SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Lecture 5
Course project.
Arduino basics.

IAT267 Introduction to Technological Systems

Assignments

- Assignment 1: due today
  - Answers to the questions will be posted by the end of the week
- Assignment 2: Sensor research assignment
  - Available on webct
  - Due: October 20, 2011

Course Project

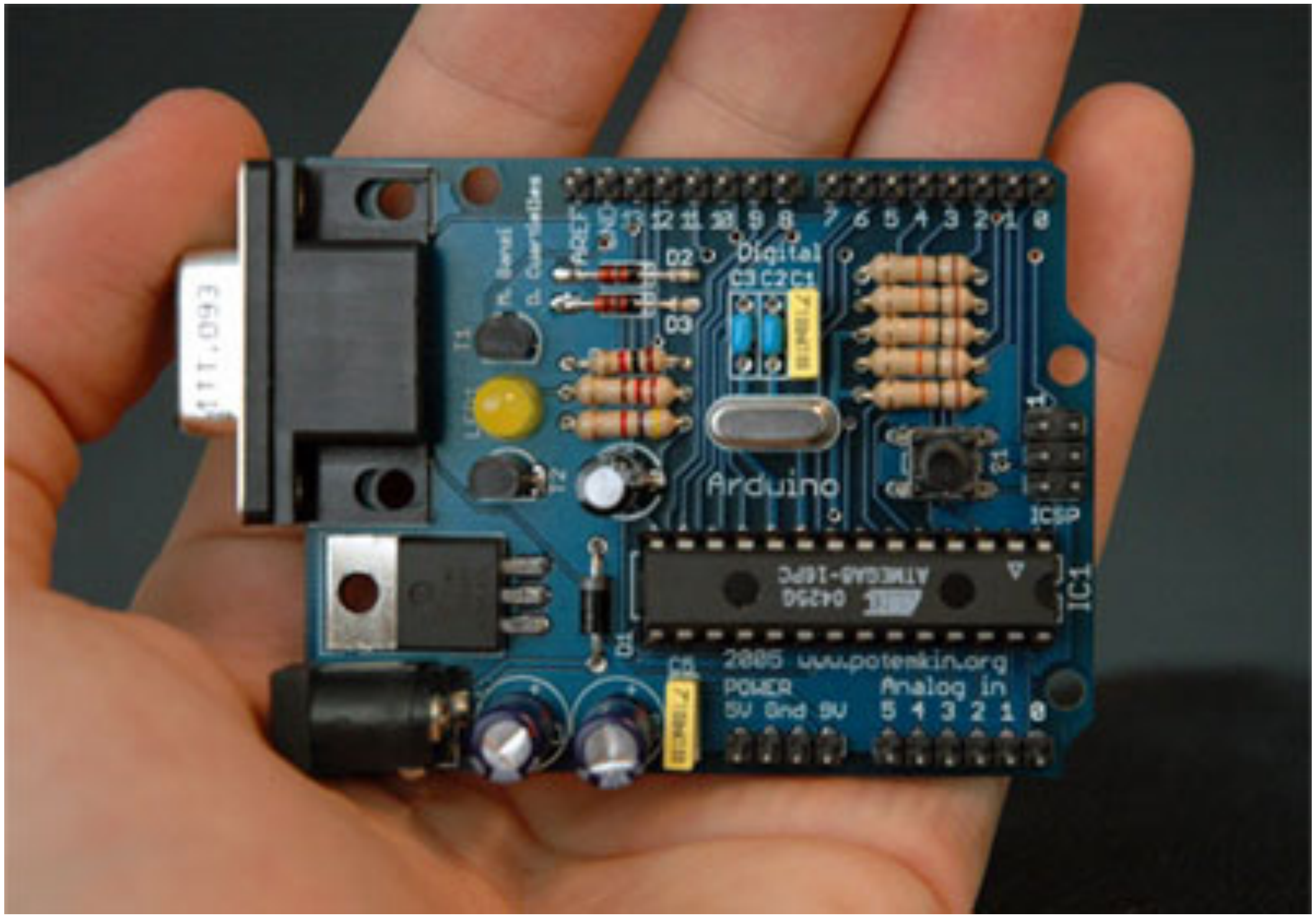
- See the ‘Course Project’ folder on webct
  - Project description
  - Project teams – sign-up
  - Resources
  - Sample student projects from past years
  - Milestones – each milestone has a small deliverable

Lecture Topics for Today

- Arduino basics
  - What is Arduino
  - Arduino family of tools
  - How to use Arduino
    - Workshop this week: hands-on activity using Arduino

Arduino Basics

What is Arduino?



Arduino: 3 Separate Tools

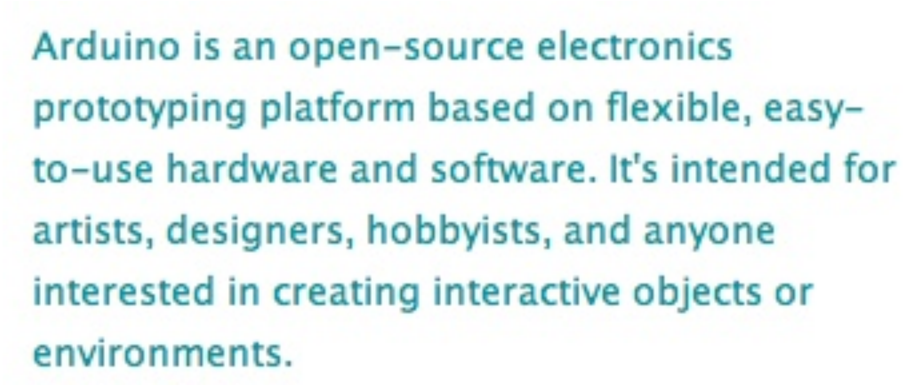
- 1. Arduino controller
  - – The hardware
- 2. Arduino working environment
  - – Simple open source IDE built in Java
- 3. Language and compiler
  - – Create code for the microcontroller

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

Arduino: Extends the Computer System

- Arduino is a tool: for enabling computers to sense and control more of the physical world
- Prototyping platform



Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.

Arduino: Microcontroller Board

- What is a microcontroller: Small, inexpensive **computing** device
- Usually employed for sensing input from the real world and controlling devices based on that input
- Easy to use with simple sensors and output devices

ATMEL
0625
ATMEGA168-20PU

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

How can we use Arduino

- Arduino can be used to develop interactive objects
- Taking inputs from a variety of switches or sensors
- Controlling a variety of lights, motors, and other physical outputs.

Arduino Projects

- Arduino projects can be stand-alone.
- Or they can communicate with software running on your computer (e.g. Flash, Processing, MaxMSP).
- The boards can be assembled by hand or purchased preassembled.
- The open-source IDE can be downloaded for free.

Why Use Arduino? (1)

- Inexpensive
- Cross-platform - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- Simple, clear programming environment

Why Use Arduino? (2)

- Programming is very easy / quick
  - Programmed via a USB cable, not serial port.

- Active community of users online, so there are lots of resources available.

Arduino: Open Source

- Open source and extensible **software**: The Arduino software is published as an open source tool, available for extension by experienced programmers.

- Open source and extensible **hardware** - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it.

