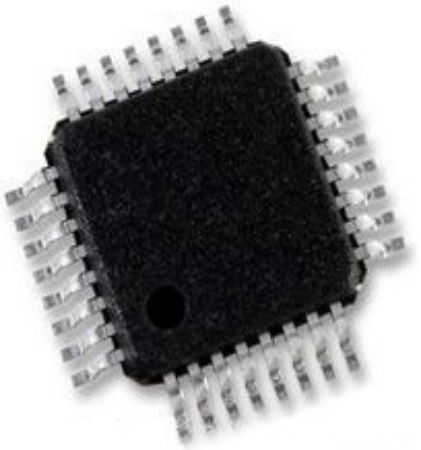
# Creative Multimedia Using Free Software and Open Source Hardware

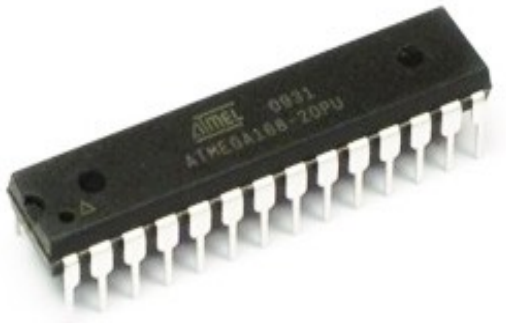
Processing, Pure Data,
Arduino and Raspberry Pi

Dana Moser
dmoser@massart.edu

http://www.curiousart.org/OSCON

# What's the difference between a microprocessor and a microcontroller?



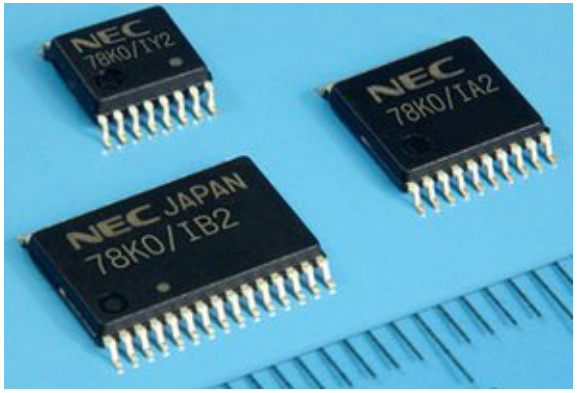
A *microprocessor:*

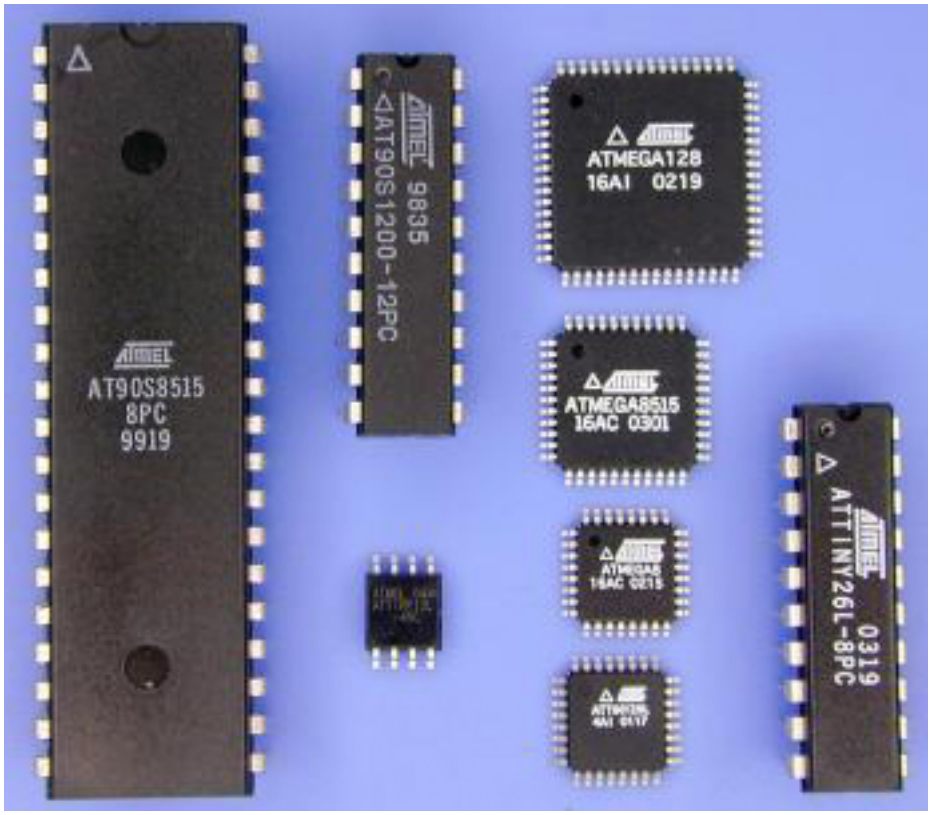
- an IC with only the Central Processing Unit (CPU)
- No RAM, ROM, or peripheral I/O on the chip. (System designers must add these externally to make them function in Desktop PC's, Laptops, notepads, tablets, etc.
- (Manufacturers include: Inte'sl Pentium, core 2 duo, i3, i5, ARM, PowerPC, AMD, etc. )

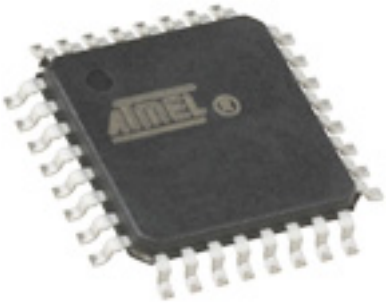
A *microcontroller:*

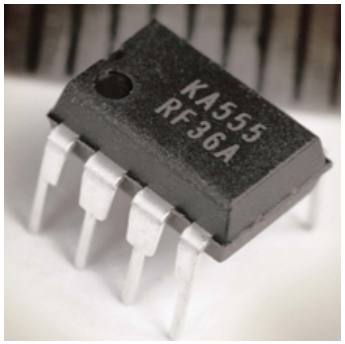
- an IC with CPU, I/O pins, a fixed amount of RAM, ROM, all embedded on a single, 'all in one' chip.
- (Manufacturers include: Microchip, ATMEL, TI, Freescale, Philips, Motorola)

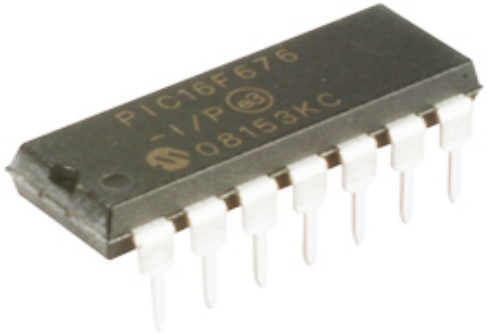
# There are many different microcontroller ICs...

