Manuals and Curriculum
(http://playground.arduino.cc/Main/ManualsAndCurriculum)
Arduino StackExchange
(http://arduino.stackexchange.com)
Board Setup and Configuration
(http://playground.arduino.cc/Main/ArduinoCoreHardware)
Development Tools
(http://playground.arduino.cc/Main/DevelopmentTools)
Arduino on other Atmel Chips
(http://playground.arduino.cc/Main/ArduinoOnOtherAtmelChips)
Interfacing With Hardware
(http://playground.arduino.cc/Main/InterfacingWithHardware)

- Output

  (http://playground.arduino.cc/Main/InterfacingWithHardware#Output)

- Input

  (http://playground.arduino.cc/Main/InterfacingWithHardware#InputTOC)

- User Interface

  (http://playground.arduino.cc/Main/InterfacingWithHardware#ui)

- Storage

  (http://playground.arduino.cc/Main/InterfacingWithHardware#Storage)

- Communication

  (http://playground.arduino.cc/Main/InterfacingWithHardware#Communication)

- Power supplies

  (http://playground.arduino.cc/Main/IntWithHW-

  PwrSup)

- General

  (http://playground.arduino.cc/Main/InterfacingWithHardware#General)

Interfacing with Software
(http://playground.arduino.cc/Main/InterfacingWithSoftware)
User Code Library
(http://playground.arduino.cc/Main/GeneralCodeLibrary)

- Snippets and Sketches

  (http://playground.arduino.cc/Main/SketchList)

- Libraries

  (http://playground.arduino.cc/Main/LibraryList)

- Tutorials

  (http://playground.arduino.cc/Main/TutorialList)

Suggestions & Bugs
(http://code.google.com/p/arduino/issues/list)

```
Library Tutorial for Arduino
Author:  Alexander Brevig
Contact: alexanderbrevig@gmail.com
```

How to write libraries for the Arduino?

So you want to write a Library for the Arduino?
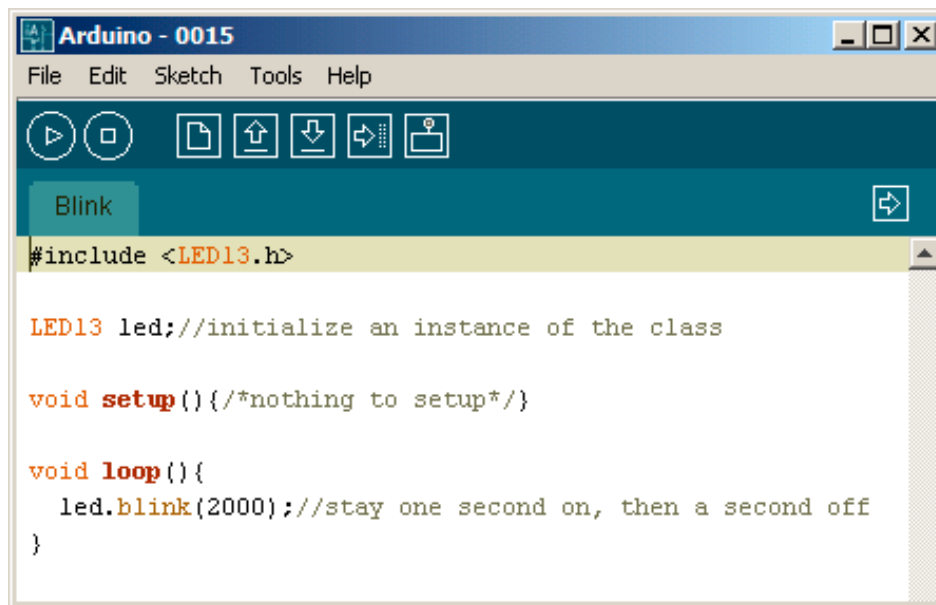-**Great**!
Here is a little tutorial on how to write one.

You will learn the basic syntax rules for c++ class declaration and definition and I will summarize the conventional Arduino function and variable naming.

For this tutorial we will implement a library called LED13 we will do some fun stuff with the LED @ pin 13
We will make it turn on, off and make it blink.