Open Source

- Open source **hardware** and **software**:
  - – if you wish you can download the circuit diagram, buy all the components and make your own board without paying anything to the makers of Arduino

1. Arduino Board

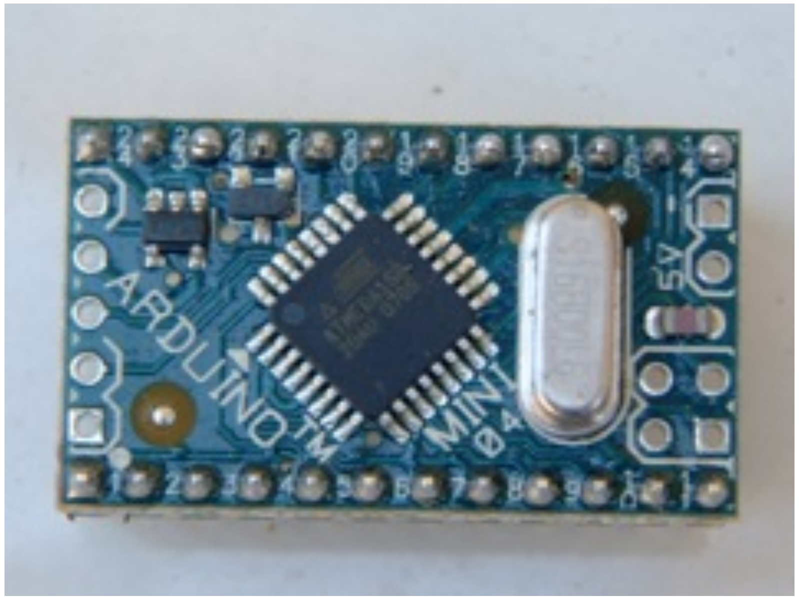
To Get Started:

Hardware

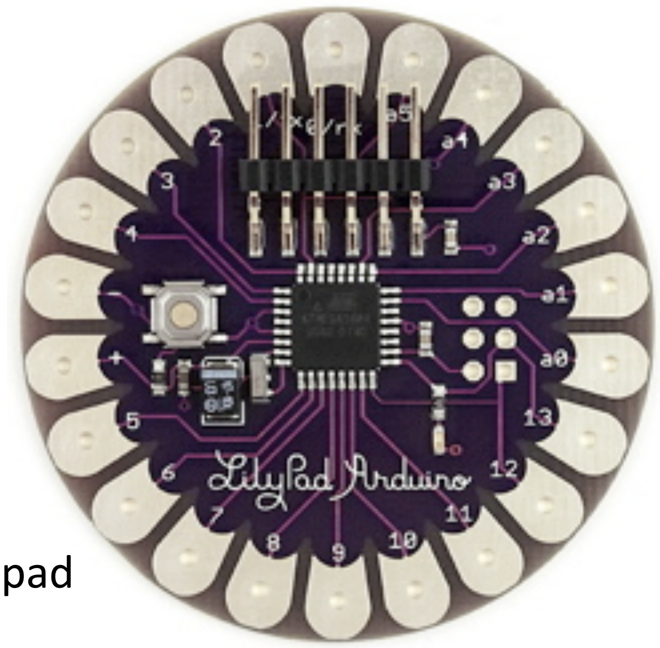
Arduino Serial

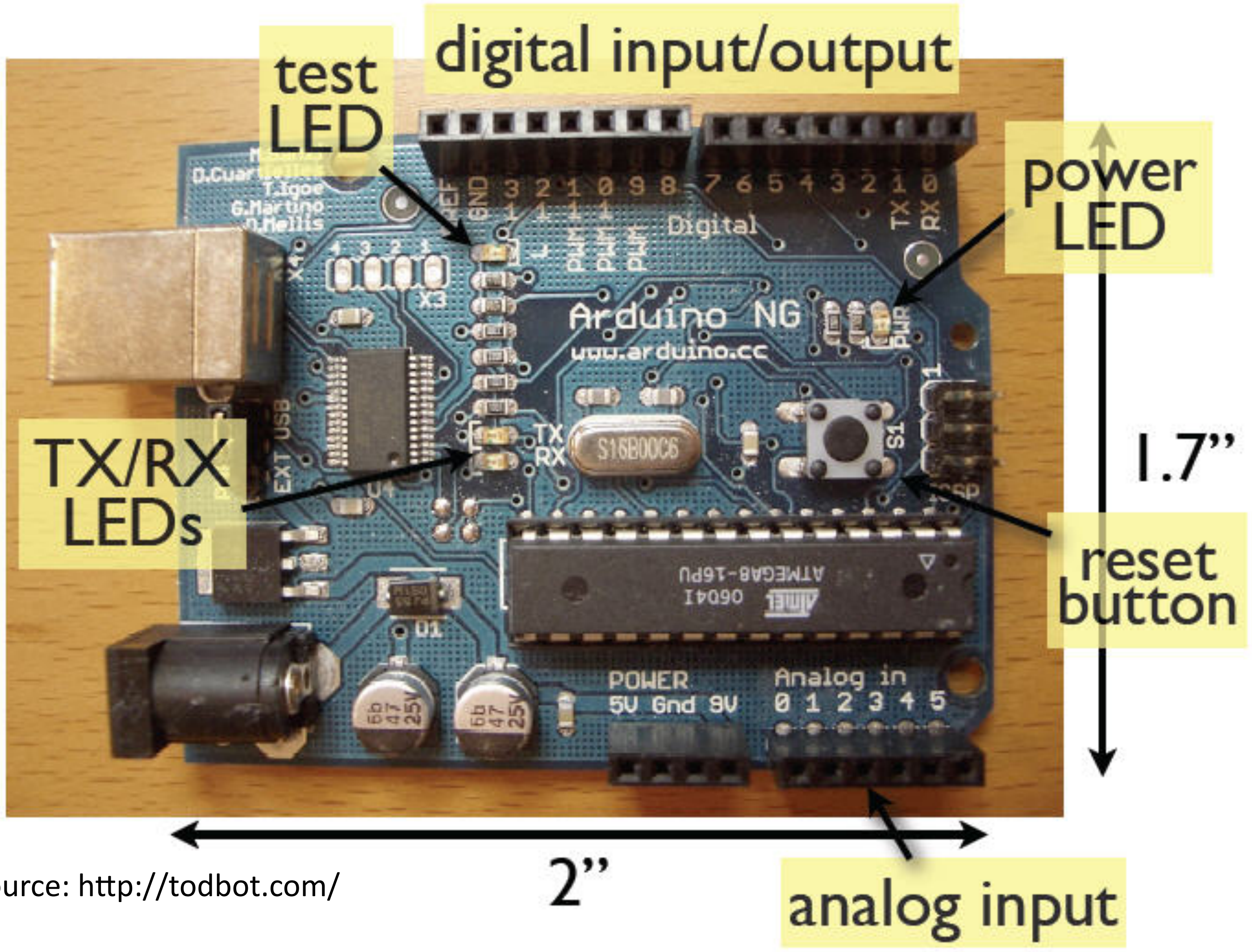
Arduino BT

Arduino Mega


Arduino Mini

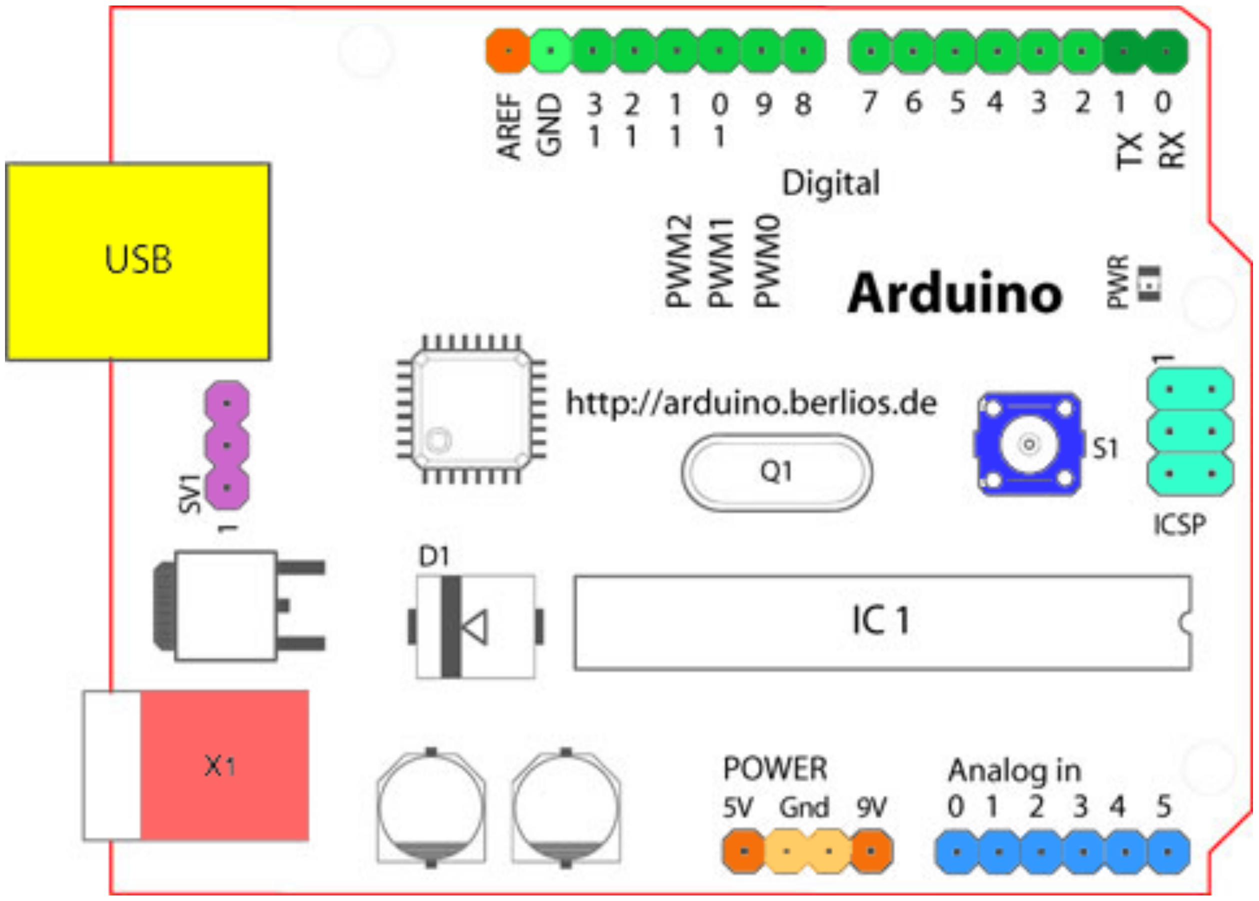
Arduino Nano


Arduino Lilypad



Source: http://todbot.com/

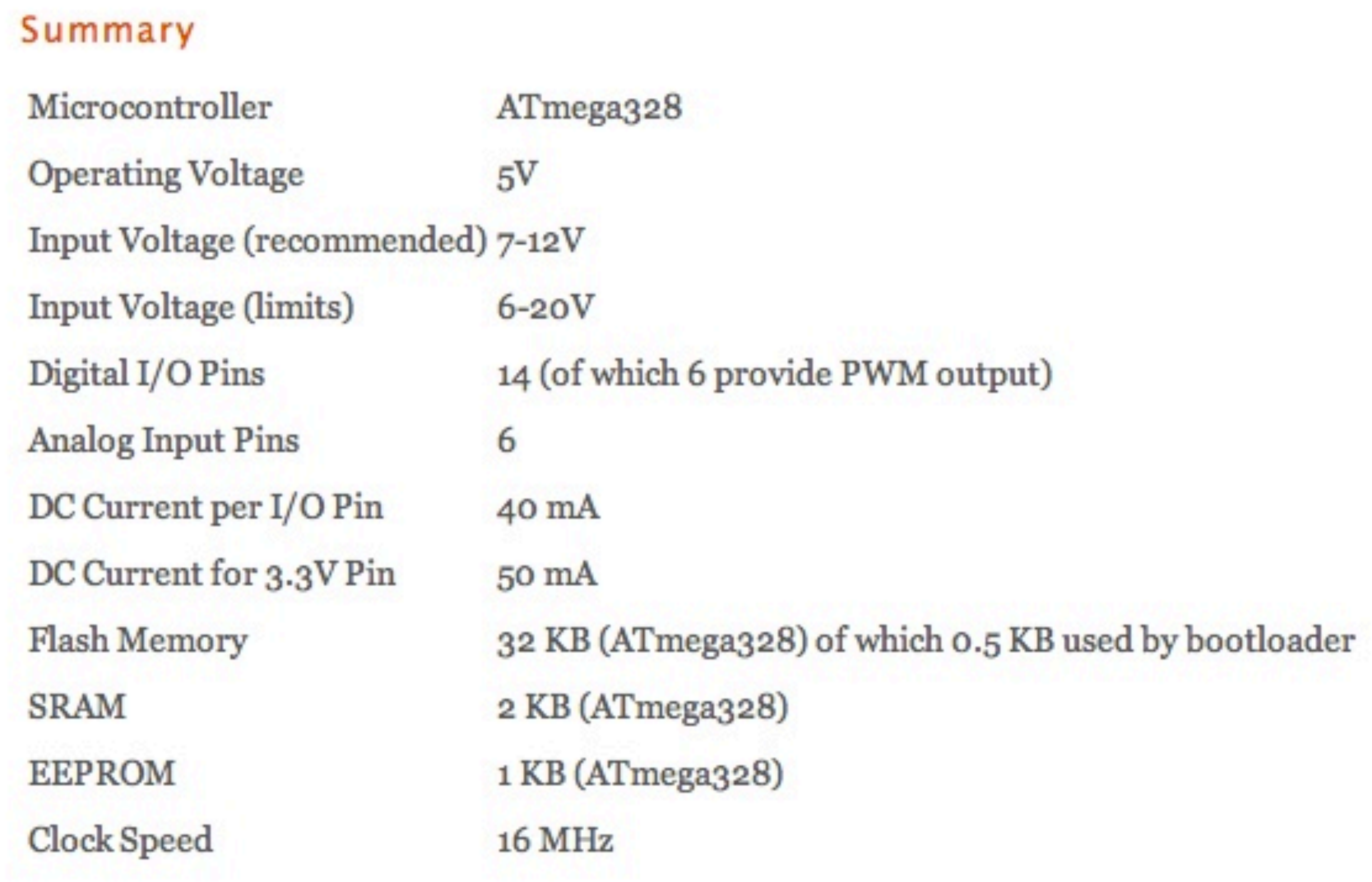
Diagram of the Arduino Board



Components of the Arduino Board

- Digital inputs: 2 to 13
- Analog inputs: 0 to 5
- Arduino uses the Atmel ATMega microcontroller
- Has a USB port to communicate with a computer
- Reset button
- TX/RX LEDs
- Connection for external power supply (9-12V DC)

Capabilities of Arduino – Arduino UNO



Summary

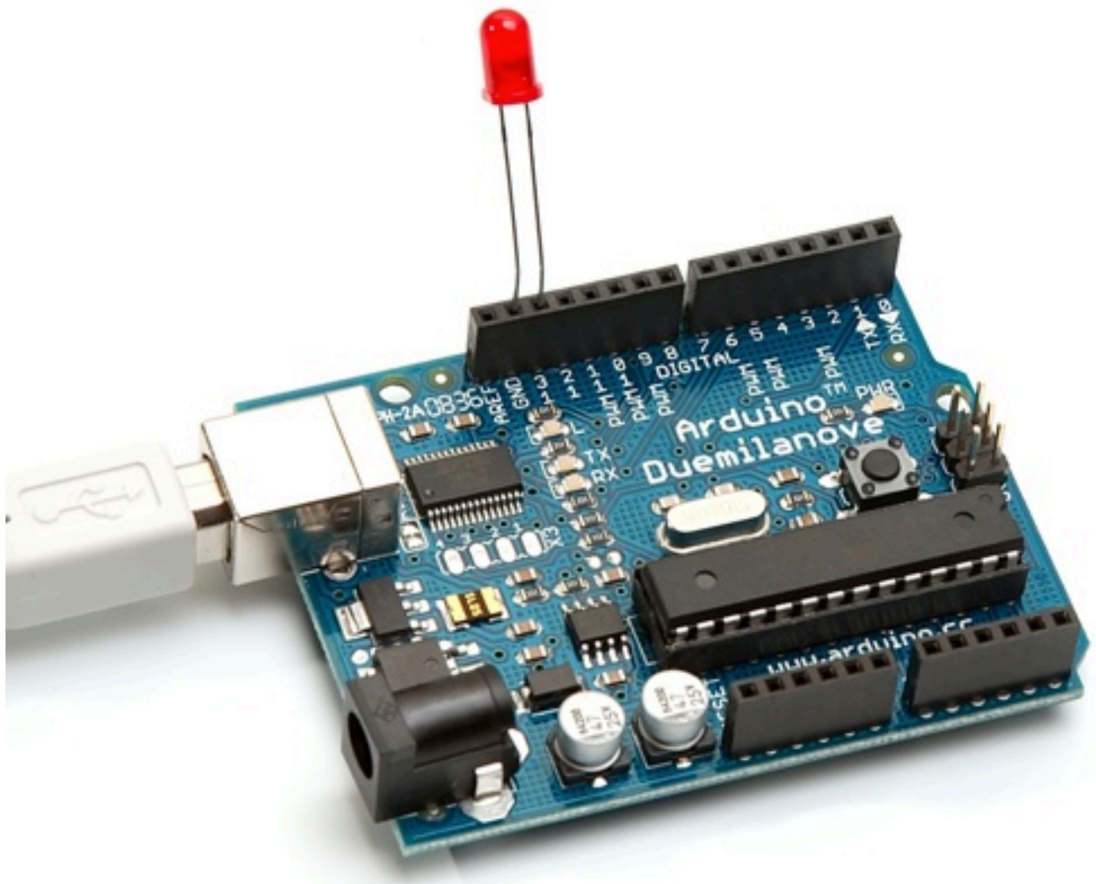
Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

