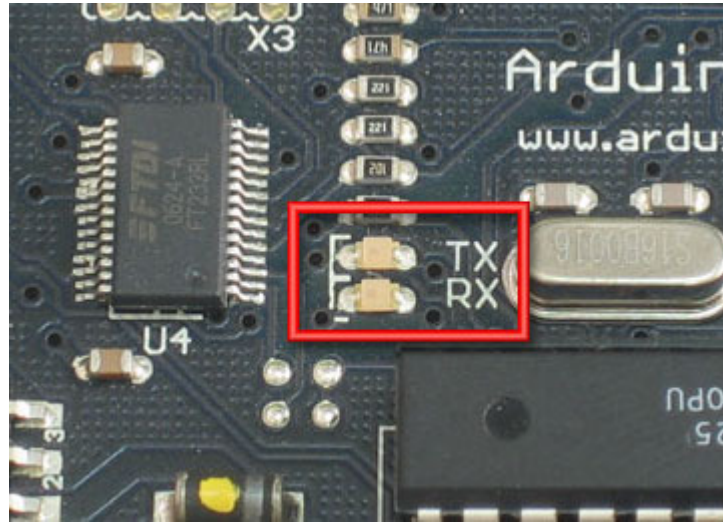


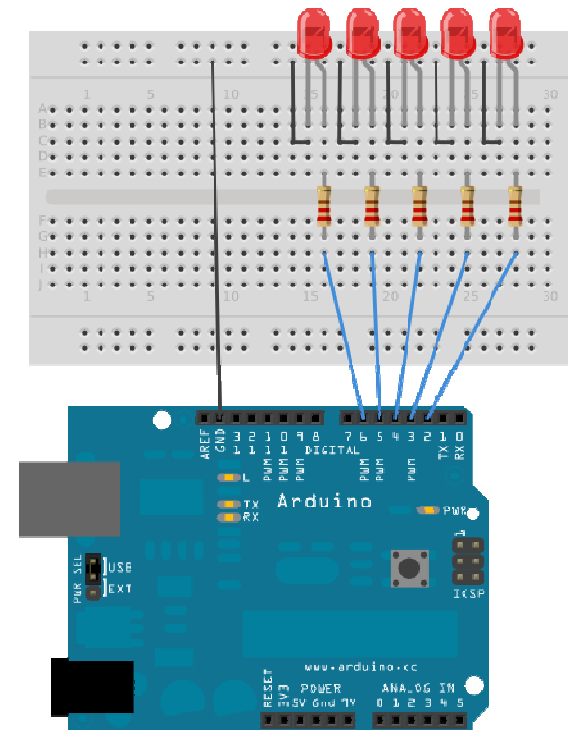


Serial Communication

Quick Recap

Exercises we did...

- Made Arduino say “Hello World!”
- Serial.print / Serial.println
- Serial.read
- Serial.write



Custom Functions

Anatomy of a C function

Datatype of data returned,
any C datatype.

"void" if nothing is returned.