We will make this:

```
Arduino - 0015
File  Edit  Sketch  Tools  Help

Blink

#include <LED13.h>

LED13 led;//initialize an instance of the class

void setup(){/*nothing to setup*/}

void loop(){
   led.blink(2000);//stay one second on, then a second off
}
```

Navigation

- What is a Library?
    - What is the Arduino conventions?
- Why do we have libraries?
- How do we write libraries?
    - Define the functionality your Library has to provide
    - Make a proof of concept sketch
    - Learn c++ class syntax
        - LED13 Header
        - LED13 Source
        - Source for this tutrial
    - Implementation
        - Selecting public, protected or private
        - Selecting datatypes / returntypes
        - Selecting variable- / functionnames
        - Writing examples
        - keywords.txt
    - Debugging
        - Remember #include <WProgram.h>
        - Delete object file and recompile between each alteration of header and/or source
        - Learn to interpret the Arduino IDE debugger / compiler.
    - Sharing your library with other people
- Links
- Information about this page

What is a Library?

A Library is a collection of instructions that are designed to execute a specific task.
Good libraries describe the tasks it tries to solve by the name of the library itself, and the names of the functions associated to the library.
This is because the library should be intuitive to use, and all names should describe, in an efficient way, all questions regarding *what*, *when*, and *where*, but should conceal the *how*.

On the Arduino the libraries often serve as a **wrapper** around the Arduino API (http://arduino.cc/en/Reference/Extended) language to enhance code *readability* and *user experience*.

Arduino IDE places it's libraries in \hardware\libraries The Library for this tutorial has the URL C:\programming\arduino-0015\hardware\libraries\LED13 on my Windows XP machine.

For the Ubuntu operating system, the libraries may be found at /usr/share/arduino/libraries.

What is the Arduino conventions?

The Arduino libraries uses a UppercaseUppercase convention for Library names, and lowercaseUppercase naming convention for the functions.
What this means is that our library has/*should* get the name LED13 (not led13 for instance), and the functions will be named **on()** and **off()**.
I could've selected **turnOn()** and **turnOff()**, but I've selected the shortest because it is as descriptive, but in addition it is shorter. [short code == less probability for typos]

It is important to remember that you do not *need* a Library. It only makes it simpler.
If you are interested in the inner mechanics of a specific library, you'll find the *MyLibrary*.cpp (if downloaded) in `/hardware/libraries/MyLibrary/MyLibrary`.cpp `change MyLibrary to`
`whatever the name of the library you want to investigate`

Why do we have libraries?

- Simplify usage and/or organization of code

- Enhance code readability

- Decentralize logic

  - Main sketch can now simply state library.action() instead of coding the equivalent in the sketch.

How do we write libraries?

[define functionality - make a proof of concept - learn c++ class syntax - implement - debug]

Define the functionality your Library has to provide

For our Library we've decided on these functions:

- on()

- off()

- blink(int time)

This should be the premise for our library.
At this point, we do not care *how* the on() should work, only that it should exist.

Make a proof of concept sketch

Before implementing a new class I usually find it very useful to test the idea in a sketch. Try a few different implementations, see what uses the least amount of space, and what seems simplest in use. Select the solution that compromises these two in a suitable way.

An experience I've made is that if one has something slightly similar to work from, that is often beneficial.
For our Library, one immediatly thinks of the Blink right? The **Hello World!** of µCs. So, go ahead and start the Arduino IDE and open the File->Sketchbook->Examples->Digital->Blink.
The first thing we'll do, is to cut the code inside loop(), and paste it into a new function called void blink().
I suspect this is not surprising, being that we *know* this code will blink the LED connected to pin 13.
Now, write blink(); inside the loop, so that the code that once was inside the loop() and now is located inside blink, gets executed (because loop calls blink).

Now, you should have a sketch looking like this:

```
1.  int ledPin = 13;          // LED connected to digital pin 13
2.
3.  void setup()              // run once, when the sketch starts
4.  {
5.    pinMode(ledPin, OUTPUT);    // sets the digital pin as output
6.  }
7.
8.  void loop()              // run over and over again
9.  {
10.    blink(); //call the code that makes the LED blink
11.  }
12.
13.  //make the LED blink
14.  void blink(){
15.    digitalWrite(ledPin, HIGH);   // sets the LED on
```

16.   delay(1000);   // waits for a second
17.   digitalWrite(ledPin, LOW);   // sets the LED off
18.   delay(1000);   // waits for a second
19. }

The next two steps will be to add the void on(); and the void off();.
These will be really simple functions, becuase they will actually just 'wrap' **digitalWrite(13,HIGH)** and **digitalWrite(13,LOW)** respectively.

