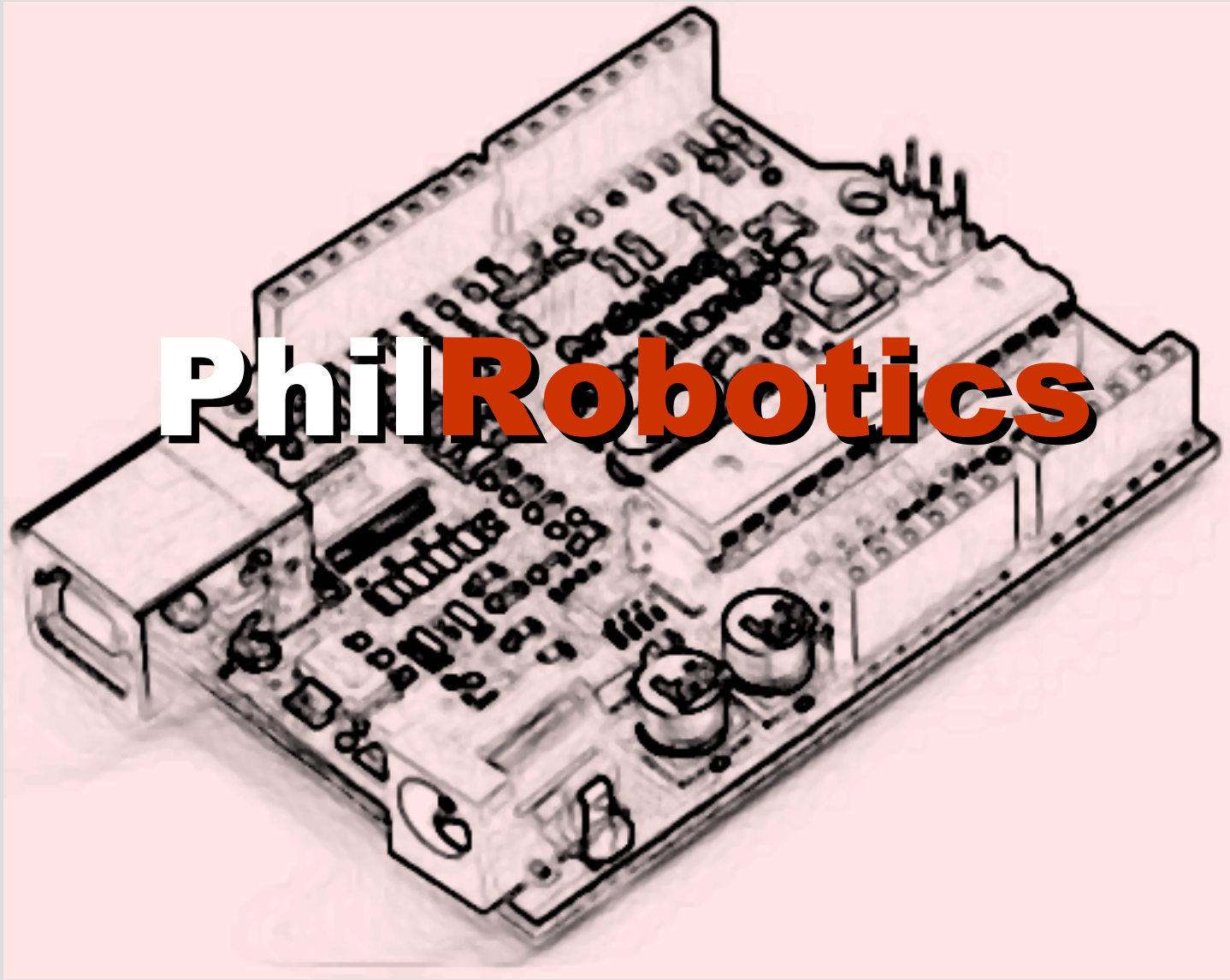
Introduction to Microcontrollers Using Arduino



Objectives

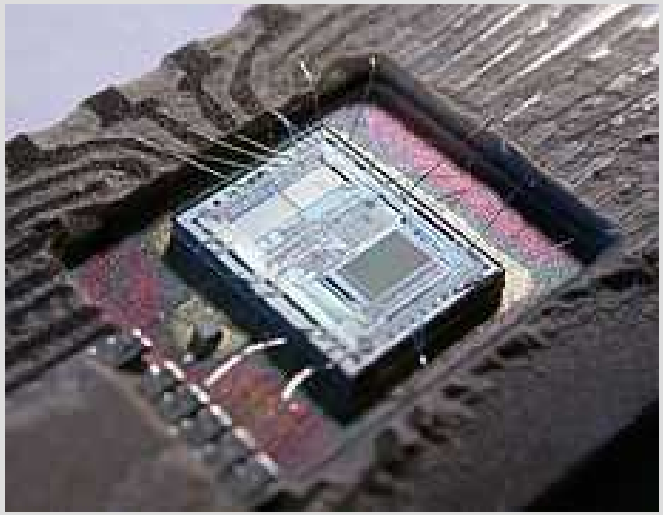
- Know ***what is a microcontroller***
- Learn the ***capabilities*** of a microcontroller
- Understand how microcontroller ***execute instructions***

Objectives

- Learn how to ***program a microcontroller*** using Arduino
  - Configure a pin (Input, Output, Analog, PWM)
  - Write Output State
  - Read Input State
  - Read Analog Input
  - Use PWM

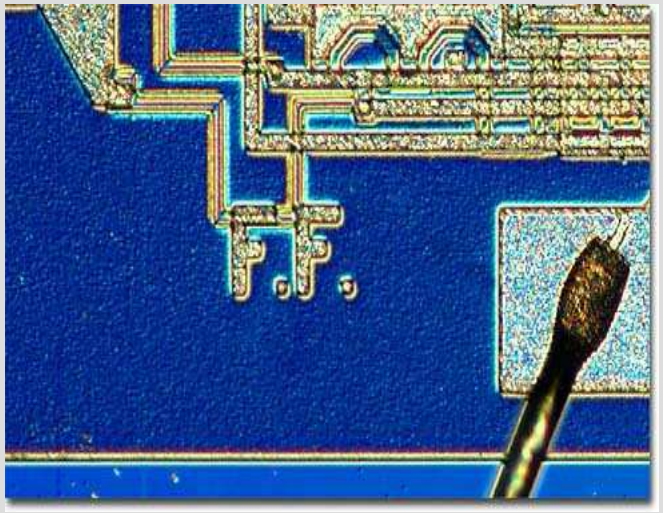
Microcontrollers

- History of Microcontrollers
  - 1971
    - Intel ***4004*** was released
    - ***First microprocessor***
    - Followed by 8008 and then 8080
  - 1975
    - Intel 8048 was released
    - Intel President, ***Luke J. Valenter***
    - ***First Microcontroller***