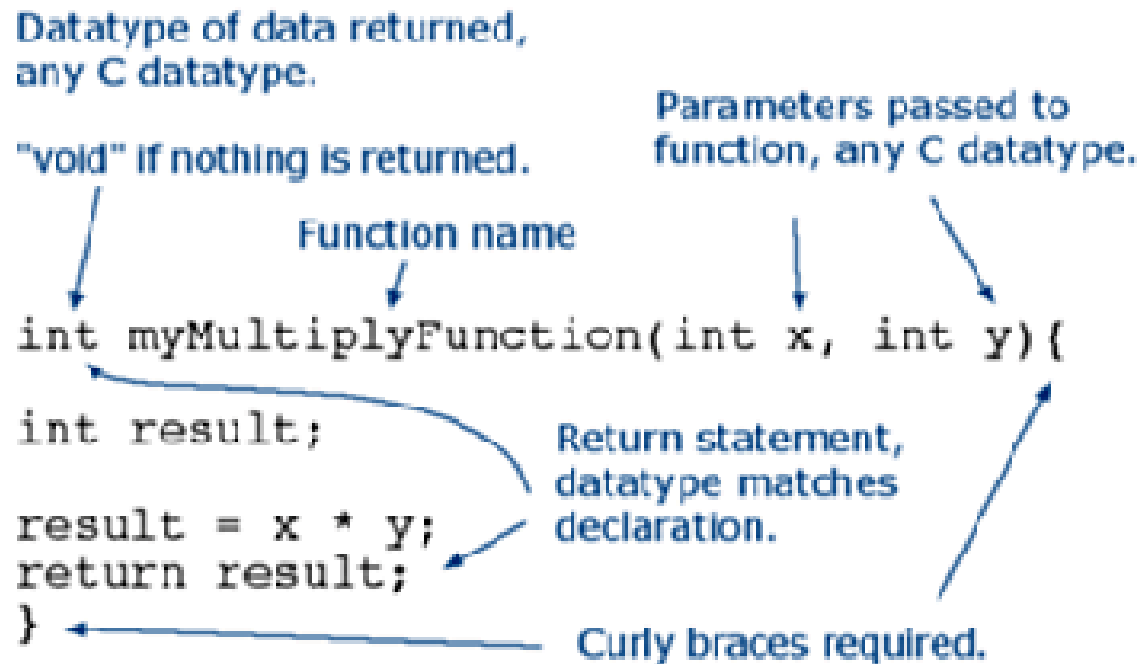
Function name

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.

The diagram illustrates the components of a C function. Annotations with arrows point to specific parts of the code: 'Datatype of data returned, any C datatype.' points to 'int' at the start of the first line. '"void" if nothing is returned.' points to the same 'int'. 'Function name' points to 'myMultiplyFunction'. 'Parameters passed to function, any C datatype.' points to 'int x, int y'. 'Return statement, datatype matches declaration.' points to 'return result;'. 'Curly braces required.' points to the opening and closing curly braces of the function body.

Custom Functions

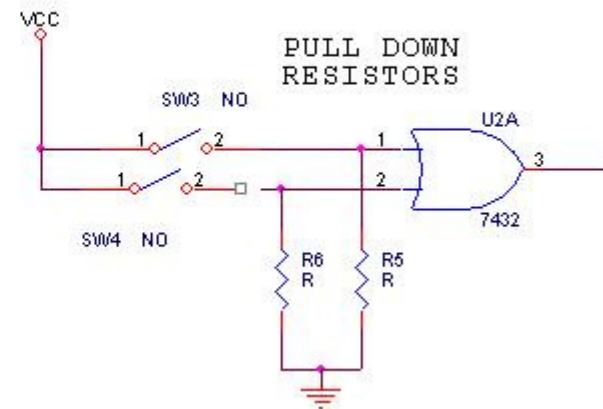
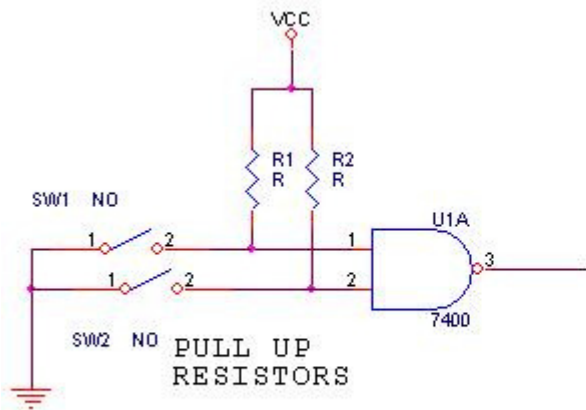
```
int blinkno = 10;  
int ledPin = 13;
```

