Generator**, a **Color Coded Resistor**, and a generic **Resistor** are connected in series. You may notice that in this circuit the generic resistor is oriented in the north-south direction while in the circuit you built earlier (Fig. 2.2), it was in its default east-west orientation. As you build your circuits, you would soon feel the need to rotate the components so as to put them in an orientation that is different from their default orientation. Section titled *Rotating Components* discusses multiple ways for rotating components. In the sample circuit you loaded (Fig. 2.6), an **Oscilloscope** is connected across the **Color Coded Resistor** and displays the

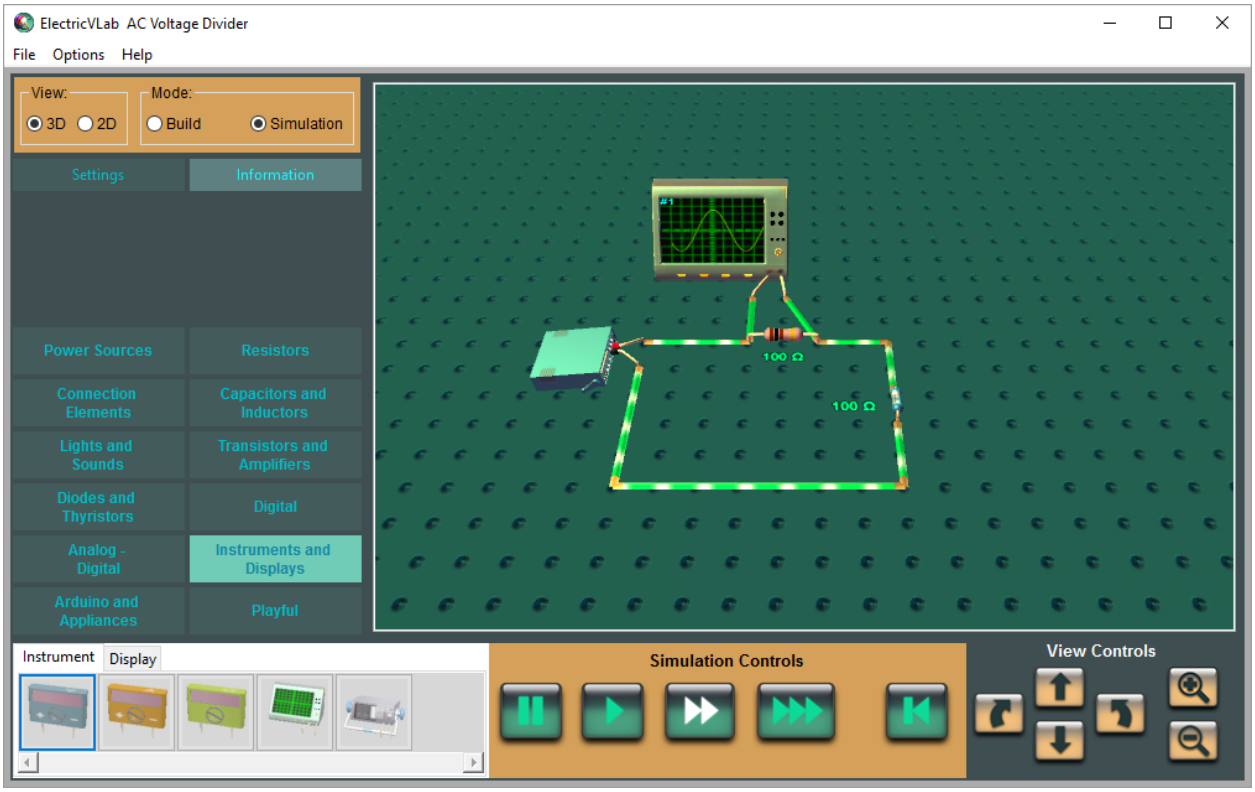



Fig. 2.6: Simple AC Circuit : AC Voltage Divider

voltage pattern across that resistor. The *Function Generator* in the circuit produces a time varying voltage pattern. Currently, it is set to produce a sinusoidal waveform at 40Hz. The **Function Generator** component in ElectricVLab is a versatile component that can be made to produce a wide variety of waveforms at arbitrary frequencies. Let us change the waveform to be triangular instead of sinusoidal. Right-click the **Function Generator** component on the **Circuit Board** and choose the **Parameters** option from the context menu to bring up the *Parameter Editor Dialog*. As you can see, it allows **Max Voltage** (Amplitude), **Frequency**, **Waveform** etc. to be varied. It can be made to generate any custom waveform you may want as well. Details of all that are provided in the section on *Function Generator*. For now, let us just change the waveform to be triangular. Click on the **Waveform** value field, i.e., the field currently showing **Sinusoid**. From the drop-down list, select **Triangle**. Close the **Parameter Editor Dialog** by clicking the **OK** button. Notice that the oscilloscope is now showing a triangular waveform.

Now click on the oscilloscope on the **Circuit Board**. That will bring up the **Oscilloscope Dialog** where the waveform is displayed at a larger scale. The dialog also provides a number of other controls for operating the oscilloscope, the details of which are discussed later in the section about *Oscilloscope Dialog*. Close the **Oscilloscope Dialog** by clicking the **X** button in the upper right corner of the dialog.

That completes our walk-through of some basic aspects of using ElectricVLab. The rest of this manual goes into more detail and covers the many other features in the software. From this point, there are a couple of different ways to continue the journey.

- You may want to continue reading this manual and get to know about the many features available in ElectricVLab and the details of their usage. We would recommend at least quickly glancing through the section titles of the rest of this manual to get a rough idea of what are all the features available and their usage. Another advantage of continuing to read the manual before building circuits is the following. The software provides multiple ways of doing some common operations like rotating/duplicating/deleting objects, deleting wires, setting the parameters etc. Under different contexts, different ways will be more convenient. So, by knowing about the different possibilities, you will be able to build the circuits much faster.

- Alternatively, you may want to jump right in and start building your own circuits and consult the relevant portions of this manual as needed.
- Another worthwhile thing to do would be to load the sample circuits provided one by one, observe their workings, and generally play with them. The sample circuits are organized under folders named **L1_OhmsLawAndResistors**, **L2_BasicLogicGates**, and so on. As mentioned earlier, use the **Load Sample Circuit** option under the **File** menu to load them. Many of the sample circuits have information notes associated with them conveying brief information about the circuit. If a circuit has any associated information notes, the **Information** button (situated directly under the **Mode** panel) gets gently highlighted. Pressing the **Information** button brings up the *Circuit Information Dialog* which displays the information notes about the circuit, if any. By the way, if you want to attach any information notes about any circuit you build, enter the notes in the *Circuit Information Dialog*. When you save the circuit, those information notes gets saved with it.

CHAPTER

THREE

# FEATURE AND USAGE DETAILS

The previous chapter provided a brief tutorial on ElectricVLab by walking you through the building of a simple example circuit. This chapter goes into more detail about using ElectricVLab and its available features.

## 3.1 Components

As of this writing, the following components are available in ElectricVLab.

- Power Sources
  - Battery
    - Battery (Box shaped)
    - Battery (Cylindrical)
  - AC Voltage Source
    - *Function Generator*
    - Sweep Generator
  - Current Source
  - Regulator
    - 78XX Positive Voltage Regulator
    - 79XX Negative Voltage Regulator
- Resistors
  - Ordinary
    - Resistor
    - Color Coded Resistor
    - Potentiometer
    - Rheostat
  - Sensor
    - Photoresistor
    - Thermistor
- Connection Elements
  - Ground

    - Switch
        - Single Throw Switch
        - 2-State Double Throw Switch (Middle connector otherside)
        - 2-State Double Throw Switch (Middle connector sameside)
        - *Momentary Switch*
        - 3-State Double Throw Switch (Middle connector otherside)
        - 3-State Double Throw Switch (Middle connector sameside)
    - Distributor
    - Fuse
    - Relay
        - Single Pole Single Throw (SPST) Relay
        - Single Pole Double Throw (SPDT) Relay
        - Double Pole Single Throw (DPST) Relay
        - Double Pole Double Throw (DPDT) Relay
- Capacitors and Inductors
    - Capacitor
        - Capacitor (2 Variants)
        - Electrolytic Capacitor
    - Inductor
        - Cylindrical Inductor
        - Toroidal Inductor
    - Transformer
        - Transformer
        - Center-Tapped Transformer
- Lights and Sounds
    - Incandescent Lamp
        - Bud Bulb
        - Bulb
    - Fluorescent Lamp
        - Spiral Light
        - Tube Fluorescent Lamp
    - Sound Maker
        - Bell
        - Horn
- Transistors and Amplifiers
    - BJT (Bipolar Junction Transistor)

        - NPN Transistor
        - PNP Transistor
    - JFET (Junction Field Effect Transistor)
        - N-Channel JFET
        - P-Channel JFET
    - MOSFET (Metal-Oxide-Semiconductor Field Effect Transistor)
        - N-MOSFET
        - P-MOSFET
    - OpAmp (Operational Amplifier)
- Diodes and Thyristors
    - Diode
    - Zener Diode
    - Light Emitting Diode (LED)
        - Green LED
        - Red LED
    - Thyristor
        - SCR (Silicon Controlled Rectifier)
- Digital
    - Gate
        - AND Gate, OR Gate, Inverter (NOT Gate), NAND Gate, NOR Gate, XOR Gate, 3-Input NAND Gate, 4-Input NAND Gate
    - Flipflop
        - D Flipflop
        - JK Flipflop
    - Counter
        - 4-Bit Counter
    - Input/Output
        - Digital Input
        - Digital Output
        - Clock
    - 7400 Series
        - 7400 Quad NAND Gate
        - 7402 Quad NOR Gate
        - 7404 Hex Inverter
        - 7408 Quad AND Gate
        - 7410 Triple 3-Input NAND Gate