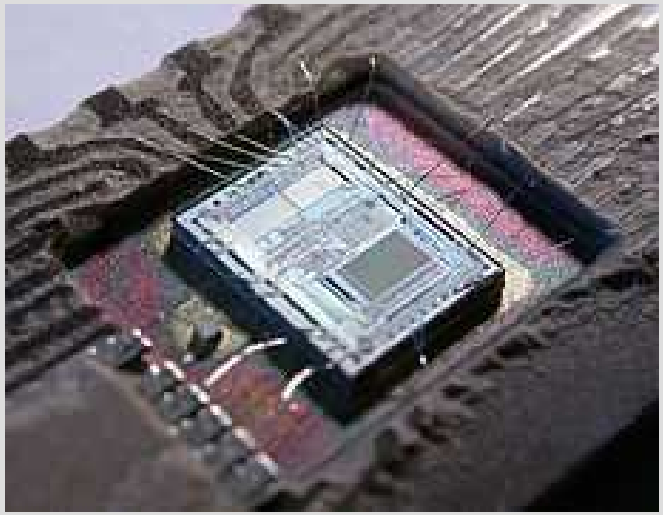
Microcontrollers

- History of Microcontrollers
  - 1976
    - ***Zilog, founded by Federico Faggin***
    - Z80
    - Gameboy, TI Calculators
  - 1979
    - Decided to shift to embedded space
    - ***Introduced Z8***



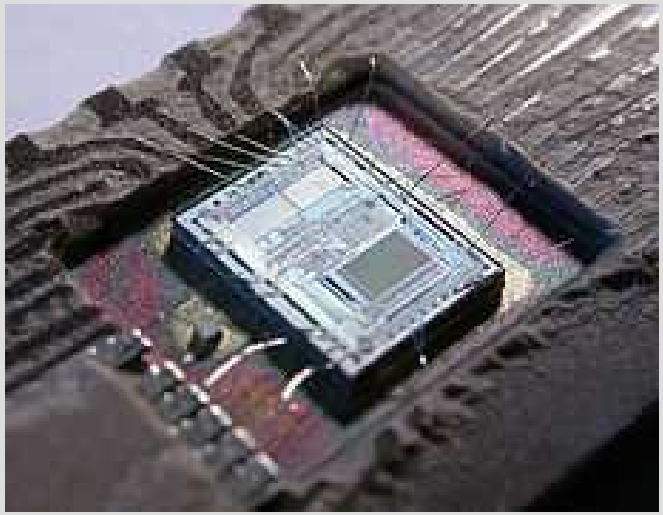
Microcontrollers

- History of Microcontrollers
  - 1993
    - Microchip
    - PIC16x84 was released
    - ***First with EEPROM*** program memory
  - 1993
    - Atmel
    - First microcontroller ***with Flash memory***



Microcontrollers

- History of Microcontrollers
  - At present
    - Several companies followed the use of ***flash memory***



Microcontrollers

- Abstract Definition
  - An IC (Integrated Circuit)
  - Configurable Pins
  - Programmable Function
  - With ***Built-in Memory***
    - RAM
    - ROM

Microcontrollers

- Abstract Definition
  - An IC (Integrated Circuit)
  - Configurable Pins
  - Programmable Function
  - With ***Built-in Memory***
    - RAM
    - ROM
  - Has ***Built-in Peripherals***
    - Timer
    - ADC
    - UART/SPI/I2C
    - PWM
    - Comparators

Popular Microcontrollers

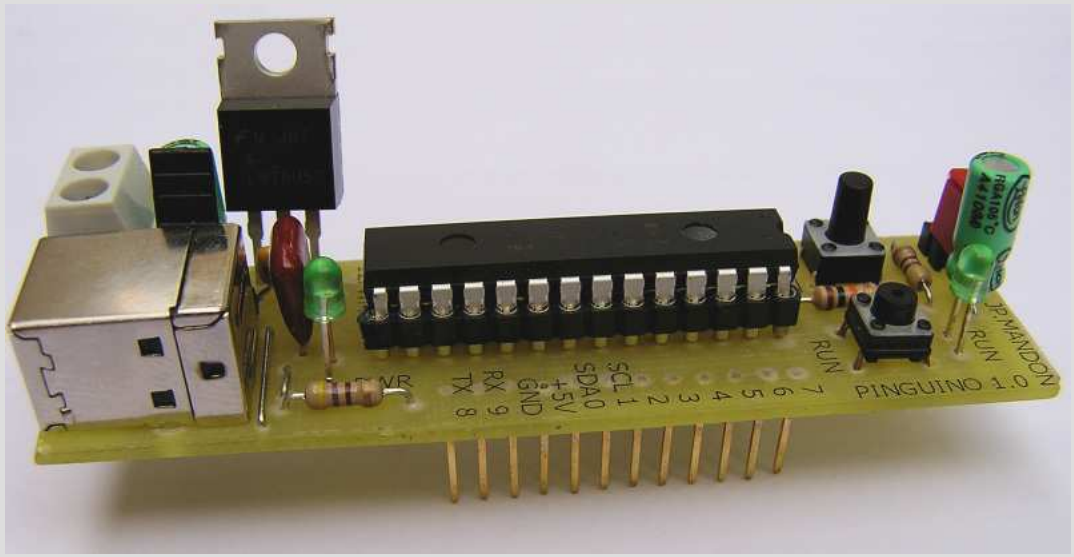
- PIC16F84a
  - Developed by Microchip
  - ***Widely used by hobbyist*** in the late 90's
  - Very basic function
  - Later replaced by pin compatible PIC16F628a

Popular Microcontrollers

- PIC16F877a
  - Developed by Microchip
  - Greater pin count than PIC16F84a
  - Has ADC, PWM, UART
  - and is ***self programmable***

Popular Microcontrollers

- PIC18F2550
  - Developed by Microchip
  - ***With USB connectivity***
  - Used on Pinguino (Arduino equivalent for PIC)



