Microcontrollers

Press Play: Interactive Device Design | June 30, 2011

Basic Sensor Circuit

Resistors | Voltage Divider | Sensor Circuits

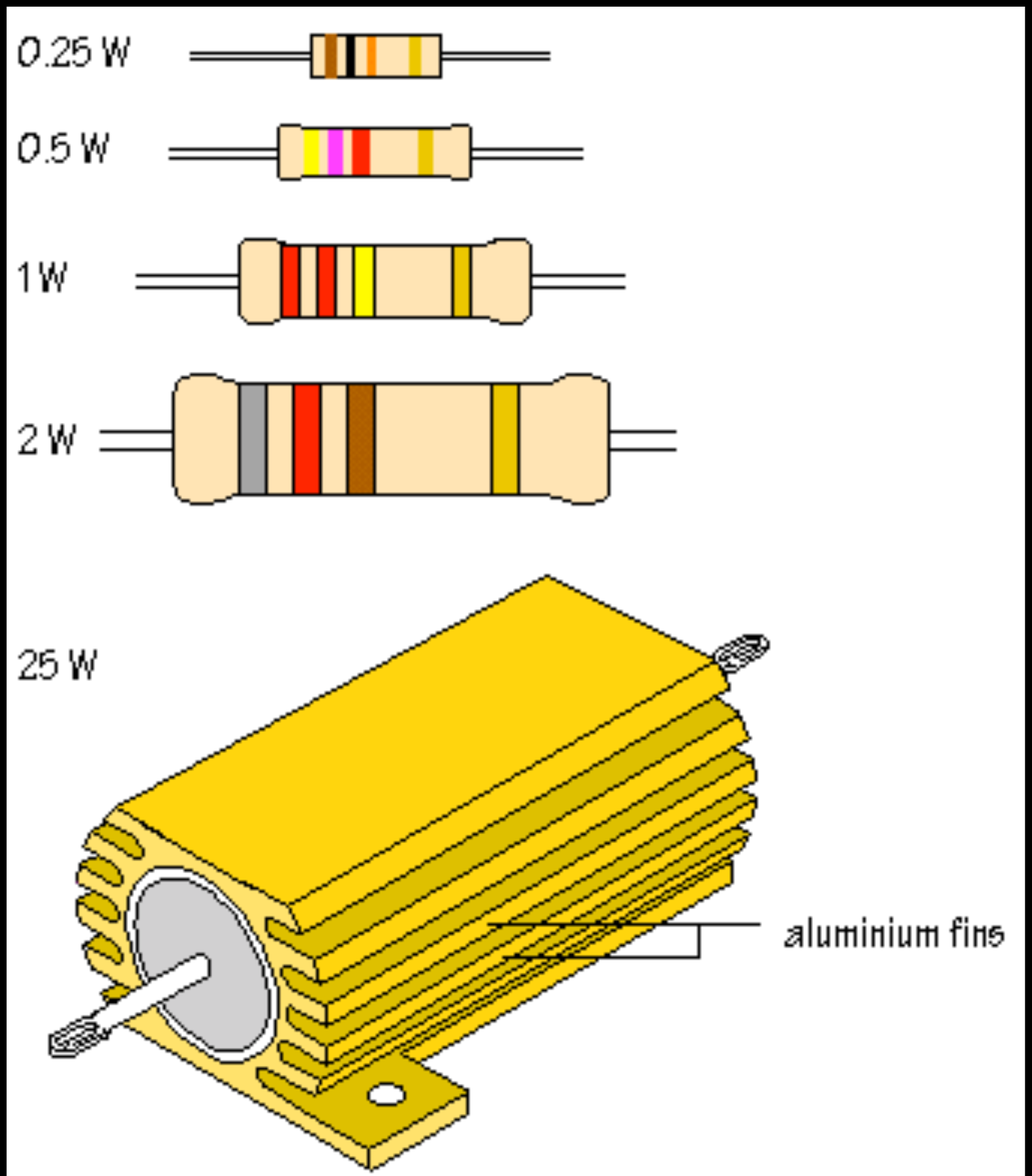
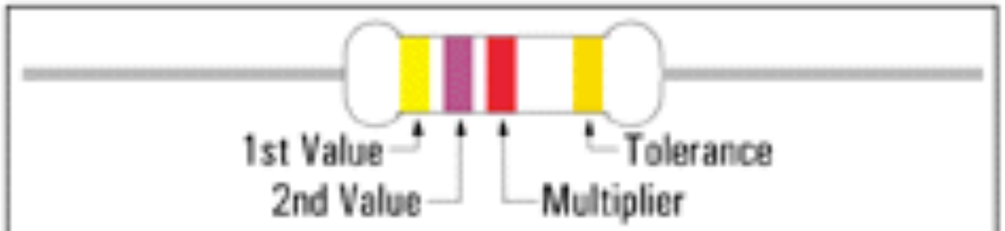