        - 7420 Dual 4-Input NAND Gate
        - 7432 Quad OR Gate
        - 7447 BCD to 7-Segment Decoder
        - 7483 4-Bit Full Adder
        - 7485 4-Bit Comparator
        - 7486 Quad XOR Gate
        - 7490 Decade Counter
        - 7495 Shift Register, Parallel In, Parallel Out, Serial Input
        - 74139 Demultiplexer
        - 74147 10-Line to 4-Line Priority Encoder
        - 74148 8-Line to 3-Line Priority Encoder
        - 74153 Dual 4-Line to 1-Line Multiplexer
        - 74157 Quad 2-Line to 1-Line Multiplexer
        - 74193 Synchronous 4-Bit Up/Down Binary Counter
    - Other
        - Priority Encoder
        - Decoder
        - Multiplexer
        - Demultiplexer
        - Seven Segment Decoder
- Analog-Digital (Mixed)
    - Analog to Digital
        - 4 Bit ADC (Analog to Digital Converter)
    - Digital to Analog
        - 4 Bit DAC (Digital to Analog Converter)
    - Timer
        - 555 Timer
- Instruments and Displays
    - Instrument
        - Voltmeter
        - Ammeter
        - Ohmmeter
        - *Oscilloscope*
        - *Spectrum Analyzer*
    - **Display**
        - Seven Segment Display (Common Cathode Type)

        * Seven Segment Display (Common Anode Type)
- Arduino and Appliances
  - *Arduino* (Interfaces ElectricVLab with physical Arduino board)
  - Appliance
    * Cooker, Fan, Iron Box
- Playful
  - Firework
    * Firework Chrysanthemum, Firework Crosette, Firework Peony, Firework Peony Strobe
    * Firework Random, Firework Rings, Firework Serpentine
    * Firework Spider, Firework Strobes, Firework Willow
    * Rocket
  - Misc
    * Robot, Flame, Water Jet, Fountain, Spark Ring,
    * Balloons, Disco Light, Sparkle Sphere, Graphic Equalizer
    * Jacob's Ladder, Laser Generator, Plasma Ball, Wind Tunnel

Many of these components have parameters whose values you can modify. To modify the values of a component's parameters, use the *Parameter Editor Dialog*, which can be invoked using any of the methods discussed in the section on *Bringing Up the Parameter Editor Dialog for a Component*.

## 3.2 File Menu

The following menu options are available under the **File** menu.

### 3.2.1 New Circuit

Select this menu to create a new circuit. The newly created circuit is a blank circuit board. When ElectricVLab starts, a new circuit gets created automatically. So, there is no need to invoke the **New Circuit** menu just after starting ElectricVLab.

The newly created circuit does not have a name. The circuit can be given a name while saving it. When a circuit has a name, the name gets displayed on the title bar of the main window.

If you invoke the **New Circuit** menu when the currently loaded circuit has some unsaved edits, a prompt asking whether you want to save those edits comes up.

### 3.2.2 Load My Circuit

Select this menu when you want to load a circuit that is not one of the sample circuits that come with ElectricVLab installation. In other words, select this menu when you want to load a circuit that you saved yourself. Selecting this menu item brings up the *Load Circuit Dialog* which allows you to select the circuit you wish to load.

### 3.2.3 Load Sample Circuit

Some prebuilt circuits get bundled with ElectricVLab as samples. They get installed as part of ElectricVLab installation. Select this menu when you want to load any of those sample prebuilt circuits. Selecting this menu item brings up the *Load Circuit Dialog* which allows you to select the circuit you wish to load.

### 3.2.4 Save

Picking this menu item saves the currently displayed circuit. If this circuit has been saved previously, ElectricVLab simply overwrites the file. Otherwise, it brings up the *Save Circuit Dialog* which allows you to select the name of the file and the folder into which the circuit should be saved.

Another thing to note is that the prebuilt sample circuit files that come with ElectricVLab are not allowed to be overwritten. This is to make sure they remain in their unmodified form for reference sake. With a prebuilt circuit loaded, both the **Save** and **Save As** menu items bring up the *Save Circuit Dialog* allowing you to save the circuit into a different file. Effectively, this results in making a copy of the installed circuit which you can then edit as you wish.

### 3.2.5 Save As

Selecting this menu item brings up the *Save Circuit Dialog* which lets you specify the file and folder to save the circuit into.

The name of the file into which the circuit gets saved is taken to be the name of the circuit. The circuit name gets displayed on the title bar of the main window.

### 3.2.6 Take Screenshot

This menu item is for taking screenshots of your circuit. Picking this menu item opens a dialog that lets you enter a name for the screenshot, select where on disk you want to save the image, and pick the image file format you wish to save it in. The available image format choices are *bmp*, *jpg*, and *png* and are chosen by clicking on the **Save As Type** drop-down list. Click on the **Save** button to save the screenshot image to disk and close the dialog.

### 3.2.7 Create Bill of Materials

Select this menu when you want to create a file that lists the different kinds of components present in your circuit and how many of each kind are there. When you click on the menu, a standard file save dialog comes up that lets you select the folder and file name. The file created is a plain text file which you can open in any text editor such as Notepad.

### 3.2.8 Export SPICE Netlist

This feature is intended for those who want to use *SPICE* electronic circuit simulation software. SPICE takes circuit descriptions in *Netlist* format. The ability to export SPICE netlist from ElectricVLab allows you to interactively build a circuit using ElectricVLab and then export the circuit into *Netlist* format for the purpose of feeding it as input to SPICE for further experimentation.

Selecting the **Export Spice Netlist** menu item brings up a dialog which lets you specify the name and location of the *Netlist* file to be created. *Netlist* files created always have the ".cir" extension.

One caveat is that not all components of ElectricVLab have SPICE equivalents and also that in some cases the exported netlist may not meet all the needs of an expert SPICE user. However, the exported file can be a good starting point even in those cases.

### 3.2.9 Exit

Selecting this menu item exits ElectricVLab.

## 3.3 Options Menu

### 3.3.1 Audio

This menu option can be used to enable/disable audio.

### 3.3.2 Display Labels

This menu option can be used to specify whether or not component labels should be displayed.

### 3.3.3 Preferences

Selecting this menu item brings up the *Application Preferences Dialog*.

## 3.4 Help Menu

### 3.4.1 User Manual

Selecting this menu item displays this user manual.

### 3.4.2 About

Selecting this menu item displays general information pertaining to ElectricVLab.

## 3.5 Build Tools

When the program is in *Build* mode, the **Build Tools** panel gets displayed under the **Circuit Board**. It contains buttons to activate the tools for building and modifying circuits. You can see the names of the tools by mousing over the buttons in the panel. Specifically, the tools available are the **Object Tool**, the **Wire Tool**, the **Parameter Tool**, and the **Delete Tool**. The rightmost two buttons on the **Build Tools** panel are not really tools per se, but instead provide *Undo and Redo* functionality which can be very useful to correct any accidental mistakes that happen while building circuits.

Any of the build tools can be activated by clicking on its button in the **Build Tools** panel. At any given time in *Build* mode, only one tool can be active. Activating a tool automatically deactivates the previously active tool. Which tool is currently active is indicated by highlighting the corresponding button. Also, when the cursor is over the **Circuit Board**, a smaller version of the active button's icon gets drawn as part of the cursor, thereby providing another indication of which tool is currently active. When you are done using a tool, you can deactivate it by pressing the *Esc* key. Whenever any build tool is deactivated, the **Object Tool** automatically gets activated since it is the default tool for *Build* mode.

The following sections discuss the different **Build Tools** in detail.

### 3.5.1 Object Tool

The **Object Tool** is for performing operations related to objects (circuit components) such as placing them, rotating them, duplicating them, deleting them, etc. Any of the following actions result in the activation of the **Object Tool**.

- Clicking on the **Object Tool** button located in the **Build Tools** panel.
- Clicking on any component icon on the **Component Catalog**.
- Pressing *Esc* key to deactivate any other *Build* mode tool.

When you want to place a new component on the **Circuit Board**, click the corresponding icon on the **Component Catalog**. That results in the creation of an instance of that component. The created component gets "attached" to the cursor. That is, as you move the cursor over the **Circuit Board** (without holding down mouse buttons), the component moves with it. When the component is at the desired location, you can do either of the following.

- Click the left mouse button. This places the component on the **Circuit Board** at the location of the cursor.
- Holding down the left mouse button, move the mouse. This rotates the component in increments of 90 degrees. When the component is oriented the way you want, release the mouse button to place it.

In both of the above two cases, once the mouse button is released, the object gets "detached" from the cursor and gets placed on the **Circuit Board**.

While a component is attached to the cursor and is moving with the mouse, using the keyboard, you can do the following:

- Rotate the component clockwise by pressing the *">"* key, and counterclockwise by pressing the *"<"* key.
- Delete the component by pressing the *Delete* key. This way, you don't have to switch to the *Delete Tool* to delete components. You can delete them with the **Object Tool** itself.
- Abort the current operation being performed by hitting the *Esc* key. It also detaches the component from the cursor.

To move a component that was previously placed on the **Circuit Board**, left-click on that component while the **Object Tool** is active, but no component is currently attached to it. That click action attaches the clicked component to the cursor. Once attached, you can move, rotate, and delete the attached component the same way we discussed earlier for the case of new component created by clicking on a component icon in the **Component Catalog**.

Right clicking on a component that is already on the **Circuit Board** brings up a context menu for the component. Some common options available are **Center in View**, **Duplicate**, **Rotate Clockwise**, **Rotate Counterclockwise**, and

**Delete**. For components that have editable parameters, the context menu will also contain the **Set Parameters** option. Selecting that option brings up the *Parameter Editor Dialog*. For some components (e.g., **Voltmeter**, **Ammeter**, **OpAmp**, transistors), the context menu also includes the **Mirror** option. Selecting the **Mirror** option replaces the component with a mirrored version of it. For example, selecting the **Mirror** option for an **Ammeter** replaces it with an **Ammeter** that has the plus and minus terminals swapped.

An important feature of the **Object Tool** is that you can also move, rotate, duplicate, and delete whole sections of the circuit instead of just individual components. To do so, hold down the left mouse button and drag a rectangular region, as shown in Fig. 3.1. When you complete the drag operation, all of the encompassed components and wires get selected and a context menu pops up displaying options such as **Duplicate**, **Delete**, **Center in View**, **Move**, and **Set Wire Color**.

