

# Microcontrollers

Press Play: Interactive Device Design | April 4, 2011

# Homework debrief

Examples | Critique (I like, I wish)

Finding the answer vs. being told | Questions

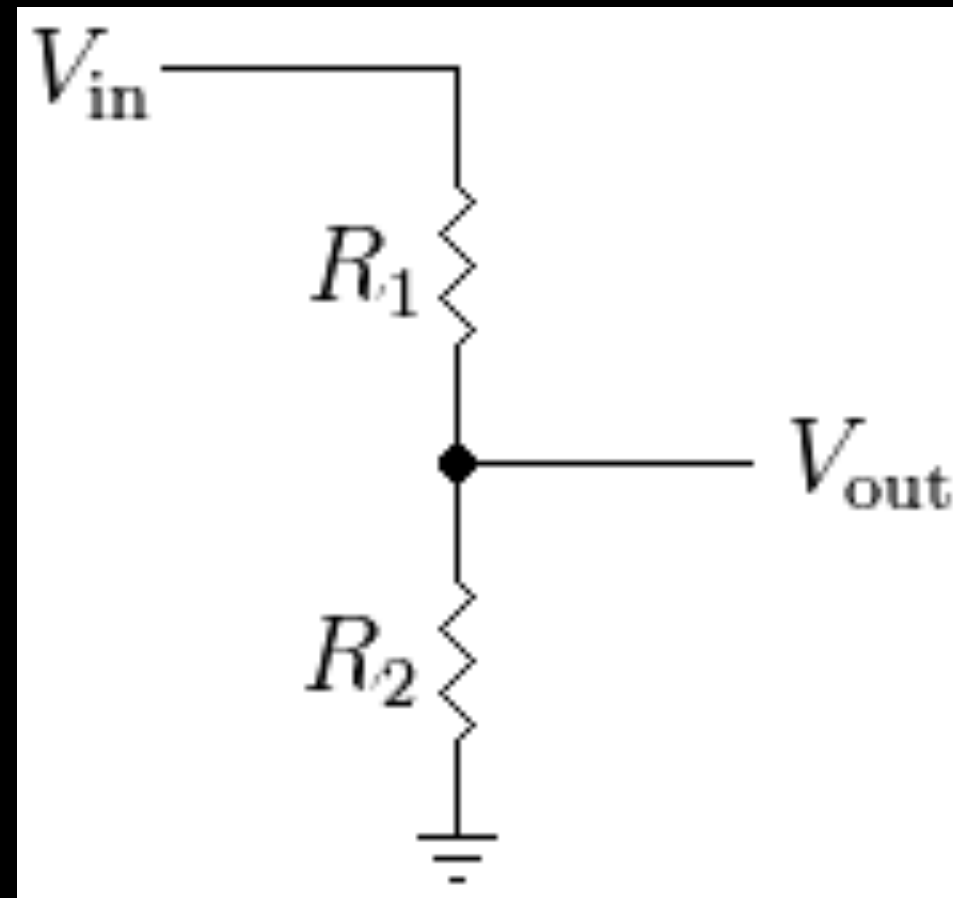
# Lab Prep!

