Lesson 4 Digital Pins and Constants

The Big Idea:

This lesson is the first step toward learning to connect the Arduino™ to its surrounding world. You will connect lights to your Arduino™ and then write sketches to turn them on and off in any desired pattern. In the process you will learn how to configure the digital pins of the Arduino™ in order to control devices and to turn those devices on and off. You will also learn how to use a solderless bread-board to connect electronic devices together and to the Arduino.™ Later lessons expand this connection ability to control motors, make sounds, detect light, and receive and transmit messages.

Background:

In Lesson 3, sketches used the serial port to send text from the Arduino™ Uno to the computer running the Arduino™ IDE. In Lesson 4, you will learn to make things happen by taking advantage of Arduino™ pins.

A *pin* is a connection with which the Arduino™ can be wired to external devices — everything from motors and switches to display panels.

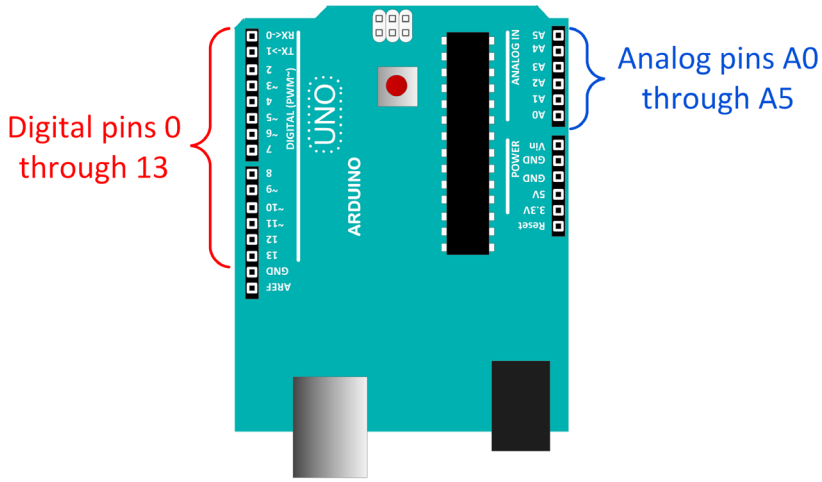


Figure 4-1. Arduino™ Uno with digital and analog pins called out

The Arduino™ Uno has two kinds of pins for receiving and sending information: analog and digital. The six analog pins, pictured on the upper right side of the Arduino™ in Figure 4-1, are the subject of a future lesson. This lesson is about digital pins, of which the Arduino™ has 14. These are numbered from 0 through 13 and are found along one side of the board.

A digital pin has only two states; on or off. The names for these states are `HIGH` and `LOW`. A `HIGH` state means that a volt meter connected to that pin would measure +5 volts. `LOW`, by contrast, means a measurement of zero volts.

Suppose a pin is HIGH. Where do the +5 volts come from? This depends on the mode of that pin. A digital pin can be set to detect the presence or absence of +5 volts coming from outside the Arduino.™ This voltage can come from a battery or some sort of sensing device. A digital pin that is set to detect the presence or absence of +5 volts from an outside source is said to be in the INPUT mode. Such a pin can detect signals from the outside world.

But an Arduino™ sketch itself can set a pin to HIGH or LOW. A pin that can have its voltage set from within a sketch is said to be in the OUTPUT mode. Pins in OUTPUT mode are used to turn devices on and off, to send signals, to control motors, and to generate sounds.

Table 4-1. Summary of modes and states of pins

Pin Mode	Pin Status	Meaning*
OUTPUT	HIGH (on)	The Arduino™ raises the voltage of the pin to +5 volts, meaning that devices connected to this pin have access to electricity. A light, for example, could come on, or a motor could start to turn.
	LOW (off)	The Arduino™ sets the voltage of the pin to zero volts. A light connected to this pin would go dark; a motor would stop.
INPUT	HIGH (on)	The presence of +5 volts is detected on this pin. This voltage is coming from outside the Arduino™ and can be from a switch or a sensor. Some sensors are +5 volts when nothing is detected.
	LOW (off)	The voltage of the pin is determined to be zero. This may reflect a button being pushed or a sensor detecting a signal.