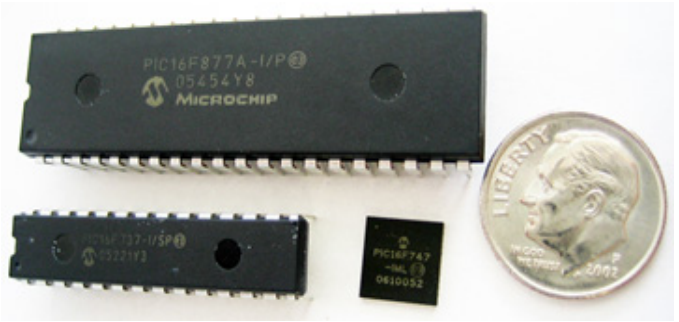


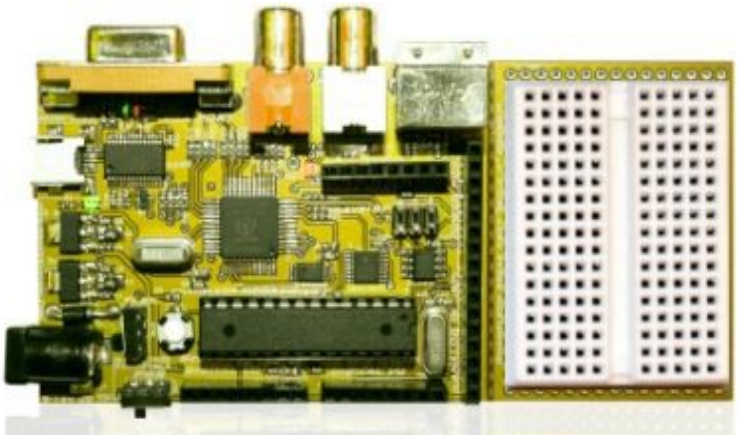



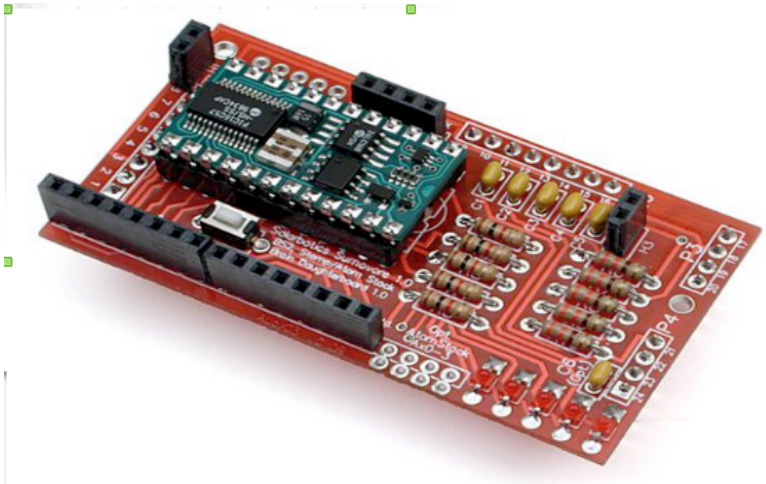


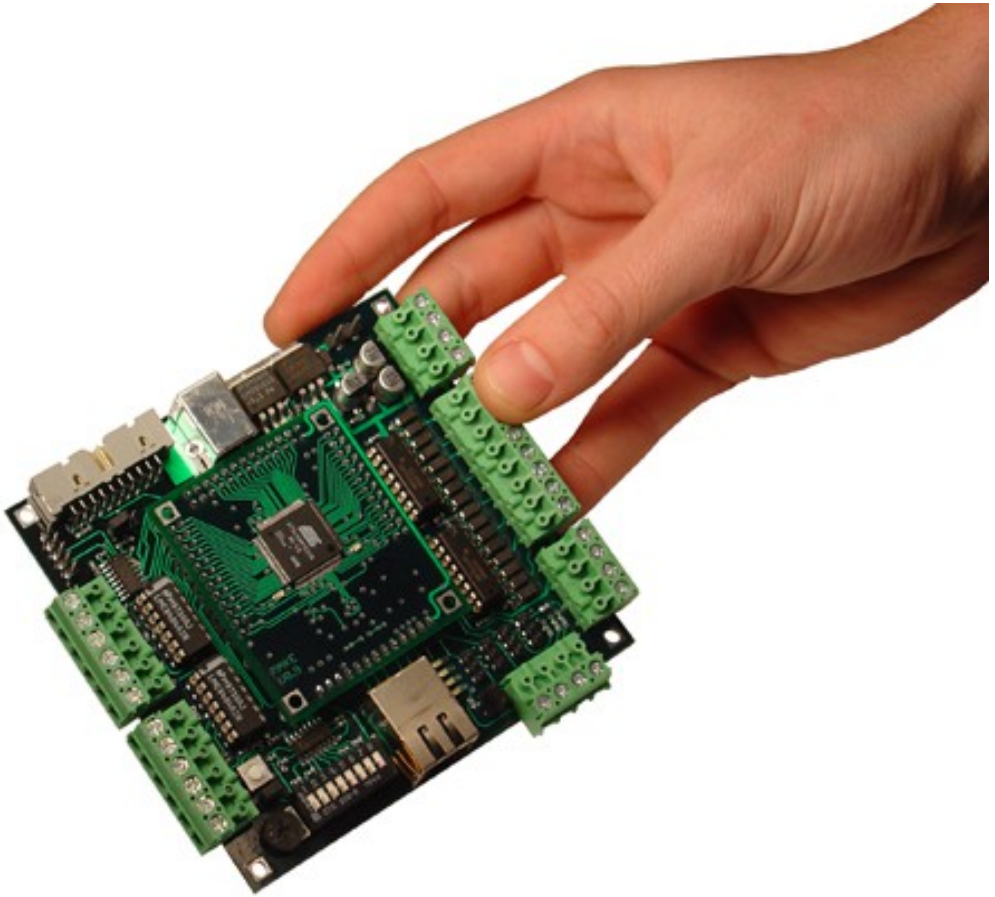



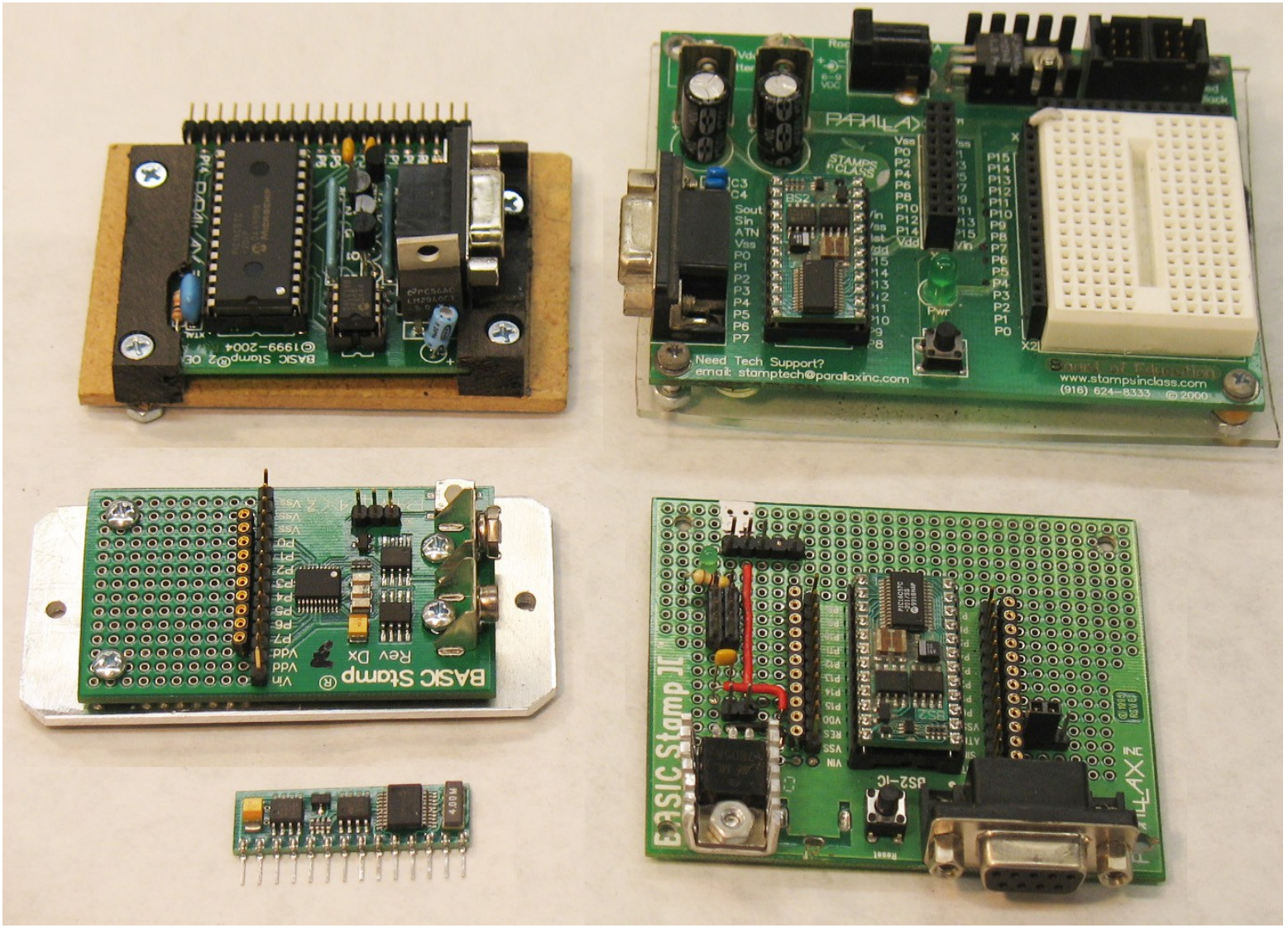
Many are available on handy “development boards”.



