Lesson 6 Serial Input

The Big Idea:

Information coming into an Arduino™ sketch is called *input*. This lesson focuses on text in the form of characters that come from the user via the Serial Monitor. The Serial Monitor can also be used to send text back to the Arduino.™

Background:

To read the incoming text, the programmer needs to be able to detect that a character is ready to be read and have a way to retrieve the waiting character. To do so, three new programming statements are introduced. These are:

1. `Serial.available()` returns the number of characters waiting to be read from the serial port.
2. `Serial.read()` retrieves a character from the serial port and assigns it to a character variable.
3. `if(<condition>)` is a means of performing some programming statements if the condition evaluates to true.

Later lessons focus on methods used to transmit information to a sketch running on the Arduino,™ including push buttons, switches, sensors of all sorts, even gravity and movement. The Arduino™ has many ways of responding to that information, including movement, lights, sounds, and text on liquid crystal panels.

About Serial

`Serial.print` and `Serial.println` have been used without examining what *serial* means. Think for a moment about Samuel Morse, who is frequently credited with developing the first practical electrical telegraph in the United States in the 1830s. That telegraph consisted of wire that allowed operators to send and receive messages one character at a time. For example, to transmit the word "hello" first the letter "h" was sent, then the "e," then the "l" (twice), and finally the "o." Each character was identified by a series of pulses, some long and some short. Each pulse was sent one at a time.

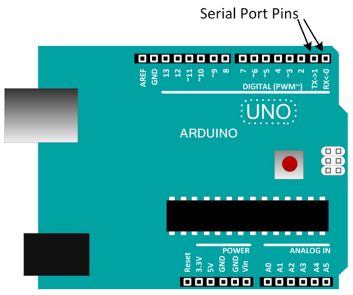
Table 6-1. Serial communication via Morse Code

Letter	Pulses (in order)
h	short short short short
e	short
l	short long short short
l	short long short short
o	long long long

This is called *serial communication* because each letter, in fact each pulse for each letter, is sent one at a time, in order, first to last.

If, however, the word "hello" were simply written on a card and that card handed to the recipient, all letters would be sent and received at the same time. This is called *parallel communication*.

Serial has the advantage over parallel that only a few wires are needed to send as much information as could be wanted.



The Arduino™ Uno has one serial port. When it is not being used to communicate with the IDE's Serial Monitor, it can transmit data via pin 1 and receive it via pin 0. A close look at these ports will reveal they are identified as TX and RX. (Other Arduino™ single-board computer models may have more than one. The Arduino™ Mega, for example, has four serial ports.)

Figure 6-1. Arduino™ Uno with serial port called out

About characters

So far the Arduino™ has sent messages to the Serial Monitor using letters, such as in "Hello, world!" But computers don't really know anything about letters. Rather, to a computer everything is a number. In order for a letter to appear, two things must happen. First, the computer must be given, or select, the number to be sent representing a letter and then be given the instructions necessary to read or display that number as a letter.

Table 6-2. Data types

Data Type	Description	Examples
`char`	character, a single letter or number to be interpreted as text	`char my Char;` `myChar` can be set to ‘a,’ ‘3,’ ‘!,’ ‘B,’ and any other character.
`int`	an integer	`int myInt;` `int` can be set to 3, 19, -212, and any other integer.
`String`	a collection of characters	`String myStr;` `myStr` can be set to "The answer is 42," "Heck no, I won't go," or any other collection of characters.

This lesson examines the relationship of characters to the integers used to represent them inside the Arduino.™ The capital letter *A*, for example, is stored in the Arduino™ as the integer 65. Whether the Arduino™ chooses to display it as 65 or the character *A* is determined by how the variable to contain it is declared. Either way, what is communicated is the number 65. In binary format this number is 01000001. The serial port sends each bit, in order, one at a time. This serial communication of the bits of the number 65, which, in turn, translate to the letter *A* is, for the Arduino,™ the equivalent of Samuel Morse's short and long pulses of the telegraph.

About `if` testing

The sketch in this lesson uses a statement, `Serial.available()`, that returns an integer greater than or equal to 0. This integer is the number of characters waiting to be read from the serial port after the [Send] button is clicked.