*The meanings in this table are merely possibilities that reflect what commonly happens. What actually happens depends on the device and how it is wired to the Arduino.™ For example, in these lessons push buttons are usually connected in such a way as to produce +5 volts on a pin in INPUT mode when the button is not being pushed. The voltage drops to zero when the button is pushed.

This lesson will confine itself to digital pins in the OUTPUT mode. It also introduces some new electronic components and the schematic diagram.

Table 4-2. Vocabulary

Term	Definition
breadboard	As a verb, to construct an electronic circuit for purposes of testing. The components and wires of such circuits are plugged onto a special device designed for this purpose called a **bread-board**.
constant	A predefined value that is used in Arduino™ sketches to describe the state of a pin and the mode of a pin.
current	The number of electrons moving per unit of time. The unit of measure is the ampere.
`HIGH`	A constant meaning the voltage of a digital pin is +5.
`INPUT`	A constant meaning the mode of a pin is set to sense the voltage being applied from an outside source.
jumper wires	The wires that connect components to each other and to the Arduino™ on the bread-board.
light-emitting diode (LED)	An electronic device that lights up when it is properly connected to an Arduino™ pin set to the `OUTPUT` mode and `HIGH`.
`LOW`	A constant meaning the voltage of a digital pin is zero.
Ohm's Law	An equation that defines the mathematical relationship of voltage, current, and resistance.
`OUTPUT`	A constant meaning the mode of a pin is being set by the Arduino™ sketch. This voltage may be used by an outside device connected to this pin.
parameter	A parameter is a special kind of variable used by a method to refer to data provided as input. A value placed inside parentheses for some C-language commands or method.
pictorial	A picture or drawing; for purposes of this book, it is specifically a picture or drawing of the components wired and connected to digital pins on the Arduino.™
pin	A pin is a connection with which the Arduino™ can be wired to external devices — everything from motors and switches to display panels.
resistance	The tendency of materials to resist the movement of electrons. Metal has low resistance; glass has very high resistance.
resistor	A component that attempts to inhibit the flow of electrons, among other uses. In this lesson, a resistor is used to limit the electrical current that goes through an LED.
schematic	A drawing shorthand used by engineers to show how components are wired together.
voltage	The force of electricity, sometimes referred to as the determination of electrons to move. Lightning, for example, has very high voltage—several hundred million volts. A double-A battery, by contrast, has merely +1.5 volts.

Description:

This lesson introduces electronic components for the first time—in particular, the light-emitting diode (LED) and the resistor. It also includes some drawings of how these components are to be wired and connected to digital pins on the Arduino.™ A picture or drawing of this connection is called a *pictorial*.

The nice thing about a pictorial wiring diagram is that it is easy to duplicate. Just connect the wires as shown. The trouble with the pictorial, however, is that showing even modestly complex circuits becomes very difficult. Engineers use a kind of drawing shorthand, called the *schematic*, for showing how components are wired together. A schematic is concise; it does an excellent job of showing how electricity is expected to move through a circuit. Complex circuits are much easier to diagram. More to the point, it is far easier for a programmer or engineer to understand complex circuitry by reading a schematic rather than a pictorial.

But schematics are also abstract. They reflect how the electricity flows, not how the parts and wires are physically arranged. Reading a schematic is a learned skill. As components are introduced, their images are accompanied by their schematic symbol. For the next few lessons, wiring diagrams will be presented in both pictorial and schematic form. Later lessons present wiring diagrams in schematic form only.

Components

This lesson introduces the light-emitting diode (LED) and the resistor. These are added to the big component that's been in use since Lesson 1, the Arduino™ itself. Each has a pictorial and a corresponding schematic symbol.

Light-emitting diodes

An *LED* is a semiconductor device that lights up when properly connected to a source of electrical current. A proper connection means that the anode is positive relative to the cathode. For an Arduino™ this usually means the cathode is connected to ground (GND).