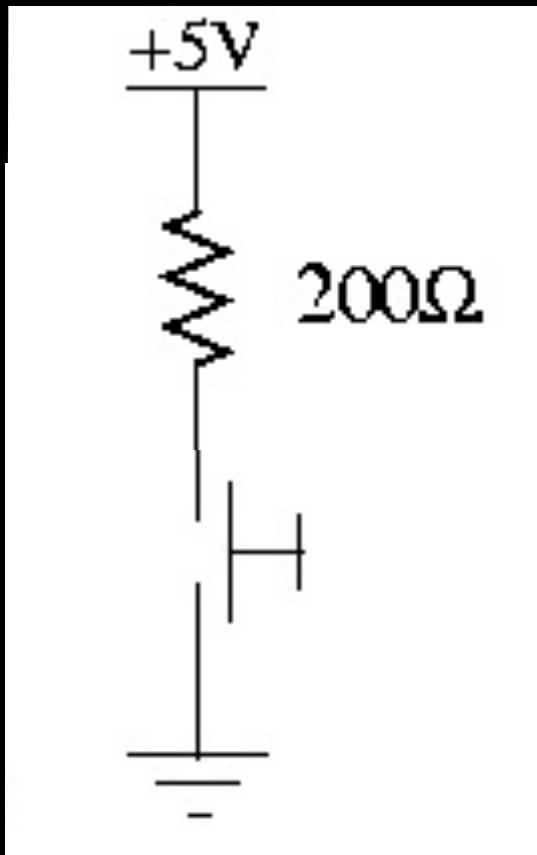
Frankenlight

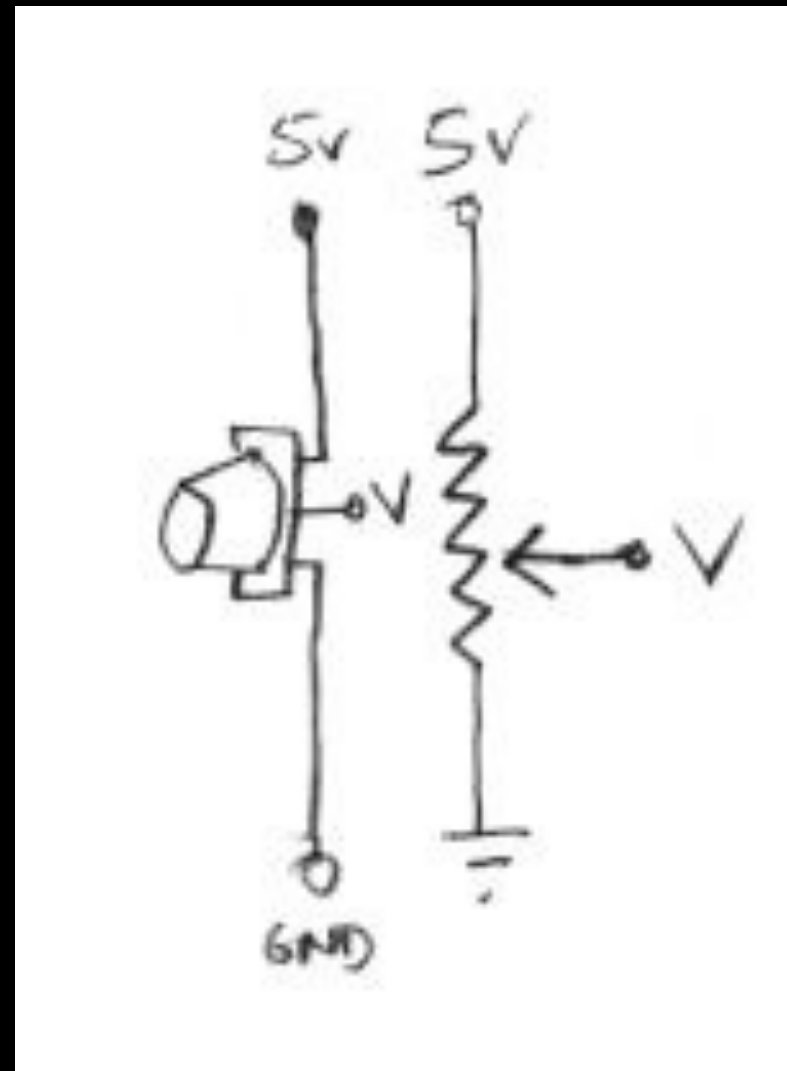
Arduino Programming