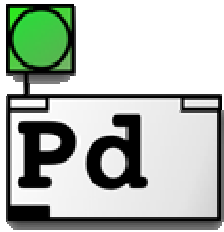
```
void setup() {  
  pinMode(ledPin, OUTPUT);  
}
```

```
void loop()  
{  
  blink(blinkno, 100, 500);  
}
```

```
void blink(int x, int y, int z){  
  for (int i = 0; i < x; i++){  
    digitalWrite(ledPin, HIGH);  
    delay(y);  
    digitalWrite(ledPin, LOW);  
    delay(z);  
  }  
  blinkno = 0;  
}
```

One more Thing:





Alternate Development Environments

Processing

Flash

PD

MAX/MSP

Serial out from Arduino

- `Serial.print(x, BYTE);`
- `Serial.println (x, BYTE);`
- `Serial.write(x);`

Processing and Arduino

Arduino

Based on Wiring
Implements C/C++

Coding :

```
Serial.println("hello world");  
  Serial.print("i = ");  
  Serial.print(i);  
  Serial.println();
```

```
int foo[] = { 0, 1, 2 };
```

Processing

Based on Visualization
Implements Java

```
println("hello world");  
println("i = " + i);
```

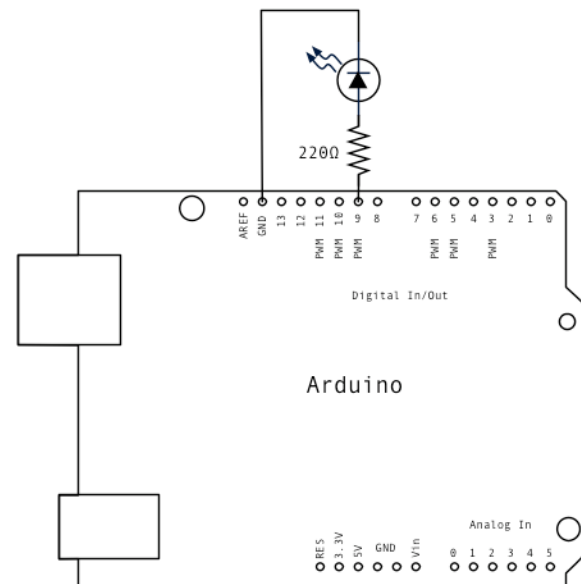
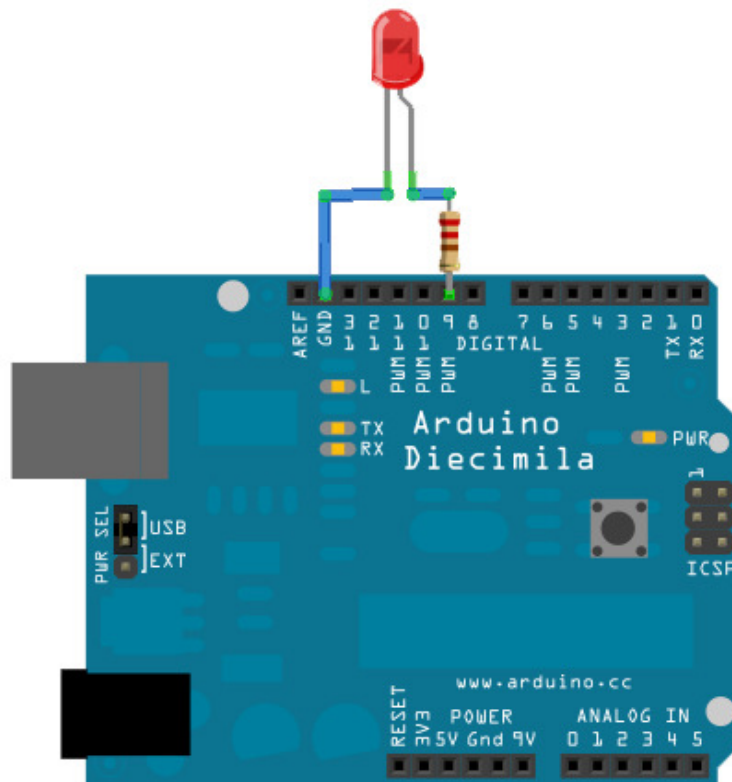
```
int foo[] = { 0, 1, 2 };
```