Image from www.doctronics.co.uk/images/res18.gif

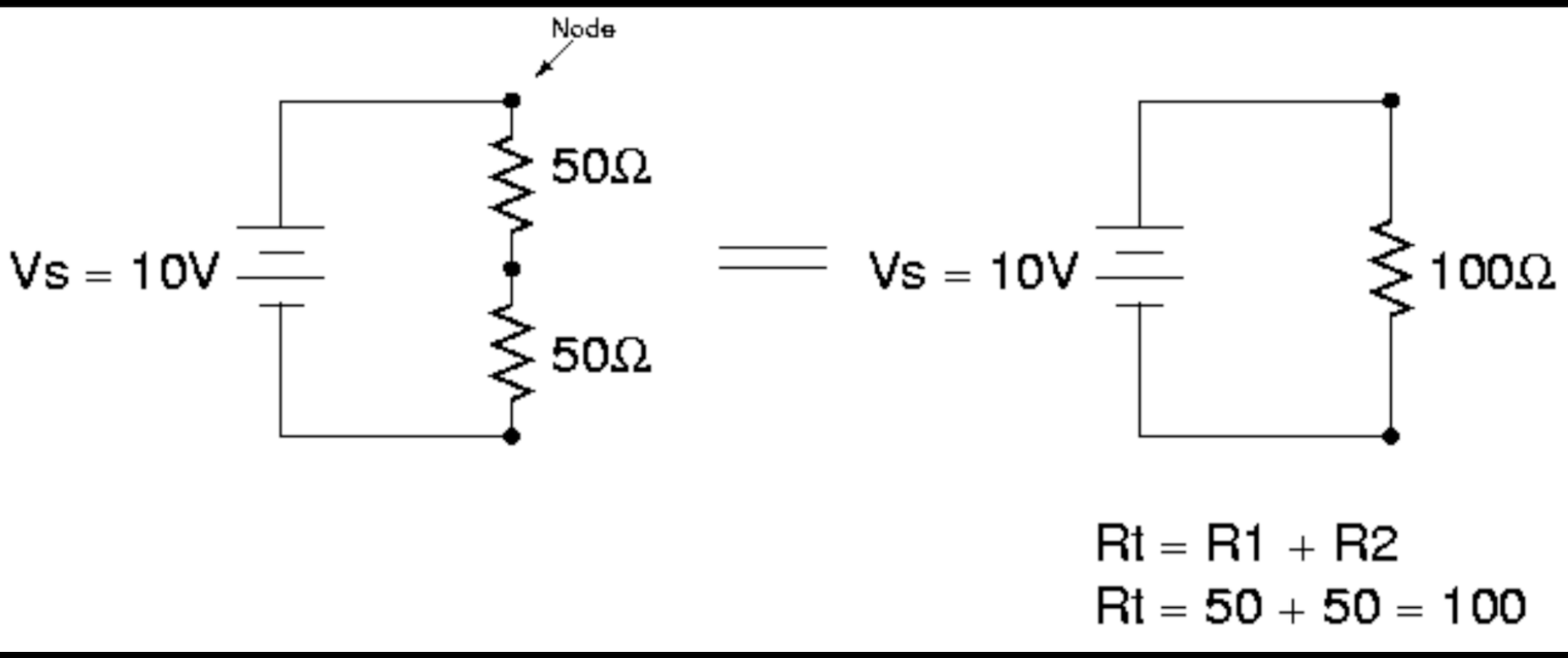
READING RESISTANCE VALUES



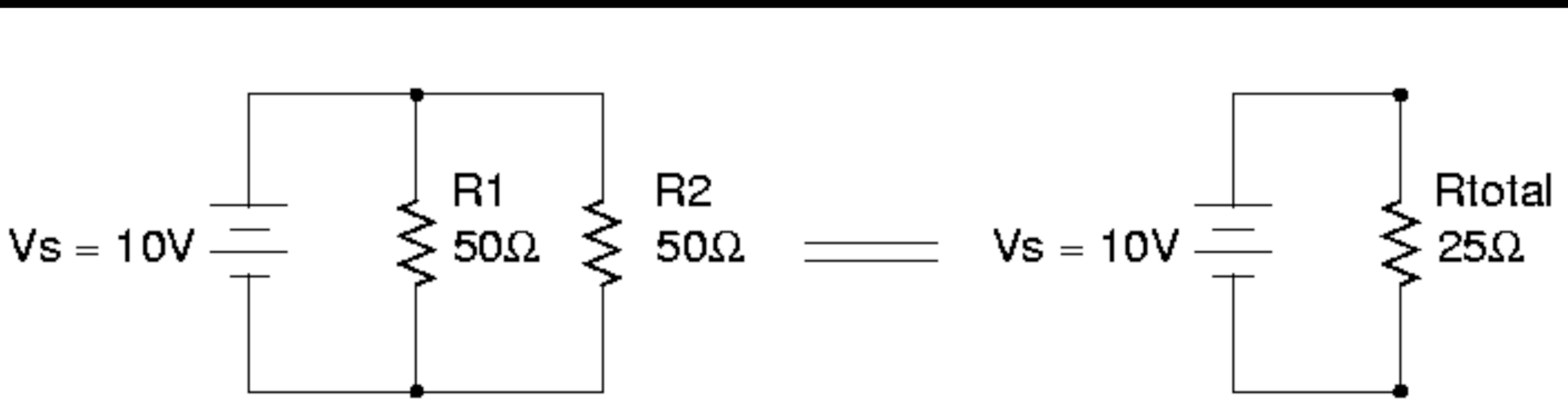
Example: 4 7 x100 –5%
4700Ω–5%

COLOR	VALUE	MULTIPLIER	TOLERANCE
Black	0	1	-
Brown	1	10	–1%
Red	2	100	–2%
Orange	3	1K	-
Yellow	4	10K	-
Green	5	100K	–.5%
Blue	6	1M	–.25%
Violet	7	10M	–.1%
Gray	8	100M	–.05%
White	9	1000M	-
Gold	-	1/10	–5%
Silver	-	1/100	–10%
None	-	-	–20%