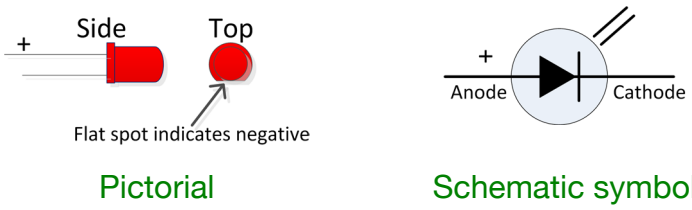
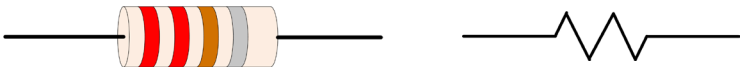


Figure 4-2. LED shown in pictorial and schematic forms

Resistors

A *resistor* is a component that attempts to inhibit the flow of electrons. A resistor has many uses, but in this case it is used to limit the electrical current that goes through an LED. Most resistors are made from some carbon compound. They come is a wide variety of values; color stripes indicate the value of an individual resistor.

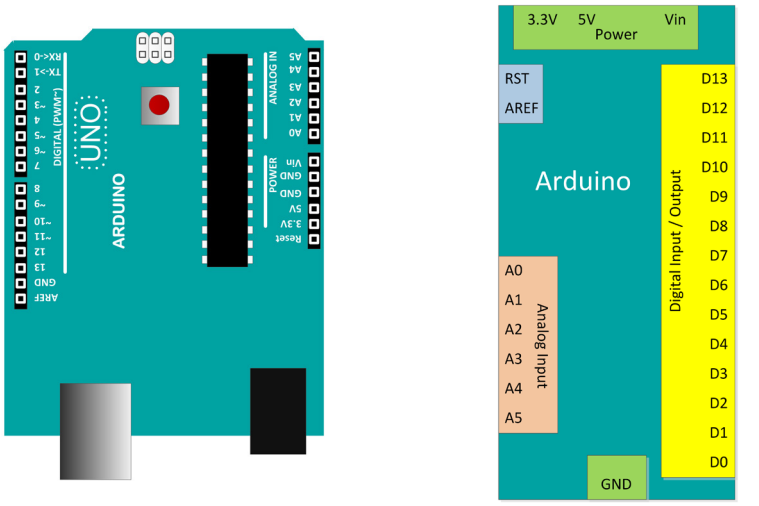


Pictorial Schematic symbol

Figure 4-3. Resistor shown in pictorial and schematic forms

Arduino™

Important	Notice that, as with the other components, each connector on the physical Arduino™ Uno board has its counterpart on the schematic diagram.



Pictorial diagram Schematic diagram

Figure 4-4. Arduino™ Uno shown in pictorial and schematic forms

Tools

Bread-board

A bread-board is a prototyping tool used to temporarily connect electronic components to each other to prototype circuits. It is not a true electronic component, and, thus, it does not appear in schematics or have a schematic symbol.

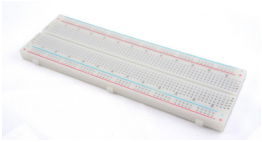


Figure 4-5. Bread-board

Jumper wires

For purposes of this book, jumper wires connect components to each other and to the Arduino™ on the bread-board. Jumper wires come in a variety of sizes and colors, which vary to distinguish the objects to which they are connected. Male-to-male jumper wires are used to make the electrical connections between components. Female-to-female jumper wires simply extend male jumpers too small for a job.

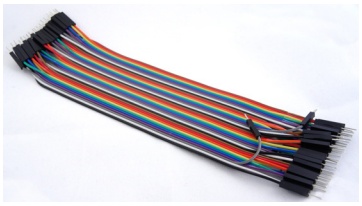


Figure 4-6. Jumper wires

Programming a Digital Pin

In this lesson digital pins are used to light LEDs. This requires setting each pin to the status of `OUTPUT`.

Two new C-language statements are required, `pinMode()` and `digitalWrite()`. Both of these statements require parameters. These are the pin number and desired mode for `pinMode()` and pin number and status for `digitalWrite()`.

`pinMode()` *and* `digitalWrite()`

```
pinMode( pinNumber, mode)
```

`pinNumber`: integer that specifies which pin is to be accessed.

`mode`: constant specifying the pin's direction.

`OUTPUT`: program can set the pin's voltage to +5V or 0v.