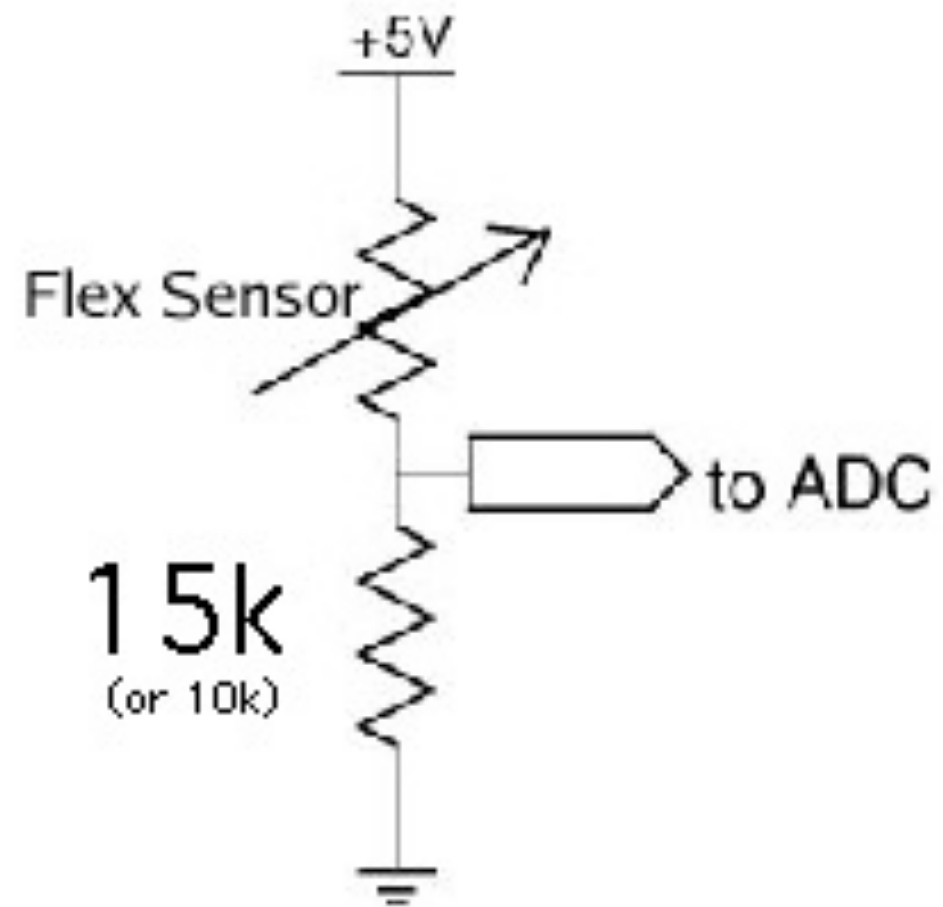
# Basic Sensor Circuit

Button circuit | Voltage divider circuit



