intel
Sponsors of Tomorrow.
ENGINEERS
EXCITING NEXT GENERATION INNOVATORS NEEDED EVERYWHERE ENABLING REAL WORLD SOLUTIONS
What will the future look like?
You decide.

Software Basics

Arduino

Robotics Week Instructor

Joy Shetler, PhD

Overview

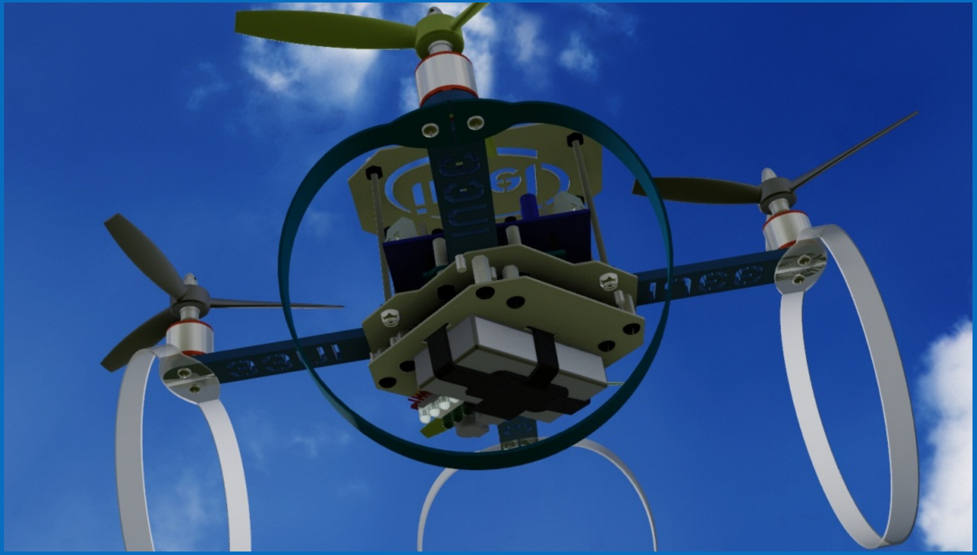
Microcontroller **Programming**

Arduino Software and Libraries

Installing Software

Arduino/C Language

Summary



The engine is the heart of an airplane, but the pilot is its soul. ~Walter Raleigh

Microcontroller Programming

Usually, programs must fit in the available on-chip memory. The "language" of computers is "0" and "1" or "off" and "on".

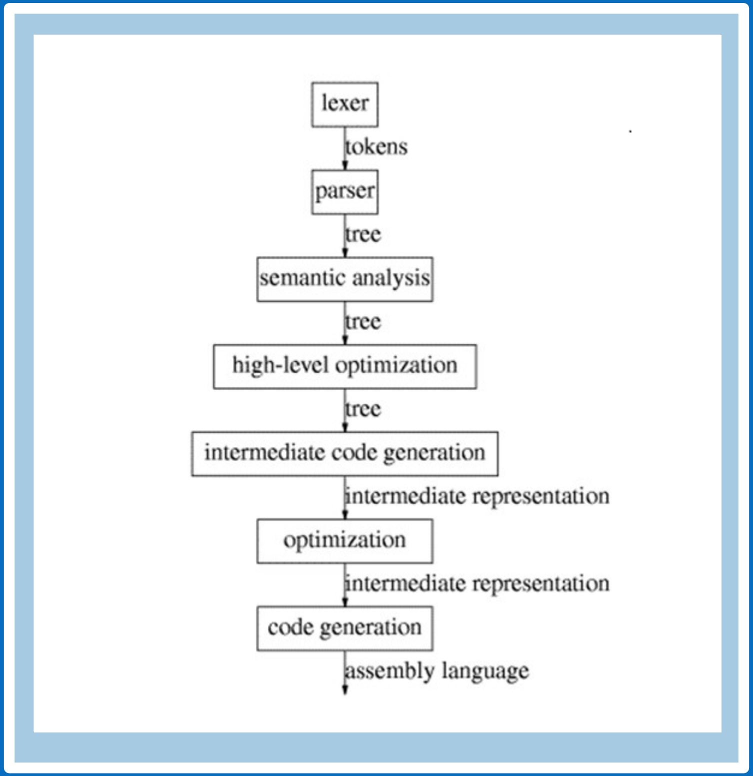
Compilers and Assemblers can be used to convert high-level code and/or assembler code into a compact machine code for storage.

Interpreters are often used to bridge the gap between a standard programming language and the actual hardware.