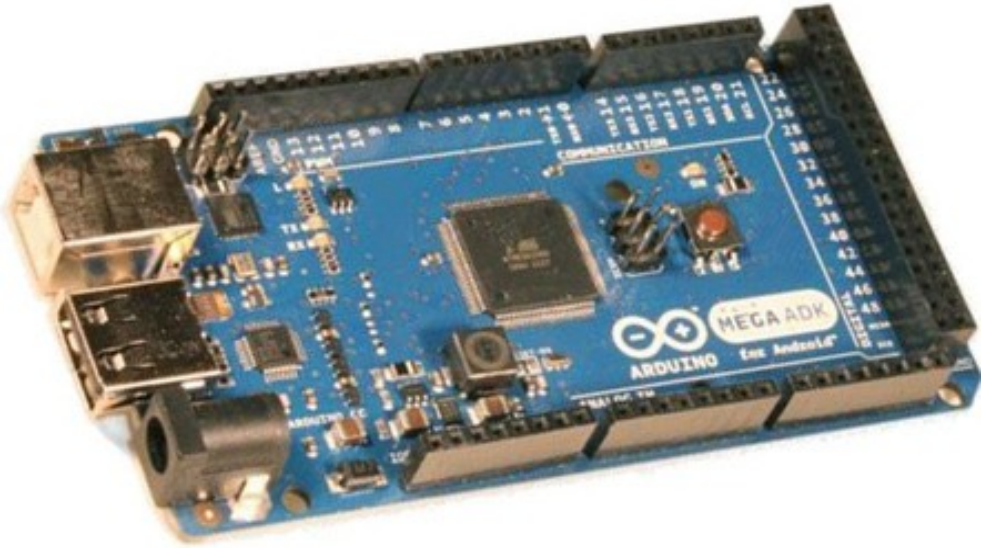


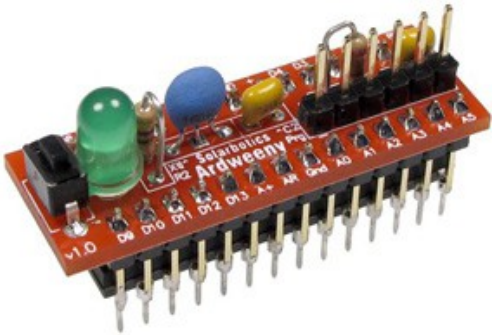


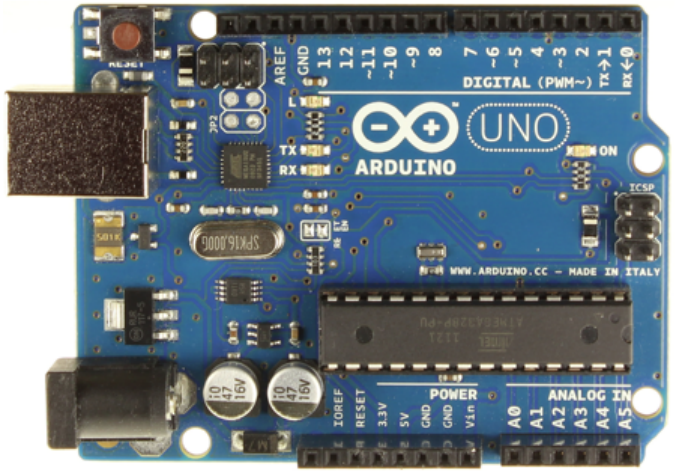
# Parallax “Basic Stamp” :

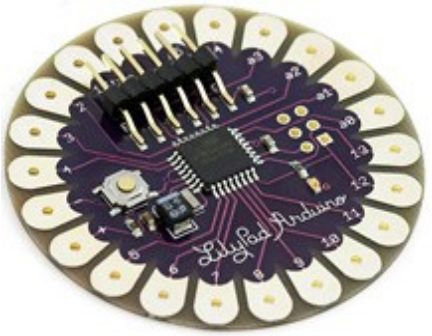


# Arduino boards...



The *Arduino* is a development platform using microcontrollers from Atmel's ATmega series.

The UNO for example, uses the ATmega 328:

- 28 pin, 8 bit IC
- Runs at 16 MHz (but chip can run at 20 MHz).
- 32K Bytes of flash memory, 2K SRAM
- 20 I/O pins: 14 digital, 6 analog (10 bit ADC)
- regulated at 5 and 3.3 volts

# Requirements for the microcontroller:

## 1. Power

- It's an electrical component, so of course you have to give it power. But like many ICs, the voltage used to operate it needs to be controlled relatively precisely.

## 2. I/O

- Input and output, some way to communicate with the chip. This is generally done through some kind of connection to the chip's pins. Breadboards are handy.

## 3. Programming Interface

- Some way to write programs and download them to the chip and run them.

# Requirements for the microcontroller:

## 1. Power:

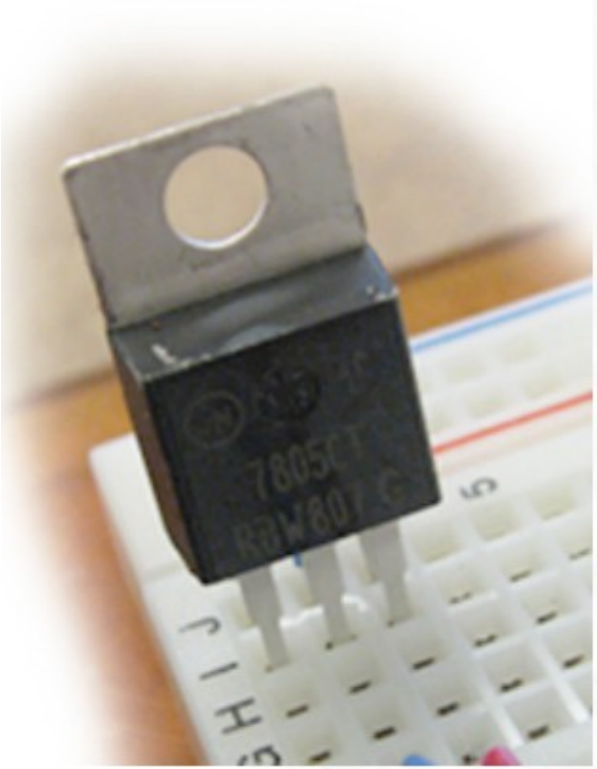
There are three commonly used ways for controlling the voltage supplied to microcontrollers:

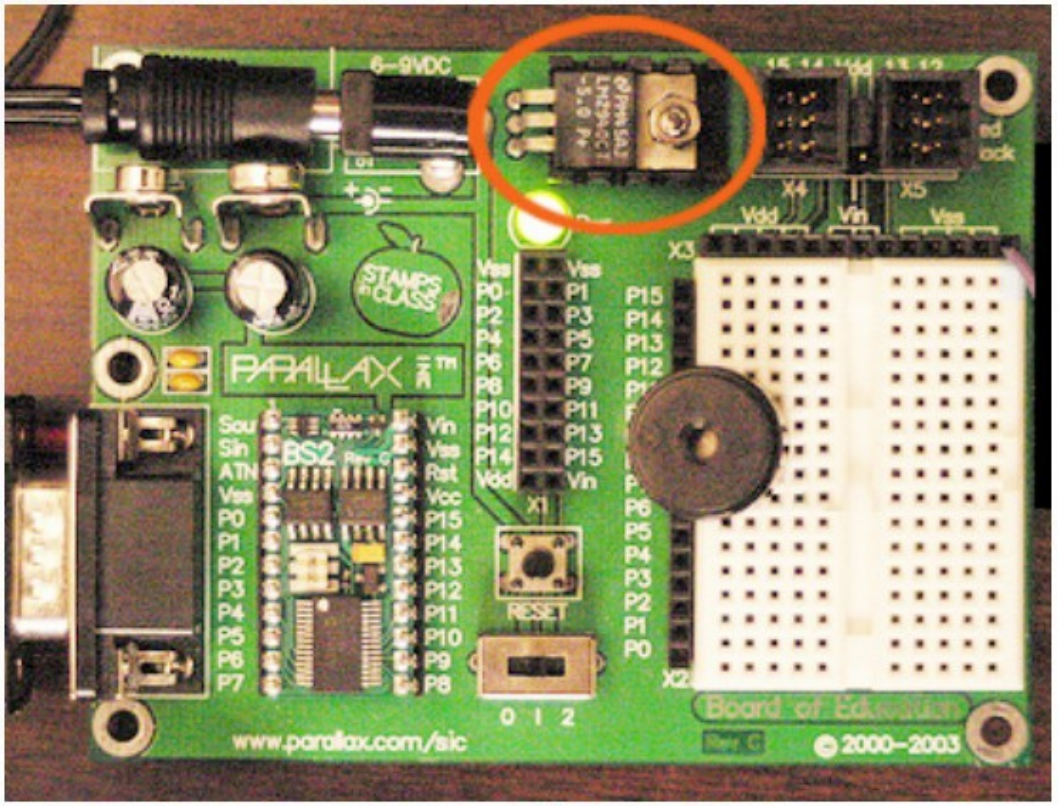
I. A voltage regulator

II. A regulated power supply

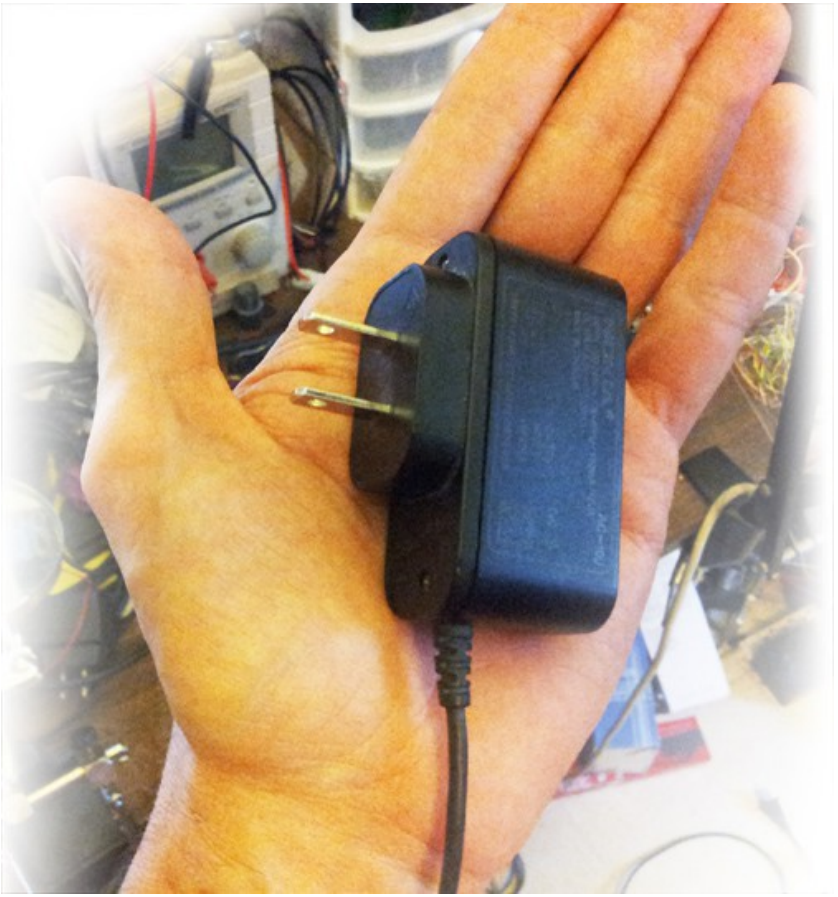
III. Battery power

# Voltage regulator

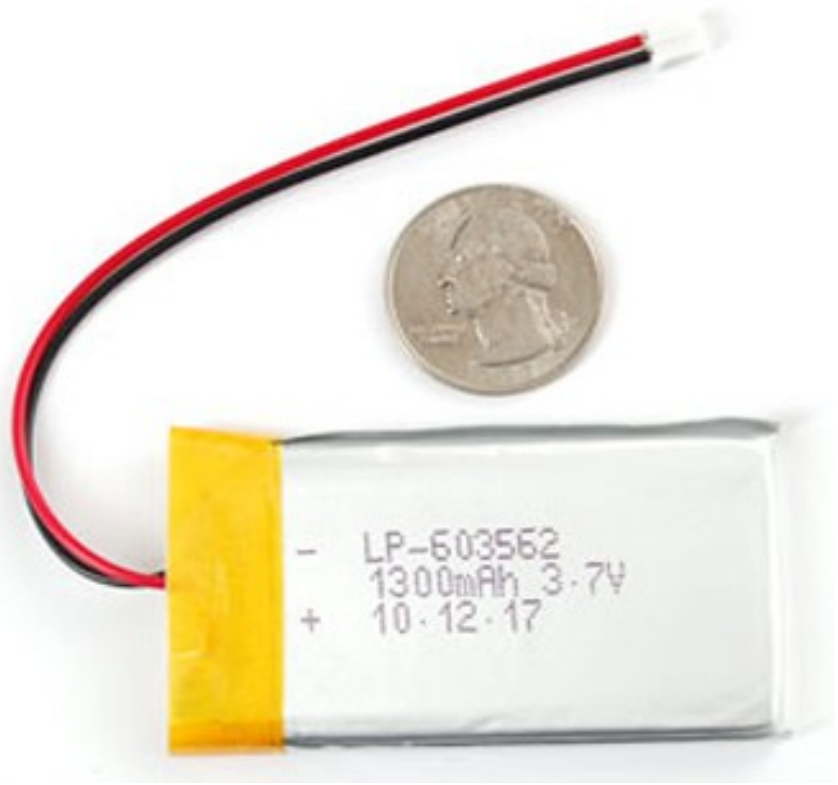


# Regulated power supply

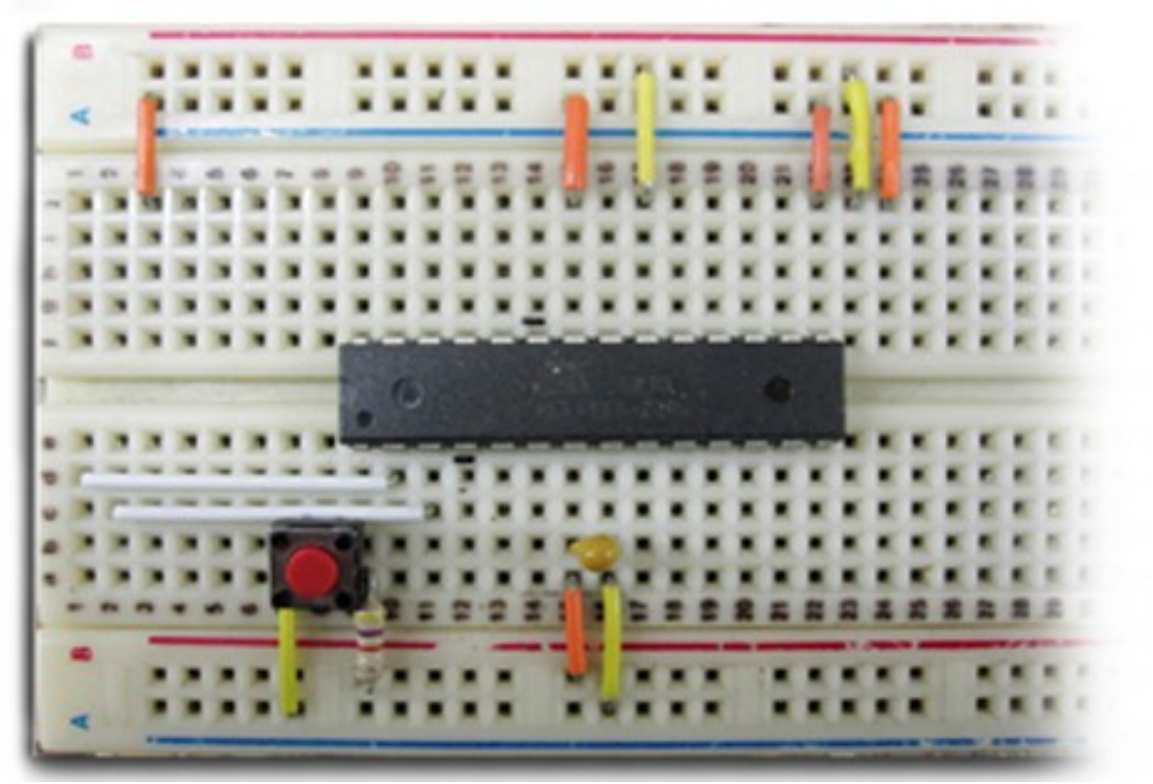
# Battery pack

# Requirements for the microcontroller:

## 2. I/O:

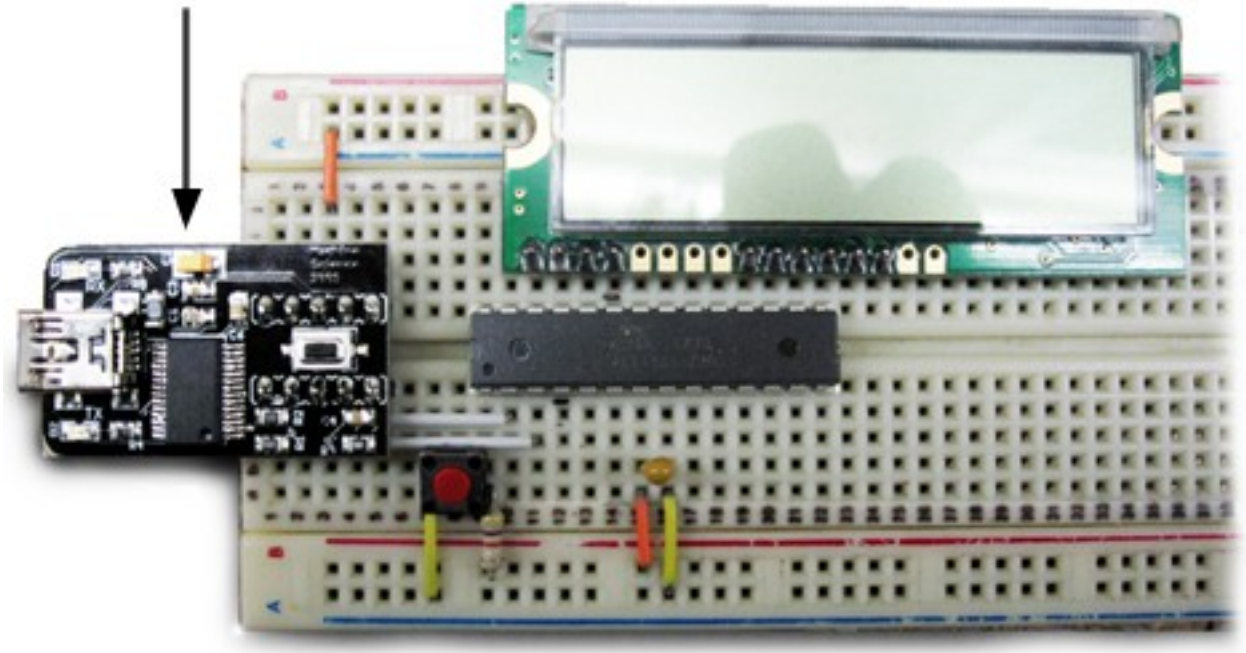
We will be using the same microcontroller used on the Arduino board, a popular chip made by the Atmel corporation -the “Atmega” 168 or 328. (These are 28 pin ICs with identical pin