Arduino Sketches

Get to know how sketches work, and how they are uploaded to an Arduino.

In the getting started guide (Windows, MacOS, Linux), you uploaded a sketch that blinks an LED. In this tutorial, you'll learn how each part of that sketch works.

A *sketch* is the name that Arduino uses for a program. It's the unit of code that is uploaded to and run on an Arduino board.

Comments

The first few lines of the Blink sketch are a *comment*:

```
COPY
1/*
2
3 * Blink
4
5 *
6
7 * The basic Arduino example.  Turns on an LED on for one second,
8
9 * then off for one second, and so on...  We use pin 13 because,
10
11 * depending on your Arduino board, it has either a built-in LED
12
```

```
13 * or a built-in resistor so that you need only an LED.

14

15 *

16

17 * http://www.arduino.cc/en/Tutorial/Blink

18

19 */
```

Everything between the

```
/*
```

and

```
*/
```

is ignored by the Arduino when it runs the sketch (the

```
*
```

at the start of each line is only there to make the comment look pretty, and isn't required). It's there for people reading the code: to explain what the program does, how it works, or why it's written the way it is. It's a good practice to comment your sketches, and to keep the comments up-to-date when you modify the code. This helps other people to learn from or modify your code.

There's another style for short, single-line comments. These start with

```
//
```

and continue to the end of the line. For example, in the line:

```
COPY

1int ledPin = 13;               // LED connected to digital pin 13
```

the message "LED connected to digital pin 13" is a comment.

Variables

A *variable* is a place for storing a piece of data. It has a name, a type, and a value. For example, the line from the Blink sketch above declares a variable with the name

```
ledPin
```

, the type

`int`, and an initial value of 13. It's being used to indicate which Arduino pin the LED is connected to. Every time the name `ledPin` appears in the code, its value will be retrieved. In this case, the person writing the program could have chosen not to bother creating the `ledPin` variable and instead have simply written 13 everywhere they needed to specify a pin number. The advantage of using a variable is that it's easier to move the LED to a different pin: you only need to edit the one line that assigns the initial value to the variable.

Often, however, the value of a variable will change while the sketch runs. For example, you could store the value read from an input into a variable. There's more information in the Variables tutorial.

Functions

A *function* (otherwise known as a *procedure* or *sub-routine*) is a named piece of code that can be used from elsewhere in a sketch. For example, here's the definition of the `setup()` function from the Blink example:

```
COPY

1void setup()

2{

3

4  pinMode(ledPin, OUTPUT);      // sets the digital pin as output

5}
```

The first line provides information about the function, like its name, "setup". The text before and after the name specify its return type and parameters: these will be explained later. The code between the `{` and `}` is called the *body* of the function: what the function does.

You can *call* a function that's already been defined (either in your sketch or as part of the Arduino language). For example, the line

```
pinMode(ledPin, OUTPUT);
```

calls the `pinMode()` function, passing it two *parameters*: `ledPin` and `OUTPUT`. These parameters are used by the `pinMode()` function to decide which pin and mode to set.

pinMode(), digitalWrite(), and delay()

The `pinMode()` function configures a pin as either an input or an output. To use it, you pass it the number of the pin to configure and the constant INPUT or OUTPUT. When configured as an input, a pin can detect the state of a sensor like a pushbutton; this is discussed in the Digital Read Serial tutorial. As an output, it can drive an actuator like an LED.

The `digitalWrite()` functions outputs a value on a pin. For example, the line:

```
COPY

1digitalWrite(ledPin, HIGH);
```

set the `ledPin` (pin 13) to HIGH, or 5 volts. Writing a LOW to pin connects it to ground, or 0 volts.

The `delay()` causes the Arduino to wait for the specified number of milliseconds before continuing on to the next line. There are 1000 milliseconds in a second, so the line:

```
COPY

1delay(1000);
```

creates a delay of one second.

setup() and loop()

There are two special functions that are a part of every Arduino sketch:

`setup()` and `loop()`. The `setup()` is called once, when the sketch starts. It's a good place to do setup tasks like setting pin modes or initializing libraries. The `loop()` function is called over and over and is heart of most sketches. You need to include both functions in your sketch, even if you don't need them for anything.

Exercises

1. Change the code so that the LED is on for 100 milliseconds and off for 1000.

2. Change the code so that the LED turns on when the sketch starts and stays on.

See Also

- setup()
- loop()
- pinMode()
- digitalWrite()
- delay()