{Languages that don't directly support the hardware functions might use an interpreter.}

```
10010001101001101100100000100100110011111
01101100000100001111000011110100110010110
00001101001111010010011111100111000001101
10111110011000001110000110110011001011100
00111100111110101111001011001011000001110
10011011111000001110011111000011001011100
00111010111000001110100110111110000011110
01110111111101011000001110100110111111001
00110000111110011011101000001001001100000
11010001101111111000011001011000001111001
11011111110101100000110010111011101101010
11011111111001100000110100111101001000010
```

Microcontroller Programming



Compiler – Takes input in a high-level language and produces output in assembly language (or machine code).

Assembly Language

Most basic programming language

Lacks High-level conveniences

Operates on the physical CPU

Not portable between families of processors.

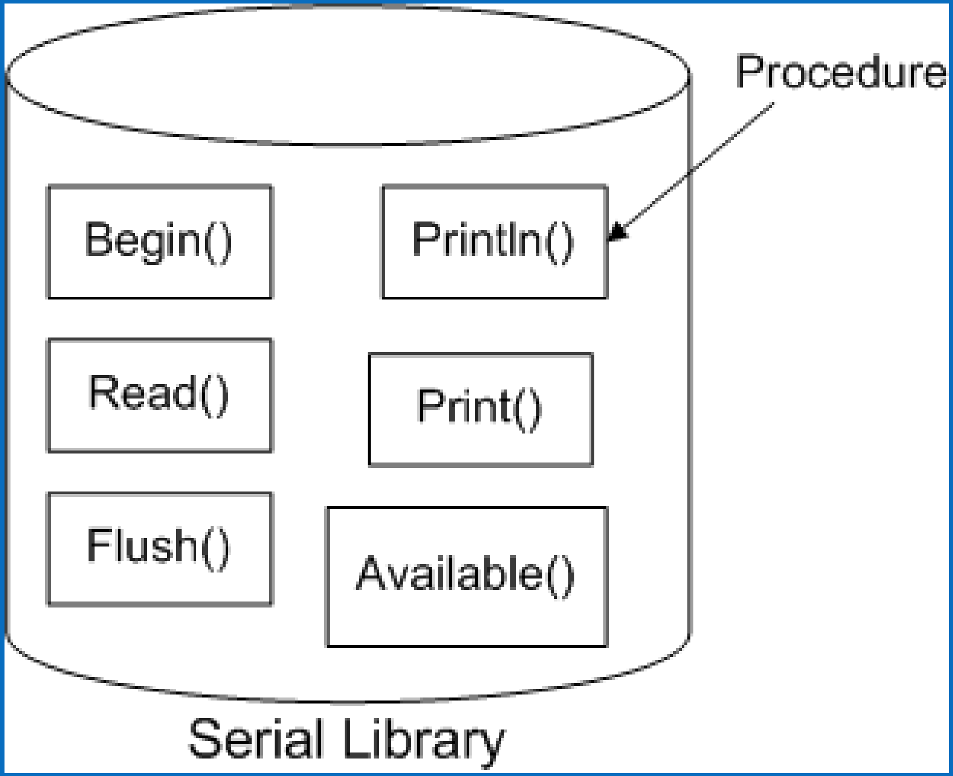
Assembler – Takes the assembly language and converts it to machine code.

Arduino Software

Core Functions

– Simple Programs that demonstrate basic Arduino Commands.

- Example functions are available on the Arduino website.



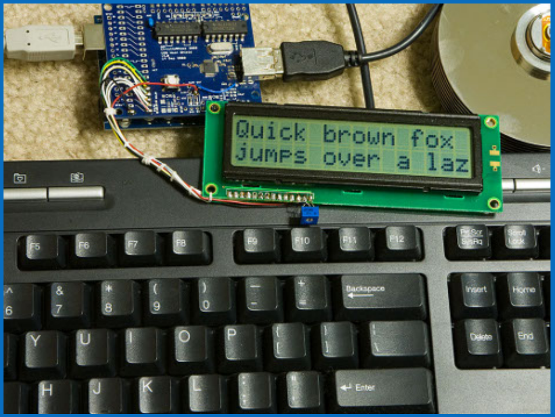
Libraries

These "procedures" can also be a starting point for creating a new "procedure" or "function".

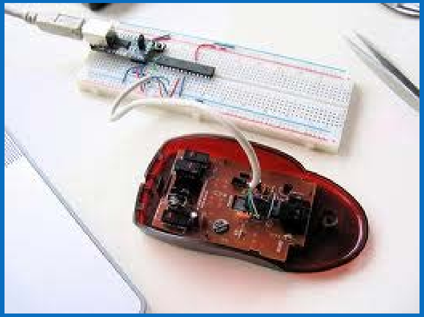
[More on this later…]