designations, but the 328 has more memory.) The hardware interface we'll be using is perhaps the simplest (and cheapest): connecting to it on a standard breadboard.



# Requirements for the microcontroller:

## 2. I/O:

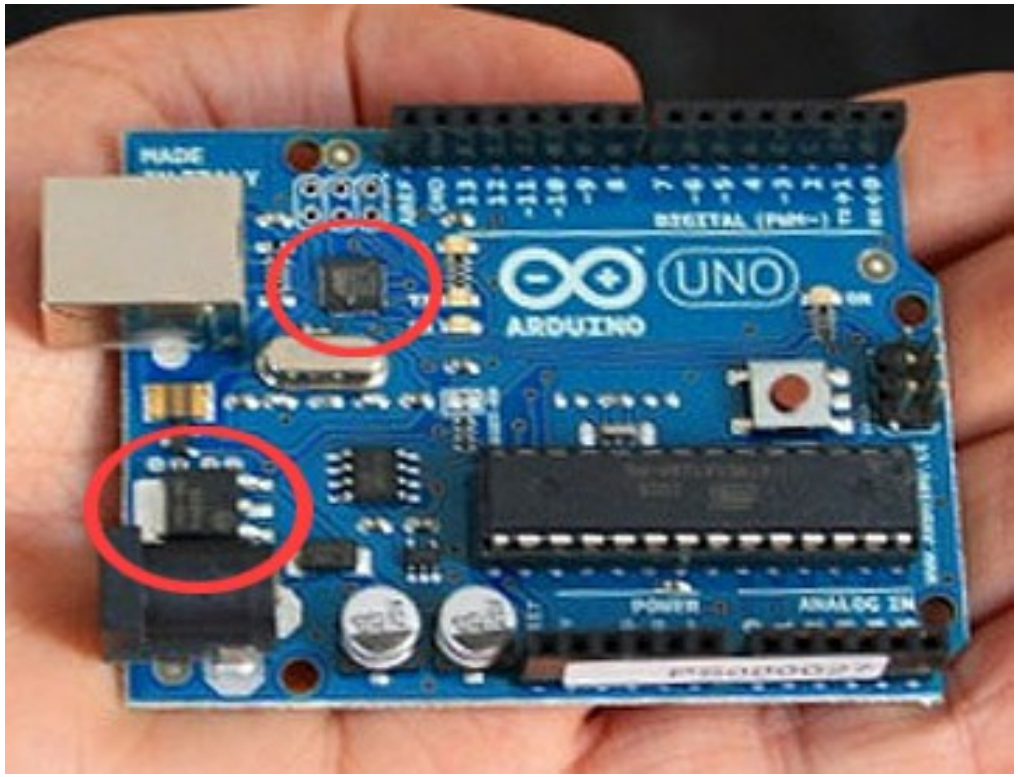
The USB serial communication with the chip is negotiated by an on-board chip made by a company called FTDI. There are free FTDI drivers that can be downloaded for almost every computer platform. Once installed, the computer will be able to send/receive serial communication via USB connection to an FTDI chip.