What is a pin?

- A **pin** provides an **input** or **output** through which the controller can communicate with components.

- Small wires can be inserted into the pin connectors

LED Connected to Pin 13 and GND



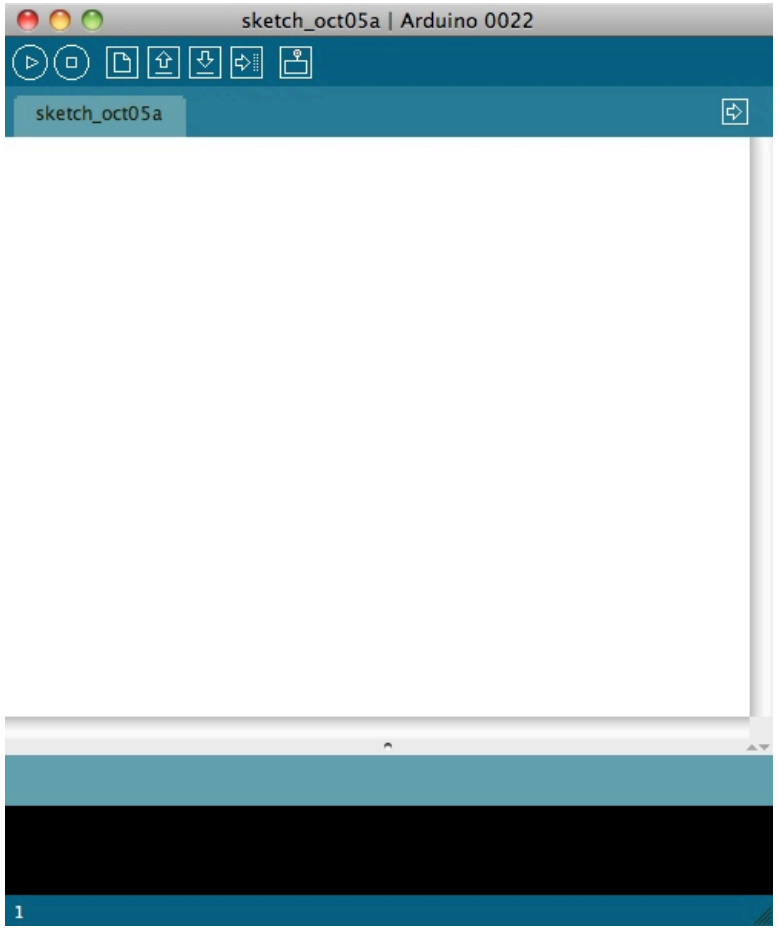
Digital vs. analog pins

- **Digital pins**:
  - Have two values that can be read or written to them: high and low
    - High: means that 5 V (Volts) is being sent either from the controller or from a component
    - Low: means that the pin is at 0 Volts.
  - Any kind of binary information can be read or written to a digital pin.

Analog Pins

- Can have a wide range of information sent to them (analog pins are inputs)
- These pins are what we use to input information that has a range of values, e.g.:
  - The position of a dial
  - The distance of an object from an infrared sensor