Arduino Core Functions

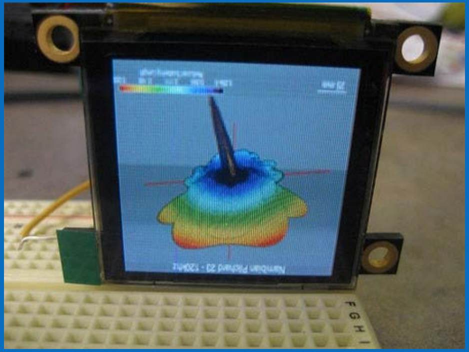
Basics – Getting familiar with the Arduino – some digital, some analog, some blinking, some basic *stuff*



Keyboard- keyboard operations



Mouse – controls cursor movement



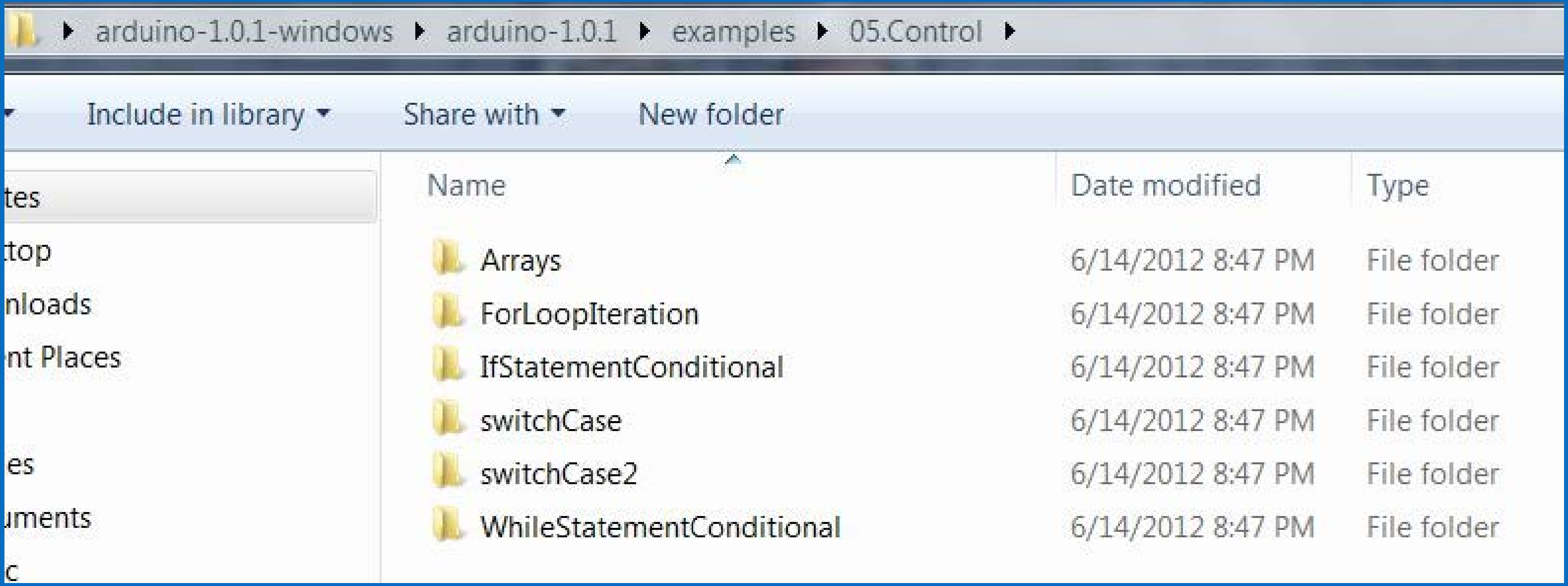
Display

Arduino Core Functions

Programming Core Functions

Control Structures –

IF, For Loop, While Loop, Cases



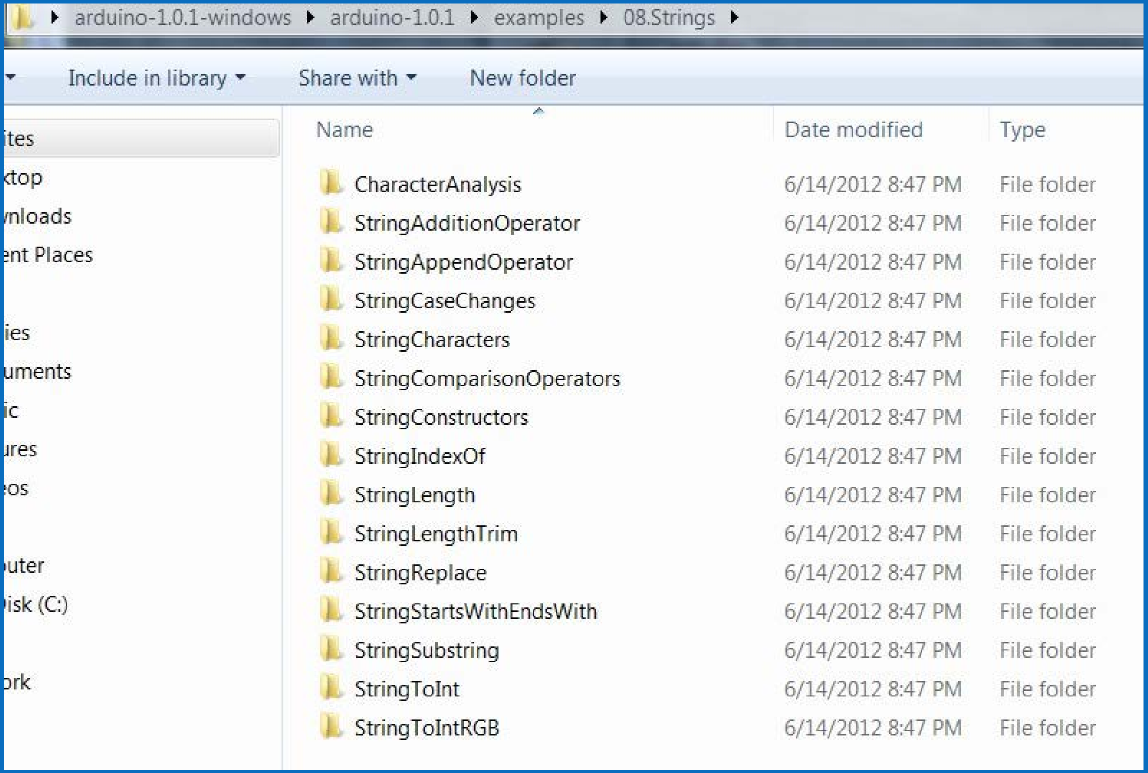
Name	Date modified	Type
Arrays	6/14/2012 8:47 PM	File folder
ForLoopIteration	6/14/2012 8:47 PM	File folder
IfStatementConditional	6/14/2012 8:47 PM	File folder
switchCase	6/14/2012 8:47 PM	File folder
switchCase2	6/14/2012 8:47 PM	File folder
WhileStatementConditional	6/14/2012 8:47 PM	File folder

Look at the code in the arduino-1.0.1 /examples/0.5Control folder!

Arduino Core Functions

Programming Core Functions

Strings – working with characters, text, phrases, etc.