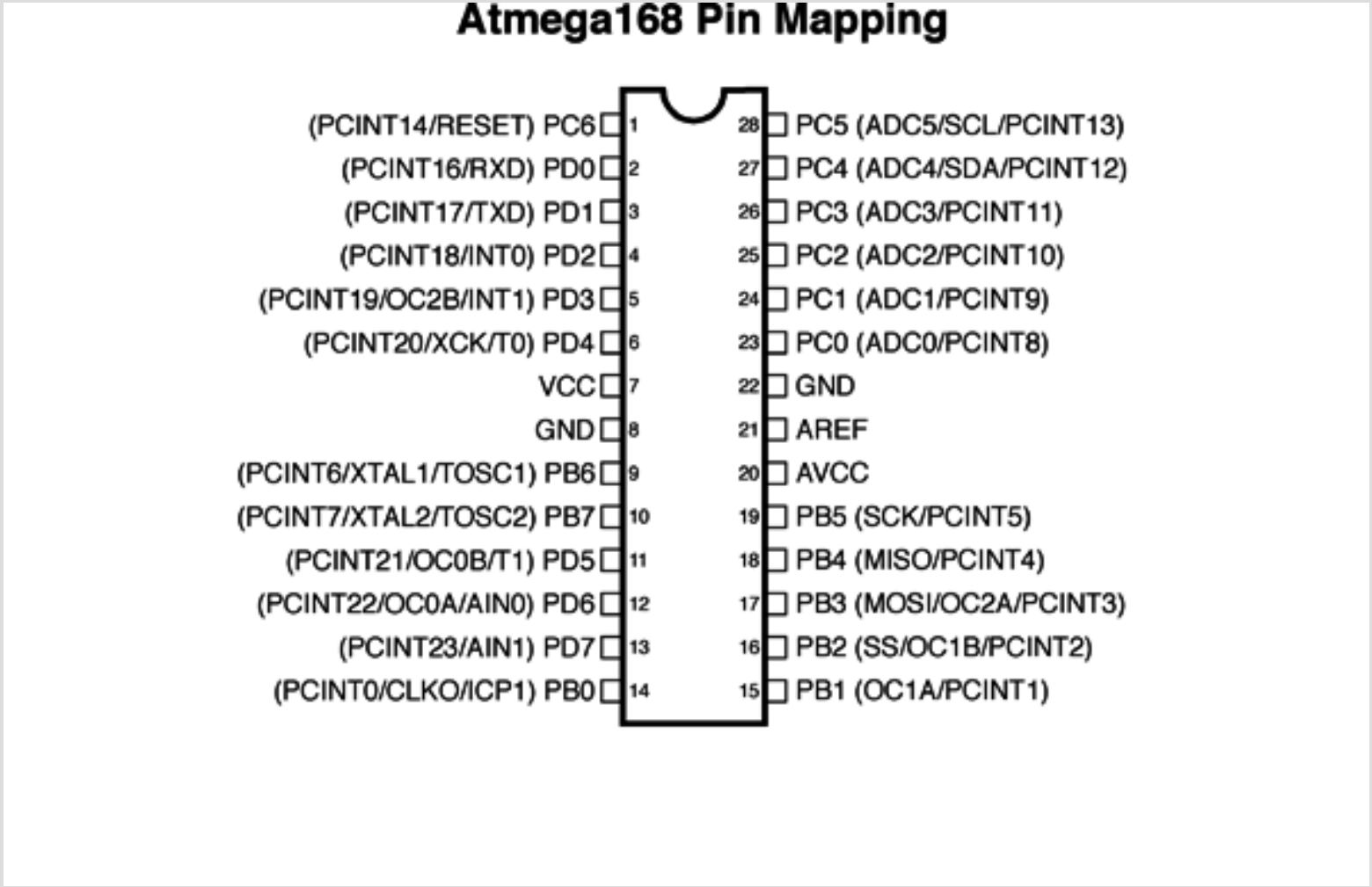
Popular Microcontrollers

- AtMega168
  - Developed by Atmel
  - Widely used ***microcontroller on Arduino***

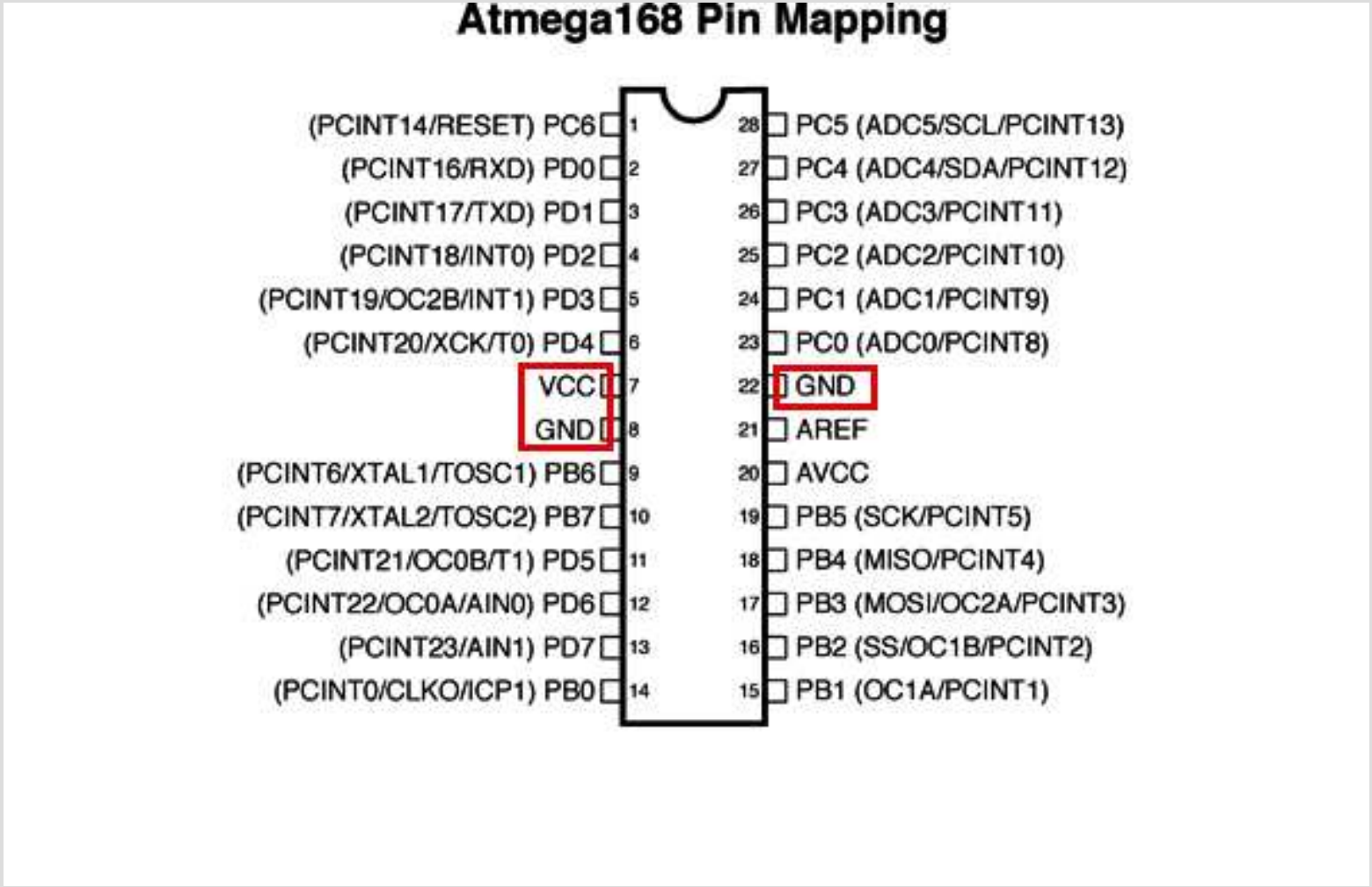
Atmega168

- Microcontroller Developed by Atmel
- ***8-bit Microcontroller***
- 28pins
- 16Kbytes Flash Memory
  - ***“Self-Programmable”***