The keyword `if` is a test. The result of that test must be true or false. In this lesson the number returned from `Serial.available` is tested to see if it is greater than zero. If it is, then the statements between the curly braces are executed. Otherwise, these statements are skipped.

Lesson 8 covers `if` in more detail.

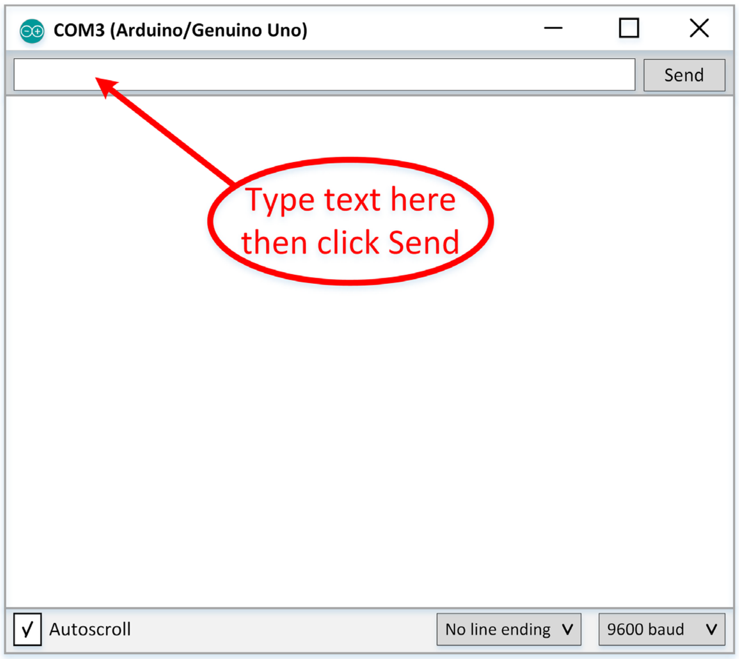
Table 6-3. Vocabulary

Term	Definition
`BIN`	BIN is binary, the number base used by computers.
`DEC`	DEC is decimal, which is in common use by people. Its base is 10 symbols, 0 through 9.
`HEX`	HEX is hexadecimal, the numbering scheme used by computer science to economically represent binary values. It has 16 symbols, 0 through 9 plus A, B, C, D, E, and F.

Term	Definition
input	Information going into an Arduino™ sketch from the Serial Monitor.
parallel communication	Communication method in which multiple bits of data are conveyed simultaneously.
serial communication	Comunication method in which data is conveyed one bit at a time.
Unicode	The parent container of computer characters. It includes the commonly used ASCII characters plus characters from most of the world's major languages.

Description:

Sending text to the Arduino from the Serial Monitor is pretty simple. The desired text is entered into the Text Input Box. The contents of this box are sent to the Arduino one character at a time when the [Send] button is clicked.



The new programming statements shown in Table 6-4 are used to read characters from the serial port.

Table 6-4. Programming statements used to read characters from the serial port

Programming Statement	Description
`Serial.available()`	Returns the number of characters waiting to be read
`Serial.read()`	Retrieves the character

Example 6-1 illustrates the use of `Serial.available()` and `Serial.read()` plus the character data type to first determine if a character is ready to be read and, if it is, retrieve it and store it to a character variable.

Example 6-1. Use of `Serial.available()` and `Serial.read()`

```
char myChar;  // declare variable to contain a character
if(Serial.available()){  // test, is character waiting?
    myChar = Serial.read();  // yes, retrieve it and
                             // assign value to myChar
```

Notice the character is not retrieved and assigned to `myChar` unless the serial port indicates at least one is available. This is what the `if(<condition>)` does. The condition, in this case, is `Serial.available()`.

Goals:

By the end of this lesson readers will:

1. Know how to accept text from the user and store it to a variable.
2. Know how to design and program an interactive program, one that responds to input from the user.
3. Know how to capture and display an entry as a character and as an integer.

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

Quan-tity	Part	Image	Notes	Catalog Number
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.	---

Procedure:

Part I: Set up, upload, and run the first program.