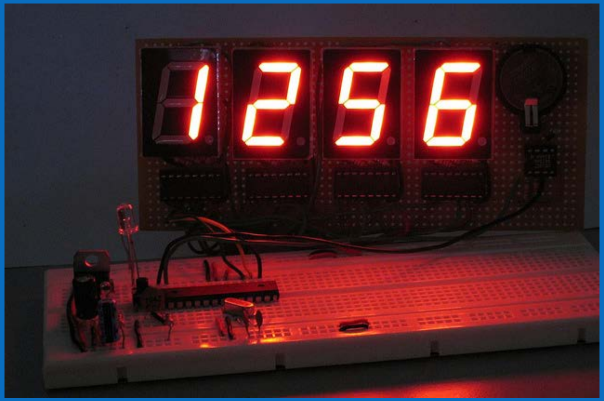
Name	Date modified	Type
CharacterAnalysis	6/14/2012 8:47 PM	File folder
StringAdditionOperator	6/14/2012 8:47 PM	File folder
StringAppendOperator	6/14/2012 8:47 PM	File folder
StringCaseChanges	6/14/2012 8:47 PM	File folder
StringCharacters	6/14/2012 8:47 PM	File folder
StringComparisonOperators	6/14/2012 8:47 PM	File folder
StringConstructors	6/14/2012 8:47 PM	File folder
StringIndexOf	6/14/2012 8:47 PM	File folder
StringLength	6/14/2012 8:47 PM	File folder
StringLengthTrim	6/14/2012 8:47 PM	File folder
StringReplace	6/14/2012 8:47 PM	File folder
StringStartsWithEndsWith	6/14/2012 8:47 PM	File folder
StringSubstring	6/14/2012 8:47 PM	File folder
StringToInt	6/14/2012 8:47 PM	File folder
StringToIntRGB	6/14/2012 8:47 PM	File folder

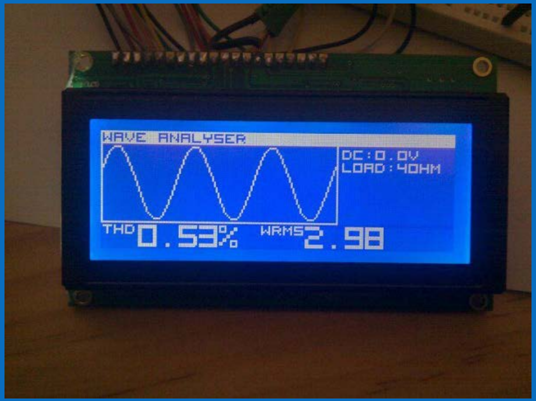
Pick a Strings Example folder and check out the contents and a file!