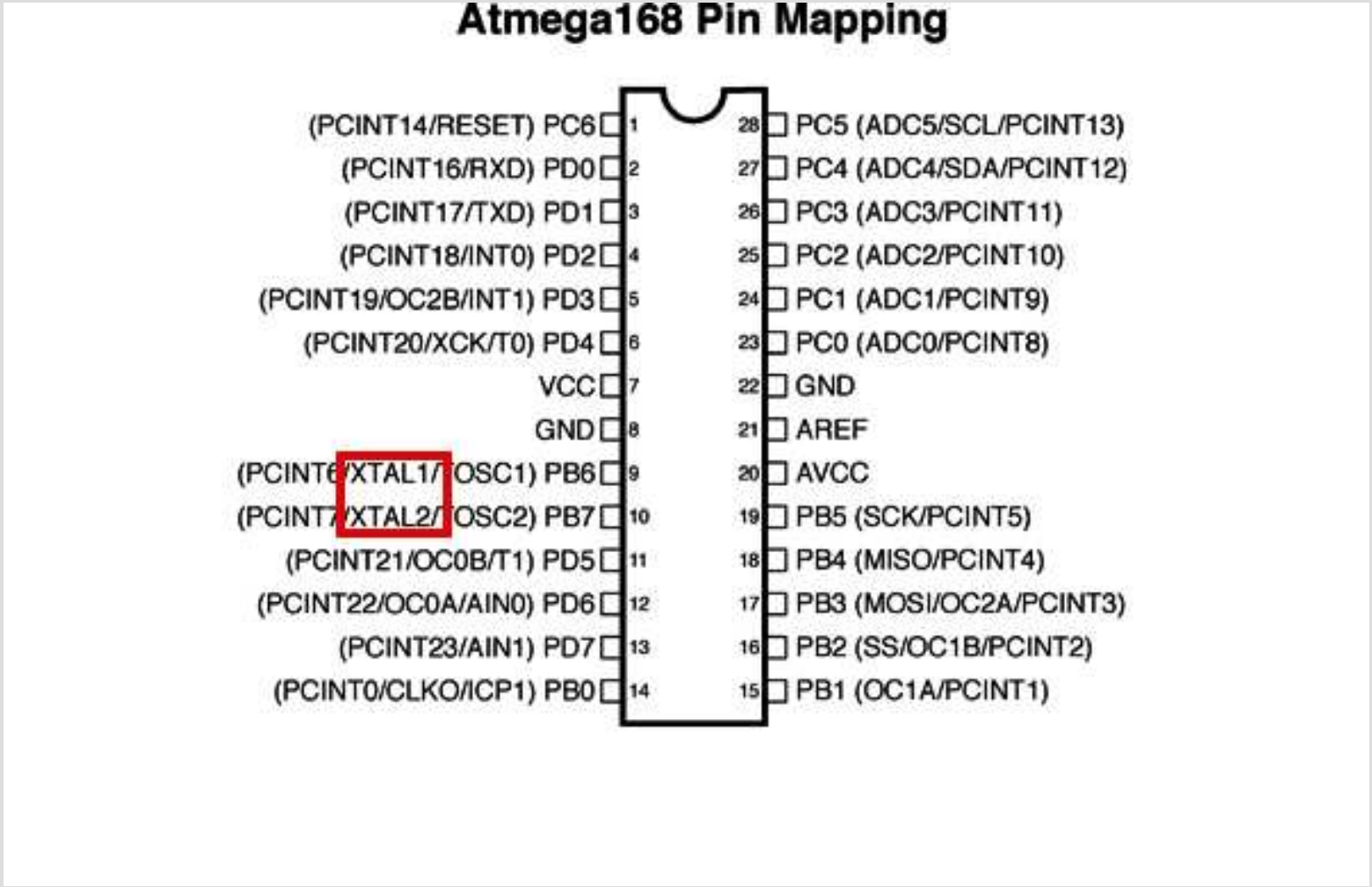
Atmega168


Atmega168 Pin Mapping
(PCINT14/RESET) PC6 1
(PCINT16/RXD) PD0 2
(PCINT17/TXD) PD1 3
(PCINT18/INT0) PD2 4
(PCINT19/OC2B/INT1) PD3 5
(PCINT20/XCK/T0) PD4 6
VCC 7
GND 8
(PCINT6/XTAL1/TOSC1) PB6 9
(PCINT7/XTAL2/TOSC2) PB7 10
(PCINT21/OC0B/T1) PD5 11
(PCINT22/OC0A/AIN0) PD6 12
(PCINT23/AIN1) PD7 13
(PCINT0/CLKO/ICP1) PB0 14
28 PC5 (ADC5/SCL/PCINT13)
27 PC4 (ADC4/SDA/PCINT12)
26 PC3 (ADC3/PCINT11)
25 PC2 (ADC2/PCINT10)
24 PC1 (ADC1/PCINT9)
23 PC0 (ADC0/PCINT8)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK/PCINT5)
18 PB4 (MISO/PCINT4)
17 PB3 (MOSI/OC2A/PCINT3)
16 PB2 (SS/OC1B/PCINT2)
15 PB1 (OC1A/PCINT1)