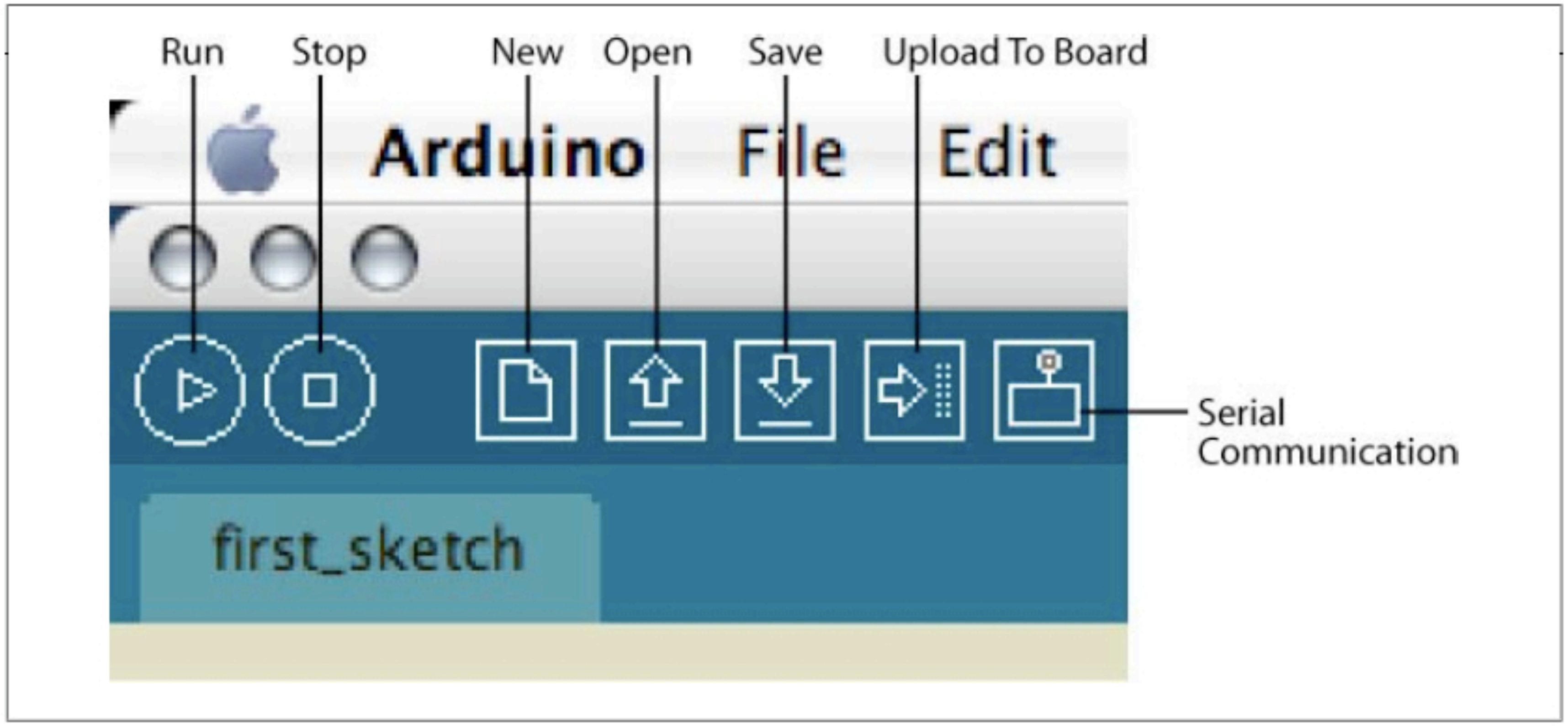
2. Arduino programming environment



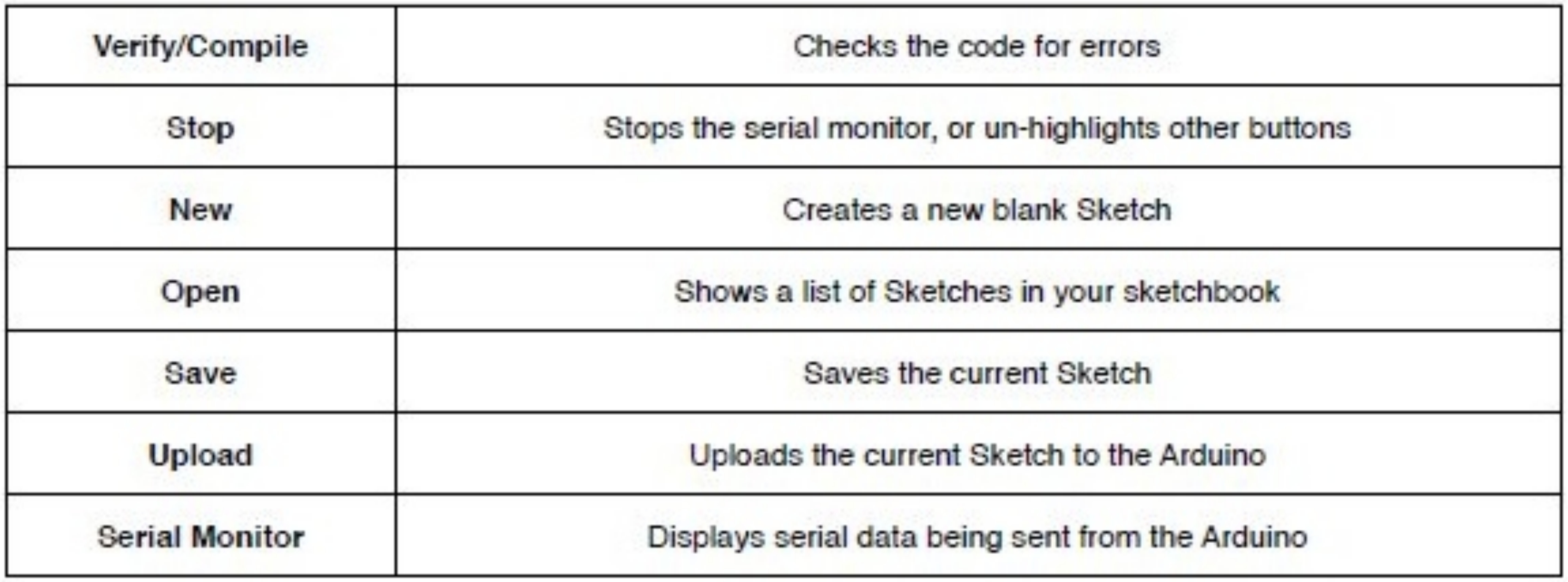
How is Arduino programmed?

- Write programs on your PC
- Download them into the Arduino board
- Arduino board can then be used by itself

The Arduino IDE



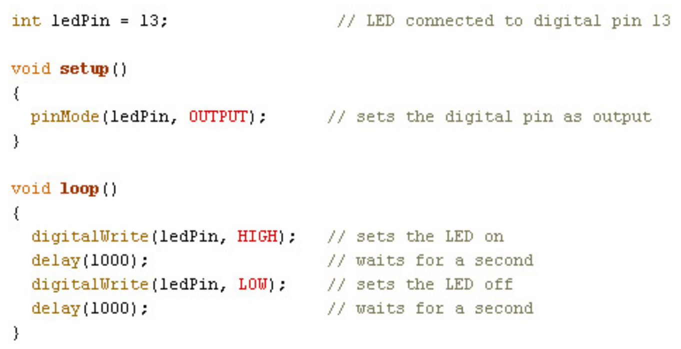
Arduino IDE Menu Options



Verify/Compile	Checks the code for errors
Stop	Stops the serial monitor, or un-highlights other buttons
New	Creates a new blank Sketch
Open	Shows a list of Sketches in your sketchbook
Save	Saves the current Sketch
Upload	Uploads the current Sketch to the Arduino
Serial Monitor	Displays serial data being sent from the Arduino