Arduino Core Functions

Functions which support basic “hardware” operations.

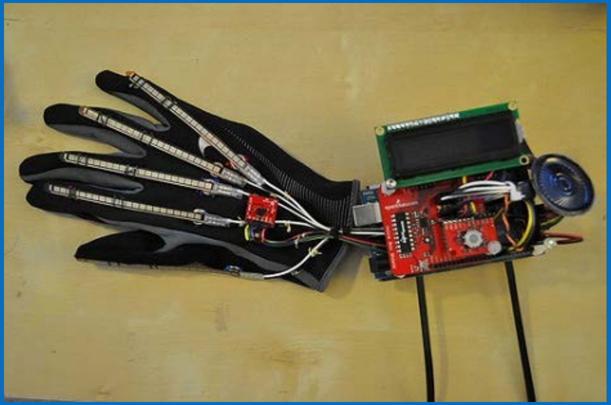
Digital





Analog

Communication



Sensors

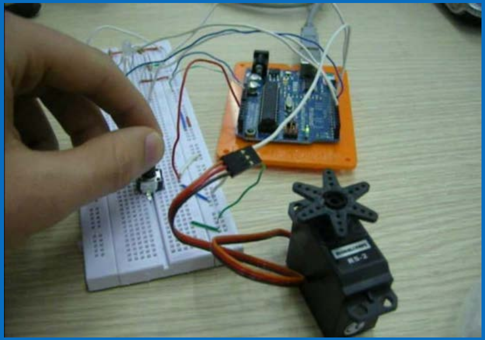
Arduino Libraries

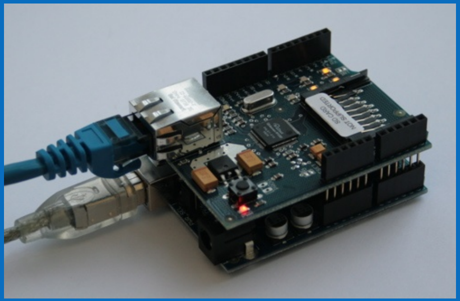
Libraries – collection of “procedures” or “functions” which are all related.

Procedures – “list of things to do”

Sketches often depend on libraries.

Servo

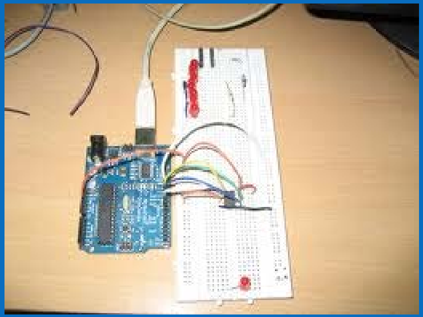




Ethernet

EEPROM
and

SPI



Liquid Crystal

Arduino Libraries

Suppose you want to control a “motor” (or “rotor” on a quadcopter??)?

This might take a lot of work and time to learn all about how to code for this type of application!

Maybe someone has already written the code and you don’t have to write it yourself!?!

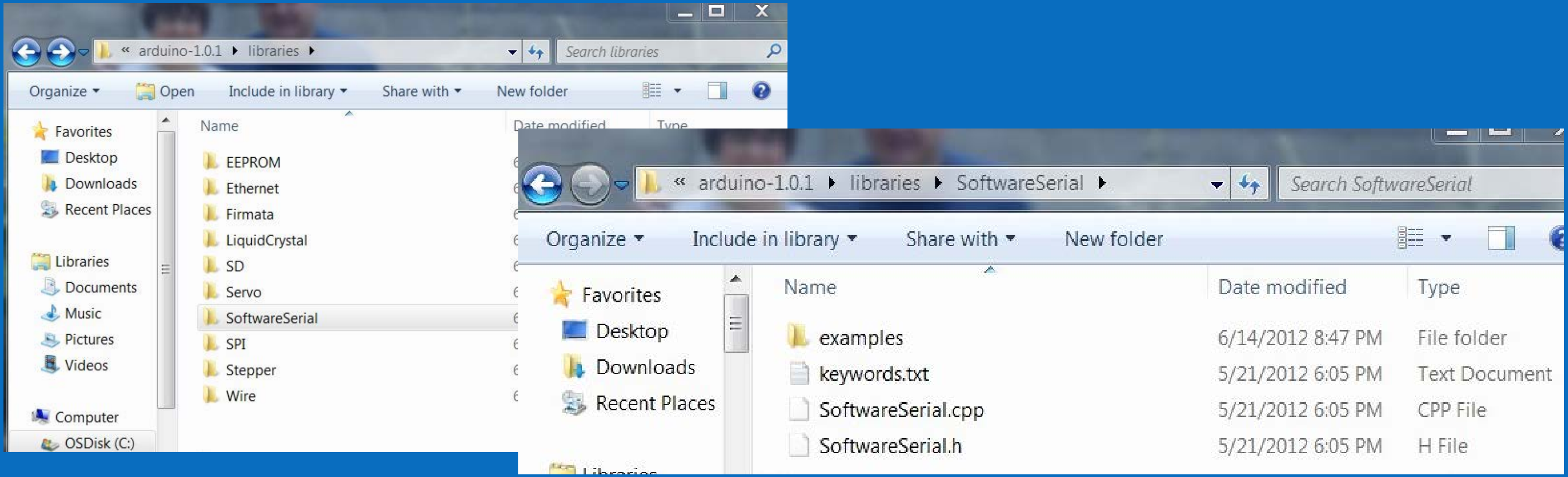
Or you can take their code and modify it to do what you want to do!