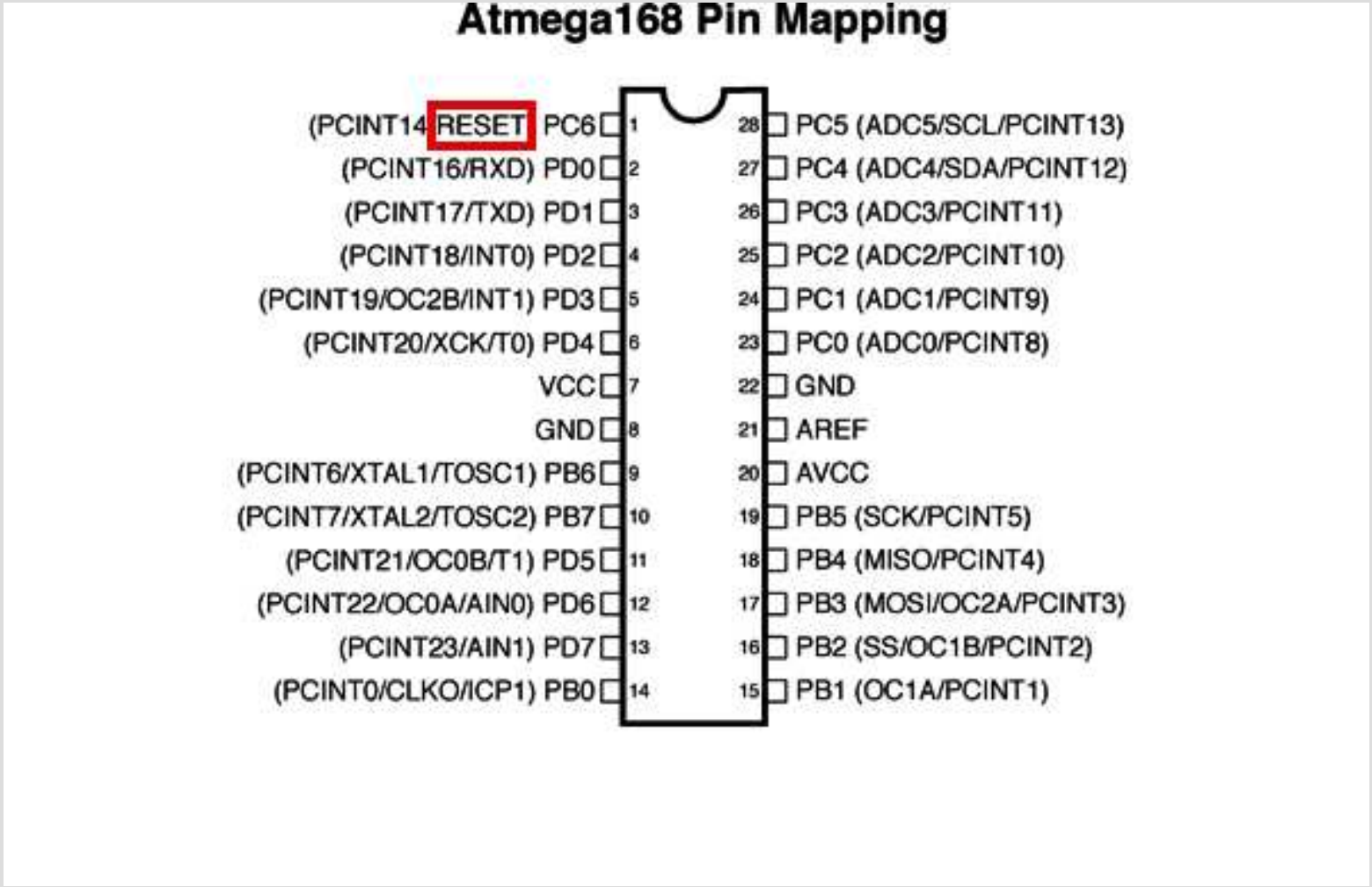
Atmega168


Atmega168 Pin Mapping
(PCINT14/RESET) PC6 1
(PCINT16/RXD) PD0 2
(PCINT17/TXD) PD1 3
(PCINT18/INT0) PD2 4
(PCINT19/OC2B/INT1) PD3 5
(PCINT20/XCK/T0) PD4 6
VCC 7
GND 8
(PCINT6/XTAL1/TOSC1) PB6 9
(PCINT7/XTAL2/TOSC2) PB7 10
(PCINT21/OC0B/T1) PD5 11
(PCINT22/OC0A/AIN0) PD6 12
(PCINT23/AIN1) PD7 13
(PCINT0/CLKO/ICP1) PB0 14
28 PC5 (ADC5/SCL/PCINT13)
27 PC4 (ADC4/SDA/PCINT12)
26 PC3 (ADC3/PCINT11)
25 PC2 (ADC2/PCINT10)
24 PC1 (ADC1/PCINT9)
23 PC0 (ADC0/PCINT8)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK/PCINT5)
18 PB4 (MISO/PCINT4)
17 PB3 (MOSI/OC2A/PCINT3)
16 PB2 (SS/OC1B/PCINT2)
15 PB1 (OC1A/PCINT1)

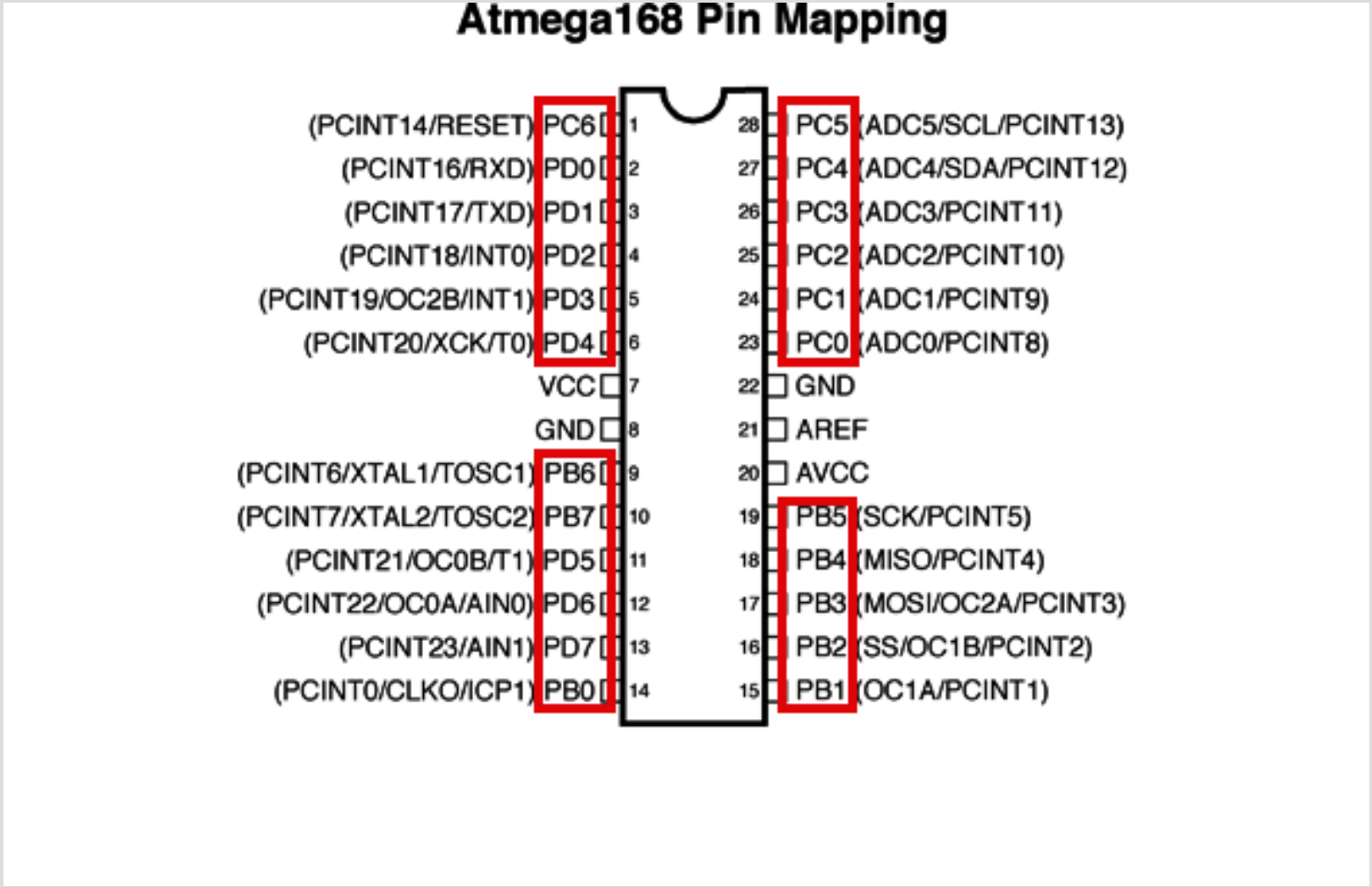
Atmega168


Atmega168 Pin Mapping
(PCINT14/RESET) PC6 1
(PCINT16/RXD) PD0 2
(PCINT17/TXD) PD1 3
(PCINT18/INT0) PD2 4
(PCINT19/OC2B/INT1) PD3 5
(PCINT20/XCK/T0) PD4 6
VCC 7
GND 8
(PCINT6/XTAL1/TOSC1) PB6 9
(PCINT7/XTAL2/TOSC2) PB7 10
(PCINT21/OC0B/T1) PD5 11
(PCINT22/OC0A/AIN0) PD6 12
(PCINT23/AIN1) PD7 13
(PCINT0/CLKO/ICP1) PB0 14
28 PC5 (ADC5/SCL/PCINT13)
27 PC4 (ADC4/SDA/PCINT12)
26 PC3 (ADC3/PCINT11)
25 PC2 (ADC2/PCINT10)
24 PC1 (ADC1/PCINT9)
23 PC0 (ADC0/PCINT8)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK/PCINT5)
18 PB4 (MISO/PCINT4)
17 PB3 (MOSI/OC2A/PCINT3)
16 PB2 (SS/OC1B/PCINT2)
15 PB1 (OC1A/PCINT1)