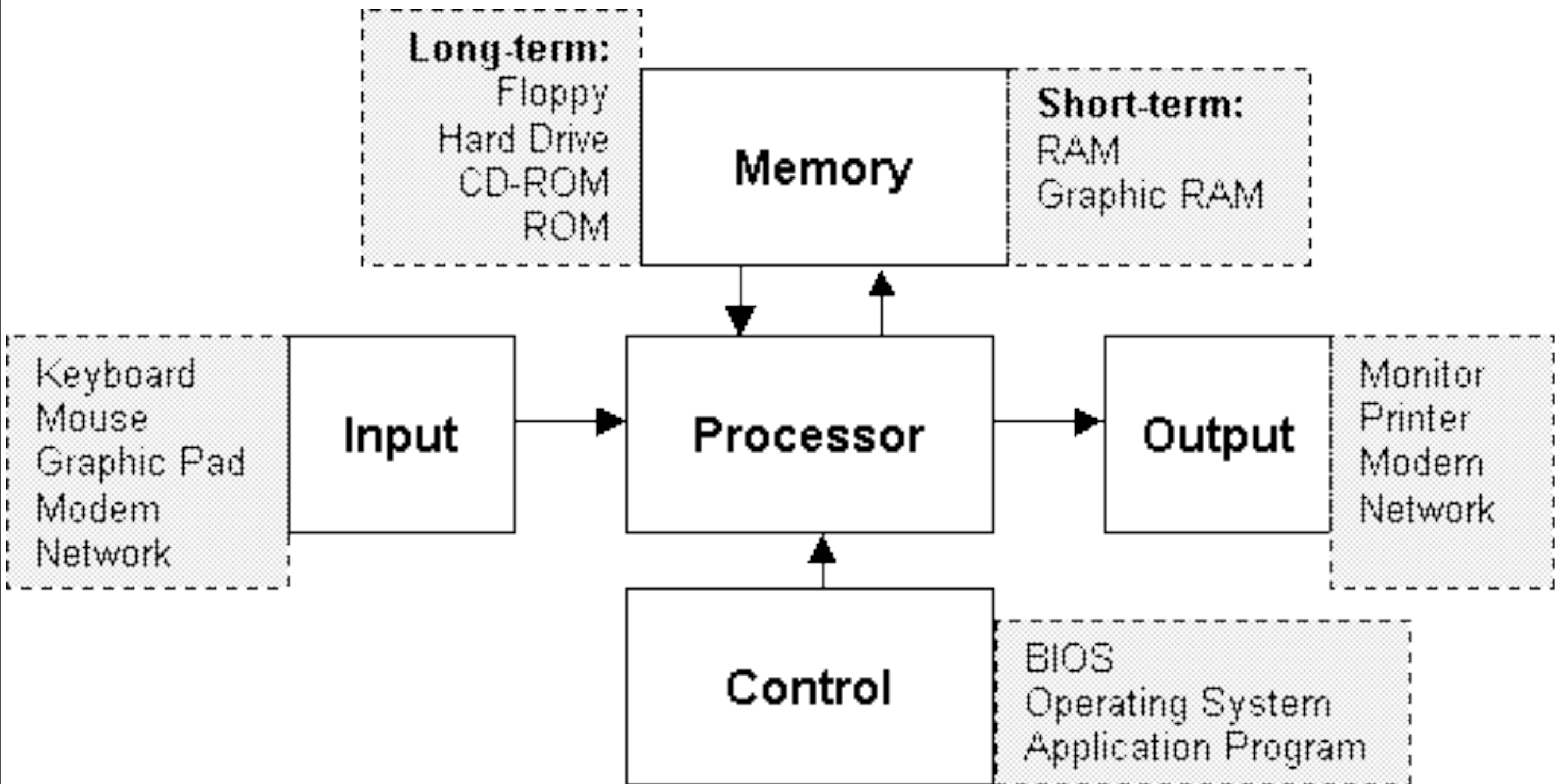








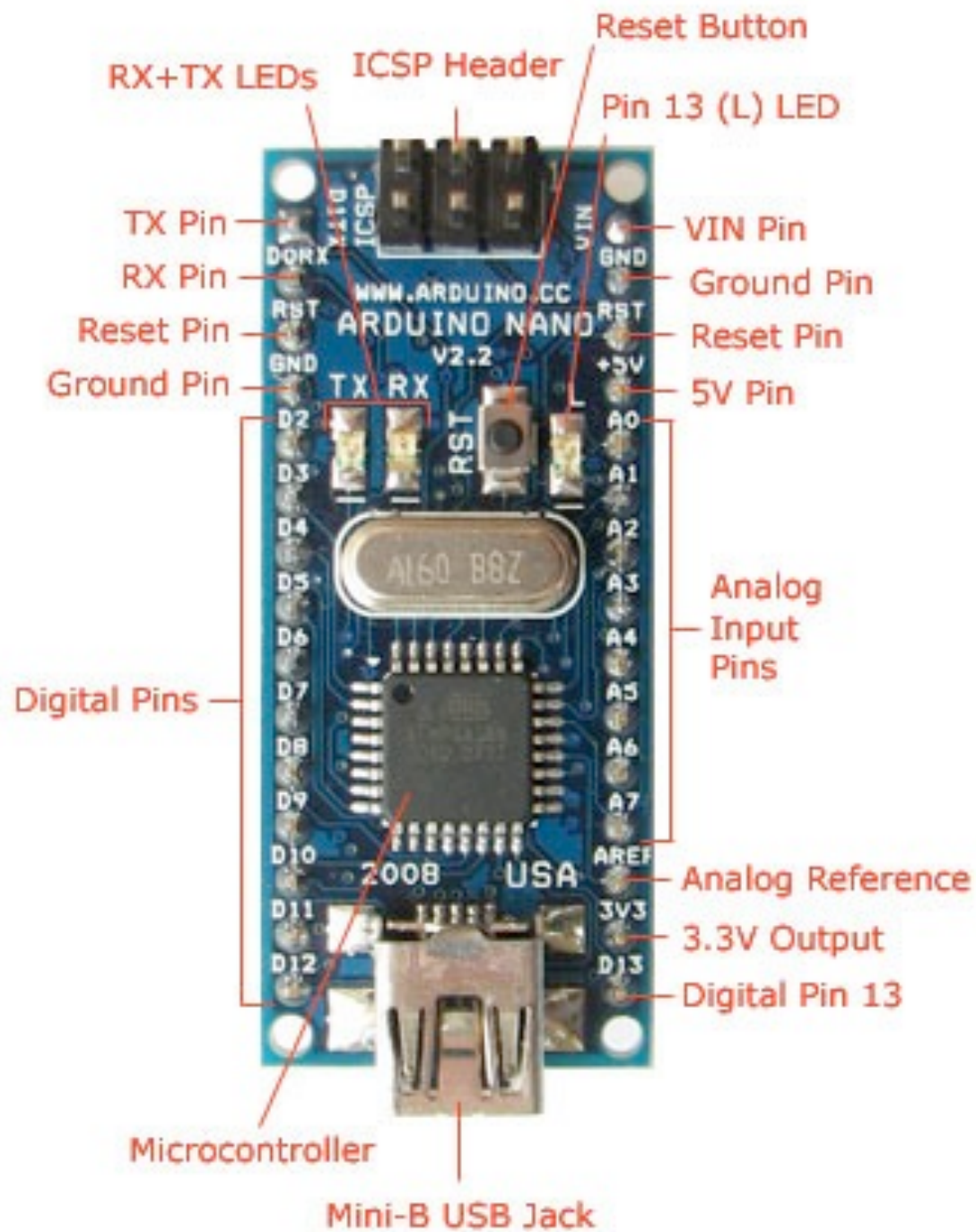
Microcontrollers are  
very small computers