Go ahead and try to write these functions.
You should end up with something like:

1. //turn the LED on
2. void on(){
3.   digitalWrite(ledPin,HIGH); //set the pin HIGH and thus turn LED on
4. }
5.
6. //turn the LED off
7. void off(){
8.   digitalWrite(ledPin,LOW); //set the pin LOW and thus turn LED off
9. }

We have one more thing to fix.
As the code is now, the blink() will always use two seconds, but we wanted to pass the number of microseconds as an argument. Additionally we'll want to remove the redundant code that turns a led on and off. We'll replace those redundant calls with on() and off(). You probably know how to write functions that has parameters, but here it is:

1. //make the LED blink
2. void blink(int time){
3.   on();   // sets the LED on
4.   delay(time/2);   // waits for a second
5.   off();   // sets the LED off
6.   delay(time/2);   // waits for a second
7. }

Now you have enough code to implement all the desired functionality.

Complete Concept Sketch

```
 1.  int ledPin = 13;            // LED connected to digital pin 13
 2.
 3.  void setup()               // run once, when the sketch starts
 4.  {
 5.    pinMode(ledPin, OUTPUT);     // sets the digital pin as output
 6.  }
 7.
 8.  void loop()               // run over and over again
 9.  {
10.    blink(2000);  //call the code that makes the LED blink
11.  }
12.
13.  //turn the LED on
14.  void on(){
15.    digitalWrite(ledPin,HIGH); //set the pin HIGH and thus turn LED on
16.  }
17.
18.  //turn the LED off
19.  void off(){
20.    digitalWrite(ledPin,LOW); //set the pin LOW and thus turn LED off
21.  }
22.
23.  //make the LED blink
24.  void blink(int time){
25.    on();                         // sets the LED on
26.    delay(time/2);         // waits for a second
27.    off();                        // sets the LED off
28.    delay(time/2);         // waits for a second
29.  }
```

[Get Code] (http://playground.arduino.cc/Code/Library?action=sourceblock&num=4)

Download the source for this tutorial

The **next** step is to:

Learn c++ class (http://en.wikipedia.org/wiki/C%2B%2B_classes) syntax

There are a lot of resources on this topic, but generally the class are split into two files.

1. The declaration, referred to as the *header file*. 'LED13.h' will indicate that the file declared the class LED13. *Declaration* is the process of defining what the class should do.
2. The implementation, referred to as the *source file*. 'LED13.cpp' indicates that the file implement the declared functions and variables from "LED13.h". *Implementation* is the process of writing the code, that determines how the declared functions are imlemented.

LED13 Header

```
1.  #ifndef LED13_H
2.  #define LED13_H
3.
4.  #include <WProgram.h> //It is very important to remember this! note that if you are using Arduino
    1.0 IDE, change "WProgram.h" to "Arduino.h"
5.
6.  class LED13 {
7.  public:
8.       LED13();
9.       ~LED13();
10.      void on();
11.      void off();
12.      void blink(int time);
13. };
14.
15. #endif
```

[Get Code] (http://playground.arduino.cc/Code/Library?action=sourceblock&num=5)

'Code Analysis:'

- Lines 1 and 2 are referred to as an include guard (http://en.wikipedia.org/wiki/Include_guard) It basically prevents the code from being included into the program binary multiple times.
- Line 4 includes the WProgram code to our library. This enables us to use the pinMode, digitalWrite, delay etc.
- Line 6 is the beginning of the class itself.
- Line 7 uses the public (http://msdn.microsoft.com/en-us/library/hzk8a7d3.aspx) keyword
- Line 8 and 9 are called constructor and destructor, they are responsible for setting up your library (construct), and delete it (deconstruct)
- Line 10 declares the function 'void on();' , this will turn the LED on
- Line 11 declares the function 'void off();' , this will turn the LED off
- Line 12 declares the function 'void blink(int time);' , this will blink the LED in 'time' milliseconds.
- Line 13 is the end of the class.

- Line 15 ends the #ifndef (http://www.fredosaurus.com/notes-cpp/preprocessor/ifdef.html) preprocessor directive

LED13 Source

You will see that this source will be very much like the concept sketch.
However, I've changed some datatypes, and the rest is just the c++ syntax.