Resistors in series combine as their sum



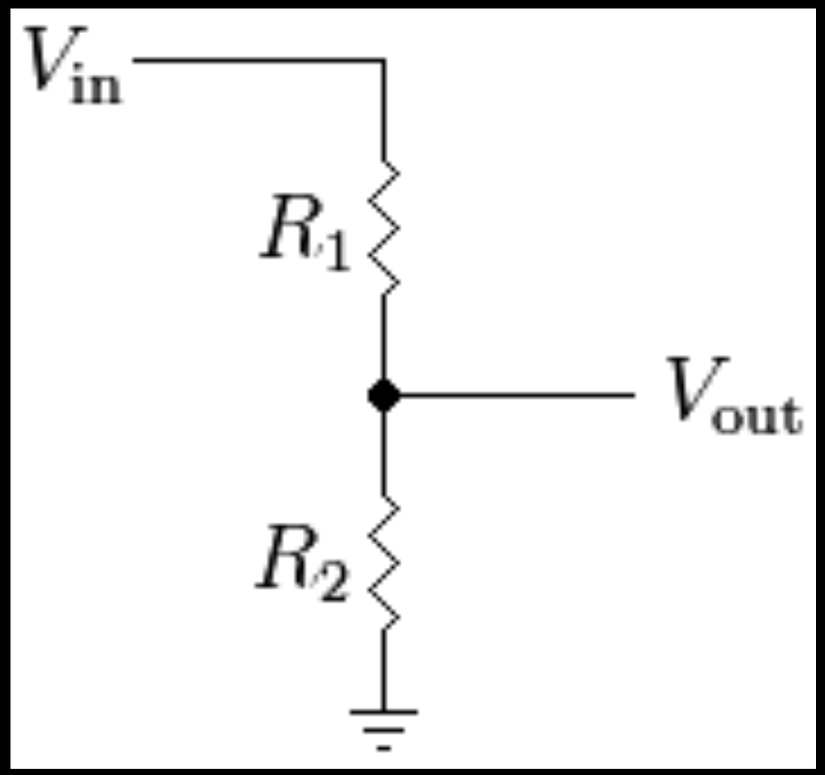
Resistors in parallel combine as a ratio



For Parallel Circuits:

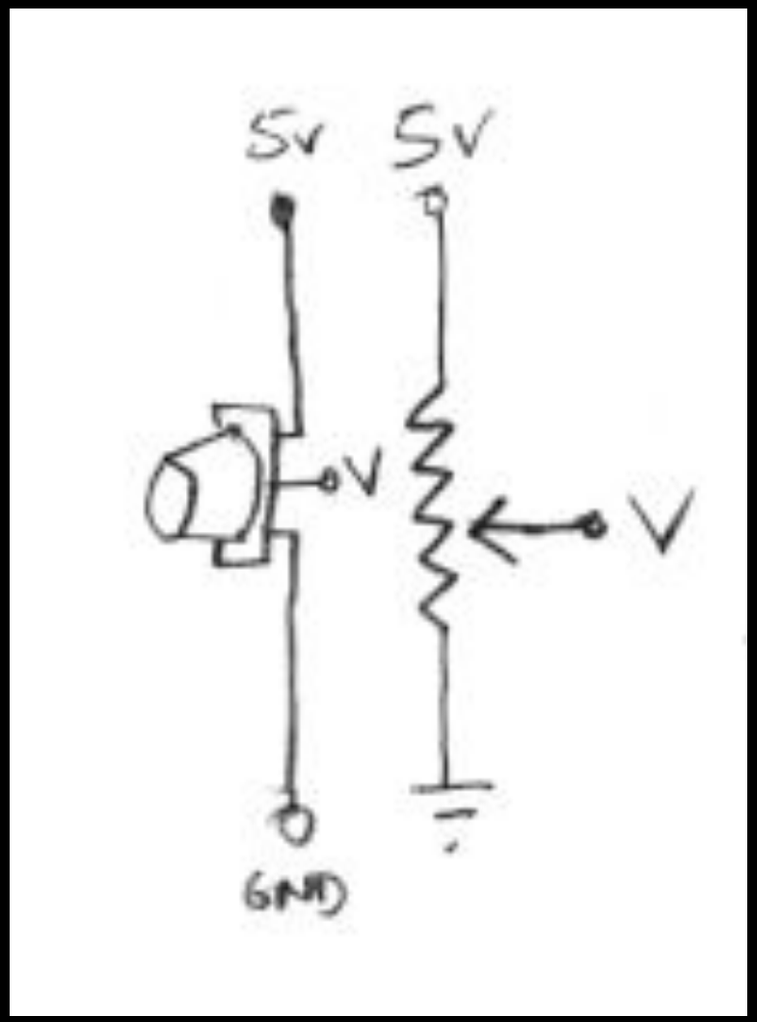
$$R_{total} = (R1 * R2) / (R1 + R2)$$

A Voltage Divider Circuit

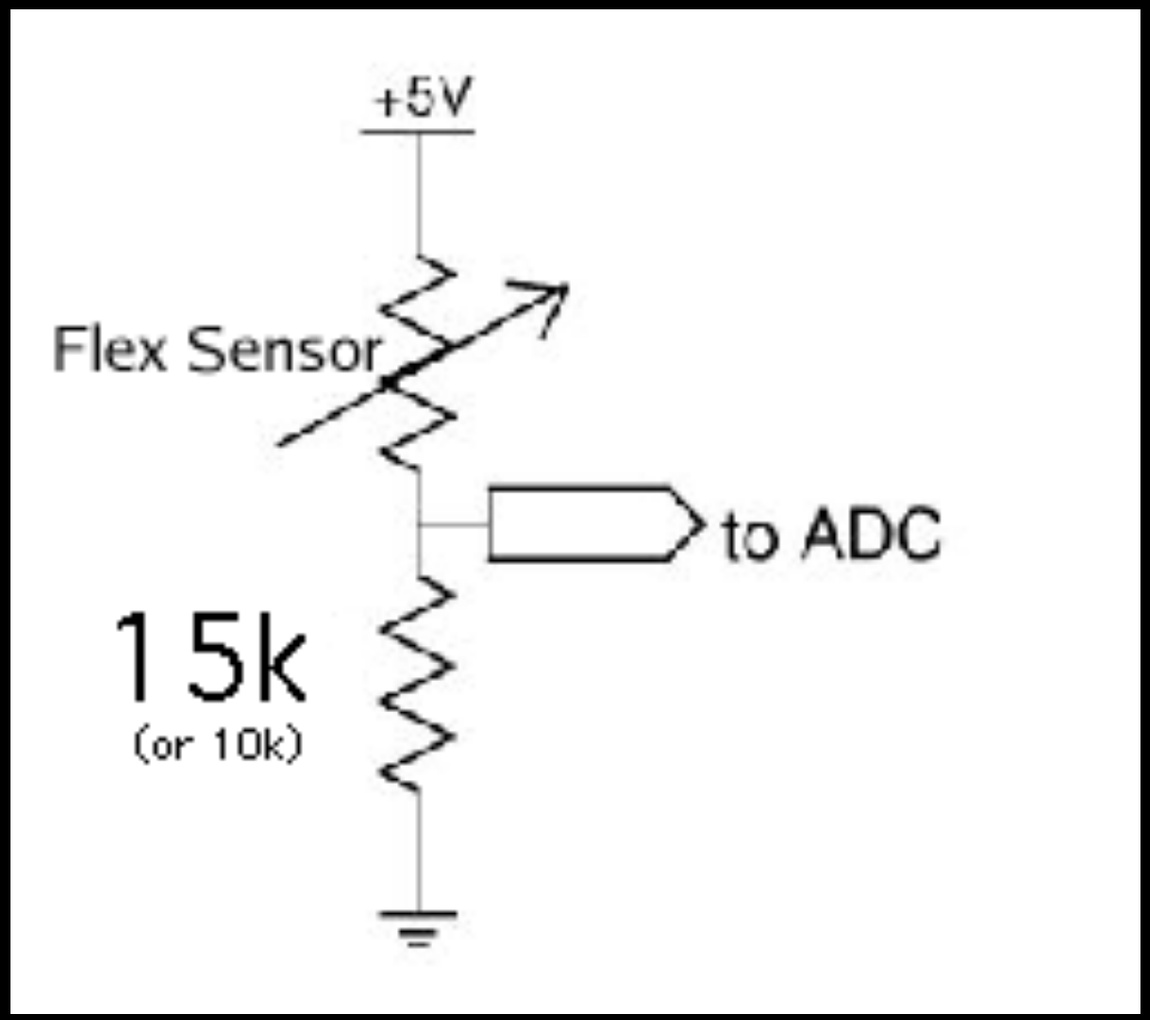


$$V_{out} = \frac{R2}{R1 + R2} \times V_{in}$$

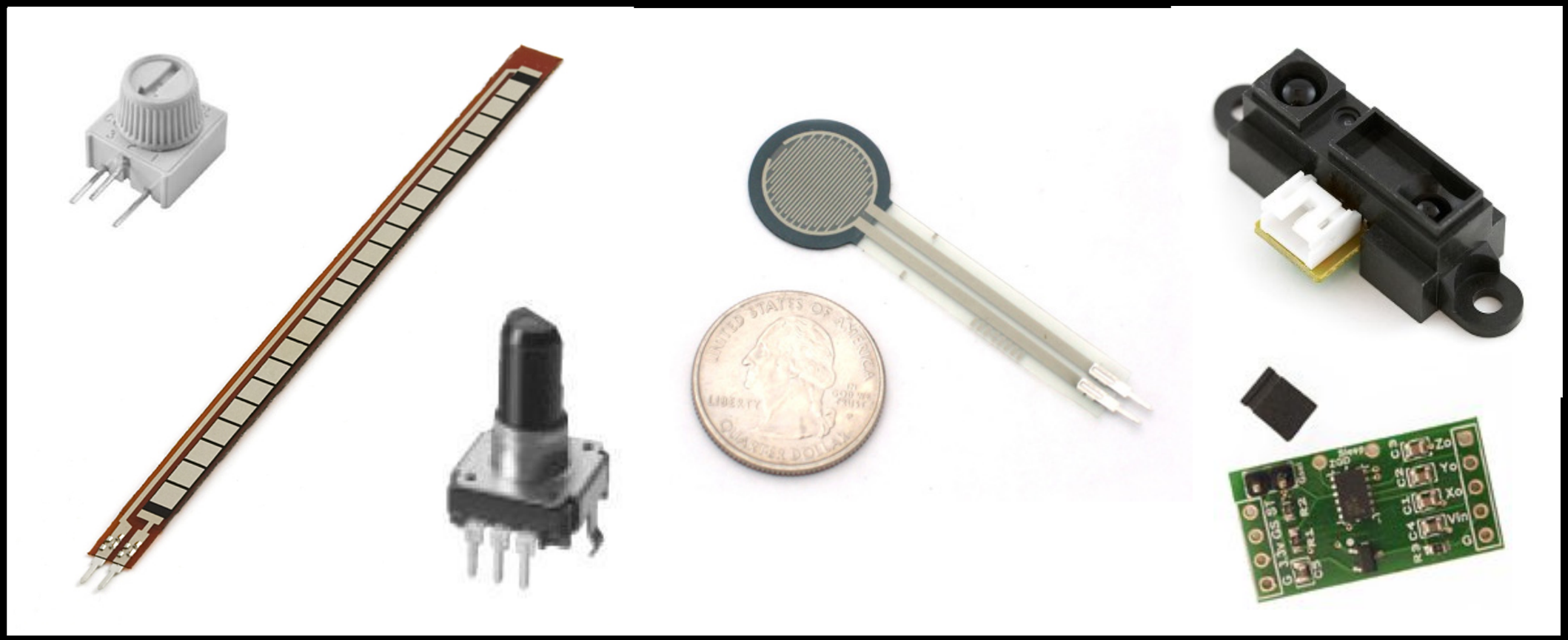