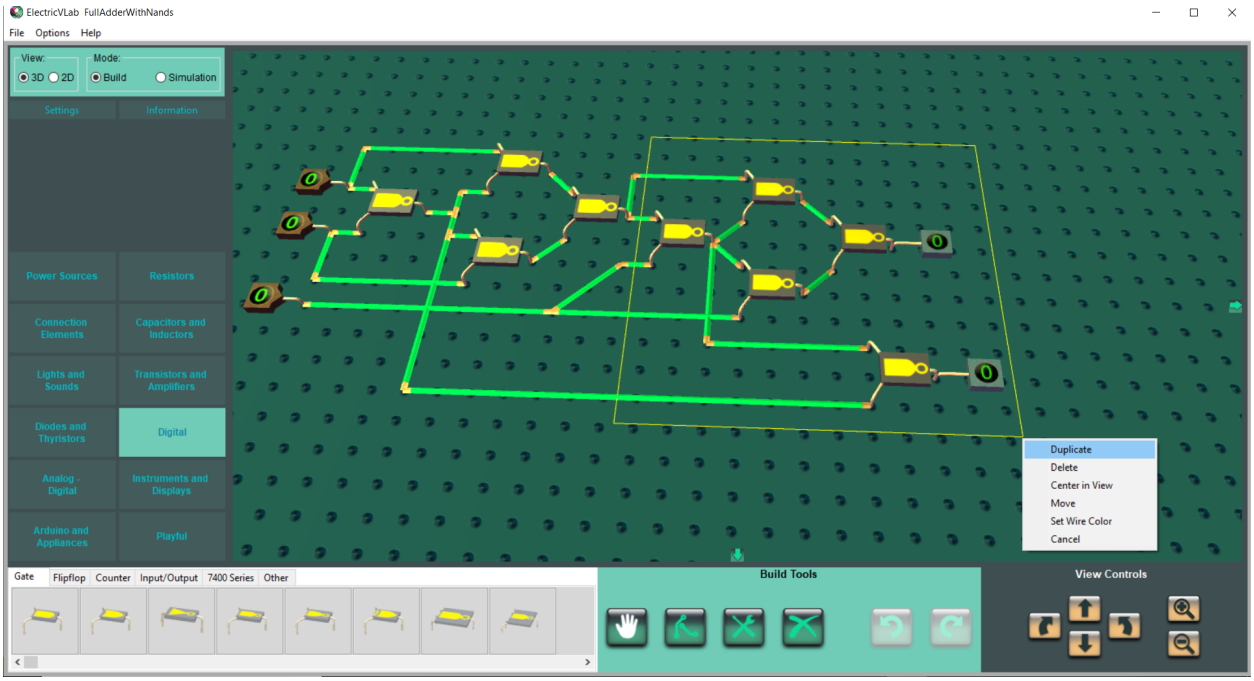



Fig. 3.1: Multiple Component Operations

If you select **Center in View**, the view gets adjusted so that the dragged rectangular region is at the center of the **Circuit Board** window. If you select **Duplicate**, a copy of the selected components and wires gets made. Once the duplicates are created, they get "attached" to the cursor in the same way individual objects do. So, you can move the duplicates by moving the mouse. When the duplicates are at the location you want, you can place them by clicking the left mouse button. While the duplicates are attached to the cursor, you can rotate them using the "<" and ">" keys. The **Move** option is similar to **Duplicate** except that no copy is made; the components and wires of the circuit inside the rectangle themselves get moved. Picking the **Delete** option will delete the selected components and wires. Picking the **Set Wire Color** option lets you change the color of all the wires inside the rectangle.

### 3.5.2 Wire Tool

The **Wire Tool** allows you to connect components by placing wires between any two holes on the **Circuit Board**. To activate the **Wire Tool**, click on its button in the **Build Tools** panel. To place a wire, press down and hold the left mouse button at the hole on the **Circuit Board** where you want the wire to start. Drag the mouse to the hole at which you want to end the wire, and release the mouse button. This results in the creation of a wire between the two holes with exposed copper at both of its ends. While the **Wire Tool** is active, you can place any number of wires, one after another. When you are done placing wires, you can deactivate the **Wire Tool** the same way you would deactivate any other build tool, that is, either by pressing the *Esc* key or by clicking the button for some other *Build* mode tool.

You don't have to switch to the *Delete Tool* to delete wires. You can delete them using the **Wire Tool** itself instead. Mousing over the wire you want to delete while holding down the *Ctrl* key will highlight the wire. When the wire is highlighted, left click to delete it.

Also, while using the **Wire Tool**, right-clicking on a wire brings up the context menu for it via which you can delete the wire or change its color.

Instead of setting the colors of the wires individually, you can make use of the **Color of New Wires** preference setting in the *Application Preferences Dialog*. That way, you can choose the color for a bunch of wires you are going to place next.

You may find that connecting the components with wires is easier in *2D View* since the connection points are more clearly visible in that view. To switch between *3D View* and *2D View*, click the corresponding radio button located near the top left corner of the main window.

One thing to be noted regarding wires is that ElectricVLab considers an electrical connection to be made only at the exposed copper terminals and not at the unexposed interior parts of the wire. This is illustrated in Fig. 3.2 where the two wires forming the X shape on the left were each placed with a single drag operation. Hence they have exposed copper terminals at their ends and not at the point where they cross. Hence, there is no electrical contact between those two wires; one wire just goes over the other. On the right is another X shape consisting this time of four wires, all terminating at the crossing point of the X shape creating an electrical connection. The fact that wires are allowed to go over one another as in the case of the X shape on the left gives you greater freedom when laying out circuits.

### 3.5.3 Parameter Tool

The **Parameter Tool** is for setting the parameters of circuit components. This tool can be activated by clicking the **Parameter Tool** button in the **Build Tools** panel. Many circuit components (but not all) have parameters whose values you can modify. While the **Parameter Tool** is active, whenever you mouse over a component that has modifiable parameters, the component gets highlighted. Clicking on the component when it is highlighted brings up the *Parameter Editor Dialog* which allows you to modify the values of the parameters.

### 3.5.4 Delete Tool

The **Delete Tool** can be activated by pressing its button in the **Build Tools** panel. While the **Delete Tool** is active, you can keep deleting any number of wires or components one after another. Mouse over the component or wire on the **Circuit Board** you want to delete, and when it is highlighted, click the left mouse button to delete it.

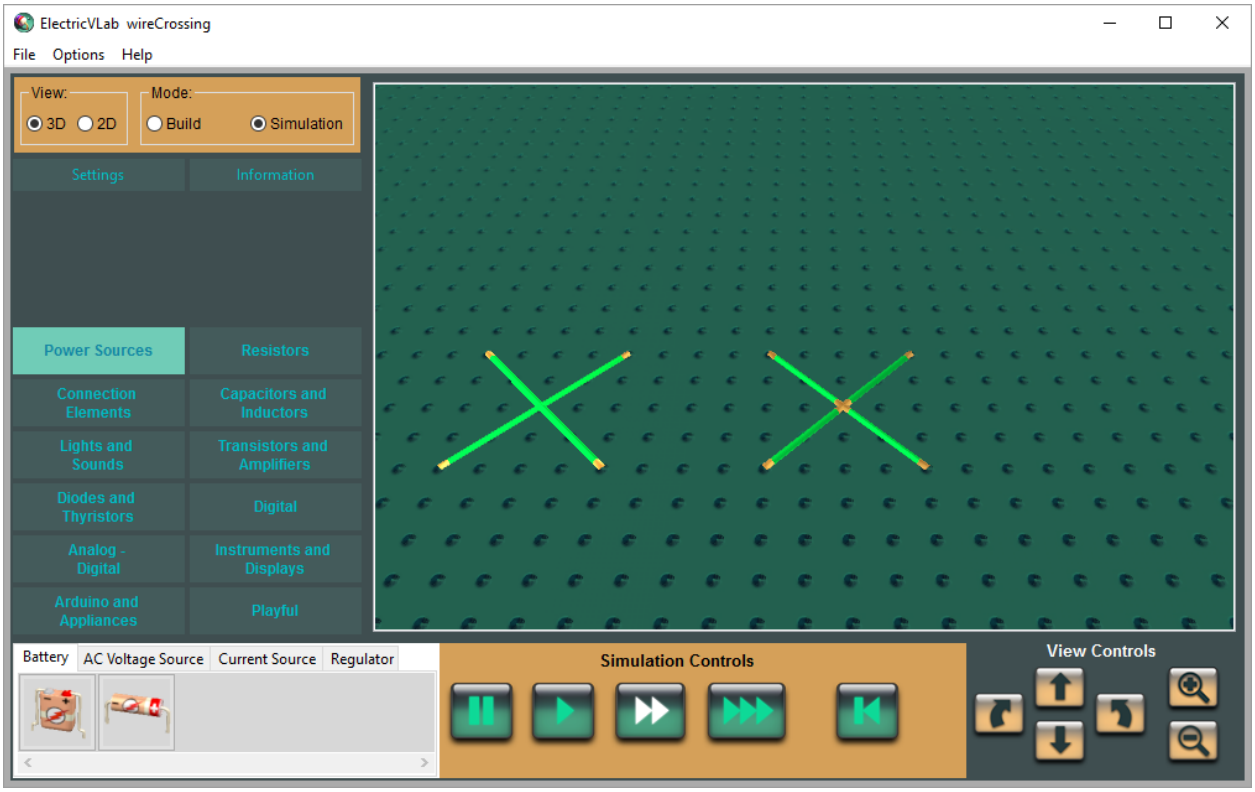



Fig. 3.2: Wires Crossing without Electrical Contact on the Left. Wires Connecting on the Right.

### 3.5.5 Undo and Redo

The rightmost two buttons on the **Build Tools** panel, **Undo** and **Redo**, allow you to correct the mistakes you may have made. After performing an operation, you can undo it by pressing the **Undo** button. And you aren't limited to just one undo. You can undo as many operations as you want. You can also undo the undo operations by pressing the **Redo** button.

*The Undo/redo* lists get cleared when you save the circuit, switch modes, or exit ElectricVLab.

When there is no operation to *undo* or *redo*, the corresponding button gets disabled. When you initially load a circuit, both the **Undo** and **Redo** buttons will be disabled since no circuit editing activity has yet been performed on the circuit since its loading.

## 3.6 Simulation Mode Interaction

When the program is in *Simulation* mode, the circuit gets simulated, allowing you to observe its behavior. Animations and visual effects make the experience enriching and fun. Current can be seen flowing through the wires and light bulbs glow. You are also able to interact with the circuit in the following ways:

- Mousing over any terminal of a component or at a junction of two or more wires displays, next to the cursor, the voltage at that point and at that moment.
- Mousing over any wire displays, next to the cursor, the amount of current currently flowing through the wire.
- You can change the state of some components, such as a switch or a **Digital Input** component, by clicking on them. In a switch with just two states, namely *open* and *closed*, clicking on the switch toggles it from the *open*

state to the *closed* state and vice versa. In a switch with more than two states (e.g., **3-State Double Throw Switch**), each click takes it to the next state. By repeatedly clicking on such a switch, you can take it through all of its states.