There are several contributed libraries available, besides those available in the “core distribution”. These have been contributed by other users.

And, maybe, you come up with some clever application and write your own library to add to the collection!

Arduino Libraries

What’s in a library?



Name
- EEPROM
- Ethernet
- Firmata
- LiquidCrystal
- SD
- Servo
- SoftwareSerial
- SPI
- Stepper
- Wire

Name	Date modified	Type
examples	6/14/2012 8:47 PM	File folder
keywords.txt	5/21/2012 6:05 PM	Text Document
SoftwareSerial.cpp	5/21/2012 6:05 PM	CPP File
SoftwareSerial.h	5/21/2012 6:05 PM	H File

Folder which contains files.

Most files will end in .cpp (C++ code file) and .h (C++ header file)

Files may also end in .o (which means these files are C++ compiled Objects).

Note: If you modify a library, you may need to delete the .o files, to force the Arduino IDE to recompile the modified .cpp files into fresh .o files.

Installing Software

Lots of online tutorials are available.

Windows

http://arduino.cc/en/Guide/Windows

Mac OS X

http://arduino.cc/hu/Guide/MacOSX

Linux

http://www.arduino.cc/playground/Learning/Linux

- ArchLinux
- Debian
- Fedora
- Gentoo
- openSUSE
- Puppy
- Pussy
- Slackware
- Ubuntu
- Xandros (Debian derivative) on Asus Eee PC
- All, "the hard way"

Want to use a different IDE?

http://www.wikihow.com/Write-Arduino-Software-in-C

Arduino Development Environment

Connects to the Arduino hardware to upload programs and communicate with them.