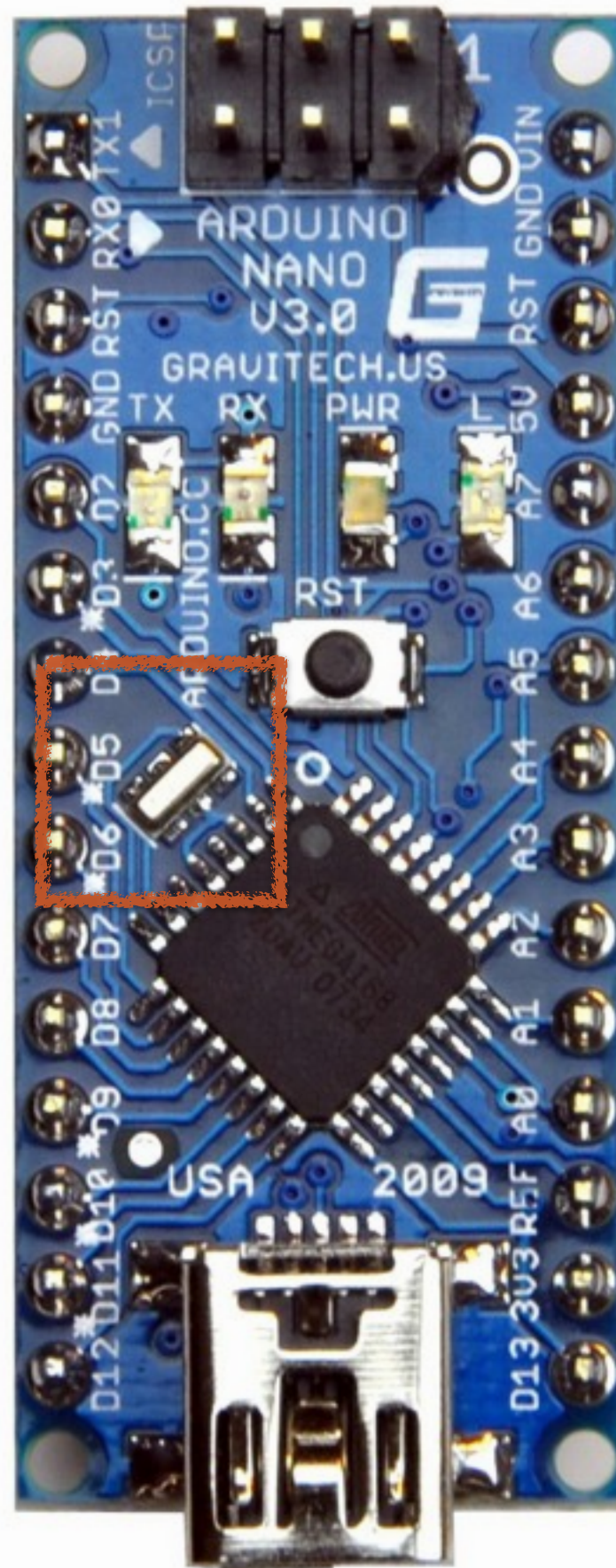
# Microcontroller Architecture

Clock | Program Memory | Data Memory | Registers | Code

# Physical Hardware

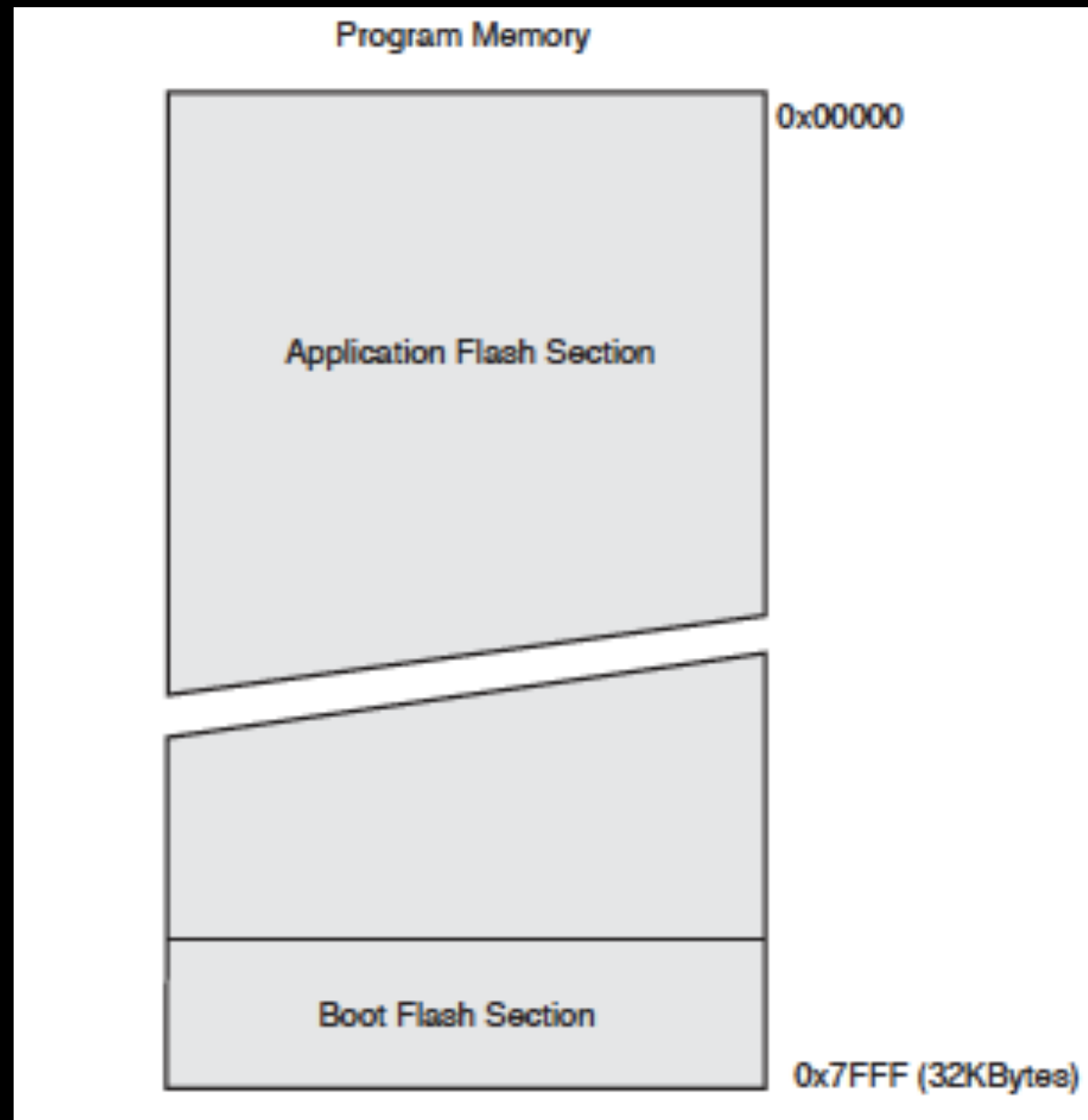


Clock





# Program Memory



# Data Memory

	Bit								Value in Memory (hex)      (decimal)	
	MSB							LSB		
	7	6	5	4	3	2	1	0		
\$0000	1	0	0	1	1	0	1	0	0x9A	154
\$0001	0	1	1	0	1	1	1	1	0x6F	111
\$0002	0	0	1	1	1	1	1	0	0x3E	62
	...									
\$07FD										
\$07FE										
\$07FF										
Address										

## Bits and Bytes:

- ❑ 1 byte = 8 bits, 256 unique values for each byte
- ❑ All the information in the microcontroller is stored in byte-size chunks; we represent each byte of information as a two-digit hexadecimal number.
- ❑ 11110011 in binary = 243 in decimal = F3 in hexadecimal
- ❑ b11110011 = 0xF3
- ❑ Memory addresses are hex, as well, but preceded with \$, e.g. \$03DF.