`INPUT`: program can detect a voltage applied to the pin.

```
example: pinMode( 2, OUTPUT); // sets pin 2 to output
   digitalWrite( pinNumber, status)
```

`pinNumber:`	integer that specifies which pin is to be accessed.
`status:`	constant specifying whether that pin is to be set to +5v or 0 volts.
	`HIGH`: sets pin to +5 volts.
	`LOW`: sets pin to 0 volts.
`example:`	`digitalWrite( 3, HIGH); // sets pin 3 to 5v.`

Electronics and Ohm's Law

Underlying everything the Arduino™ does are *voltage*, *current*, and *resistance*. These will appear again and again, so taking some time now to understand them will make future challenges easier. Electricity is all about electrons — how many there are, how badly they want to get from one place to another, and how difficult it is for them to move.

Table 4-3. Components of Ohm's Law

Voltage:	The difference in electric potential between two points in space, measured in volts. How badly some electrons want to get from one place to another.
Current:	A flow of electric charge, measured in amperes (amps). How many electrons want to move.
Resistance:	The degree of opposition an electrical current will encounter when passing through an electrical conductor, measured in ohms. How much trouble electrons will encounter in attempting the move.

The relationship of these is described by Ohm's Law:

Ohm's Law: $V = IR$

where:
V is voltage, in volts
I is current, in amps
R is resistance, in ohms

The LED is a device made from semiconducting material, usually silicon. This silicon in an LED is modified to conduct electrical current easily but only in only one direction. A problem occurs when the LED is connected to power because the LED has a low resistance. A quick look at Ohm's law shows that if the resistance (R) is low, the current (I) will be high.

For that reason, an LED is always connected to power in series with a resistor, usually 220 ohms for a source of +5 volts. This is the value of the resistor used in these lessons.

Goals:

By the end of this lesson the reader will:

1. Know how to identify the pins by number on the Arduino™ Uno.
2. Know how to configure any particular pin as `INPUT` or `OUTPUT`.
3. Know the meaning of and how to use the constants `INPUT` and `OUTPUT`.
4. Know how to set the output of a digital pin to `HIGH` or `LOW`.
5. Know the meaning of and how to use the constants `HIGH` and `LOW`.
6. Be able to identify an LED, including which lead is the anode (+) and which is the cathode (-).
7. Be able to identify a 220-ohm resistor and know that it is used in this case as a way of limiting how much current passes through the LED.
8. Be able to connect an LED to an Arduino™ pin and control that LED with an Arduino™ sketch.

Materials:

Quantity	Part	Image	Notes	Catalog Number
1	Arduino™ Uno		Single-board computer. This board is delicate and should be handled with care. When you are not using it, keep it in a box or plastic bag.	3102
1	USB Cable		This is the standard USB adapter cable with the flat connector on one end and the square connector on the other.	2301
1	Computer with at least one USB port and access to the Arduino™ website, http://www.arduino.cc	---	The operating system of this computer must be Windows, Macintosh OS/X, or Linux.	---
6	Light-emitting diodes (LEDs)		Single color, about 0.02 amps rated current, diffused.	1301
6	220 ohm resistors		¼ watt, 5% tolerance. Color code is red–red–brown-gold.	0102
1	Bread-board		Used for prototyping.	3104
As-req'd.	Jumper wires		Used with bread-boards for wiring the components.	3105

Procedure:

Part I: Set up and test a set of six LEDs connected to pins 2 through 7 of the Arduino.™