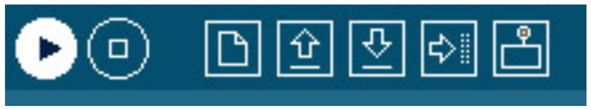
Development Cycle

- Edit code

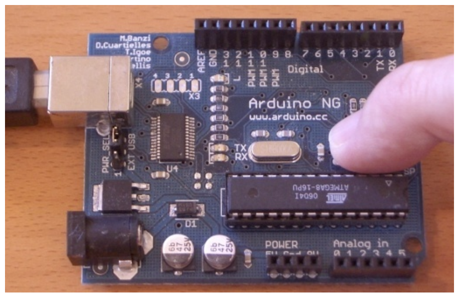


```
int ledPin = 13;                // LED connected to digital pin 13

void setup()
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

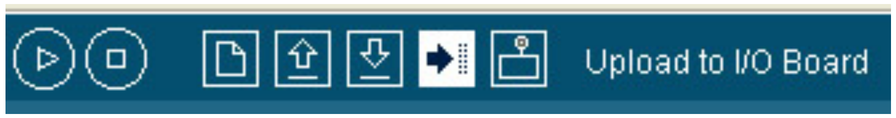
- Compile



- Reset board



- Upload

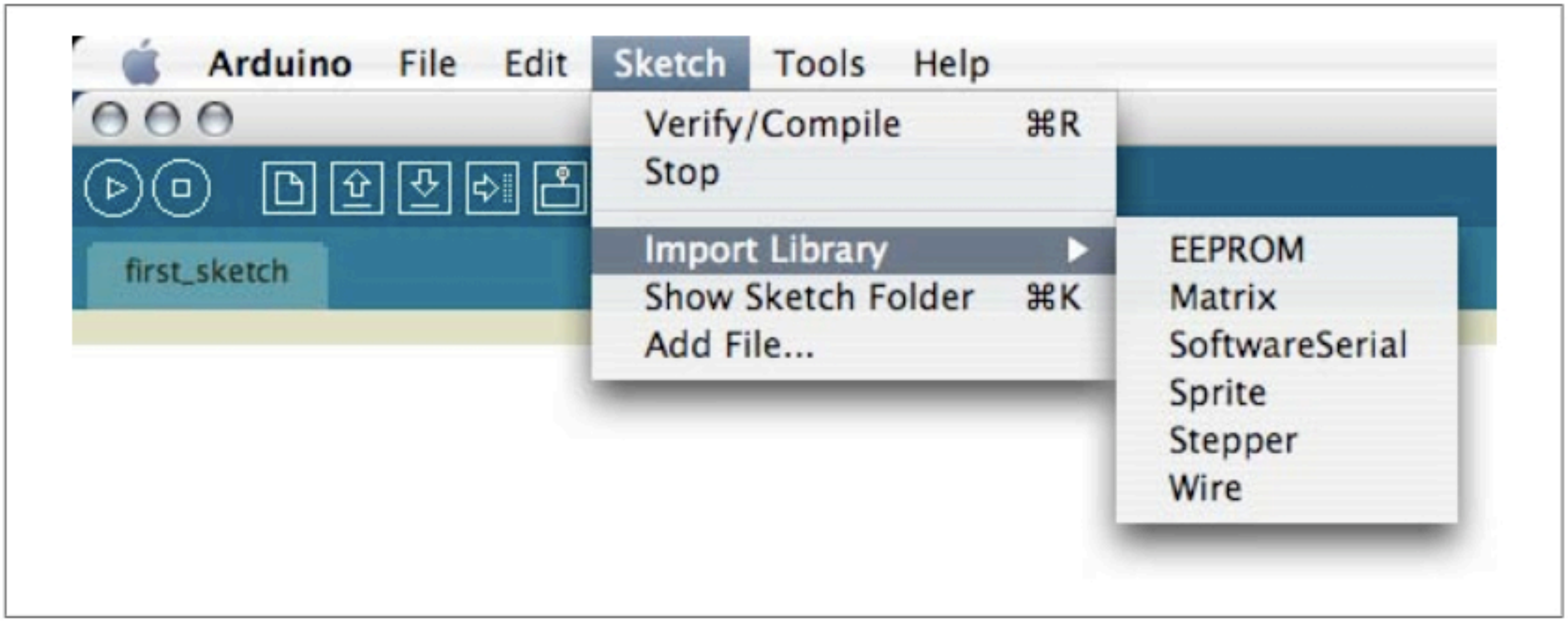


- **Run** button: does not in fact run the code; it checks for errors and compiles the code
- **Stop** button: stops the IDE from listening on the serial port
- **New** button: creates a new application
- **Save**: saves your project

- **Upload to board**: actually uploads the code to the board, assuming that the board is properly connected and all the drivers are properly installed

- **Serial communication**: opens the Serial Monitor window – used in cases when we want feedback from the board (data sent serially to computer)

‘Sketch’ menu of the toolbar



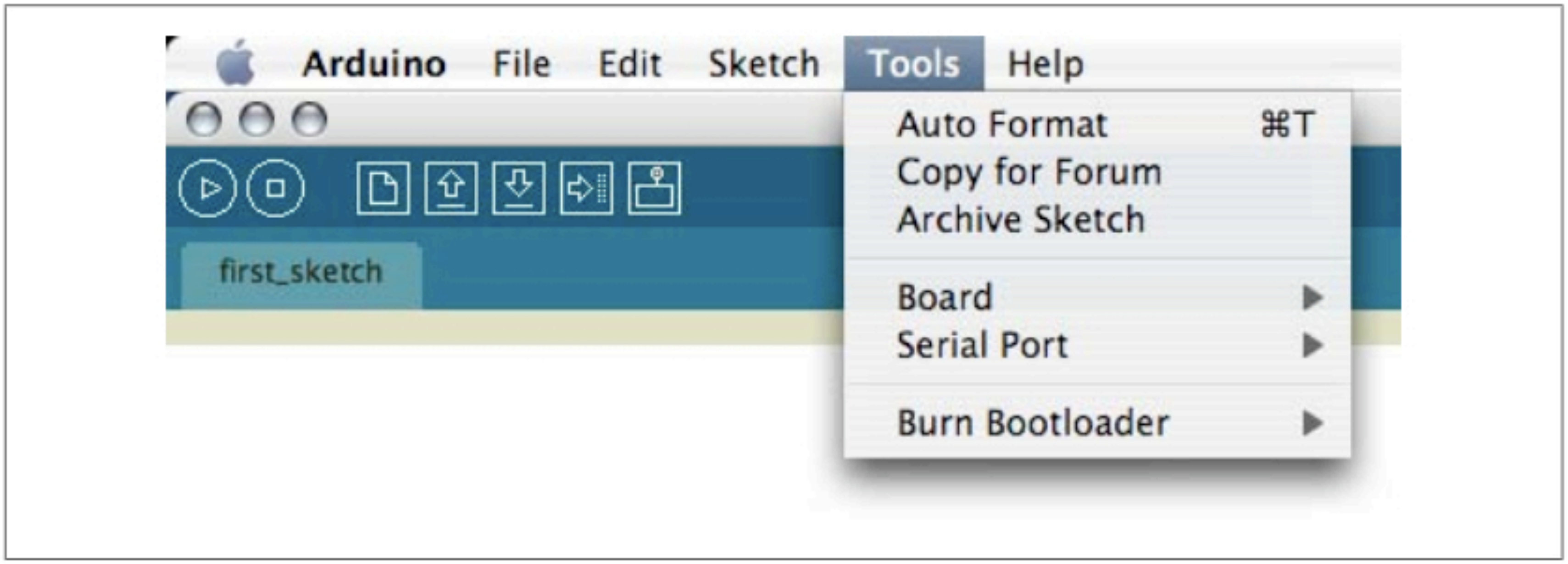
‘Sketch’ menu of the toolbar

- **Import Library**: allows you to import functionality from a library created for a specific purpose
  - E.g., sound, working with motors, communication