## I/O Registers:

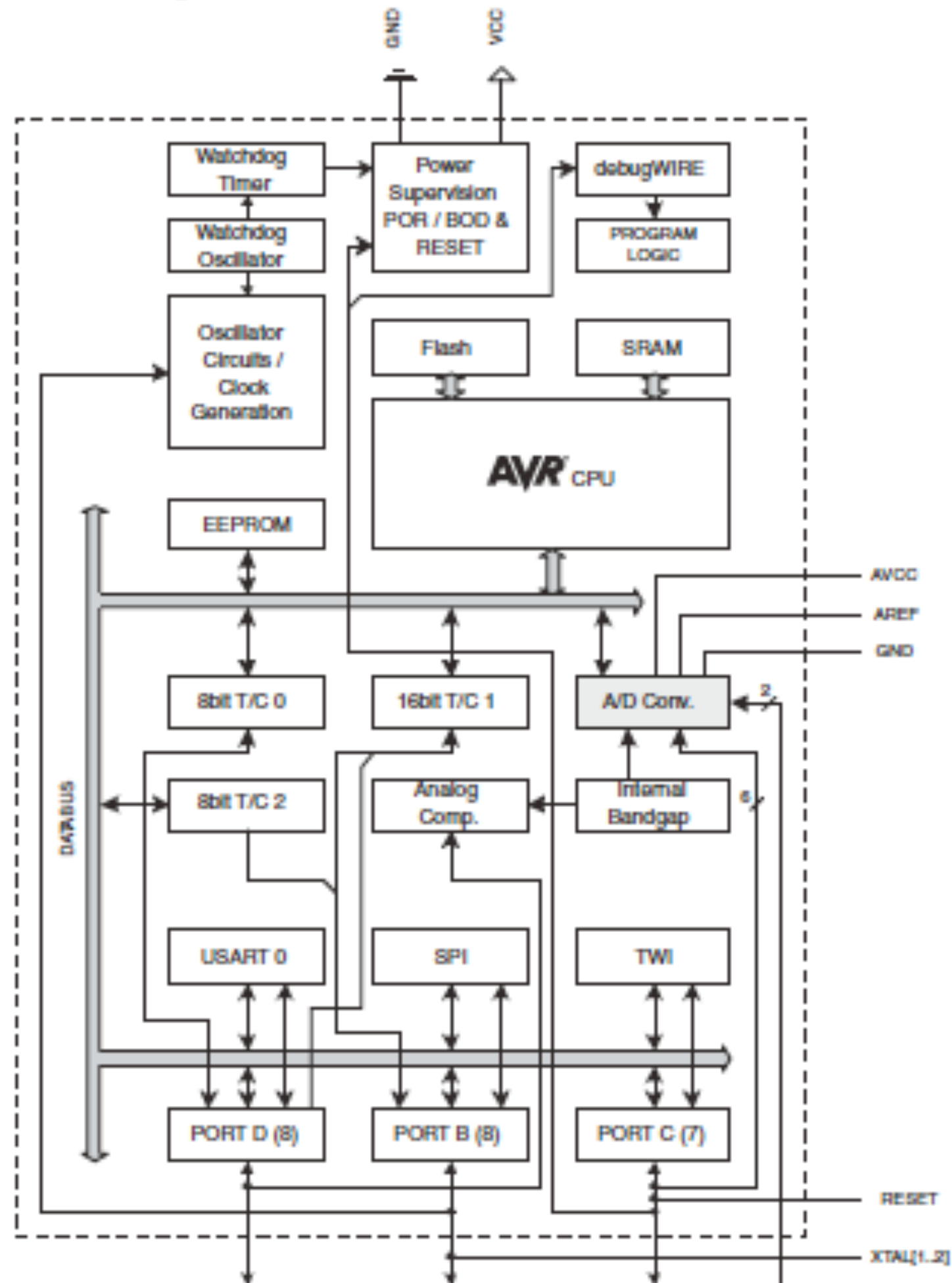
PORT B: (PB7-PB0) 8-bit bi-directional I/O

PORT C: (PC 7, 6) 8-bit bi-directional I/O

PORT D: (PD7-0) 8-bit bi-directional I/O

PORT F: (PF7-4, PF1, PF0): analog inputs to A/D converter  
(can be used at 8-bit bi-directional I/O)

Figure 2-1. Block Diagram



## Data Direction Registers (DDR):

Since the I/O pins are configurable to be either input or output, the controller needs some place to store the directionality of each bit.

These are stored in the Data Direction Registers. Like all the other registers, the DDRs have 1's and 0's, but its 1's and 0's indicate whether the corresponding port pin is an input (0) or output (1).