1. Connect the Arduino™ to the host computer and open the Arduino™ IDE.
2. Type in the heading comments, then declare the variables to be used.

Snippet 6-1.

```
/* Lesson6SerialRead
   <author>
   <date>
*/

int incomingInteger;
char incomingCharacter;
```

3. Add the `setup()` method. It need only initialize the serial port so that communications can be established with the Serial Monitor.

Snippet 6-2.

```
...
void setup(){
    Serial.begin(9600);
}
```

4. Add the `loop()` method with the test for an incoming character and the ability to read and print that character.

Snippet 6-3.

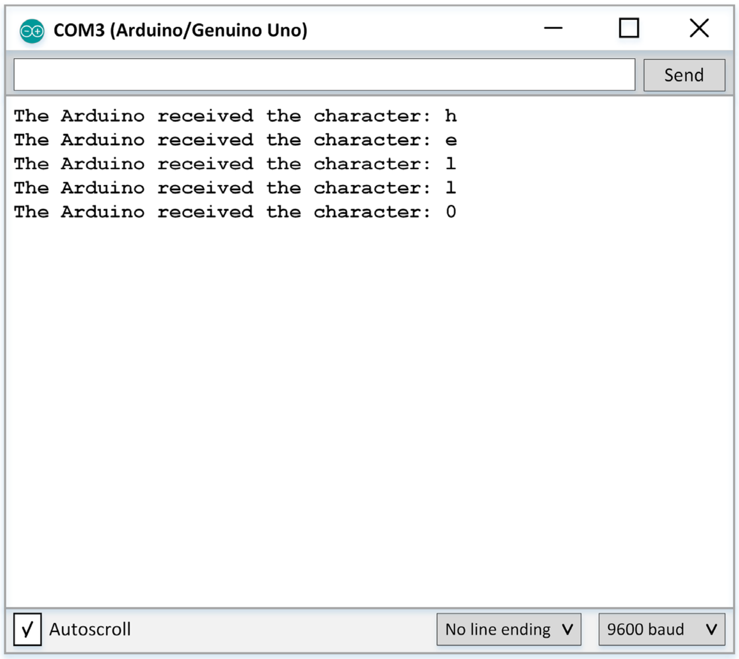
```
...
void loop(){
    // are characters waiting to be read?
    if(Serial.available() > 0){
        // Yes, so read next one as a character
    incomingCharacter = Serial.read();

    Serial.print("The Arduino™ received the character: ");
    Serial.println(incomingCharacter);
    }
}
```

6

5. Save the sketch, upload it to the Arduino™ and open the Serial Monitor.

6. Type the word `hello` into the text input box, then click [Send]. The Serial Monitor should look like this:



```
The Arduino received the character: h
The Arduino received the character: e
The Arduino received the character: l
The Arduino received the character: l
The Arduino received the character: o
```

7. Add some programming statements to display the numbers that are sent to the Arduino™ to be interpreted as characters. This is done by first assigning the value of the read character to an integer variable, then printing that value. Remember, the Arduino™ knows nothing of characters, really. It simply knows to print a character if the variable that contains it is of type `char`. But if the variable is of type `int` the actual number will be printed.