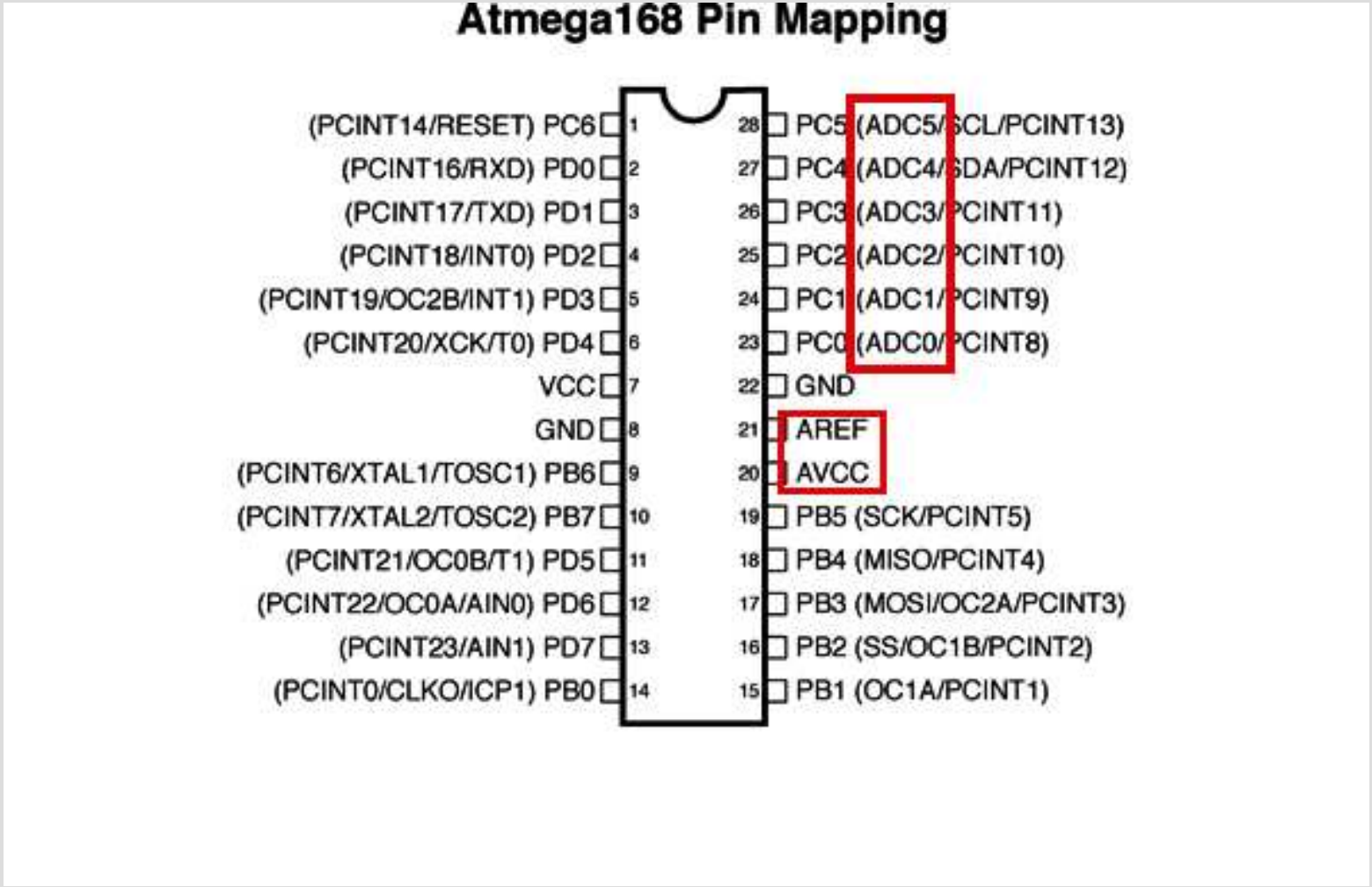
Atmega168


Atmega168 Pin Mapping
(PCINT14/RESET) PC6 1
(PCINT16/RXD) PD0 2
(PCINT17/TXD) PD1 3
(PCINT18/INT0) PD2 4
(PCINT19/OC2B/INT1) PD3 5
(PCINT20/XCK/T0) PD4 6
VCC 7
GND 8
(PCINT6/XTAL1/TOSC1) PB6 9
(PCINT7/XTAL2/TOSC2) PB7 10
(PCINT21/OC0B/T1) PD5 11
(PCINT22/OC0A/AIN0) PD6 12
(PCINT23/AIN1) PD7 13
(PCINT0/CLKO/ICP1) PB0 14
28 PC5 (ADC5/SCL/PCINT13)
27 PC4 (ADC4/SDA/PCINT12)
26 PC3 (ADC3/PCINT11)
25 PC2 (ADC2/PCINT10)
24 PC1 (ADC1/PCINT9)
23 PC0 (ADC0/PCINT8)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK/PCINT5)
18 PB4 (MISO/PCINT4)
17 PB3 (MOSI/OC2A/PCINT3)
16 PB2 (SS/OC1B/PCINT2)
15 PB1 (OC1A/PCINT1)