# Requirements for the microcontroller:

## 2. I/O:

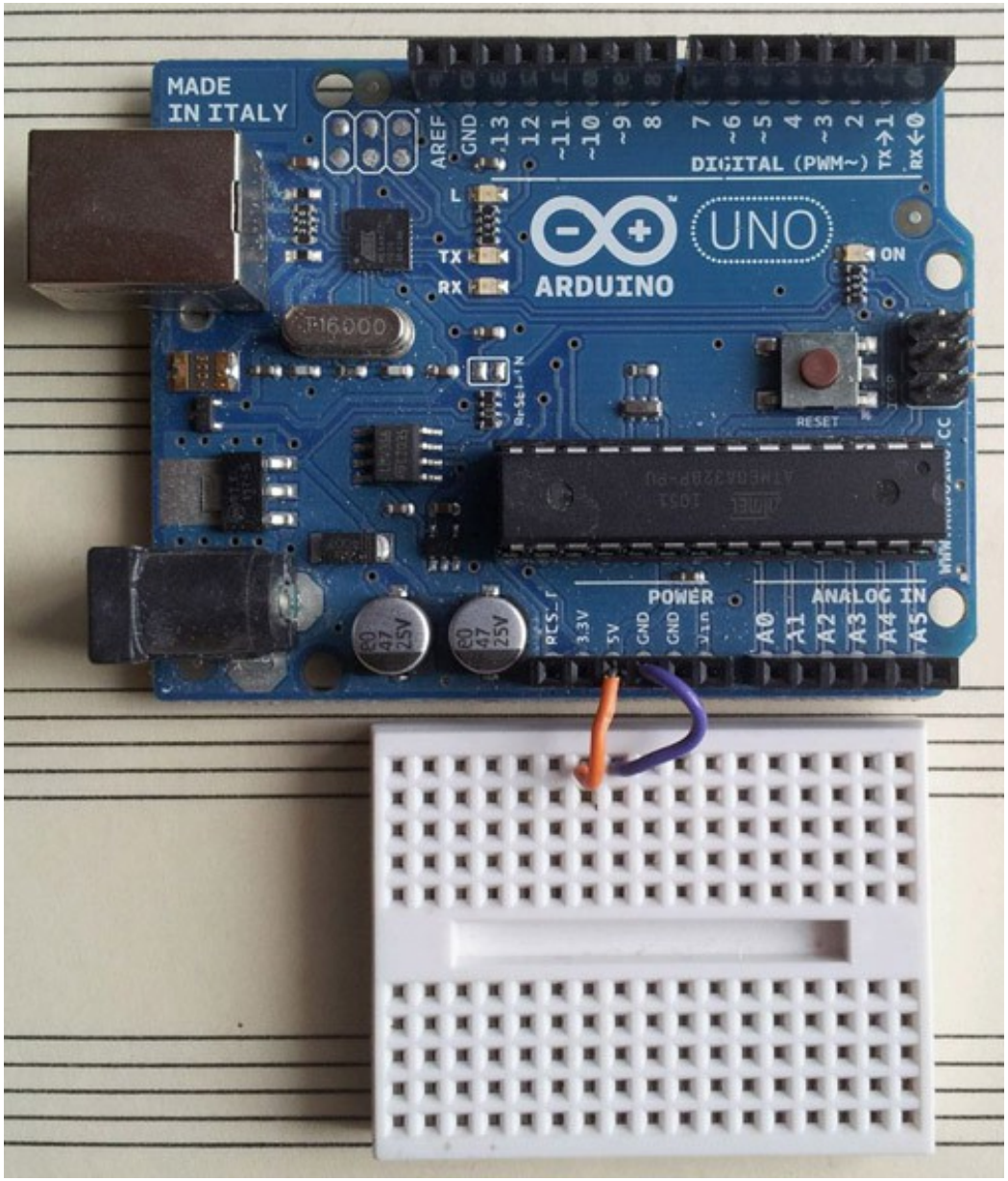
You might wish to do development for the class using your own Arduino board. (Notice it has onboard a voltage regulator as well as a serial FTDI chip attached to the USB connection on the board.)



# Requirements for the microcontroller:

## 2. I/O:

The Arduino board has a limitation for developers: a limited number of pin connections (only one hole per pin.) This can be overcome in a number of ways, including just expanding connections to an adjacent breadboard as pictured here. You can also buy expansion boards (called “shields”) that socket directly onto the Arduino.
(Note: 6 analog and 14 digital pins for I/O)



# Requirements for the microcontroller:

## 3. Programming Interface:

Lastly, programming any computer (including a microcontroller) involves writing code with some kind of text editor, having some mechanism for translating that code into machine-executable instructions, and then getting it to run on the actual hardware (for example, by downloading it to a chip.) Software that combines these processes (including a text editor) into a convenient interface is referred to as an “IDE” (Integrated Development Environment.)

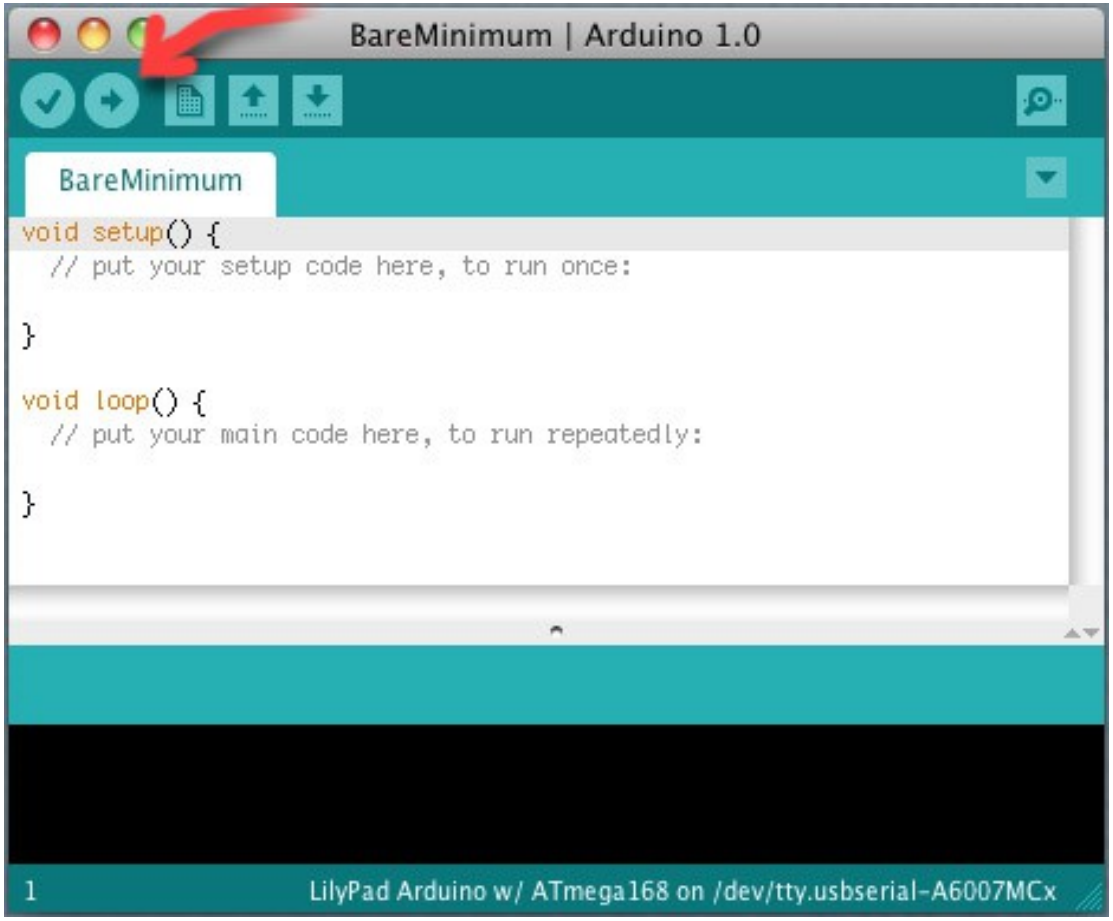
The Atmel corporation makes available a programming environment for the Atmega series of chips called “AVR Studio”. It's free and works perfectly well, but setting it up can be a complicated process. -And it's only available for Windows.

The Arduino development environment is freely available for OSX, Windows and Linux (http://www.arduino.cc), and is the one we are using in the context of this course.

# Requirements for the microcontroller:

## 3. Programming Interface:

Running the Arduino application opens a text editor for editing code. Clicking on the forward arrow button compiles the code and downloads it to the chip.

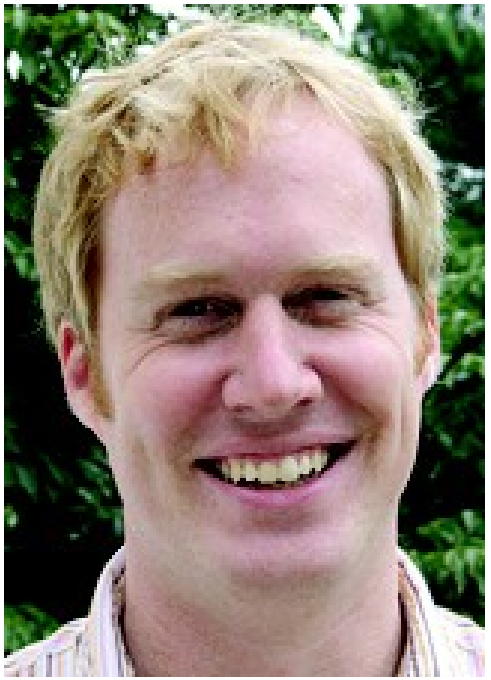


# Requirements for the microcontroller:

## 3. Programming Interface:

If there is an error in the code (usually something simple like leaving out a semicolon) the compiler will complain with an error message and highlight the first line of the problem.