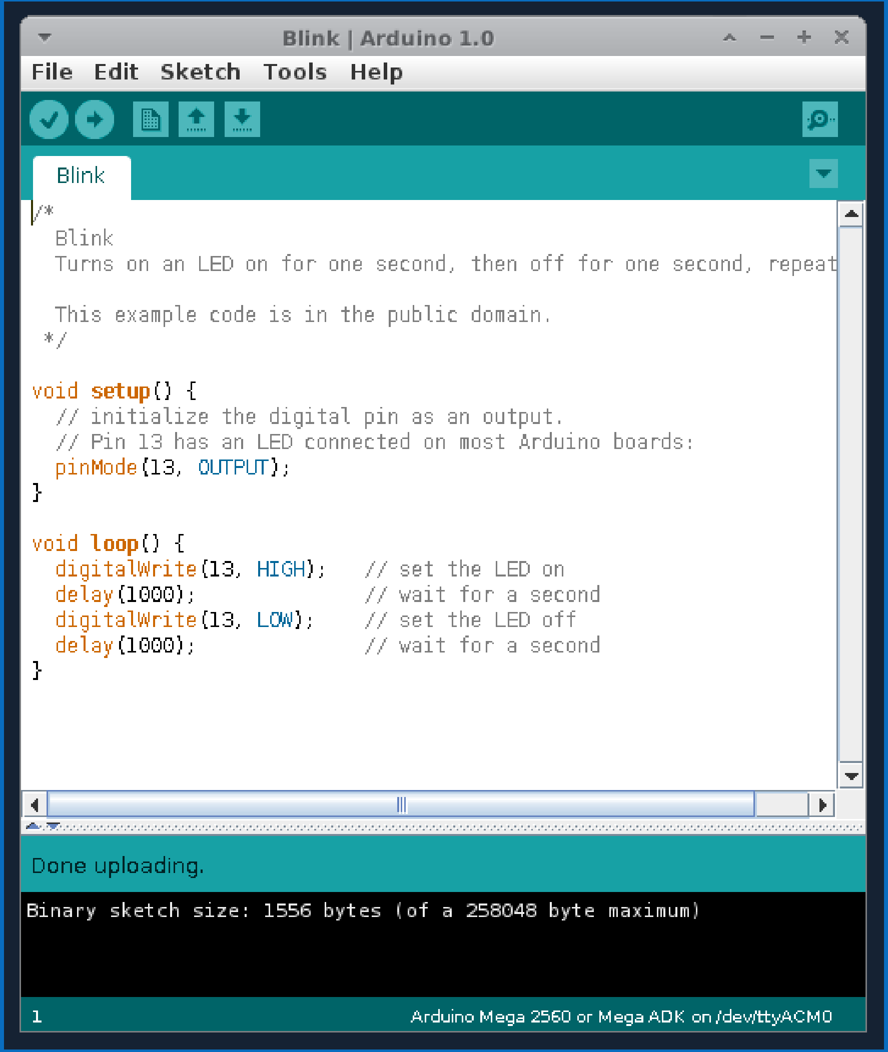
text editor - for writing code

message area - gives feedback while saving and exporting and also displays errors.

text console - displays text output by the Arduino environment including complete error messages and other information

toolbar - buttons for common functions > verify and upload programs, create, open, and save sketches, and open the serial monitor.

a series of menus



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeat

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

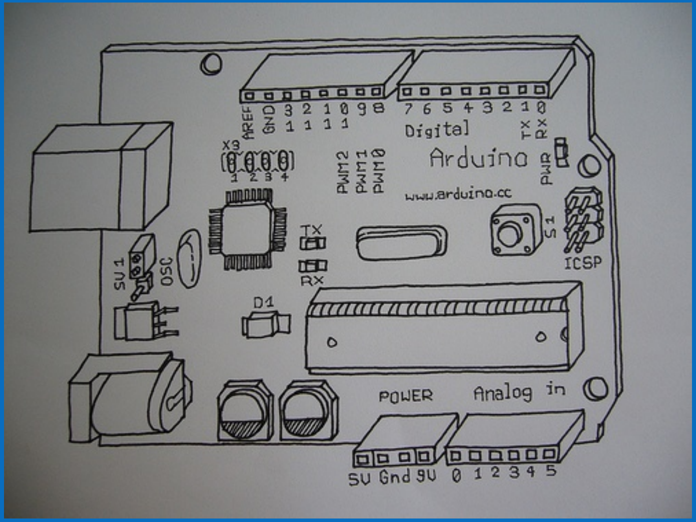
Done uploading.

```
Binary sketch size: 1556 bytes (of a 258048 byte maximum)
```

Arduino Development Environment

Arduino Sketches

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.