- Right-clicking on a component brings up the context menu for it. If the component has modifiable parameters, one of the options in the context menu will be **Parameters**. Selecting that option brings up the *Parameter Editor Dialog*, allowing you to modify the values of the parameters. Another common menu option would be **Center in View**. Selecting that option adjusts the view such that the component is at the center of the circuit board window.
- Holding down the left mouse button, you can move the scroller handle of the rheostat to change the resistance value.

The display screen of an oscilloscope component on the **Circuit Board** displays the waveform of the voltage across its terminals. However, since the oscilloscope's screen size is rather small, it can be a bit hard to see in detail. To get a bigger view of the waveform, click on the oscilloscope. This brings up the *Oscilloscope Dialog* which shows an enlarged view of the waveform together with cursors, value labels, etc. The dialog also provides many more controls to adjust the behavior of the oscilloscope. For details, see the section on *Oscilloscope Dialog*.

Clicking on a spectrum analyzer component brings up the *Spectrum Analyzer Dialog* which displays the spectrum analysis results.

### 3.6.1 Simulation Controls

In *Simulation* mode, the **Build Tools** panel gets replaced by the **Simulation Controls** panel. The **Simulation Controls** panel contains the following buttons from left to right:

- The **Pause** button. Clicking the **Pause** button pauses the simulation. Clicking the button again unpauses the simulation.
- The **Low Speed** button for simulating the circuit in slow speed which is one-tenth the default **Medium Speed**.
- The **Medium Speed** button which is the default speed.
- The **High Speed** button, which simulates the circuit at a speed ten times the speed of the default **Medium Speed**.
- The **Reset** button. Clicking the **Reset** button restarts the simulation from the beginning.

## 3.7 View Controls

The buttons on the **View Controls** panel are useful for adjusting the 3D view of the **Circuit Board**. The view buttons can be used in both *Build* and *Simulation* modes. For details about adjusting the view, see the section on *Zooming, Moving, and Rotating the View*.

## 3.8 Load Circuit Dialog

The **Load Circuit Dialog** is essentially a standard file selector dialog that lets you navigate to the folder containing the desired circuit file and then load that file. The circuit files have the extension *".lab"*.

The **Load Circuit Dialog** gets invoked when you select the **Load My Circuit** or **Load Sample Circuit** menu item under the **File** menu. The behavioral difference between the two menu items is only in the initial location the **Load Circuit Dialog** points at when it is displayed. For your convenience, the program makes a heuristic guess as to roughly where in the file system the circuit file you are intending to load is likely to be located and makes the dialog point there when it shows up. Once the dialog shows up, you can navigate to any folder you want and load any circuit file you want.

When a circuit is loaded, the view (zoom, rotation, and tilt) gets set to be the same as it was when the circuit was last saved.

## 3.9 Save Circuit Dialog

The **Save Circuit Dialog** comes up when the **Save As** menu item under the **File** menu is selected or when the **Save** menu item is selected and the circuit is being saved for the first time. This is a standard file save dialog. It lets you select the folder in which you want to save the circuit and specify a name for the file.

When the circuit is successfully saved, a message box gets displayed indicating the full path of the file into which the circuit is saved.

## 3.10 Parameter Editor Dialog

The **Parameter Editor Dialog** lets you view and change the parameter values for components on the **Circuit Board**. It can be invoked for a component that has modifiable parameters in a couple of different ways as explained in the section titled *Bringing Up the Parameter Editor Dialog for a Component*. The dialog contains two columns. The left column displays the names of the parameters and the right column displays their values. Mousing over a parameter row displays a tooltip showing a brief description of the parameter.

To change the value of a parameter, click on its value field. For some parameters like resistance, capacitance, inductance, voltage and current, the value is specified by typing in the numerical part and selecting the unit from a drop-down list. For some types of parameters, specialized editors are available. An example would be a curve parameter. Let us consider the case of a **Firework** component under the **Playful** category. It has a **Firing Rate** parameter whose value is a curve specifying how the firing rate varies with the amount of current flowing through the component. To edit a curve parameter, first click on the parameter's value field (i.e., the field displaying the text "Curve"). When you do so, a small button with ellipsis ("...") appears. Clicking on this button brings up the *Curve Editor Dialog* in which you can edit the curve.

When you are done altering/viewing the parameter values in the **Parameter Editor Dialog**, you can close the dialog by clicking the **OK** button. To abort any edits you did to the parameter values and continue to use the previous values, close the dialog by hitting the **Cancel** button.

## 3.11 Application Preferences Dialog

This dialog can be invoked by selecting the **Preferences** menu item under the **Options** menu. In this dialog, you can specify the following preferences.

- Change the color of the **Circuit Board**.
- Color for new wires: Once you set this value, the new wires you place will have this color. Suppose you want to place a bunch of red wires, and then place a bunch of green wires. You can do that by first setting this preference value to **Red** and placing the first set of wires. Then, you would change this preference value to **Green** and place the second set of wires. Of course, once a wire is placed, you can still change its color using the *Paramerer Editor Dialog*.
- Color of the symbols representing the components while in 2D view.
- Color used to display component labels.
- Current Flow Depiction: In **Simulation** mode, wires animate indicating the direction of current flow. The "Conventional Current Flow" is from the positive terminal of a battery to the negative terminal. But, "Electron

Flow" is in the opposite direction. By choosing the value for **Current Flow Depiction**, you can choose which of these two you prefer.

- Enable/Disable the display of ground indicator. Details pertaining to this option can be found in the section on *Ground Point*.

The preferences set in the **Application Preferences Dialog** pertain to the behavior of the application and are not restricted to the currently displayed circuit. Some settings that are specific to the currently displayed circuit can be modified using the *Circuit Settings Dialog*.

## 3.12 Circuit Settings Dialog

The **Circuit Settings Dialog** can be invoked by pressing the **Settings** button on the user interface window. The options in this dialog apply to only the currently displayed circuit. As of now, the following parameters can be set via this dialog.

- **Current for Fastest Flow Animation on Wires**. In *Simulation* mode, animations indicating the current flow get played on the wires. The speed of the animation increases with the amount of current. This parameter specifies the magnitude of current at which the animation attains its maximum speed.
- **Force real-time simulation**. Normally, ElectricVLab simulates and displays the circuit operation in "slow motion". That is, when 1 second of real time elapses, the circuit behavior simulated corresponds to only a small fraction of a second (Normally, something like 0.04 second). There are two reasons for taking this approach. Firstly, it is usully much more helpful to view the operation of the circuit in slow motion to understand how exactly the circuit is operating. Secondly, with the computing power available on many users' machines, it is sometimes difficult to do accurate simulations at increased speeds. However, there may be cases when you don't want this "slow motion" behavior and instead want the simulation to proceed in real-time. To do that, mark the **Force real-time simulation** checkbox in this dialog. One caveat is that it may possibly affect the accuracy of simulations in some cases.

## 3.13 Circuit Information Dialog

The **Circuit Information Dialog** can be invoked by pressing the **Information** button on the user interface window. This dialog is meant for entering and displaying information about the circuit. For a circuit you build, you may, for example, want to enter information about what the circuit does, how it works etc. When you save the circuit into a file, the information entered automatically gets saved along with it into the same file. There is no separate command to save the information entered in this dialog. To save the information, simply save the circuit itself using the **Save** or **Save As** menu item under the **File** menu.

When you load a circuit, the program checks whether there is any information associated with it. If there is, the program gently highlights the **Information** button to hint that there is some information available for the circuit and you may want to see it by pressing the **Information** button.

Suppose you pressed the **Information** button and brought up the **Circuit Information Dialog** for the currently displayed circuit. If you now want to load some other circuit, you don't need to close the **Circuit Information Dialog**. You can choose to leave it open. If the dialog is left open, when you load another circuit, the dialog gets automatically refreshed to display the information, if any, about the newly loaded circuit.

Suppose you have left the **Circuit Information Dialog** open and proceeded to perform some other actions. During the process, some other windows might happen to get displayed on top of the **Circuit Information Dialog** and obscure it. To get back to seeing that dialog, you don't need to search for it. Simply pressing the **Information** button will bring it to the front.

## 3.14 Oscilloscope Dialog

The **Oscilloscope Dialog**, as seen in Fig. 3.3, displays a graph of voltage (Y axis) as a function of time (X axis). In *Simulation* mode, if you click on an **Oscilloscope** placed on the **Circuit Board**, this dialog gets displayed.