# processing.org



Project of Ben Fry and Casey Reas

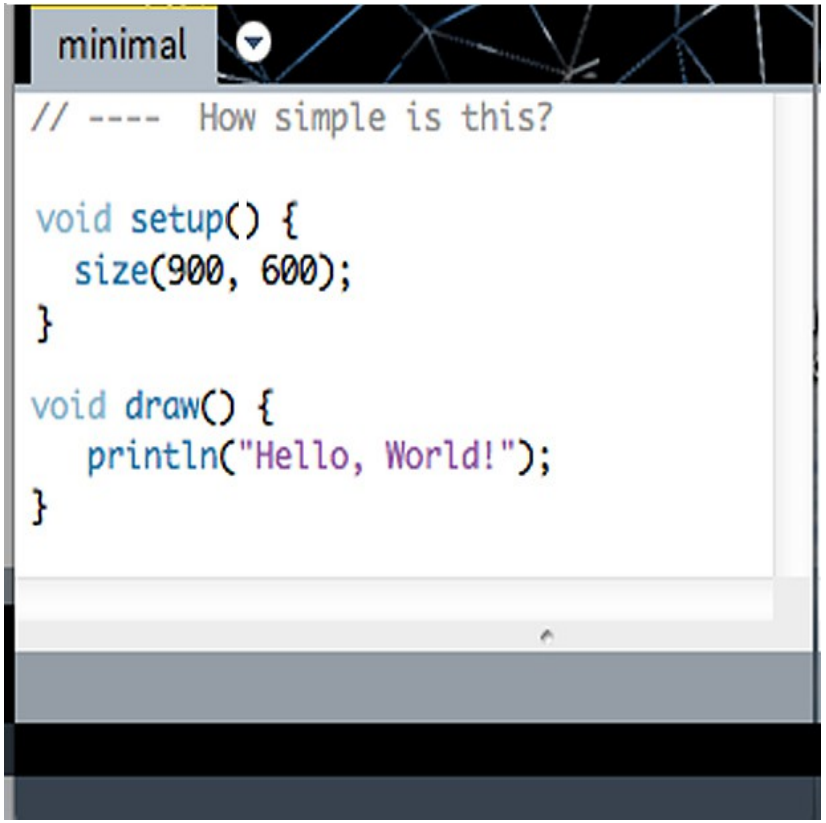
Casey Reas: *Installation at the Art Institute of Chicago*

*We stand by our mission statement:*

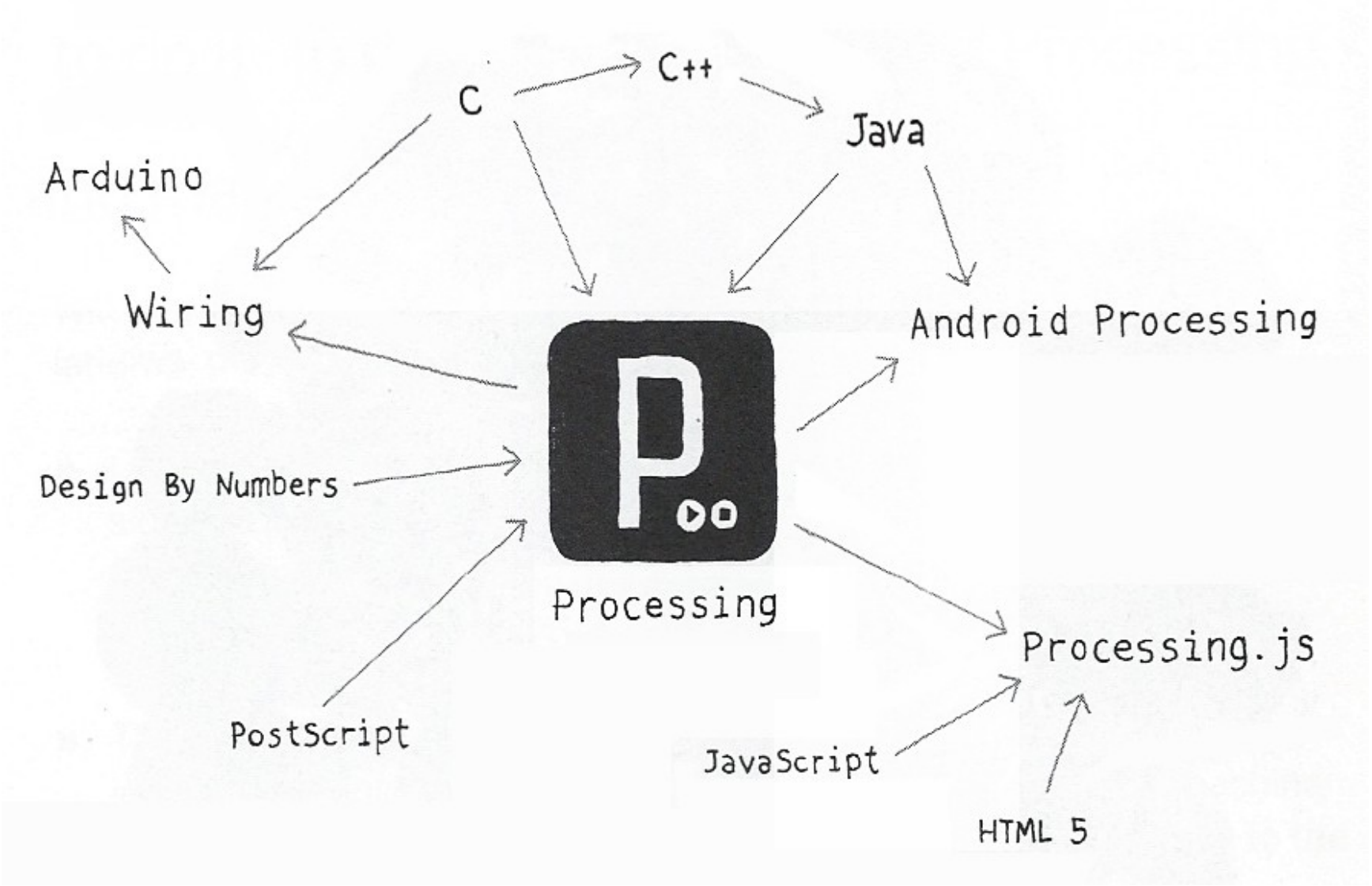
*“ Processing seeks to ruin the careers of talented designers by tempting them away from their usual tools and into the world of programming and computation.*

*Similarly, the project is designed to turn engineers and computer scientists to less gainful employment as artists and designers.”*

```
/*   WinMain()   */
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
                   LPSTR szCmdLine, int iCmdShow) {
    static char szAppName[] = "winhello";
    HWND        hwnd;
    MSG         msg;
    WNDCLASSEX  wndclass;

    wndclass.cbSize         = sizeof(wndclass);
    wndclass.style          = CS_HREDRAW | CS_VREDRAW;
    wndclass.lpfnWndProc    = WndProc;
    wndclass.cbClsExtra     = 0;
    wndclass.cbWndExtra     = 0;
    wndclass.hInstance      = hInstance;
    wndclass.hIcon          = LoadIcon(NULL, IDI_APPLICATION);
    wndclass.hIconSm        = LoadIcon(NULL, IDI_APPLICATION);
    wndclass.hCursor        = LoadCursor(NULL, IDC_ARROW);
    wndclass.hbrBackground  = (HBRUSH) GetStockObject(WHITE_BRUSH);
    wndclass.lpszClassName  = szAppName;
    wndclass.lpszMenuName   = NULL;

    /*  Register a new window class with Windows  */
    RegisterClassEx(&wndclass);
    /*  Create a window based on our new class  */

    hwnd = CreateWindow(szAppName, "Hello, world!",
                        WS_OVERLAPPEDWINDOW,
                        CW_USEDEFAULT, CW_USEDEFAULT,
                        CW_USEDEFAULT, CW_USEDEFAULT,
                        NULL, NULL, hInstance, NULL);
    ShowWindow(hwnd, iCmdShow);
    UpdateWindow(hwnd);

    /*  Retrieve and process messages until we get WM_QUIT  */

    while ( GetMessage(&msg, NULL, 0, 0) ) {
        TranslateMessage(&msg);    /*  for certain keyboard messages  */
        DispatchMessage(&msg);     /*  send message to WndProc        */
    }
```