Notice the :: , it is called the 'scope resolution operator'

1. #include "LED13.h" //include the declaration for this class
2.
3. const byte LED_PIN = 13; //use the LED @ Arduino pin 13, this should not change so make it const (constant)
4.
5. //<<constructor>> setup the LED, make pin 13 an OUTPUT
6. LED13::LED13(){
7.   pinMode(LED_PIN, OUTPUT); //make that pin an OUTPUT
8. }
9.
10. //<<destructor>>
11. LED13::~LED13(){/*nothing to destruct*/}
12.
13. //turn the LED on
14. void LED13::on(){
15.   digitalWrite(LED_PIN,HIGH); //set the pin HIGH and thus turn LED on
16. }
17.
18. //turn the LED off
19. void LED13::off(){
20.   digitalWrite(LED_PIN,LOW); //set the pin LOW and thus turn LED off
21. }
22.
23. //blink the LED in a period equal to paramterer -time.
24. void LED13::blink(int time){
25.   on();   //turn LED on
26.   delay(time/2); //wait half of the wanted period
27.   off();   //turn LED off
28.   delay(time/2); //wait the last half of the wanted period
29. }

[Get Code] (http://playground.arduino.cc/Code/Library?action=sourceblock&num=6)

Source for this tutrial

Attach:LED13.zip (http://playground.arduino.cc/uploads/Code/LED13.zip)

Implementation

Selecting public, protected or private

A common 'rule of thumb' is to make all data as private as possible. For Arduino that will often result in all variables declared private, and most functions defined public.

Selecting datatypes / returntypes

Being the Arduino has limited resources I think selecting the 'smallest' variable is the way to go. An Arduino pin will (probably) never become above 255. So in that case using a byte (http://arduino.cc/en/Reference/Byte) is sufficient, and preferable.

Selecting variable- / functionnames

The arduino API has established some cues for how to name a variable or function. Start lower case, and indicate new words with an upper case letter. 'myDescriptiveVariable'

Writing examples

Be sure to create examples that are as self describing as possible. Additionally it should be made an effort to demo most of the functions.

Blink Example

```
1.  #include <LED13.h>
2.
3.  LED13 led;//initialize an instance of the class
4.
5.  void setup(){/*nothing to setup*/}
6.
7.  void loop(){
8.    led.blink(2000);//stay one second on, then a second off
9.  }
```

[Get Code] (http://playground.arduino.cc/Code/Library?action=sourceblock&num=7)

keywords.txt

Remember to implement this file, as it makes using your library a lot easier. Our library has four names that needs highlighting.
Those are:

- LED13

- on

- off

- blink

And they are implemented in **keywords.txt** as:

```
LED13    KEYWORD1
on       KEYWORD2
off      KEYWORD2
blink    KEYWORD2
```

Debugging

Remember #include <WProgram.h>

You will get a series of errors if you've forgot to include the WProgram, *and* trying to use some of the Arduino functions or datatypes.
It is only necessary to include the WProgram.h in you library .h, your .cpp will include it because it includes your header.

Delete object file and recompile between each alteration of header and/or source

When changes are made in the header or the source, those changes are not automatically incorporated in the object file of you library. To be sure that your changes are compiled and used, delete the LED13.o and recompile a sketch in the Arduino IDE.

NOTE: After making library changes, before recompiling, you may need to delete the <LibName>.o file which can be found by searching, or in a file similar to: c:\Documents and Settings\Administrator\Local Settings\Temp\build3998508668703718780.tmp

Learn to interpret the Arduino IDE debugger / compiler.

It can be a timesaver to take an hour and causing error deliberately, so you know what the error messages are caused by for later reference.

Sharing your library with other people

One of the fastest ways to flush out bugs in a library is to let other people use the library. Often people not only find bugs but also contribute patches to fix those bugs.

- (1) Post your library online somewhere, perhaps one of

  - here in the Arduino Playground

  - the Launchpad source code hosting using the Bazaar version control system.

  - the SourceForge source code hosting using Mercurial version control system

  - the Savannah source code hosting using Mercurial version control system

  - Github using the git version control system

  - the Google Code, Google Project Hosting using Mercurial version control system

  - Bitbucket

- your own website, using your own favorite version control system.
- (2) Then
    - Edit page LibraryList (http://playground.arduino.cc/Main/LibraryList) and add a link to your library.

Links

- Arduino Reference (http://arduino.cc/en/Reference/Extended)
- Core Libraries (http://arduino.cc/en/Reference/Libraries)
- Arduino Library Tutorial (http://arduino.cc/en/Hacking/LibraryTutorial)

Information about this page

Part of AlphaBeta (http://playground.arduino.cc/Profiles/AlphaBeta) Tutorials and Resources.
Last Modified: August 23, 2014, at 04:48 PM
By: DavidCary

Share