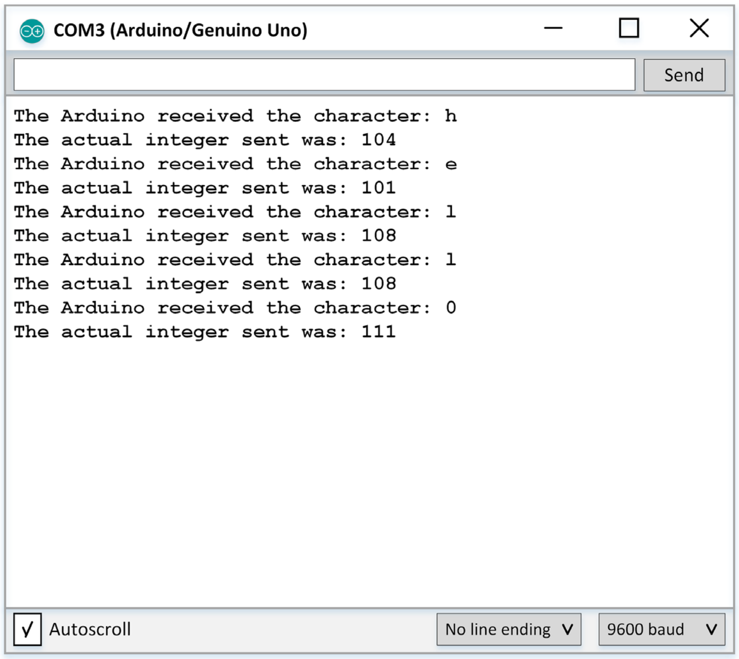
Snippet 6-4.

```
...
void loop(){
    // are characters waiting to be read?
    if(Serial.available() > 0){
        // Yes, so read next one as a character
    incomingCharacter = Serial.read();

    Serial.print("The Arduino™ received the character: ");
    Serial.println(incomingCharacter);

    // assign the character to an int variable
    // then print it.
    incomingInteger = incomingCharacter;
    Serial.print("The actual integer sent was: ");
    Serial.println(incomingInteger);
    }
```

8. Save the sketch, upload it to the Arduino™, and open the Serial Monitor. Type the word `hello` into the input text box and click [Send]. The Serial Monitor should look like this:



```
The Arduino received the character: h
The actual integer sent was: 104
The Arduino received the character: e
The actual integer sent was: 101
The Arduino received the character: l
The actual integer sent was: 108
The Arduino received the character: l
The actual integer sent was: 108
The Arduino received the character: o
The actual integer sent was: 111
```

Exercises:

Exercise 6-1. Show characters and integer

Modify the sketch `Lesson6SerialRead.ino` to print the character and its corresponding integer on one line, in the following format:

```
The character h has the integer value 104
The character e has the integer value 101
...
```

Use this table to prepare a set of characters, numbers, and punctuation. Then compare this with the ASCII Table found at http://www.asciitable.com.

Exercise 6-2. Show `String` of characters

Write an original sketch, called `Lesson6Exercise2.ino`, that reproduces the characters exactly as entered. The sentence Hello, World! typed into the input text box appears as `Hello, World!` when the [Send] button is clicked.

Exercise 6-3. Show character values and hexadecimal values

The programming statement `Serial.print` can be forced to use a specific format for displaying a character. This is done by adding a comma and a formatting instruction as follows:

```
char myChar = 'A';  // character literals are
                    //delimited by single quotes
Serial.println(myChar);  // prints the letter A
Serial.println(myChar, DEC); // prints 65
Serial.println(myChar, HEX); // prints 41
Serial.println(myChar, BIN); // prints 1000001
```

DEC, HEX, and BIN indicate a number base. DEC is decimal, which is in common use by people. Its base is 10 symbols, 0 through 9.

HEX is hexadecimal, the numbering scheme used by computer science to economically represent binary values. It has 16 symbols, 0 through 9 plus A, B, C, D, E, and F.

BIN is binary, the number base used by computers.

Write a short sketch called `Lesson6Exercise3.ino` that accepts characters from the Serial Monitor and displays both their character values and hexadecimal values. Then verify these values are correct by looking up the characters on the Unicode Character Set found at http://unicode-table.com/en/#control-character.

Unicode is the parent container of computer characters. It includes the commonly used ASCII characters plus characters from most of the world's major languages.

Complete listing 6-1. `Lesson6SerialInput`

```
/* Lesson6SerialInput
   by W. P. Osborne
   6/30/15
*/

int incomingInteger;
char incomingCharacter;

void setup(){
  Serial.begin(9600);
}

void loop(){
  // are characters waiting to be read?
  if(Serial.available() > 0){
    // Yes, so read next one as a character

    incomingCharacter = Serial.read();

   Serial.print("The Arduino™ received the character: ");
   Serial.println(incomingCharacter);

   // assign the character to an int variable
   // then print it.
   incomingInteger = incomingCharacter;
   Serial.print("The actual integer sent was: ");
   Serial.println(incomingInteger);
  }
}
```