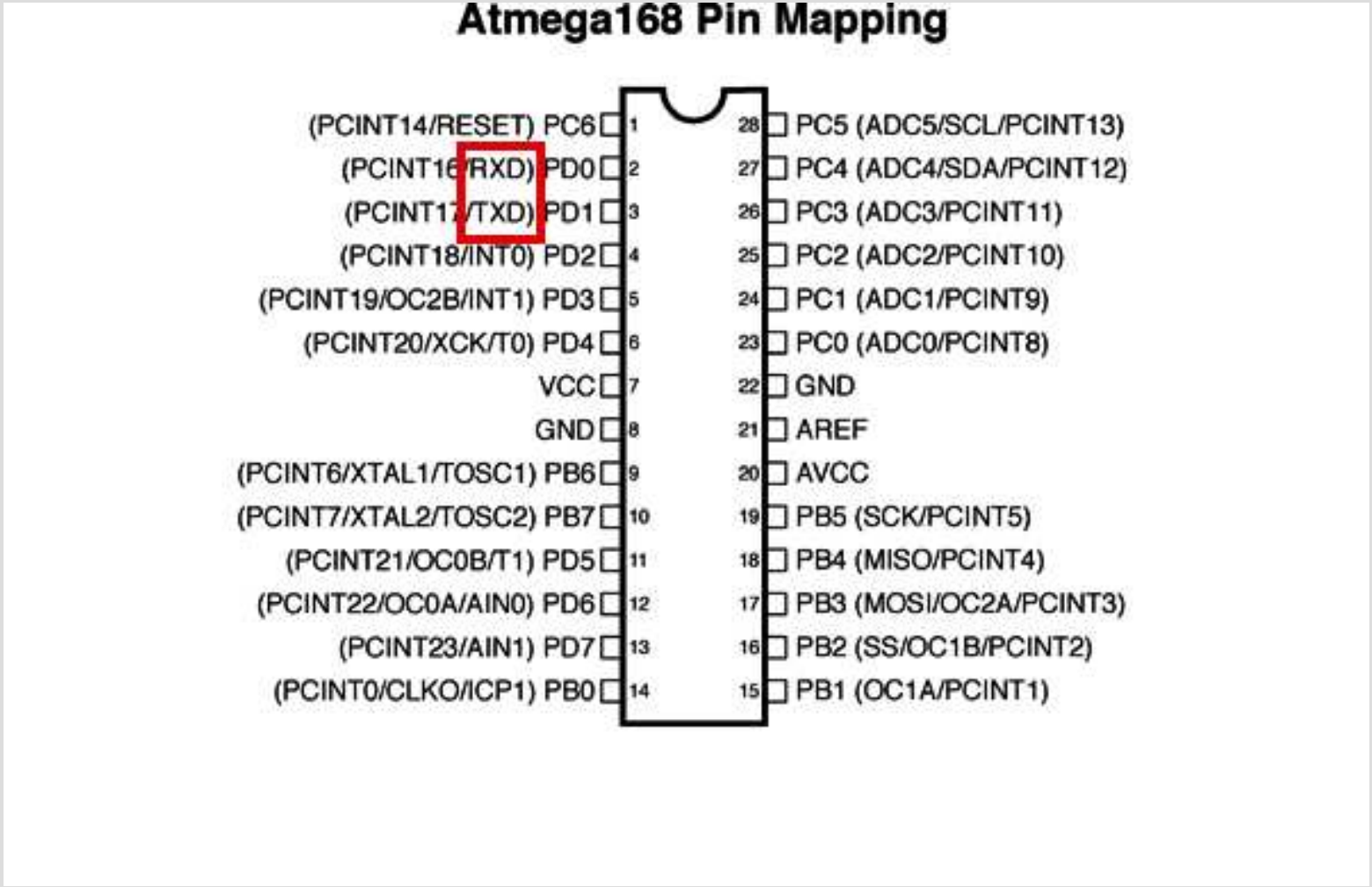
Atmega168


Atmega168 Pin Mapping
(PCINT14/RESET) PC6 1
(PCINT16/RXD) PD0 2
(PCINT17/TXD) PD1 3
(PCINT18/INT0) PD2 4
(PCINT19/OC2B/INT1) PD3 5
(PCINT20/XCK/T0) PD4 6
VCC 7
GND 8
(PCINT6/XTAL1/TOSC1) PB6 9
(PCINT7/XTAL2/TOSC2) PB7 10
(PCINT21/OC0B/T1) PD5 11
(PCINT22/OC0A/AIN0) PD6 12
(PCINT23/AIN1) PD7 13
(PCINT0/CLKO/ICP1) PB0 14
28 PC5 (ADC5/SCL/PCINT13)
27 PC4 (ADC4/SDA/PCINT12)
26 PC3 (ADC3/PCINT11)
25 PC2 (ADC2/PCINT10)
24 PC1 (ADC1/PCINT9)
23 PC0 (ADC0/PCINT8)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK/PCINT5)
18 PB4 (MISO/PCINT4)
17 PB3 (MOSI/OC2A/PCINT3)
16 PB2 (SS/OC1B/PCINT2)
15 PB1 (OC1A/PCINT1)

Atmega168


Atmega168 Pin Mapping
(PCINT14/RESET) PC6 1
(PCINT16/RXD) PD0 2
(PCINT17/TXD) PD1 3
(PCINT18/INT0) PD2 4
(PCINT19/OC2B/INT1) PD3 5
(PCINT20/XCK/T0) PD4 6
VCC 7
GND 8
(PCINT6/XTAL1/TOSC1) PB6 9
(PCINT7/XTAL2/TOSC2) PB7 10
(PCINT21/OC0B/T1) PD5 11
(PCINT22/OC0A/AIN0) PD6 12
(PCINT23/AIN1) PD7 13
(PCINT0/CLKO/ICP1) PB0 14
28 PC5 (ADC5/SCL/PCINT13)
27 PC4 (ADC4/SDA/PCINT12)
26 PC3 (ADC3/PCINT11)
25 PC2 (ADC2/PCINT10)
24 PC1 (ADC1/PCINT9)
23 PC0 (ADC0/PCINT8)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK/PCINT5)
18 PB4 (MISO/PCINT4)
17 PB3 (MOSI/OC2A/PCINT3)
16 PB2 (SS/OC1B/PCINT2)
15 PB1 (OC1A/PCINT1)