A Potentiometer Circuit



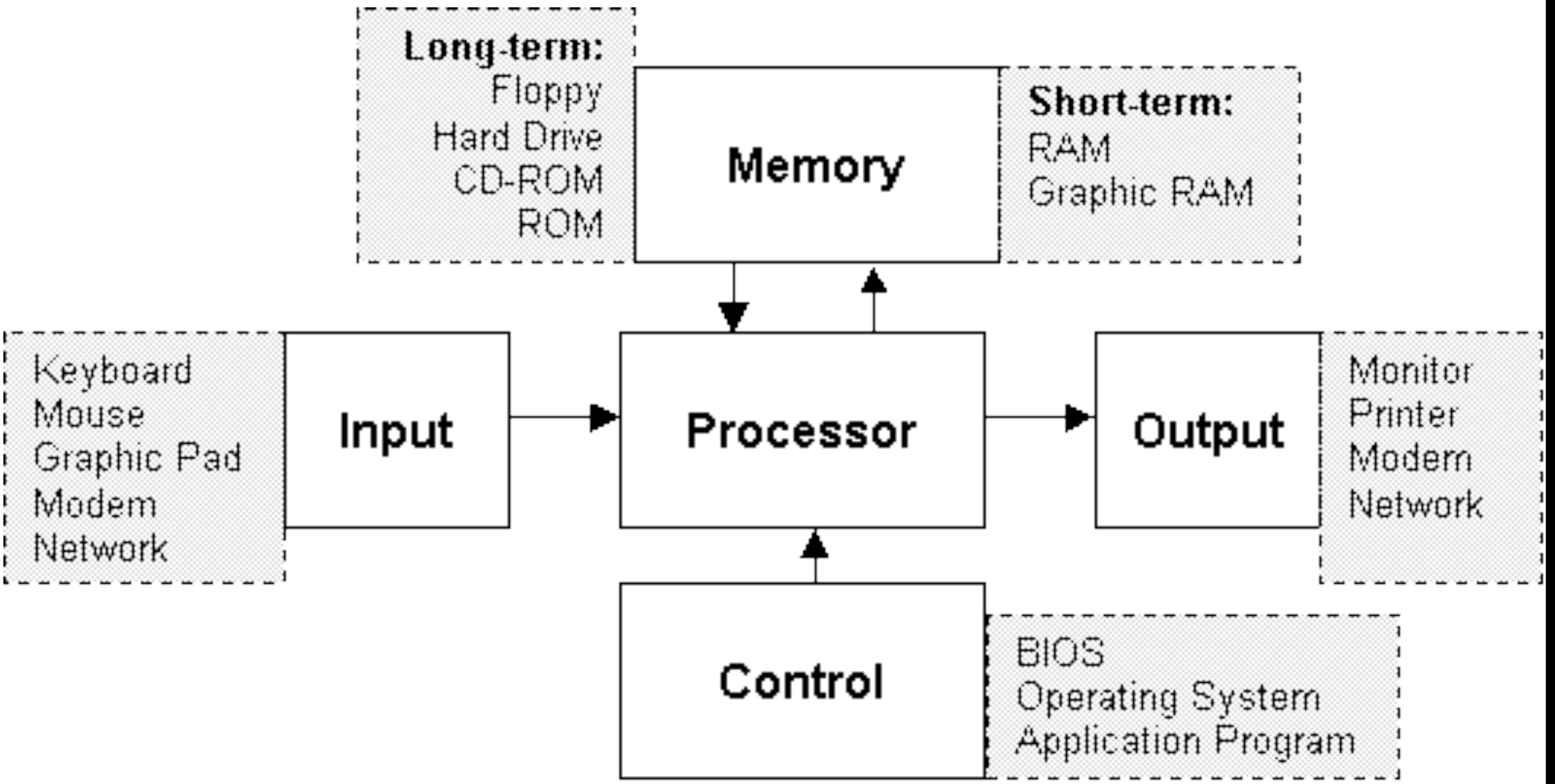
A Flex Sensor Circuit



Your Lab Kit Sensors



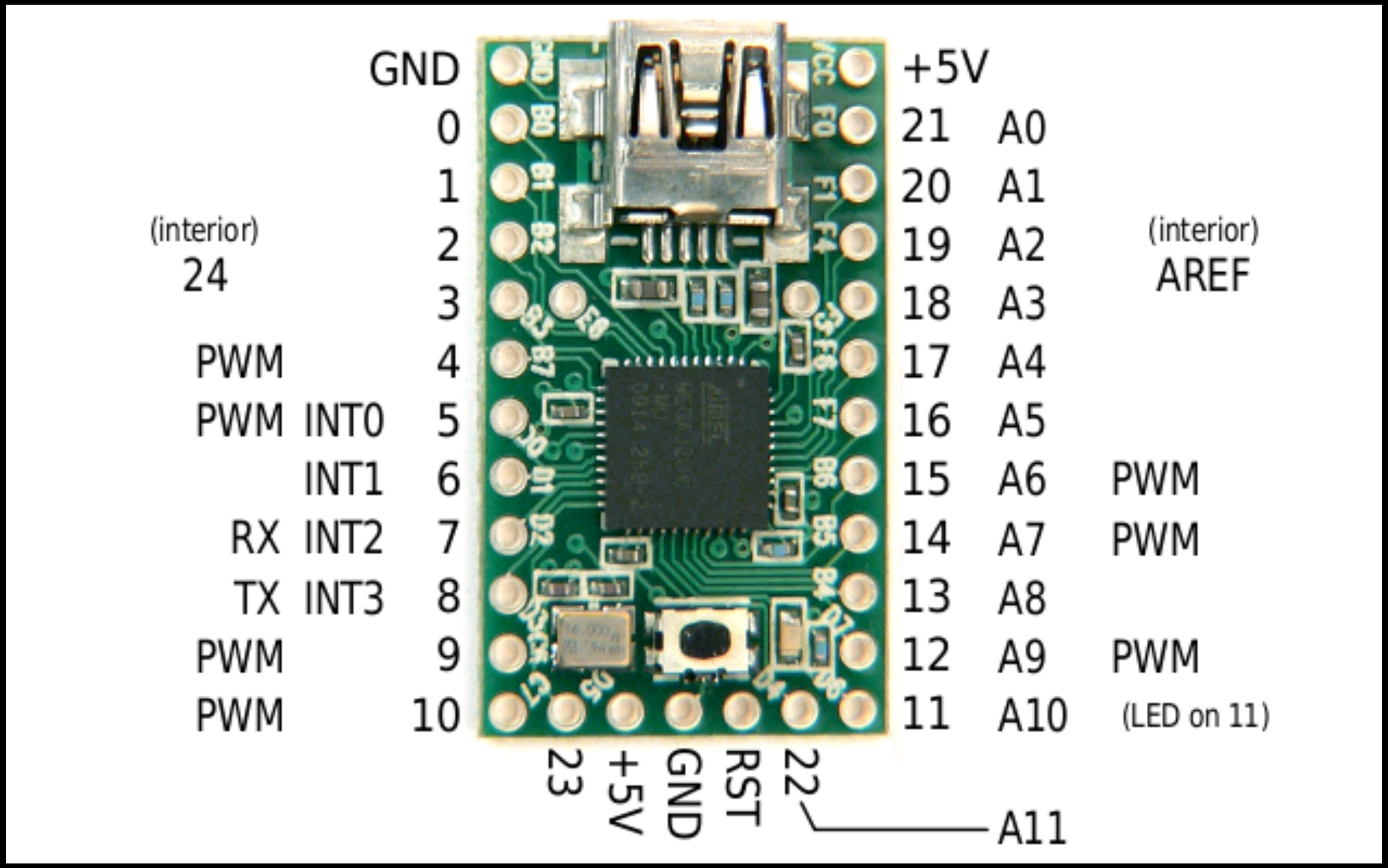
Micro-Controllers Are Very Small Computers