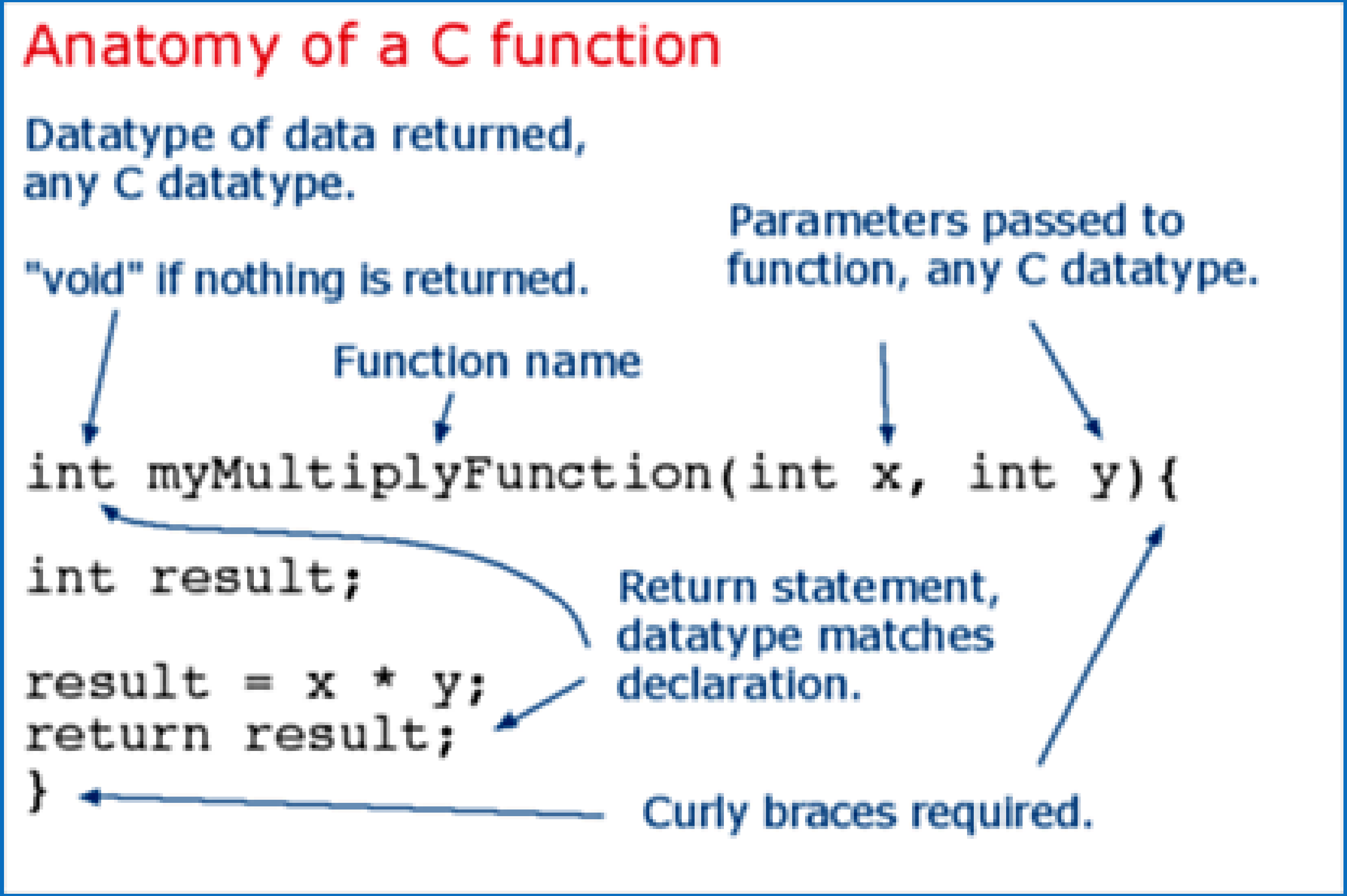


Sketchbook

A standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar.

The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog.

Arduino Programming or C Language (C++, C#, etc)



Anatomy of a C function

Datatype of data returned, any C datatype.

"void" if nothing is returned.

Function name

Parameters passed to function, any C datatype.

```
int myMultiplyFunction(int x, int y){

int result;

result = x * y;
return result;
}
```

Return statement, datatype matches declaration.

Curly braces required.

Arduino Programming
or C Language (C++, C#, etc)

Comments

Two styles

// Anything after these two slashes! Single line style

/* Anything between these two sets of symbols,

Multiple line style */

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.

Arduino Programming
or C Language (C++, C#, etc)

Structure

Functions – (also known as a procedure or sub-routine) is a named piece of code that can be used elsewhere in a sketch.

An Arduino sketch runs in two parts:

void setup()

The initialization code or sequence goes in this section of code. This code is executed once after the board is powered on or after the reset button is pressed.

void loop()

The main code of your sketch goes here. The set of instructions in this section get repeated over and over again, *until the board is switched off.*

Note – in many languages, we try to avoid "infinite loops" but in this case, we need to keep executing something on the CPU (even if it's a NOP or No Operation, which you need to put in the loop!).

Arduino Programming or C Language (C++, C#, etc)

Constants -Constants are predefined variables in the Arduino language. They are used to make the programs easier to read.

false and true, typed in lower case

HIGH, LOW, INPUT, INPUT_PULLUP, OUTPUT, typed in upper case

Take a look at the reference manual or on-line webpage (http://arduino.cc/en/Reference/Constants) for further clarification.

Variables – place for storing a piece of data, has a name, a type and a value. Often this value will change when the sketch runs.

Datatypes that are available include:

Boolean, char, byte, int, unsigned int, long, unsigned long, float, double, string, array

Arduino Programming or C Language (C++, C#, etc)

The reference manual contains a number of other data structures that you can use to write code. You can look at these descriptions in greater detail “on your own”.

Control Stuctures for controlling the logical flow of your sketch.

Arithmetic and Formulas to do complex calculations.

Comparison Operators, Boolean Operator, Compound Operators

Input and Output functions for handling “input and output”!

Time Functions for measuring elapsed time and for pausing the sketch.

Math functions that includes many common mathematical and trigonometric functions.

Random Number functions that use Arduino’s pseudo-random number generator.

Serial Communication functions to communicate with devices over the USB port using a serial communications protocol.

Start Coding!

At this point, you should have a basic idea of the "software" part of the Arduino microcontroller.

The best way to "learn" how to code, is to start coding.

The project will give you an opportunity to develop your own code.

References and Videos!!

There's a LOT of references used for the myQuad presentations and projects. The list is included in the document myQuadReferences.

This also includes a number of youtube videos and other resources available for inquiring minds!

Questions?

Project 3

Writing (or modifying) a new LED Control program.

Follow the instructions for Project 3.

Remember, you may need to use code from the previous projects to get this project to work!

Project 3

Writing (or modifying) a new LED Control program.

Congratulations!

You’ve now completed your third project and are an Arduino programmer!

intel®

Leap ahead™