  - Can be either the default libraries that come with Arduino or a library that you have created yourself
  - Line that appears in the code window:
    - #include <Stepper.h>

‘Sketch’ menu of the toolbar

- **Show Sketch Folder**: brings up the folder where all your application files are stored
  - Helpful if you want to check if a certain file is present (e.g., image file)
- **Add File**: allows you to select a file from anywhere in your operating system and save it to the folder where your application is located

‘Tools’ menu of the toolbar



SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

‘Tools’ menu of the toolbar

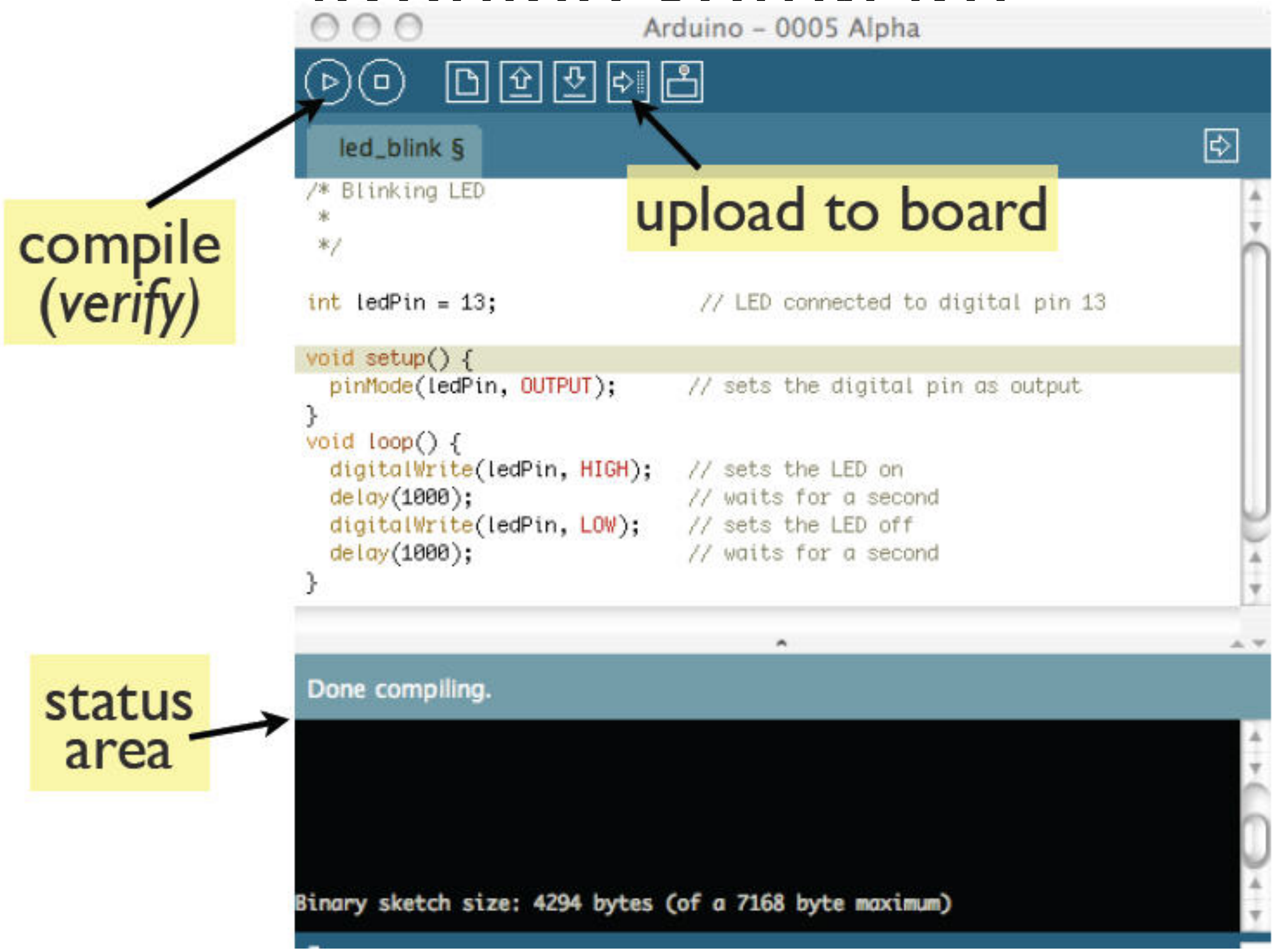
- Contains menu buttons for selecting the controller and port on which the board is connected to the computer

- **Auto Format**: formats all your code to standardize the indentations and spacing

‘Tools’ menu of the toolbar

- **Copy for Forum**: copies all the code in an application to the system clipboard if your computer in an HTML format so that it can be pasted into a web page without losing formatting.
- **Archive Sketch**: .zip file for your application
- **Burn Bootloader**: needed only if you are building your own board.

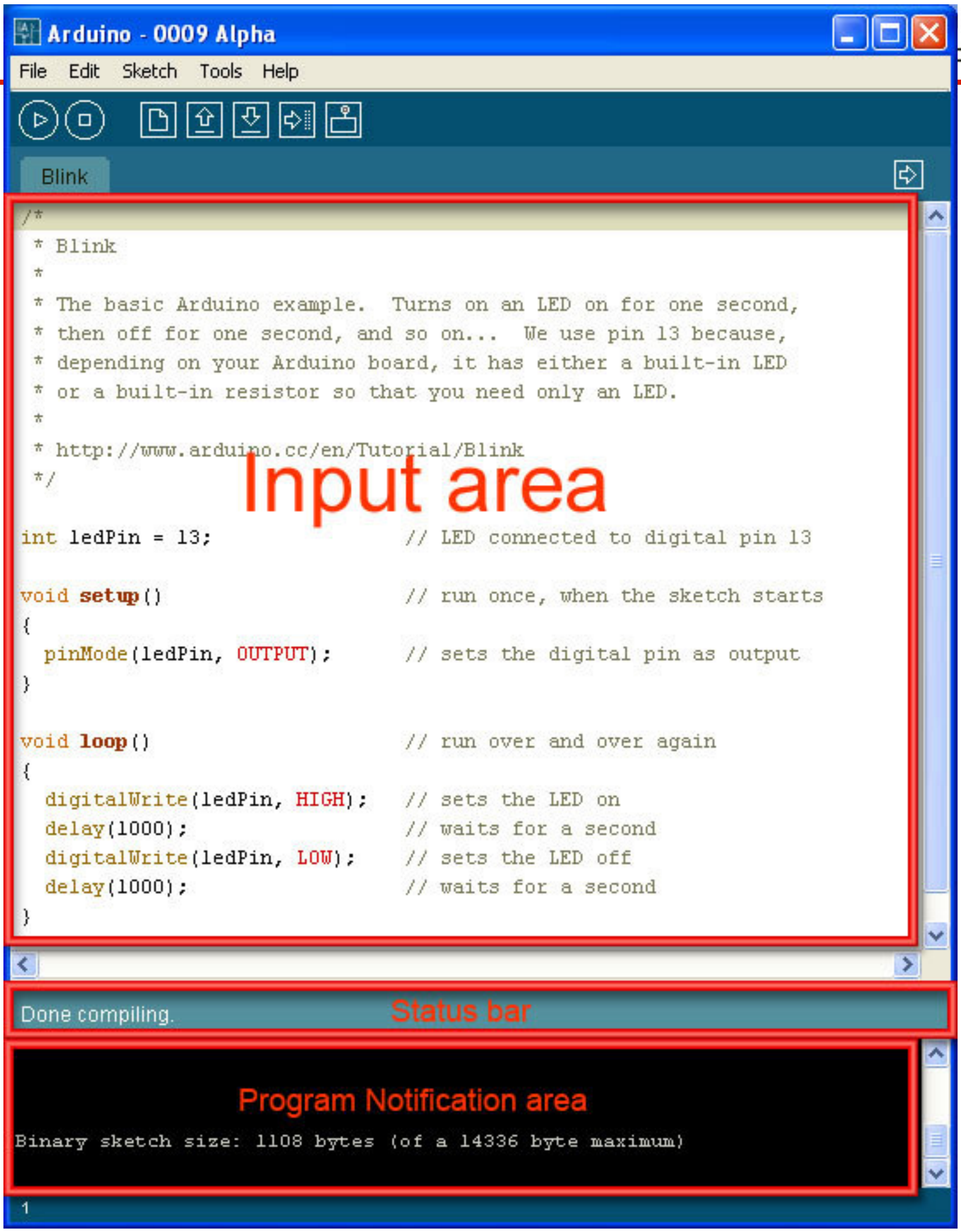
Arduino Software