Long-term:
Floppy
Hard Drive
CD-ROM
ROM
Memory
Short-term:
RAM
Graphic RAM
Keyboard
Mouse
Graphic Pad
Modem
Network
Input
Processor
Output
Monitor
Printer
Modem
Network
Control
BIOS
Operating System
Application Program

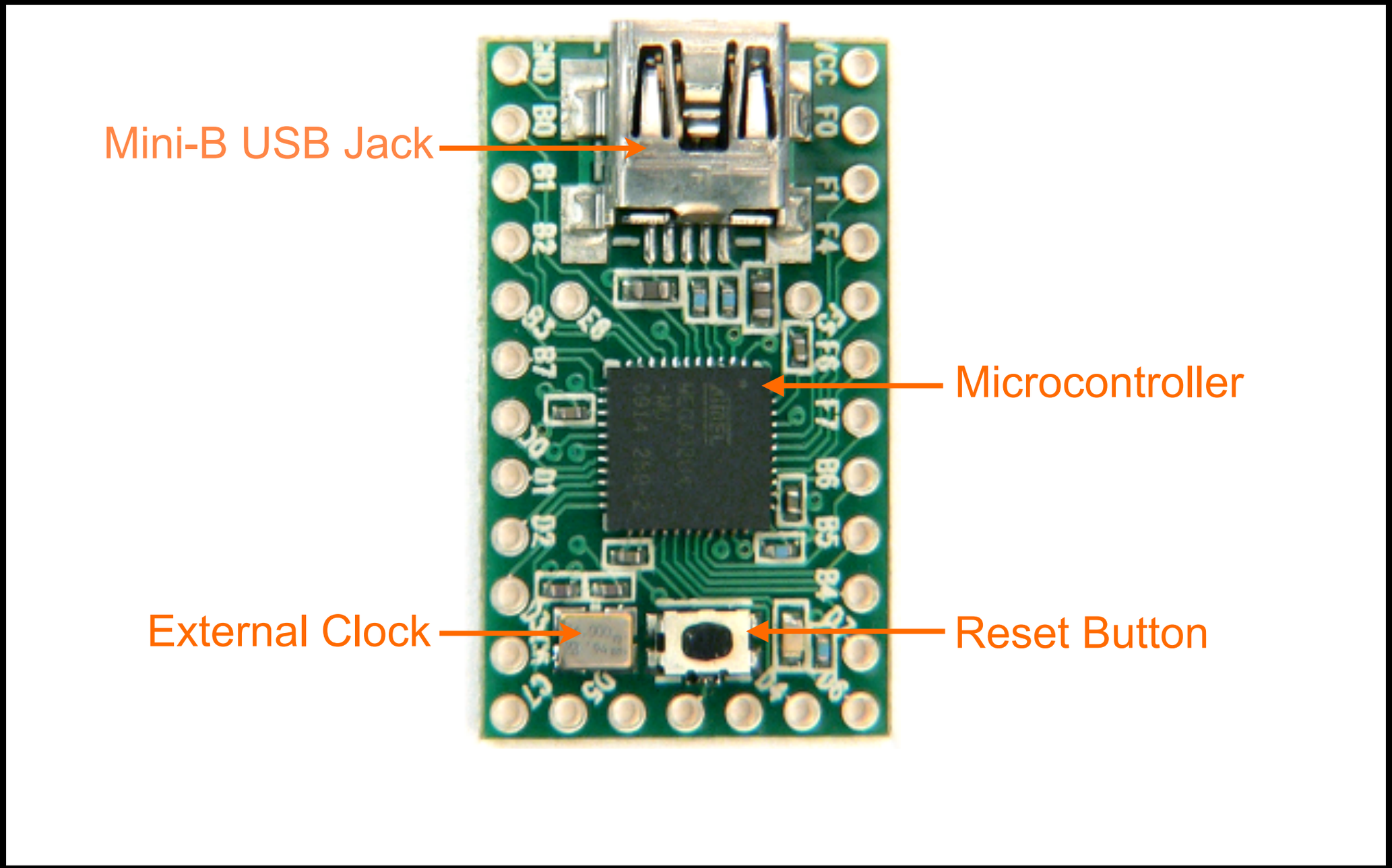