or

```
int[] foo = { 0, 1, 2 };
```

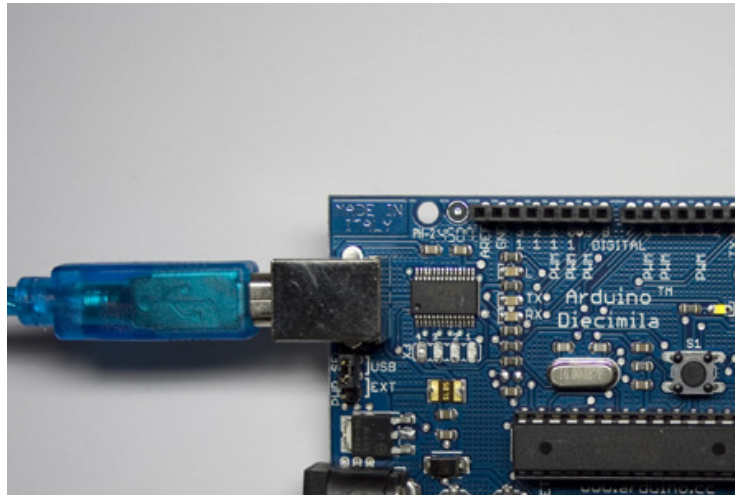

Visual Dimmer

<http://www.arduino.cc/en/Tutorial/Dimmer>



Arduino TX

- Arduino's signals : serial port : Serial Monitor
- Arduino's signals : serial port : Firmata



Firmata

- It's a protocol for communicating with Arduino / microcontrollers from software on a host computer
- Software which is capable of reading / sending data from/to serial port

<http://arduino.cc/en/Reference/Firmata>

Processing with Arduino library

- <http://www.arduino.cc/playground/Interfacing/Processing>
- Download library and copy paste into “.\processing-1.0.7\libraries” and start using processing as arduino.

Processing Language and Arduino Board

```
/****** Processing language:
attach an analog input to port 1, like potentiometer
and 2 LED's to ports 5 and 8 on arduino
*/
// import serial and arduino classes for communication with arduino
import processing.serial.*;
import cc.arduino.*;

// declare a arduino variable,

Arduino arduino;

void setup() {
  size(512, 200);

  // here we are instantiating the class Arduino into an object, now all
  // the properties of arduino are available to us from this object
  // the object uses this (our processing applet), and the bit rate of the serial
  // communication as variables for the constructor. Make sure the bitrate matches the bitrate
  // in the arduino application
  arduino = new Arduino(this, Arduino.list()[0], 115200);
}

void draw() {
  background(constrain(mouseX / 2, 0, 255));

  //we are reading the analog input from input 1 - our potentiometer and printing out from the serial below
  println(arduino.analogRead(1));

  //these two calls basically replicate the analogWrite to pins 5,8 on Arduino, using values from your mouseX data in Processing for the voltage value from 0-255
  //constrain function, limits the values from the first variable, (mouseX/2), to second and third as ranges. (0,255).
  //basically everything is almost the same, we just access these calls from the arduino object now
  //by using arduino."yourarduinocall"
  arduino.analogWrite(5, constrain(mouseX / 2, 0, 255));
  arduino.analogWrite(8, constrain(255 - mouseX / 2, 0, 255));
}
```

Other Softwares

- Flash, PD, Max/MSP:
- [http://file-error.net/1o1o1o1o1/?Physical Computing and Interaction:Arduino:Arduino VS Flash](http://file-error.net/1o1o1o1o1/?Physical+Computing+and+Interaction:Arduino:Arduino+VS+Flash)