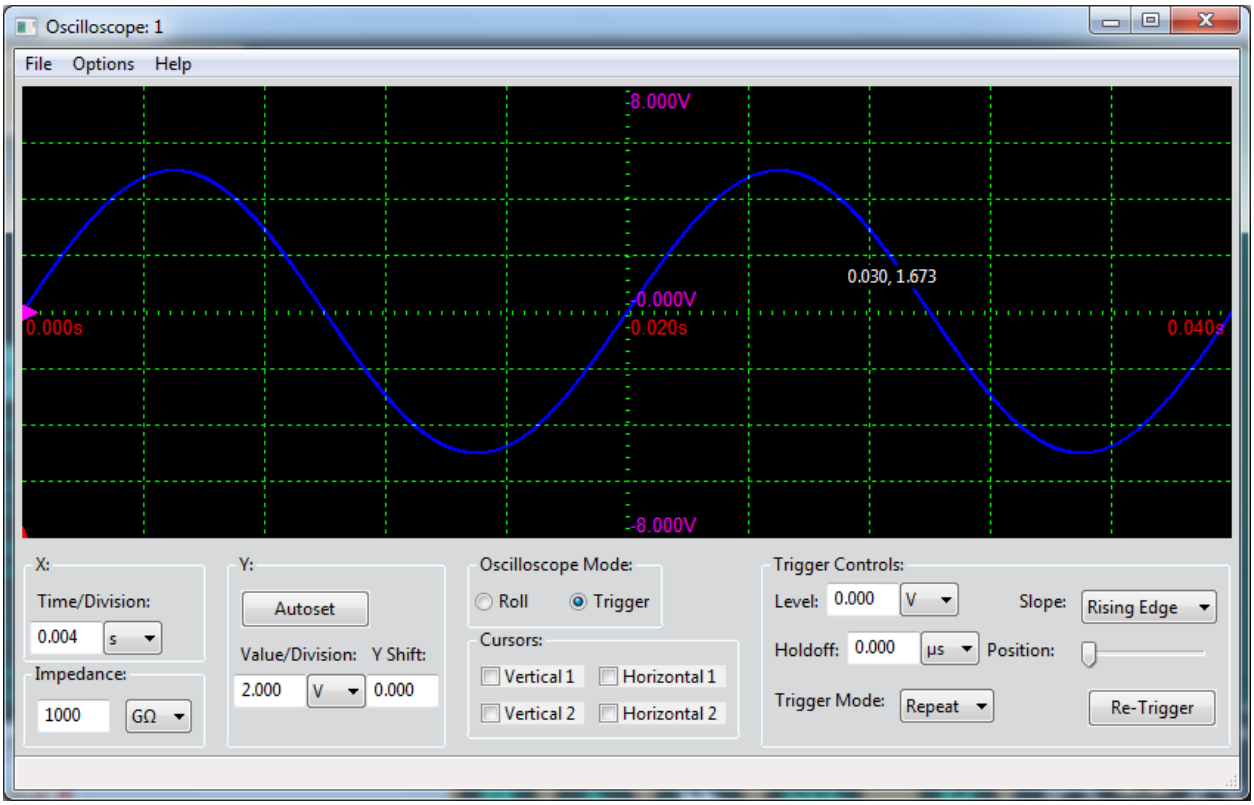



Fig. 3.3: Oscilloscope Dialog

You can specify the scale for the X axis by typing in a number for **Time/Division**. Choose the unit of time from the adjacent drop-down list, the choices being **second**, **millisecond**, and **microsecond**. The scale for the Y axis can be specified by typing in a number for **Value/Division**. Choose the unit for the Y axis from the adjacent drop-down list, the choices being **volt**, **millivolt**, and **microvolt**. The waveform can be shifted up or down by specifying a positive or negative value in the **Y Shift** field. The **Y Shift** uses the same unit as the **Value/Division**, the one indicated by the drop-down list located between them. You can hit the **Autoset** button which results in ElectricVLab automatically choosing the appropriate values for **Value/Division** and **Y Shift** in order to ensure that the waveform appears within the screen. Later, if the waveform voltage values happen to deviate significantly from what they were the last time you hit the **Autoset** button, you may need to hit the **Autoset** button again to make it choose fresh values for **Value/Division** and **Y Shift**.

The dialog allows you to add up to two horizontal cursor lines and up to two vertical cursor lines. To enable a cursor, click the corresponding checkbox under **Cursors**. You can move a cursor line to the position you desire by *dragging* it using the left mouse button.

The oscilloscope has two modes of operation named *Roll Mode* and *Trigger Mode*. You can switch between these two modes by clicking the radio buttons in the **Oscilloscope Mode** box. By default, the oscilloscope gets set to be in *Roll Mode*. In *Roll Mode*, the dialog displays a moving waveform. *Trigger Mode* is useful for getting a stable view of a repetitive waveform and make closer observations of it.

The controls in the **Trigger Controls** box on the right of the **Oscilloscope Dialog** are used when the dialog is in *Trigger Mode*. The following description assumes that you have a basic understanding of *Trigger Mode* in typical

oscilloscopes. The **Value** specifies the voltage value at which trigger occurs and the **Slope** specifies whether the trigger occurs on the rising edge of the waveform or the falling edge. Once a trigger occurs, at the minimum, an amount of time that corresponds to the width of the screen must elapse before the next trigger. You can specify an additional hold off time amount beyond that minimum value by entering the desired value in the **Holdoff** field. The unit for the hold off time can be specified using the drop-down list adjacent to it, the choices being *seconds*, *milliseconds*, and *microseconds*. By default, the waveform gets displayed so that the left edge of the screen corresponds to the trigger point. However, if you want to view the waveform before the trigger occurs, you can do that by using the **Position** slider. This slider setting specifies where on the screen you want the trigger point of the waveform to be. When the slider is all the way to the left, the waveform gets displayed so that the trigger point of the waveform is at the left edge of the screen. When the slider is all the way to the right, the waveform gets displayed so that the trigger point of the waveform is at the right edge of the screen. The **Trigger Mode** drop-down list lets you choose between **Repeat** and **Single** trigger modes. In **Single** mode, once a trigger occurs, subsequent triggers get ignored. That is, the waveform that gets drawn upon the first trigger stays on the screen. In **Repeat** mode, after a time corresponding to the width of the screen plus the specified hold off time, the next trigger point on the incoming signal causes a refresh of the waveform display. At any time, you can press the **Re-Trigger** button to make ElectricVLab look for a new trigger point in the incoming signal and refresh the display if the trigger occurs.

The **Oscilloscope Dialog** has **File** and **Options** menus. The menu items under the **File** menu allow you to save the waveform display into an image file. Selecting the **Preferences** item under the **Options** menu brings up the **Oscilloscope Preferences Dialog**. This dialog lets you customize the color of the waveform display, the background color of the screen, color of the cursors etc. Any changes you make to these preferences apply to all the oscilloscopes in ElectricVLab and not just the one for which you invoked the dialog.

The title in the title bar of the **Oscilloscope Dialog** is **Oscilloscope:X** where **X** is a number. The same number is also displayed in the upper left corner of the corresponding oscilloscope's screen on the **Circuit Board**. This numbering helps you to know which **Oscilloscope Dialog** corresponds to which oscilloscope when there are multiple oscilloscopes on the **Circuit Board** and you have multiple **Oscilloscope Dialogs** open at the same time. ElectricVLab automatically assigns a unique number to each of the oscilloscopes present on the **Circuit Board**.

## 3.15 Curve Editor Dialog

In the **Curve Editor Dialog**, a curve is represented by linear segments connecting a series of points. The dialog allows you to insert points, delete points, and move points.

There are two ways to insert a new point.

- Right click anywhere on the graph. From the context menu that comes up, choose the **Insert point** option.
- Press and hold down the left mouse button anywhere on the graph and hit the *Insert* key on the keyboard.

There are also two ways to delete a point.

- Right click on the point. From the context menu that comes up, select the **Delete point** option.
- Press and hold down the left mouse button on the point and hit the *Delete* key on the keyboard.

To move a point, move the mouse over the point and press down the left mouse button. Then, the point gets selected and highlighted. Without releasing the mouse button, move the point to the location you want. When the point is at the location you want, release the mouse button.

At the top of the dialog are fields for the minimum and maximum values for the X and Y axes. If any of these values are not allowed to be altered in a particular context, then those particular fields get disabled. Otherwise, you can enter new values in those fields.

## 3.16 Spectrum Analyzer Dialog

The **Spectrum Analyzer** displays the magnitude of a signal as a function of frequency. In *Simulation* mode, if you click on a **Spectrum Analyzer** placed on the **Circuit Board**, the **Spectrum Analyzer Dialog** gets displayed. In the dialog, the X axis of the graph corresponds to frequency measured in *Hz* (cycles/second), whereas the Y axis corresponds to signal power measured in *dBm* (Decibel-milliwatts).

In the **Center Frequency** field, you can type in the center of the frequency band you are interested in. The value specified in the **Frequency Span** field specifies the frequency range corresponding to the full width of the dialog screen. The value in the **RBW** (Resolution Bandwidth) field specifies the frequency resolution at which the spectrum is computed.

You can have ElectricVLab automatically choose the minimum and maximum magnitude values for the Y axis by checking the **Auto Magnitude Scale** checkbox. Alternatively, you can enter the values of your own choosing in the **Minimum Magnitude** and **Maximum Magnitude** fields.

The horizontal and vertical cursors can be used to make more precise magnitude and frequency measurements. They can be enabled by checking the corresponding check boxes on the dialog. The cursors can be *dragged* to any desired location on the screen using the mouse. At the location of the vertical cursor, the corresponding value of frequency gets displayed. At the location of the horizontal cursor, the corresponding value of power gets displayed.

## 3.17 Component Labels

Component labels get displayed next to the components on the **Circuit Board**. In general, a component label has two parts, a **Name** and a **Value**.

You can assign names to individual components by typing in the **Name** field of the *Parameter Editor Dialog*. Assigning names to components is optional. It can be useful for communication purposes such as while describing the operation of a circuit.

For some common components **ElectricVLab** also displays their value as part of the label. To avoid cluttering the circuit, this is done only for some common components such as resistors, capacitors, and inductors.

The color of the label text can be changed using the *Application Preferences Dialog*.

You can also toggle the display of labels between on and off using the **Display Labels** item under **Options** menu.

## 3.18 3D View and 2D View

At any time, you can switch between *3D View* and *2D View* by clicking the corresponding radio button near the top left corner of the main window.

You would normally want to use the *3D View* to visualize the behavior of the circuit in *Simulation* mode.

The *2D View* depicts the components using circuit symbols. You may find that it is easier to connect the components with wires by switching to *2D View* since the connection points are more clearly visible in this view.

The color of the symbols depicting the components in *2D View* can be changed via the *Application Preferences Dialog*.

## 3.19 Ground Point

All the voltage values in a circuit are reported relative to the voltage at a particular reference point named *ground*. When you are building a circuit, you can explicitly specify which point should be taken as ground by placing a **Ground** component there. This component can be found under the **Connection Elements** category. The voltage at the ground point is always taken to be zero. If you place multiple **Ground** components at different points in a circuit, all those points will be at zero voltage level.

When you don't specify the ground point yourself, i.e., when no **Ground** component has been added to the circuit, ElectricVLab automatically selects the ground point based on some heuristics. If you want to see which point got selected as the ground point, set the **Display Ground Indicator** option in the *Application Preferences* dialog. If that option is set, when the ElectricVlab is in *Simulation* mode, it displays a *Ground Indicator* at the point it selected to be the ground point. This *Ground Indicator* has the same shape as the normal **Ground** component, but, a slightly different color.

## 3.20 Momentary Switch

The **Momentary Switch** usually stays in its *normal state*. It toggles to the opposite state only when you point the mouse at it and hold down the mouse button. The moment the mouse button is released, it reverts back to its normal state. By default, the *normal state* is set to be *Open*. But, you can configure the normal state to be either *Open* or *Closed* by using the *Parameter Editor Dialog*.