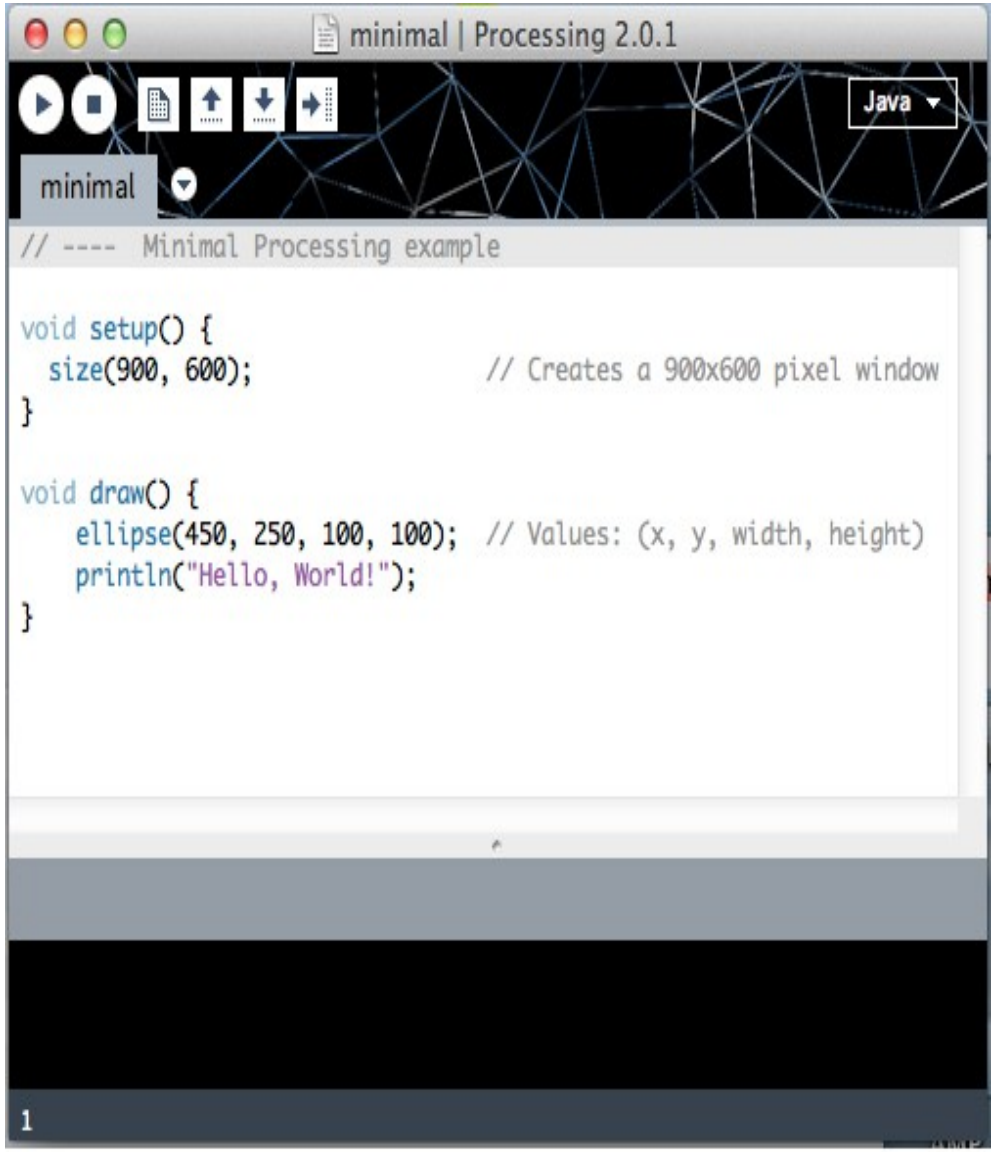


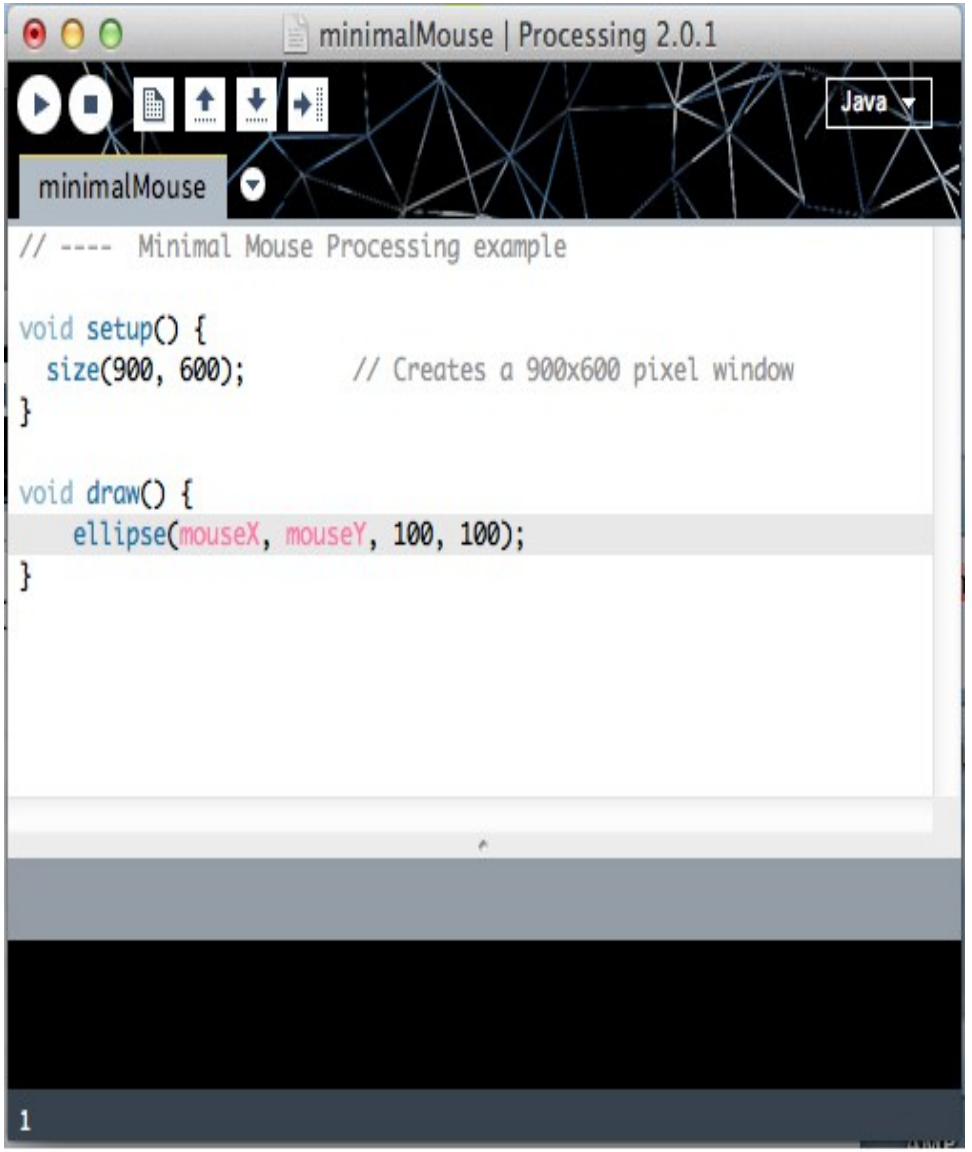

By changing the relationship with code and coding...




Processing comes from a large family of related languages.

(Graphic from *Getting Started with Processing* by Casey Reas and Ben Fry.)


minimal | Processing 2.0.1
Java
minimal
// ---- Minimal Processing example
void setup() {
size(900, 600); // Creates a 900x600 pixel window
}
void draw() {
ellipse(450, 250, 100, 100); // Values: (x, y, width, height)
println("Hello, World!");
}
1


minimalMouse | Processing 2.0.1
Java
minimalMouse
// ---- Minimal Mouse Processing example
void setup() {
size(900, 600); // Creates a 900x600 pixel window
}
void draw() {
ellipse(mouseX, mouseY, 100, 100);
}
1