1. Connect the Arduino™ to a set of six LEDs and resistors as shown in Figure 4-7.

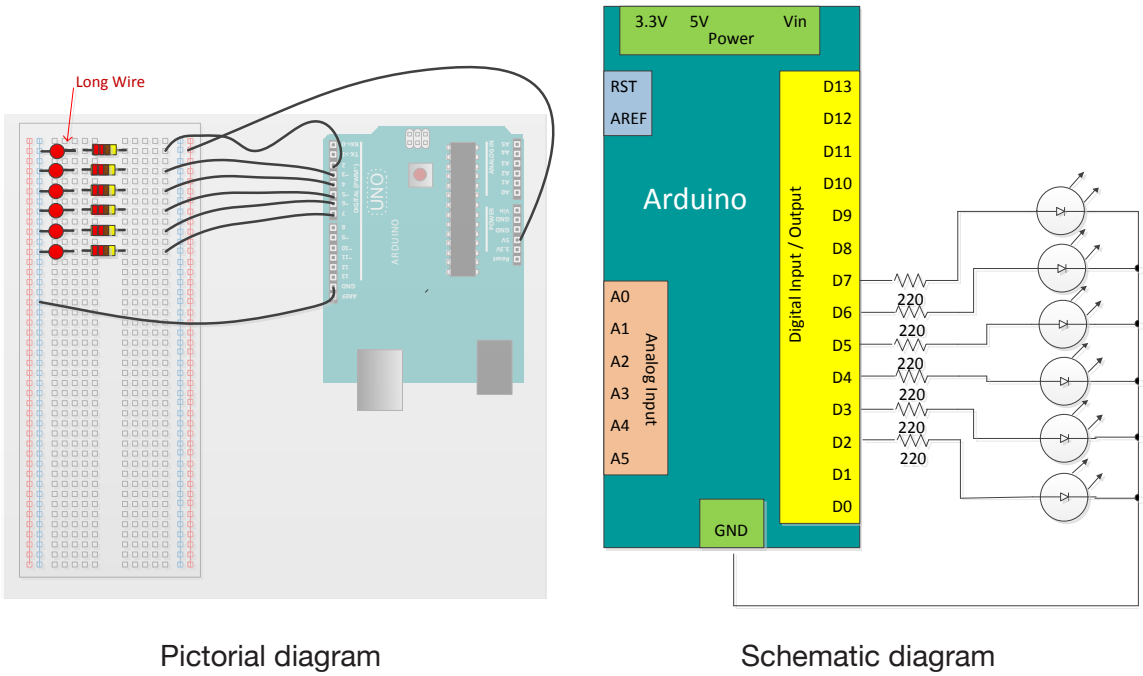


Figure 4-7. Pictorial and schematic diagrams of LED connections to the Arduino™

Important	Notice the difference between the Pictorial diagram and the Schematic diagram. The Pictorial diagram shows exactly where parts are placed on the bread-board and how wires are connected. The Schematic diagram, by contrast, shows components in abstract form. The electrical connections are clear, but the images don't match the real thing.

Schematic diagrams are the common way of illustrating how something is wired. Later lessons and projects in this book will provide only the schematic diagram, leaving the details of the wiring up to the programmer.

2. Connect the Arduino™ to the computer. Then start the Arduino™ Integrated Development Environment (IDE).

3. Enter the header comments for this lesson's sketch as shown in Snippet 4-1. Enter the programmer's (author's) name in place of W. P. Osborne. Enter the date on the next line.

Snippet 4-1.

```
/* Lesson4DigitalPins
  by W. P. Osborne
  <date>
*/
```


4. Add the `setup()` method as shown in Snippet 4-2. Within it, initialize pins 2 and 3 for `OUTPUT`. Notice that the other pins wired in step 1 are not being used yet.

Snippet 4-2.

```
...
void setup(){
    pinMode(2, OUTPUT);
    pinMode(3, OUTPUT);
}
```

5. Add the `loop()` method to cause the LEDs connected to pins 2 and 3 to blink alternately.

Snippet 4-3.

```
...
void loop(){
    digitalWrite(2, HIGH);
    digitalWrite(3, LOW);
    delay(500); // wait

    digitalWrite(2, LOW);
    digitalWrite(3, HIGH);
    delay(500);
}
```

6. Save the sketch as `Lesson4DigitalPins`. Upload to the Arduino™ and observe.

Exercises:

Exercise 4-1. LED thermometer

Make a sketch called `Thermometer` that makes all six LEDs light up, going from pin 2 through pin 7, and then go dark from pin 7 through pin 2, with 1/10th second between each change, thermometer style.

The sketch must:

- Begin with all LEDs off.
- Light the LED connected to pin 2, then wait 1/10th second.
- Leaving the LED connected to pin 2 on, light the LED connected to pin 3. Wait 1/10th second.
- Leaving the first two LEDs (pins 2 and 3) on, light the LED connected to pin 4. Wait 1/10th second, and continue until the LEDs on pins 2 through 7 are on.
- After leaving all LEDs on for 1/10th second, extinguish the LED connected to pin 7 and wait 1/10th second.
- Leaving the LED connected to pin 7 extinguished, turn off the LED connected to pin 6 and wait 1/10th second, and continue until all LEDs are off. Wait 1/10 second. By placing the code that does this in the `loop()` method, the pattern should repeat over and over.

Exercise 4-2. Pattern sketch

Create a new Arduino™ sketch that creates some sort of pattern and does something different than what has already been done in this lesson.