CHAPTER

FOUR

# COMMON USAGE TOPICS

The following sections provide some information that is very helpful to have while using ElectricVLab.

## 4.1 Zooming, Moving, and Rotating the View

At the bottom right corner of the ElectricVLab window is the **View Controls** panel. The two buttons with plus and minus magnifying glass icons are used to zoom in and zoom out, respectively. The two buttons with the right and left turn symbols are used to rotate the view clockwise and counterclockwise, respectively. Lastly, there are two buttons with up and down arrow icons which are used to tilt the view up and down, respectively.

As an alternative to using the buttons in the **View Controls** panel, you may zoom, rotate and tilt the view using the middle mouse button (mouse wheel) while the cursor is over the **Circuit Board**. Scrolling the mouse wheel will zoom the view in and out. Pressing the middle mouse button and dragging vertically tilts the view up or down, whereas pressing the middle mouse button and dragging horizontally rotates the view clockwise and counterclockwise. You can also rotate the view using 'q' or 'e' key.

There are three ways to move/scroll the view.

- By pressing 'w', 'a', 's', 'd' keys, you can scroll the the view forward, left, backward, right respectively.
- When you move the mouse near the edge of the **Circuit Board**, an arrow button gets displayed to scroll in that direction. Pressing that button scrolls the view.
- Press down the right mouse button anywhere on the **Circuit Board** and drag the mouse in the direction you want the view to move.

## 4.2 Rotating Components

In order to rotate a component, ElectricVLab needs to be in *Build* mode and the *Object Tool*, whose button contains a hand-shaped icon in the **Build Tools** panel, needs to be active. Any of the following ways can be used to rotate a component when it is selected in the **Object Tool**.

- When the component is selected, each click of the *"<"* or *">"* key rotates the component clockwise or counterclockwise, respectively, by 90 degrees.
- When the component is selected, dragging the mouse (i.e., moving the mouse while holding down the left mouse button) rotates the component in increments of 90 degrees, either clockwise or counterclockwise, depending on which way the mouse is dragged.
- With the component unselected, right click on the component to bring up a context menu from which the menu items to rotate the component clockwise and counterclockwise may be selected.

## 4.3 Bringing Up the Parameter Editor Dialog for a Component

For a component that has been placed on the **Circuit Board**, you can bring up the *Parameter Editor Dialog* in any of the following ways. However, note that though many components do have modifiable parameters, some do not. The **Parameter Editor Dialog** will come up for a component in the following cases only if it has modifiable parameters.

- In *Simulation* mode, when you mouse over a component, the component gets highlighted. Right-clicking on the component brings up the context menu for it. If the component has modifiable parameters, one of the options in the context menu will be **Parameters**. Selecting that option brings up the **Parameter Editor Dialog**.
- In *Build* mode, there are two ways to bring up the *Parameter Editor Dialog*.
  - While the *Parameter Tool* is active, simply click on the component. This will bring up the **Parameter Editor Dialog**.
  - While the *Object Tool* is active, right click on the component to bring up a context menu which will have a menu item labeled **Parameters**. Selecting that menu item brings up the **Parameter Editor Dialog**.

## 4.4 Function Generator

The **Function Generator** component is a quite versatile generator of periodic voltage signals. By default, the **Function Generator** component generates a sinusoidal voltage of 5 Volts amplitude at 50 Hz. By bringing up its *Parameter Editor Dialog*, you can vary the **Amplitude**, **Frequency**, **Waveform**, **Phase Shift**, **DC Bias**, **Duty Cycle**, and **Waveform Curve** values.

The **Waveform** value can be **Sinusoid**, **Square**, **Triangle**, **Sawtooth**, **Pulse**, or **Custom**. When you choose **Custom** waveform, you get to customize the shape of the waveform yourself. This is done by specifying the **Waveform Curve**. For that, click on the value field of the **Waveform Curve** parameter. A button with the label "**...**" appears. Clicking that button brings up the *Curve Editor Dialog* in which you can create the waveform shape you desire.

CHAPTER

FIVE

# INTERFACING ELECTRICVLAB WITH ARDUINO BOARD

By interfacing ElectricVLab with physical Arduino boards, you can build interesting systems that combine real and virtual components. If you plug an Arduino Board into a USB port of your computer, ElectricVLab will communicate with the Arduino board over the USB. The interface for the flow of signals back and forth between the Arduino board and ElectricVLab is the **Arduino** component in ElectricVLab. It can be found under the **Arduino and Appliances** category.

The **Arduino** component in ElectricVLab can be thought of as an extender for the physical Arduino board. It has four *input pins* labeled **I0**, **I1**, **I2**, **I3** and four *output pins* labeled **O0**, **O1**, **O2**, **O3** which essentially serve as additional input/output pins for the Arduino board. These pins allow the virtual circuitry of ElectricVLab to be connected to your physical Arduino board. Using the utility library provided with ElectricVLab, your Arduino program can set/get the values at these virtual pins similar to the way it can set/get the values at the board's physical pins.

The ability to connect the virtual circuitry of ElectricVLab to your physical Arduino board opens up a number of possibilities which include the following:

- You can use the virtual components of ElectricVLab to send input signals to the Arduino board. The section on *Virtual Inputs to Arduino Board from ElectricVLab* discusses this in more detail.
- You can utilize the virtual instruments of ElectricVLab such as oscilloscopes, voltmeters, spectrum analyzers, etc. to look at the signals of the Arduino board. This way, you get to view the signals without needing real instruments. For details, see the section on *Using Virtual Instruments of ElectricVLab to View Arduino Board Signals*.
- The **Arduino** component in ElectricVLab can at the same time be both receiving signals from the Arduino board and sending signals to it. The symbiotic relationship between the real (Arduino board) and the virtual (ElectricVLab) facilitates the building of powerful systems.
- Suppose you want the signal output at a pin of the Arduino board to be processed by some electronic circuit. For that, you don't need to build that electronic circuit using real hardware components. Instead, you can send that signal as input to a virtual circuit built in ElectricVLab and get it processed by that virtual circuit. Even if your ultimate goal is to really build that circuit using hardware components, it can be advantageous to first try it out using virtual circuits in ElectricVLab. Changing the circuit components, their connections, their parameter values etc. can be done much more rapidly in ElectricVLab without incurring any cost of buying the components or the risk of destroying them. Once the circuit design is stabilized, you may choose to build the circuit with hardware, or in some cases, the Arduino board and ElectricVLab combination can be the final usage setup as well.

## 5.1 Copying the ElectricVLab_Arduino Library to Appropriate Location

This section discusses the one time setup that needs to be done on your computer for interfacing Arduino boards with ElectricVLab.

For communicating with ElectricVLab from your Arduino program, you need the *ElectricVLab_Arduino* library. To avoid the inconvenience of having to download it, this library gets bundled along with the ElectricVLab software. During installation, the folder corresponding to this library gets automatically copied into the directory where you choose to install ElectricVLab. For example, if you chose to install ElectricVLab into "C:\Program Files", you can find the *ElectricVLab_Arduino* library at the location shown in Fig. 5.1.

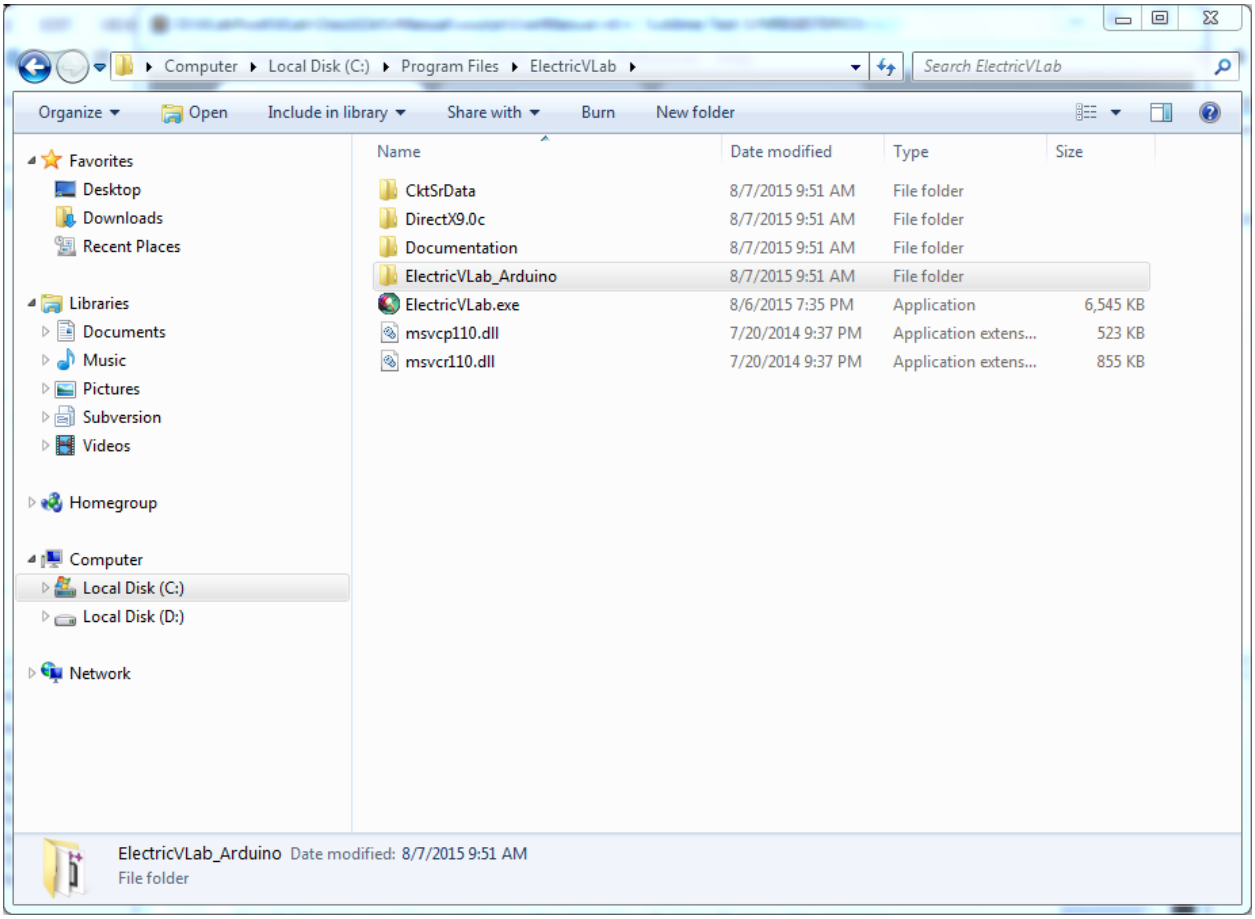



Fig. 5.1: Location where ElectricVLab_Arduino Library Folder Can be Found. Absolute path of the location will vary depending on where on the disk you choose to install ElectricVLab.