Microcontroller Architecture

Clock | Program Memory | Data Memory | Registers | Code

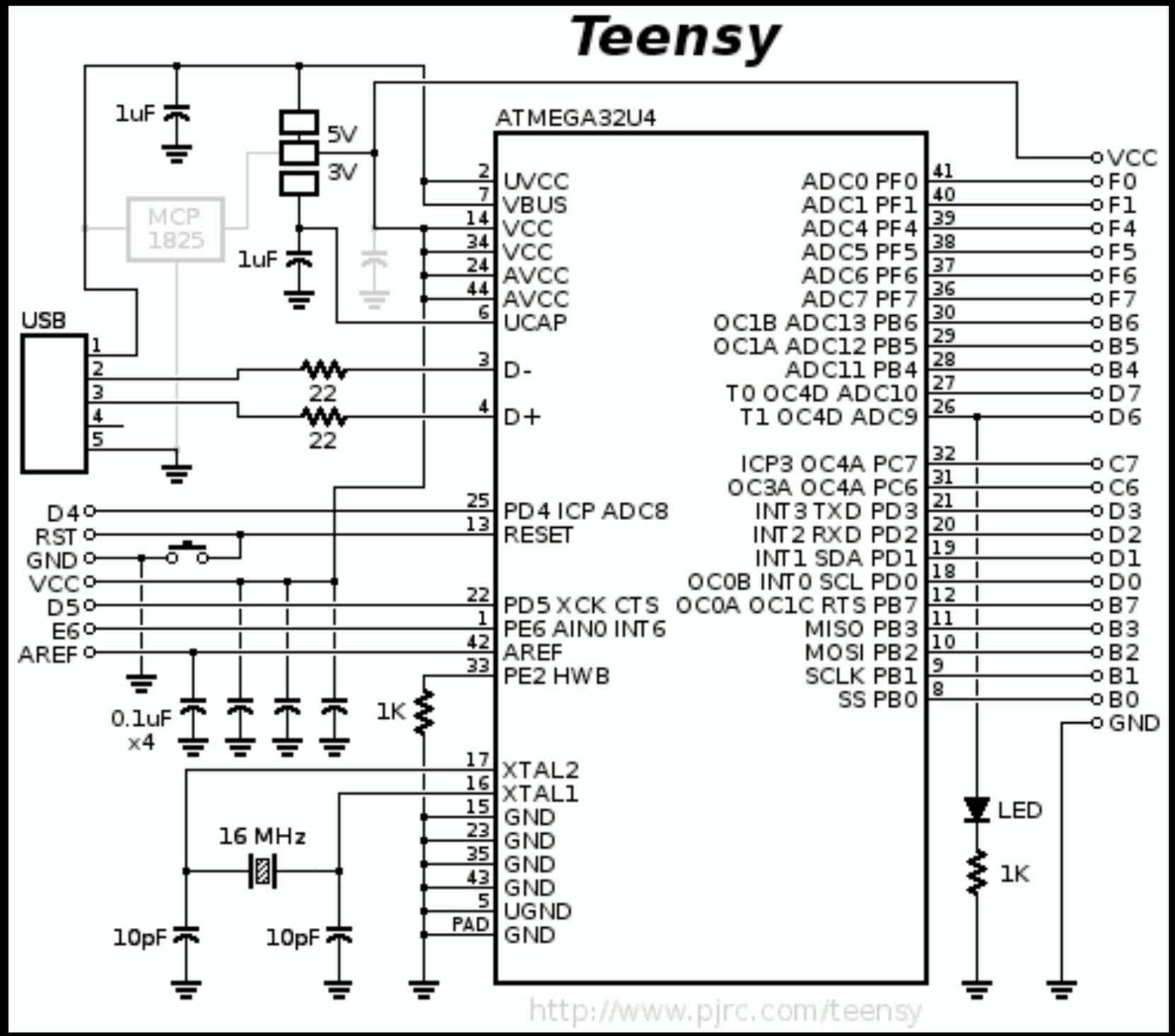
Physical Hardware:



Physical Hardware:



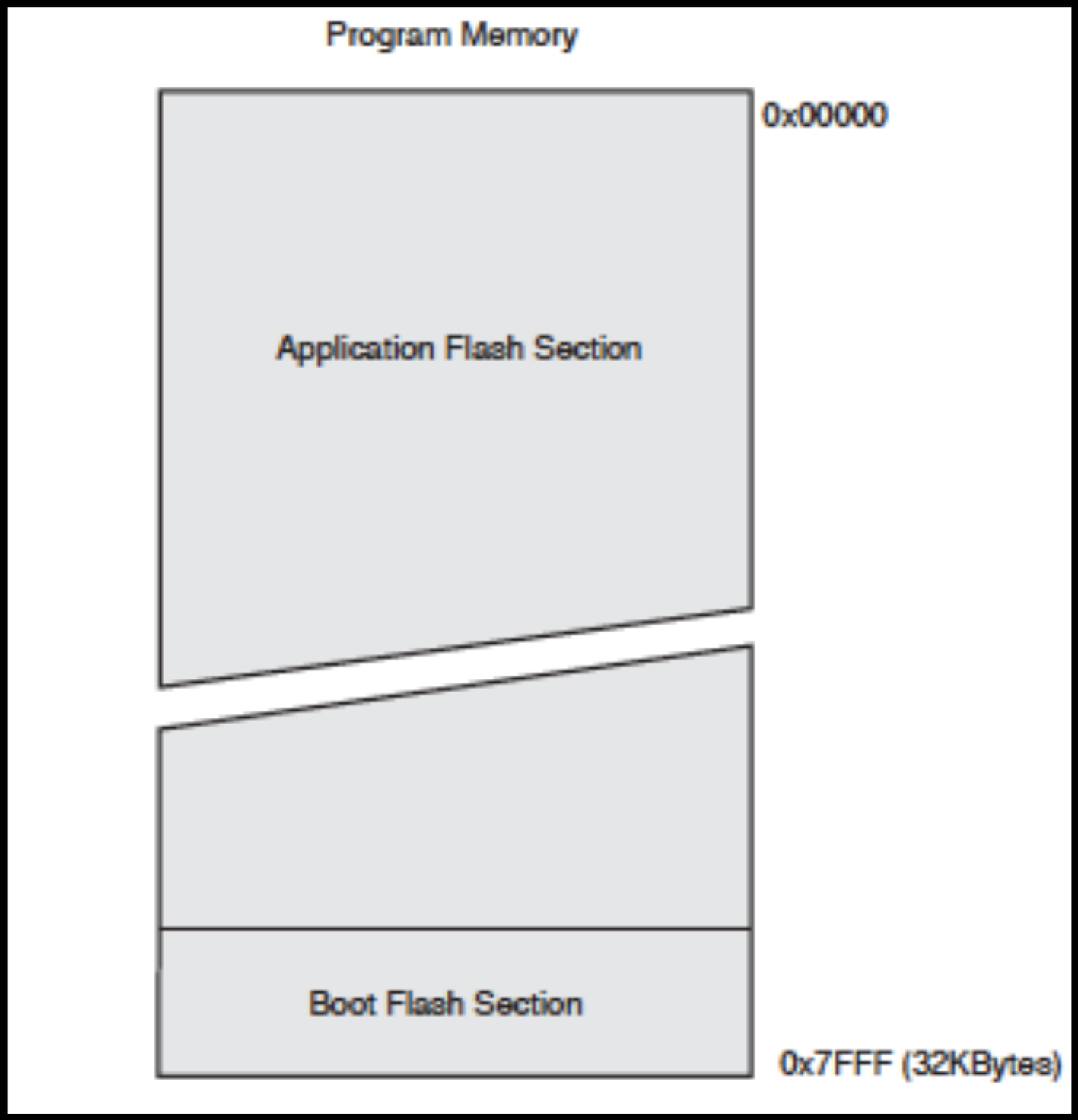
Physical Hardware:



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 decimel = F3 in hexadecimal
- b11110011 = 0xF3
- Memory addresses are hex, as well, but preceded with $, e.g. $03DF.

Program Memory:



Data Memory:

Address	Bit 7 (MSB)	6	5	4	3	2	1	0 (LSB)	Value in Memory (hex)	Value in Memory (decimal)
$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										

IO Registers:

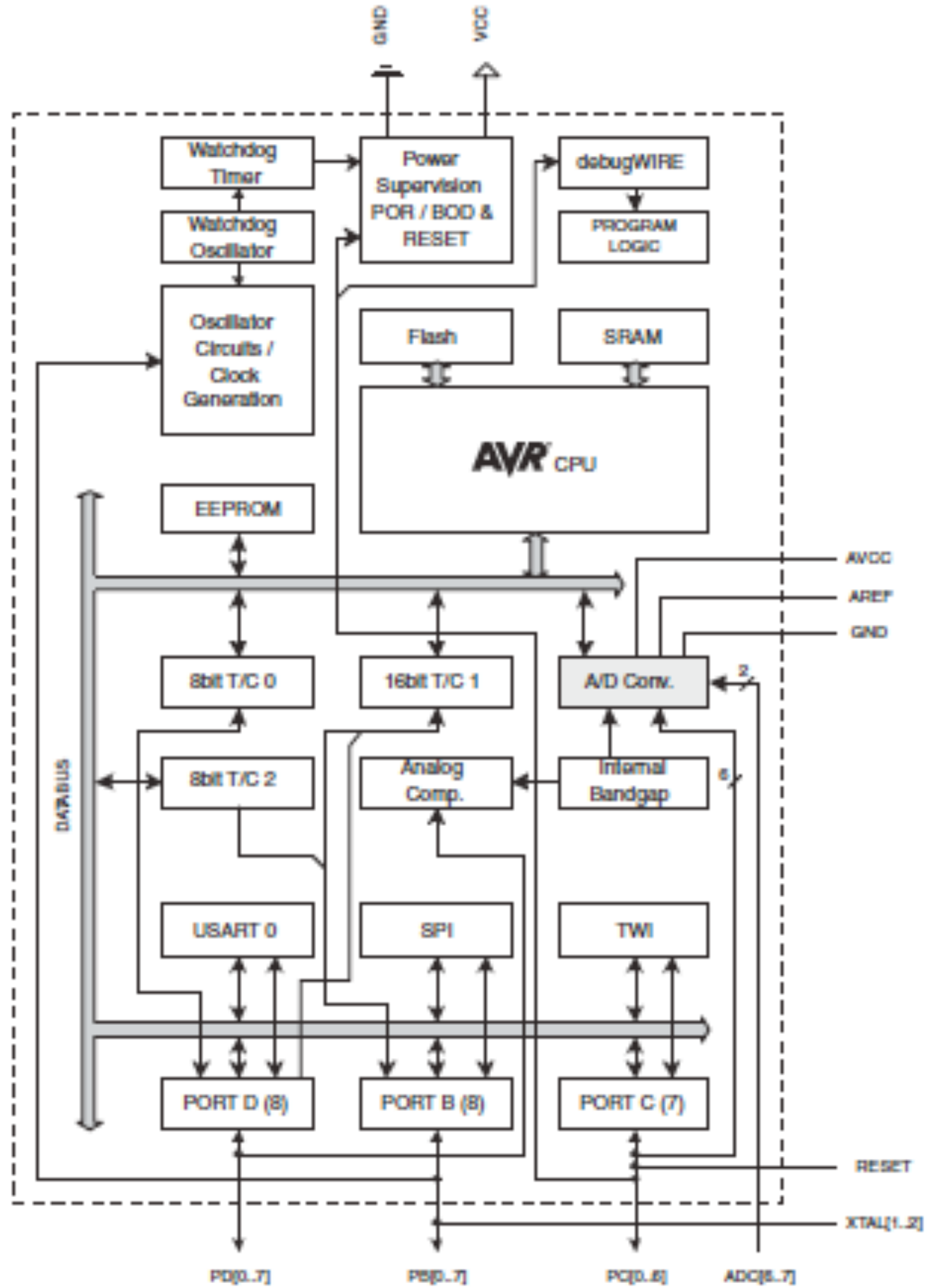
PORT B: (PB7-PB0) 8-bit bi-directional IO

PORT C: (PC 7, 6) 8-bit bi-directional IO

PORT D: (PD7-0) 8-bit bi-directional IO

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

Figure 2-1. Block Diagram



Data Direction Registers (DDR):

Since the IO 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:

Blink | Arduino 0022

Blink

```
/*
  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:
  //   Teensy 2.0 has the LED on pin 11
  //   Teensy++ 2.0 has the LED on pin 6
  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 Saving.

1

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:
  //     Teensy 2.0 has the LED on pin 11.
  //     Teensy++ 2.0 has the LED on pin 6.
  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
}
```

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 AVR over USB using the Teensy bootloader.

Flash Demonstration