## Port Features:

Analog to Digital Conversion

Pulse Width Modulation

Timers & Counters

External Interrupts

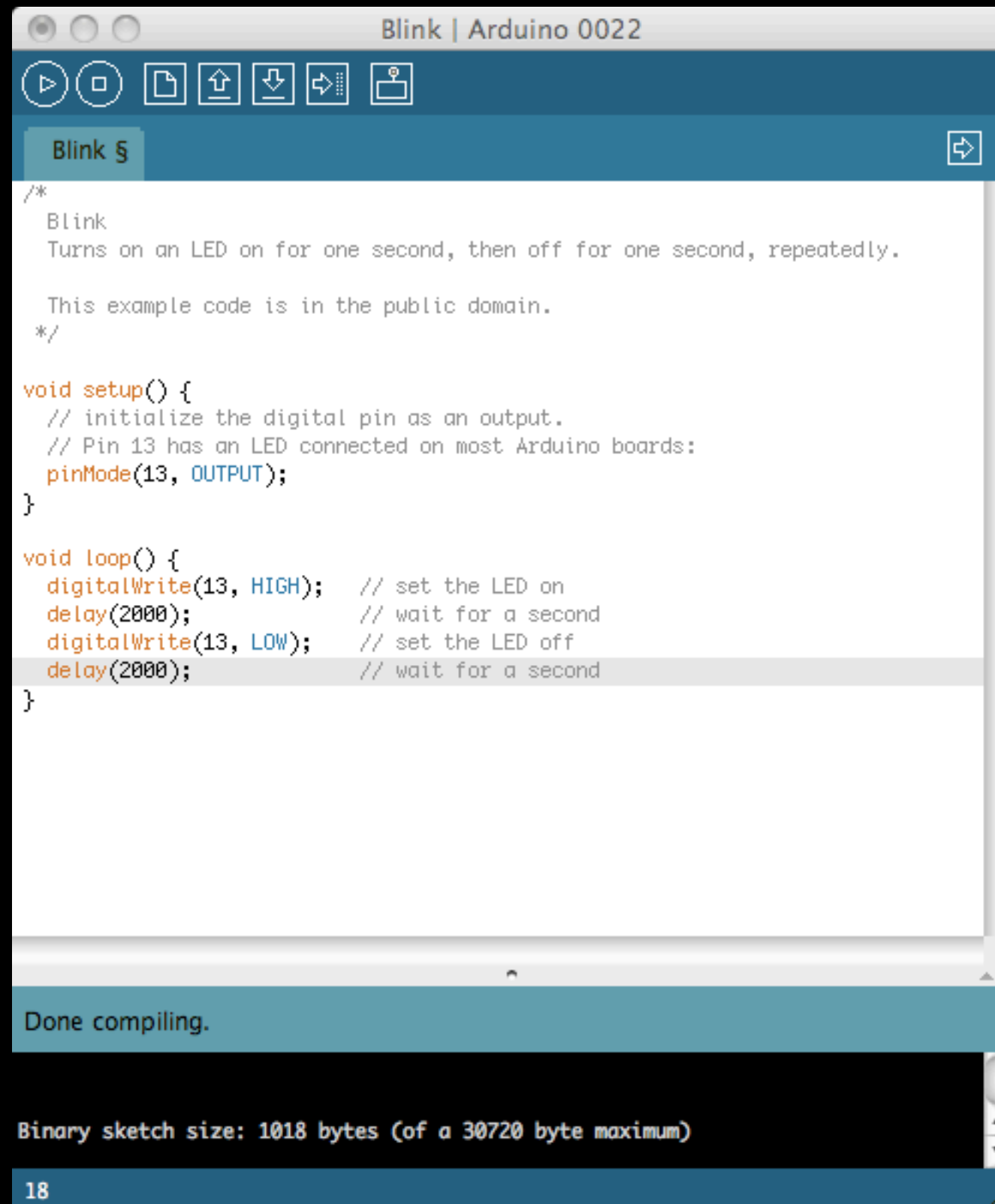
Serial Peripheral Interface

RX/TX

# Arduino Software Environment

IDE | Structure of Arduino programs | Flashing programs

# Sketch



The screenshot shows the Arduino IDE interface. The title bar reads "Blink | Arduino 0022". The toolbar contains icons for running, stopping, saving, opening, and other functions. The sketch name "Blink §" is displayed in the top right. The code editor contains the following text:

```
/*  
  Blink  
  Turns on an LED on for one second, then off for one second, repeatedly.  
  
  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(2000);             // wait for a second  
  digitalWrite(13, LOW);  // set the LED off  
  delay(2000);             // wait for a second  
}
```

At the bottom, a status bar indicates "Done compiling." and "Binary sketch size: 1018 bytes (of a 30720 byte maximum)". The line number "18" is shown in the bottom left corner.

# Sketch

```
/*
```

```
  Blink
```

```
  Turns on an LED on for one second, then off for one second, repeatedly.
```

```
  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(2000);           // wait for a second
```

```
  digitalWrite(13, LOW); // set the LED off
```

```
  delay(2000);           // wait for a second
```

```
}
```

## What happens when we flash code?

1. Code from libraries (if any) are included (linked).
2. Code is checked for errors (verified).
3. Code is “cross-compiled” into machine code (a.k.a machine code or hex code) using `avr-gcc`.
4. Code is written to the program memory of the Arduino over USB using `avrdude`.



# Flash Demonstration