The Arduino IDE however generally expects all libraries to be present under the folder named *libraries* located in the Arduino installation folder. So, copy the folder named *ElectricVLab_Arduino* (along with the files contained in it) from the ElectricVLab installation folder on your machine to the folder "<Arduino installation folder on your machine>\libraries". Once you copy, the *libraries* folder should roughly resemble what is shown in Fig. 5.2.

With the ElectricVLab_Arduino library copied to the desired location, you are all set to interface ElectricVLab with Arduino boards.

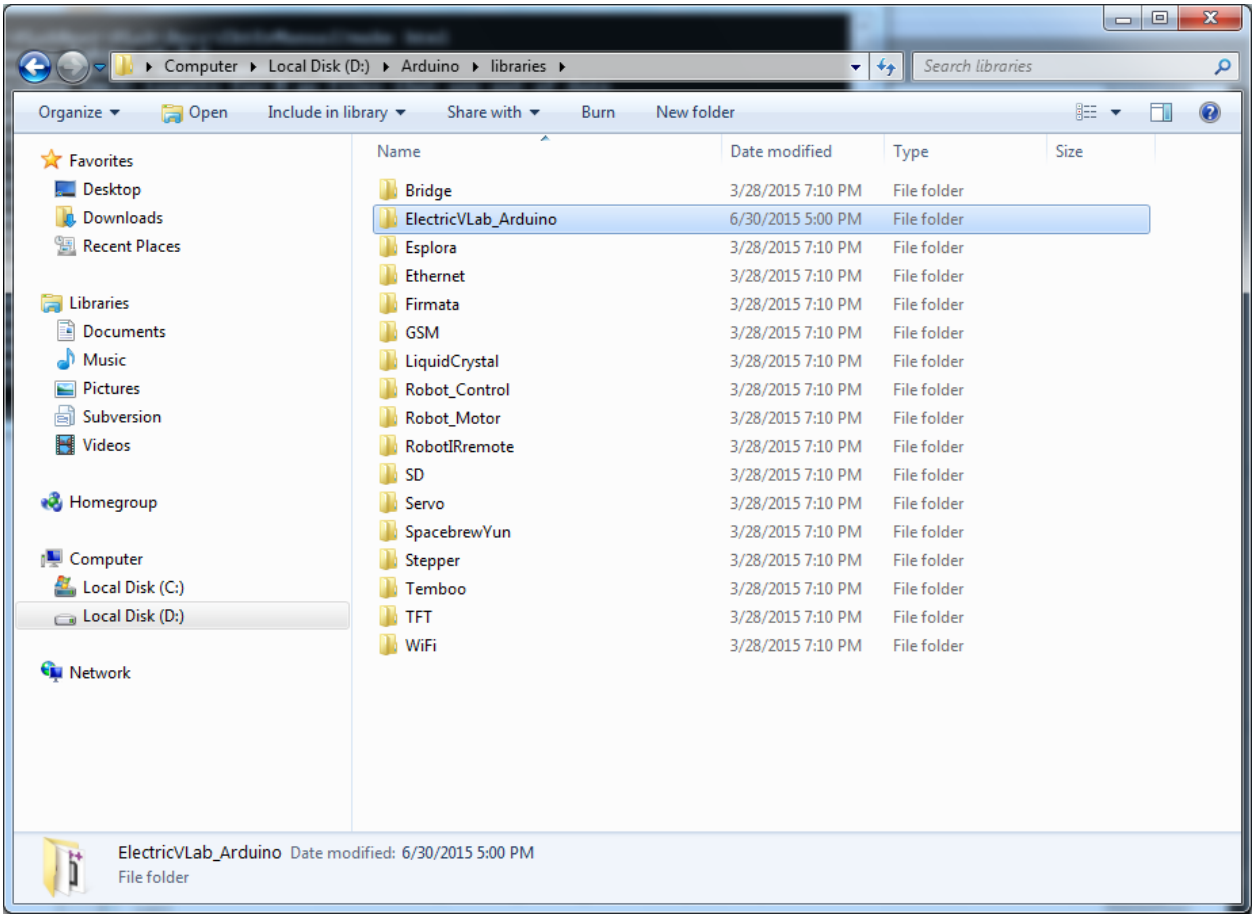



Fig. 5.2: The ElectricVLab_Arduino Location After It is Copied to be Under the Arduino IDE's *libraries* Folder.

## 5.2 Virtual Inputs to Arduino Board from ElectricVLab

Let us walk through a very simple usage case to familiarize you with the essentials of interfacing ElectricVLab with Arduino boards. Suppose you want to be able switch the LED at pin 13 on your Arduino board on and off whenever you desire. One way to achieve that would be something like the following (See e.g., [Blum2]). You would connect a hardware switch to some pin of the Arduino board. Then in your Arduino program's **loop** section, you will keep periodically reading the value at that pin using the *digitalRead* function and depending on the value, you would turn the LED on or off by calling *digitalWrite(13, HIGH)* or *digitalWrite(13, LOW)*. In reality, you would also need to take care of things like debouncing the switch.

Now, let us see how you can accomplish the same task without needing a hardware switch but using the virtual switch component available in ElectricVLab. First, build a circuit that looks like the one shown in Fig. 5.3 in ElectricVLab and save the circuit. In this circuit, a voltage source is connected to the **I1** input pin of the **Arduino** component via a switch. As mentioned earlier, the **Arduino** component can be found under **Arduino and Appliances** category. Also, some users might be surprised not to find a pull-down resistor or some such arrangement to ensure that pin **I1** is grounded when the switch is open. But, that is not necessary because switches in ElectricVLab are automatically grounded.

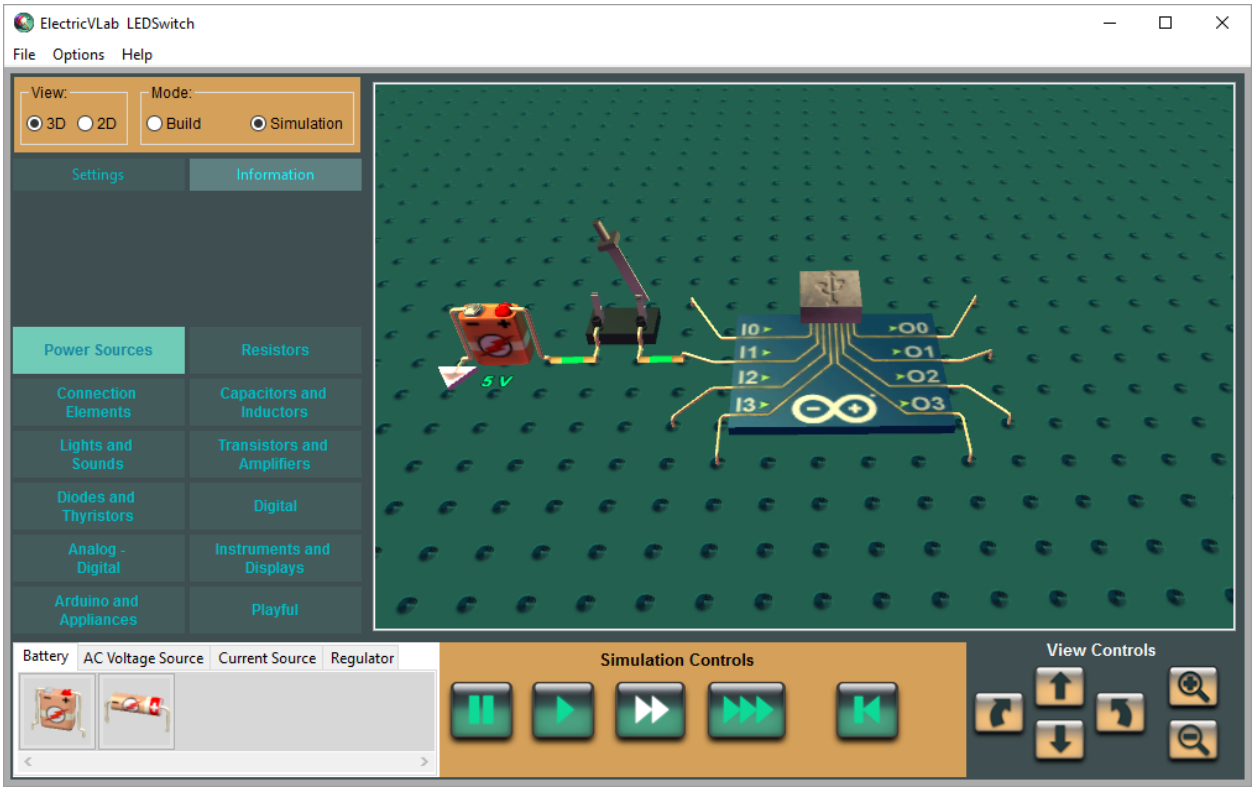



Fig. 5.3: Switching On/Off the LED on the Arduino Board from ElectricVLab.

Bring up the **Parameter Dialog** for the **Arduino** component in the circuit using any of the ways *discussed earlier*. In the dialog, you will see the following two parameters.

- **Serial Port**: This parameter specifies the serial port of the computer to which you have connected the physical Arduino board. ElectricVLab does attempt to automatically determine which port this is. However, in some cases, you may need to explicitly specify the correct port.
- **Baud Rate**: This specifies the baud rate to be used for communication between ElectricVLab and the physical Arduino board. By default, ElectricVLab chooses this to be 57600 bits/seconds. But, you can set this to other

values like 9600, 115200 etc. Generally, when you want higher frequency signals to be transmitted, choose higher baud rates. For normal usage, you can just leave it at the default value of 57600.