```
/* Blinking LED
 *
 */

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

void setup() {
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}
void loop() {
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

```
Binary sketch size: 4294 bytes (of a 7168 byte maximum)
```

SFU



```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

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

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

```
Done compiling.
```

```
Binary sketch size: 1108 bytes (of a 14336 byte maximum)
```

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

3. The Arduino language

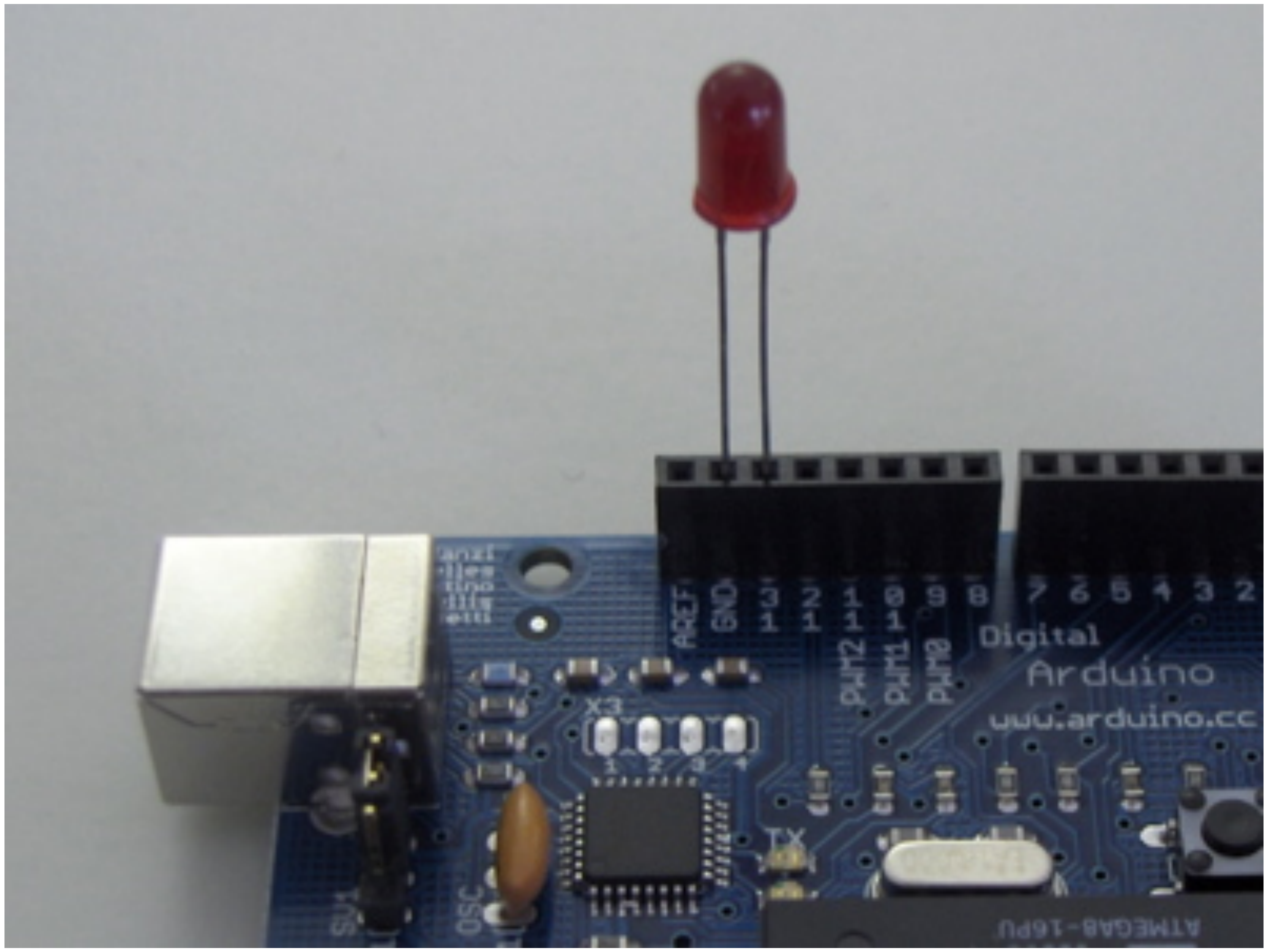
(Wiring)

Example Program: Blink

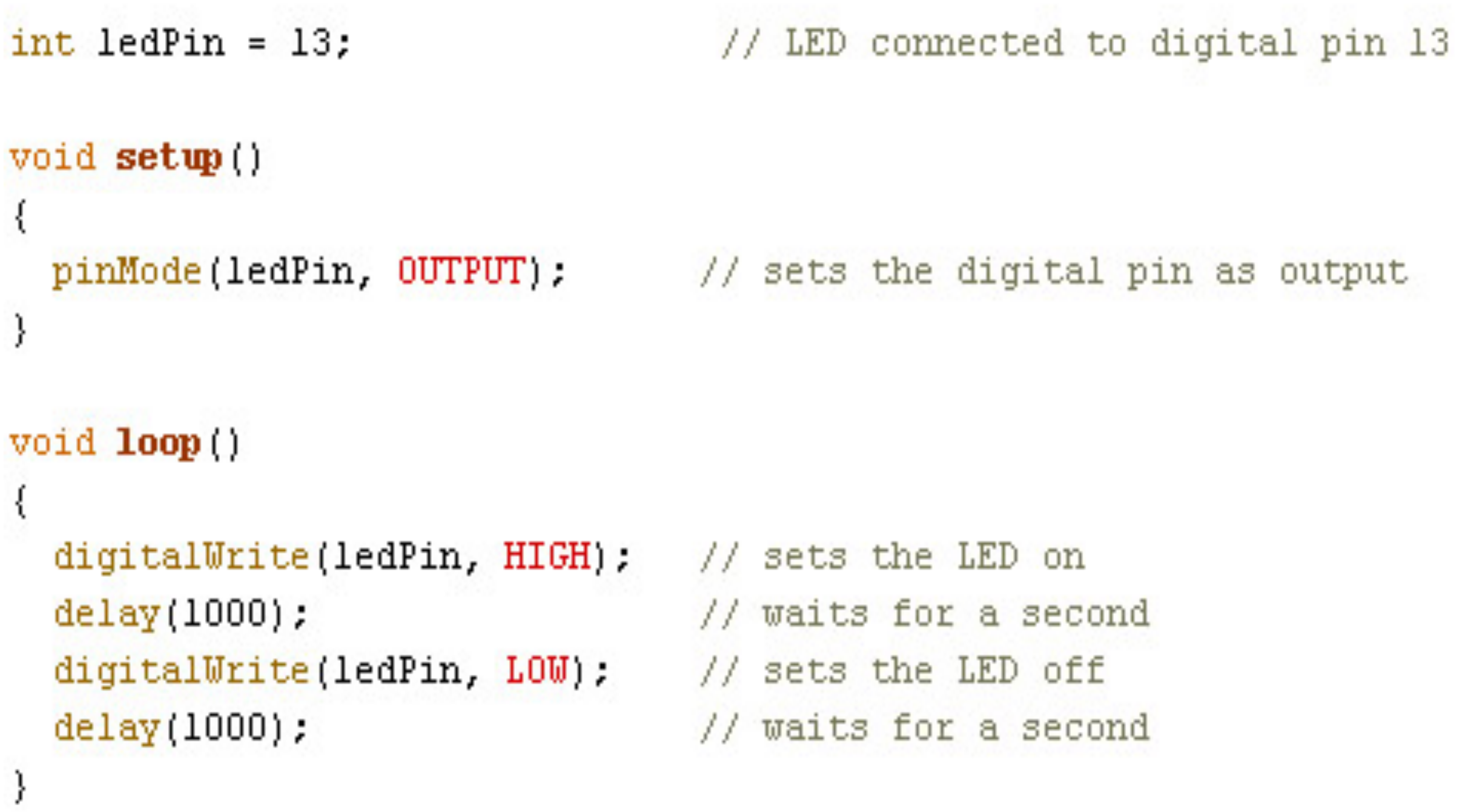
- LED connected to digital pin 13 (we choose pin 13 because depending on your Arduino board, it has either a built-in LED or a built-in resistor so that you need only an LED).

- LEDs have polarity, which means they will only light up if you orient the legs properly.

Circuit

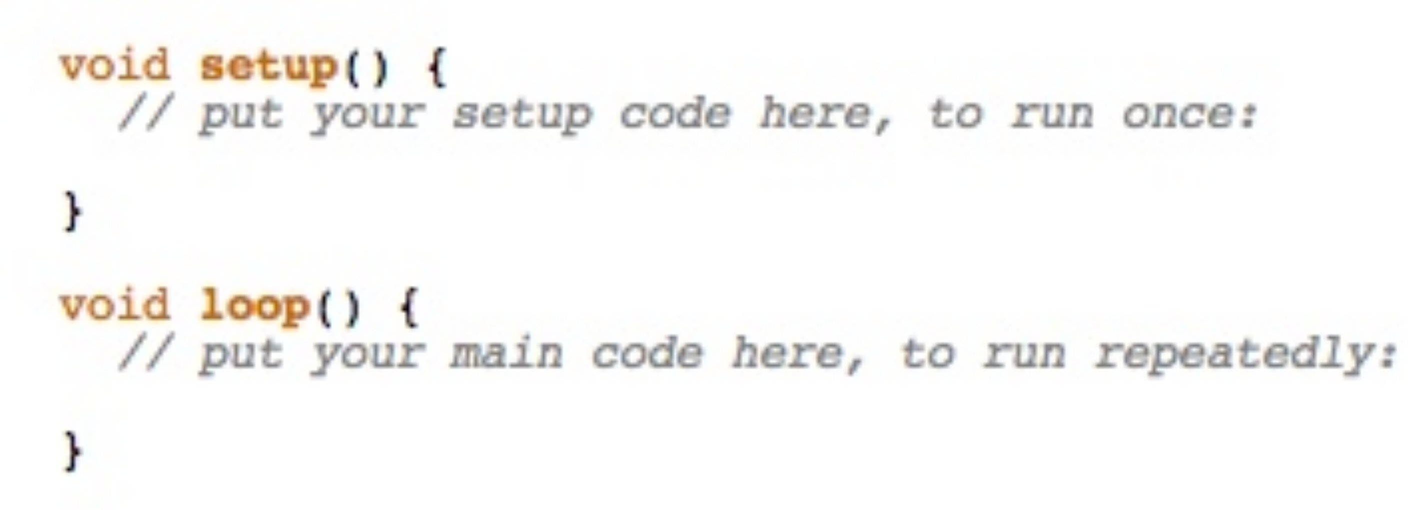

AREF
GND
13
12
11
10
9
8
7
6
5
4
3
2
PWM2
PWM1
PWM0
Digital
Arduino
www.arduino.cc
X3
TX
ATMEGA8-16PU

The code



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

void setup()
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

Minimal Code



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

setup()

- The setup() function is called when a sketch starts.
  - Use it to initialize variables, pin modes, start using libraries, etc.
  - The setup function will only run once, after each powerup or reset of the Arduino board.

SFU

SCHOOL OF INTERACTIVE ARTS + TECHNOLOGY

loop()

- Loops consecutively, allowing your program to change and respond.

- Use it to actively control the Arduino board.

Thank you

Questions?