Next, we need to write the Arduino sketch (i.e., Arduino program). Run the Arduino IDE, and type the following code.

```
#include <ElectricVLab.h> // include the header file from ElectricVLab_Arduino library

ElectricVLab eVlab; // Always needed. Instantiates ElectricVLab class.

int ledPin = 13;

void setup()
{
  eVlab.begin(57600); // Parameter is baud rate. Must match the baud rate set on
                      // Arduino component in the circuit
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  float value = eVlab.getValueAtInPin(ElectricVLab::I1); // Read the value at the
                    // virtual pin "I1" of the Arduino component in the circuit. We
                    // connected the switch to that pin.

  // When the switch is closed in the circuit, the voltage at pin "I1" will be equal
  // to the battery voltage which is 5V. When the switch is open, the voltage at pin
  // "I1" will be 0. So, let us take the middle value of 2.5 as the threshold to
  // distinguish whether the switch is open or closed.
  if (value >= 2.5)
    digitalWrite(ledPin, HIGH);
  else
    digitalWrite(ledPin, LOW);
}
```

With all the comments stripped out, the real code is just the following.

```
#include <ElectricVLab.h>

ElectricVLab eVlab;

int ledPin = 13;

void setup()
{
  eVlab.begin(57600);
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  float value = eVlab.getValueAtInPin(ElectricVLab::I1);
  if (value >= 2.5)
    digitalWrite(ledPin, HIGH);
  else
    digitalWrite(ledPin, LOW);
}
```

After having typed the Arduino code, build it (i.e., press the “Verify” button on the Arduino IDE) and upload it to the

Arduino board as usual. Run ElectricVLab (in case you had exited) and load the circuit you had built (i.e., the one shown in Fig. 5.3). Go to **Simulation** mode. Toggle the switch between on and off states by clicking on it. If you did everything right, you would see that the LED on the Arduino board glows when the switch is closed and the LED turns off when the switch is open, thus illustrating how signals from ElectricVLab can control what happens on the Arduino board.

In this example, we connected just a simple switch to one of the input pins (i.e., **I1**) of the **Arduino** component in ElectricVLab. But, you can take it much farther than this by connecting to the input pins the outputs of much more complex circuitry. Different subcircuits can simultaneously feed different signals to the different input pins of the **Arduino** component to achieve much more complex things.

## 5.3 Using Virtual Instruments of ElectricVLab to View Arduino Board Signals

In the previous section, we discussed how ElectricVLab can be used to send input signals to the Arduino board. In this section, by taking another very simple example, let us discuss how the Arduino board can send signals to ElectricVLab.

Suppose you have connected some sensor to a particular pin of the Arduino board. Say, things are not working right and you want to check whether the sensor is responding correctly. That is, you want to view how the sensor is responding to its input. One way to do that would be to feed the sensor output to a real oscilloscope. But with ElectricVLab, you can accomplish that without the needing a real oscilloscope. All you have to do is to add a line in your Arduino program to send the sensor input value to a pin of the **Arduino** component in ElectricVLab. Then, by connecting the **Oscilloscope** component to that pin, you can view the sensor output on the virtual oscilloscope available in ElectricVLab. Let us walk through this example in greater detail.

First, build a circuit like the one shown in Fig. 5.4. Save the circuit.

Next, we need to write the Arduino sketch (i.e., Arduino program). Run the Arduino IDE, and type the following code.

```
#include <ElectricVLab.h> // include the header file from ElectricVLab_Arduino library

int sensorPin = 0;  // Analog pin of the physical Arduino board to which the sensor
                    // is connected
ElectricVLab eVLab; // Always needed. Instantiates ElectricVLab class.

void setup()
{
  eVLab.begin(57600); // Parameter is baud rate. Must match the baud rate set on the
                          // Arduino component in the circuit
}

void loop()
{
  int sensorValue = analogRead(sensorPin);  // read value from sensor
  eVLab.setValueAtOutPin(ElectricVLab::O2, sensorValue);  // Send the value to the
              // "O2" pin of the Arduino component in the circuit. Oscilloscope is
              // connected to that pin.
  delay(100);  // Wait a bit before reading the sensor value again
}
```

After typing the code, as usual, press the **Verify** button on the IDE to build the Arduino sketch and then upload it to the board.

Load the circuit you built earlier, i.e., the circuit shown in Fig. 5.4. Go to *Simulation* mode. Depending on the range of values sent by the sensor, you may or may not see anything on the oscilloscope screen. Specifically, note that input

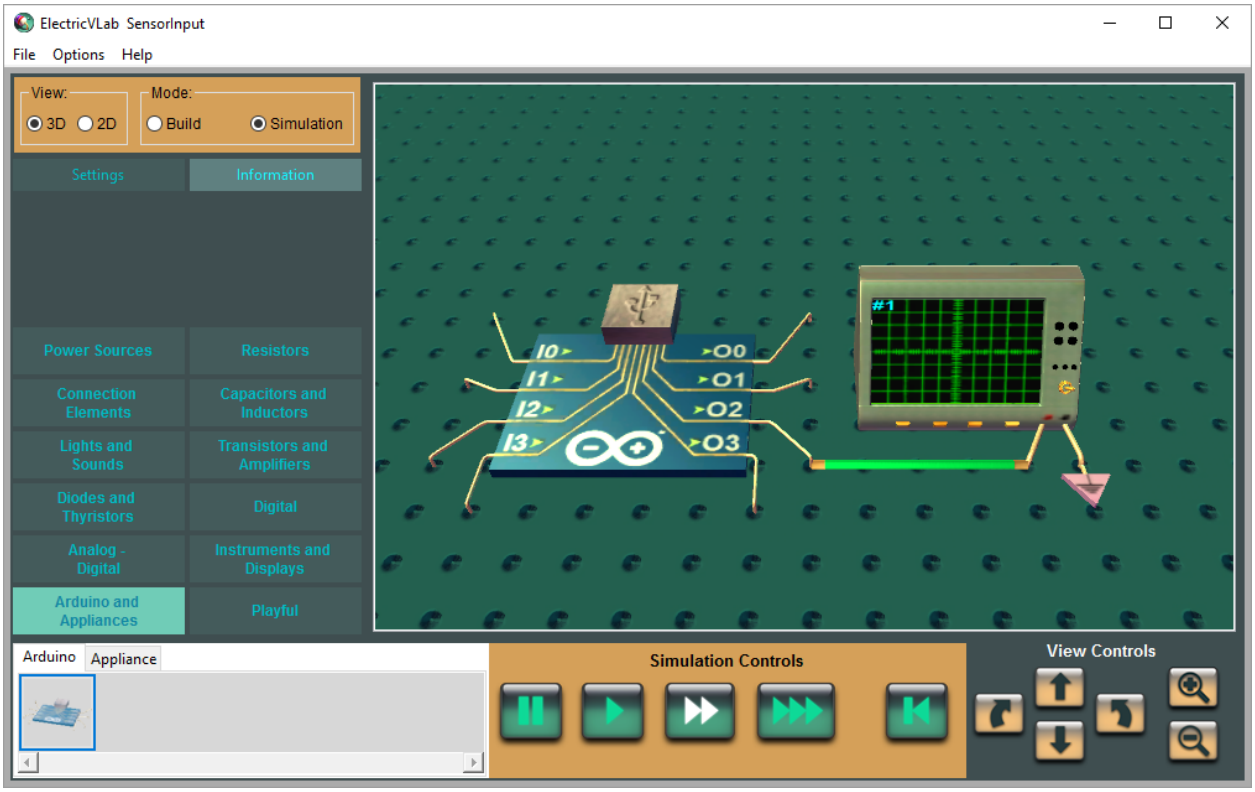



Fig. 5.4: Circuit to View the Sensor Response on Virtual Oscilloscope.

values at the analog pin typically range from 0 to 1024. So, with the default view settings of the oscilloscope, the signal waveform is likely to end up being way out of the screen. To view the signal properly, you may need to adjust the **Value/Division**, **Y Shift**, **Time/Division** settings of the oscilloscope. For that, click on the oscilloscope while in *Simulation* mode which brings up the dialog where you can adjust those settings. If the sensor values are in the range 0 to 1024, you may want to set **Value/Division** to 128 and **Y Shift** to "-512". These choices will make the sensor values of 0 and 1024 to correspond to the bottom and top of the oscilloscope screen respectively.

This example illustrated how ElectricVLab can be used to view the signal being sent from the Arduino board (which in turn was the input from a sensor). More generally, instead of just viewing the signal in ElectricVLab, you can build virtual circuitry to process that signal in myriad different ways and accomplish many different things. So, in the case of the circuit shown in Fig. 5.4, instead of connecting the output pin **O2** to just an oscilloscope, you can connect that pin to the input of a complex circuit involving the many analog and digital electronic components available in ElectricVLab.

For simplicity, first we discussed an example of signal flowing from ElectricVLab to Arduino board, and then we discussed an example of signal flowing the other way. But, signals can flow in both directions simultaneously. The virtual circuit in ElectricVLab can simultaneously be sending signals to the Arduino board and receiving and processing signals received from it. By harnessing that generality, one can build much more complex and interesting systems that blend the virtual and the real to complement each other.

# BIBLIOGRAPHY

[Blum2] Jeremy Blum, Tutorial 02 for Arduino : Buttons, PWM, Functions. https://www.youtube.com/watch?v=_LCCGFSMOr4

# INDEX


## A

Application Preferences Dialog, 23
Arduino, 30

## B

Bill of Materials, 16
Block mode, 19
Build Tools, 17

## C

center in view, 18
Circuit Information Dialog, 24
Circuit Settings Dialog, 24
click, 2
Component Label, 27
Curve Editor Dialog, 26

## D

Delete Tool, 20
drag, 2
duplicate, 18

## F

Function Generator, 30

## G

Ground, 27

## L

left-click, 2
Load My Circuit, 15
Load Sample Circuit, 16

## M

Momentary Switch, 28
mousing over, 2
Move View, 29
Multiple component operations, 19

## N

New Circuit, 15

## O

Object Tool, 18
Oscilloscope Dialog, 24

## P

Parameter Editor Dialog, 23
Parameter Tool, 20
Parameters, 29
Pause, 22

## R

Redo, 20
Reset, 22
right-click, 2
Rotate Component, 29
Rotate View, 29

## S

Save, 16
Save As, 16
Screenshot, 16
Scroll, 29
Simulation Controls, 22
Spectrum Analyzer Dialog, 26
SPICE Netlist, 16
Switch
    Change state, 21

## T

Tutorial, 2

## U

Undo, 20
Unpause, 22

## V

View, 27
View Controls, 22

## W

Wire Tool, 19

# Z

Zoom, 29