Atmega168

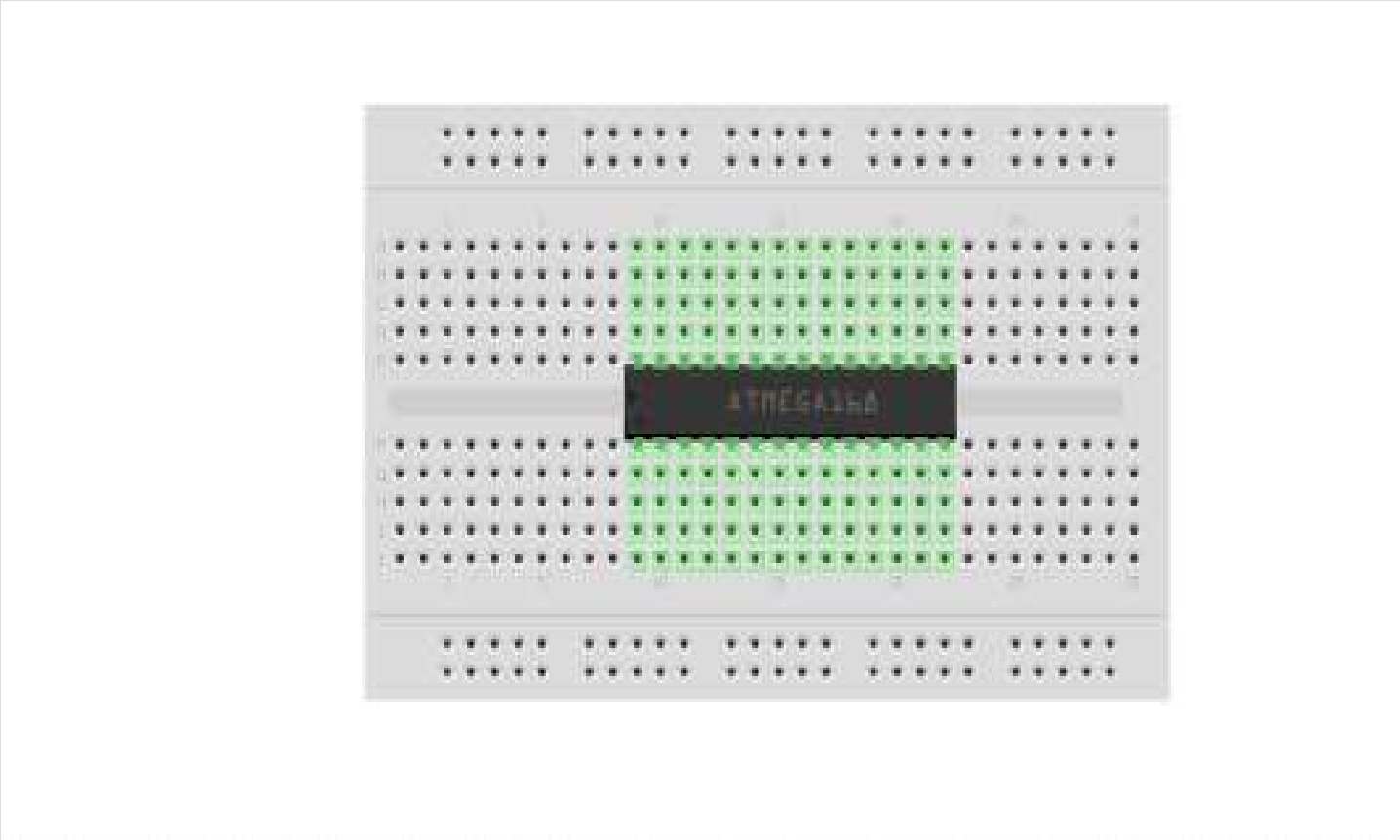

Atmega168 Pin Mapping
(PCINT14/RESET) PC6 1
(PCINT16/RXD) PD0 2
(PCINT17/TXD) PD1 3
(PCINT18/INT0) PD2 4
(PCINT19/OC2B/INT1) PD3 5
(PCINT20/XCK/T0) PD4 6
VCC 7
GND 8
(PCINT6/XTAL1/TOSC1) PB6 9
(PCINT7/XTAL2/TOSC2) PB7 10
(PCINT21/OC0B/T1) PD5 11
(PCINT22/OC0A/AIN0) PD6 12
(PCINT23/AIN1) PD7 13
(PCINT0/CLKO/ICP1) PB0 14
28 PC5 (ADC5/SCL/PCINT13)
27 PC4 (ADC4/SDA/PCINT12)
26 PC3 (ADC3/PCINT11)
25 PC2 (ADC2/PCINT10)
24 PC1 (ADC1/PCINT9)
23 PC0 (ADC0/PCINT8)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK/PCINT5)
18 PB4 (MISO/PCINT4)
17 PB3 (MOSI/OC2A/PCINT3)
16 PB2 (SS/OC1B/PCINT2)
15 PB1 (OC1A/PCINT1)

Basic Setup

- AtMega168

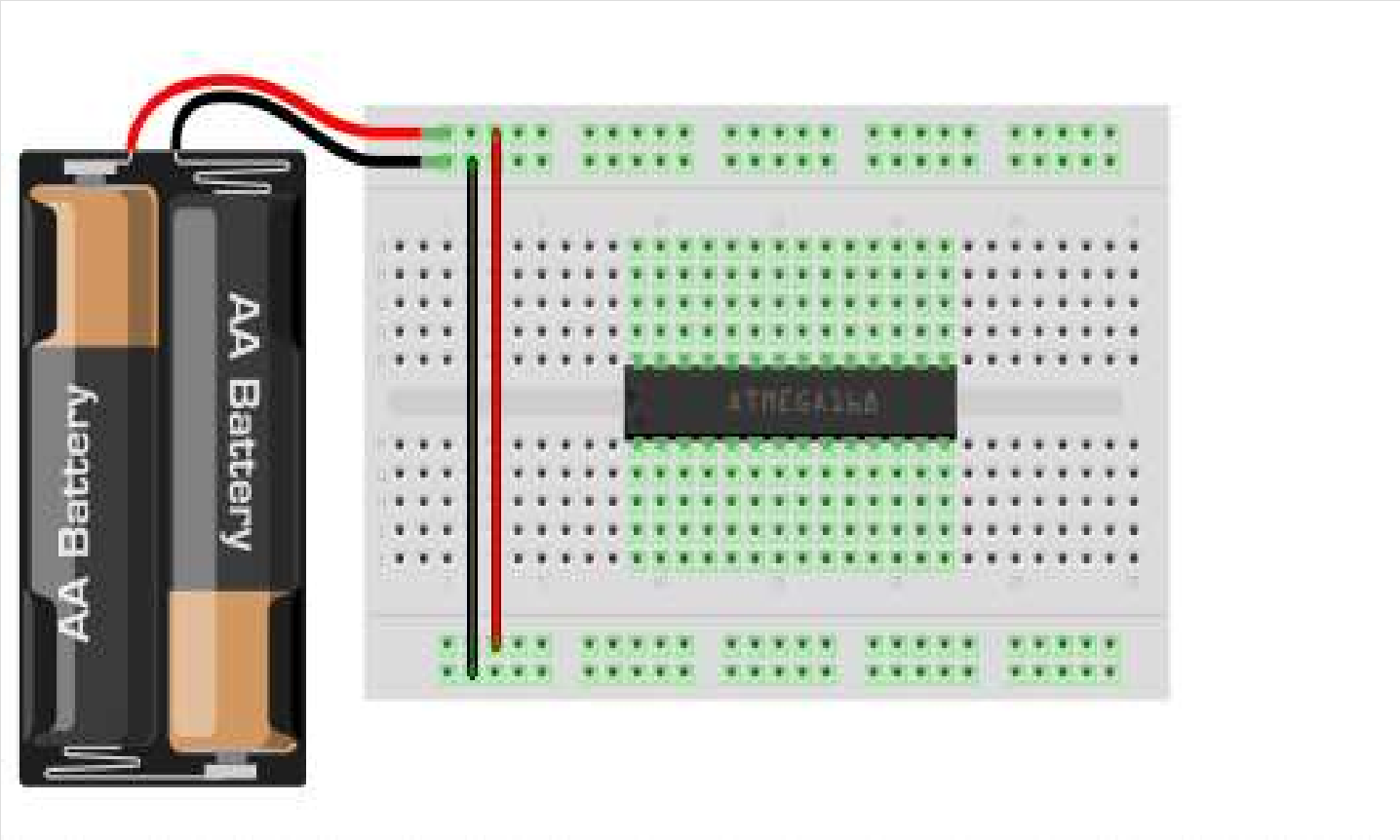
Basic Setup

- Breadboard



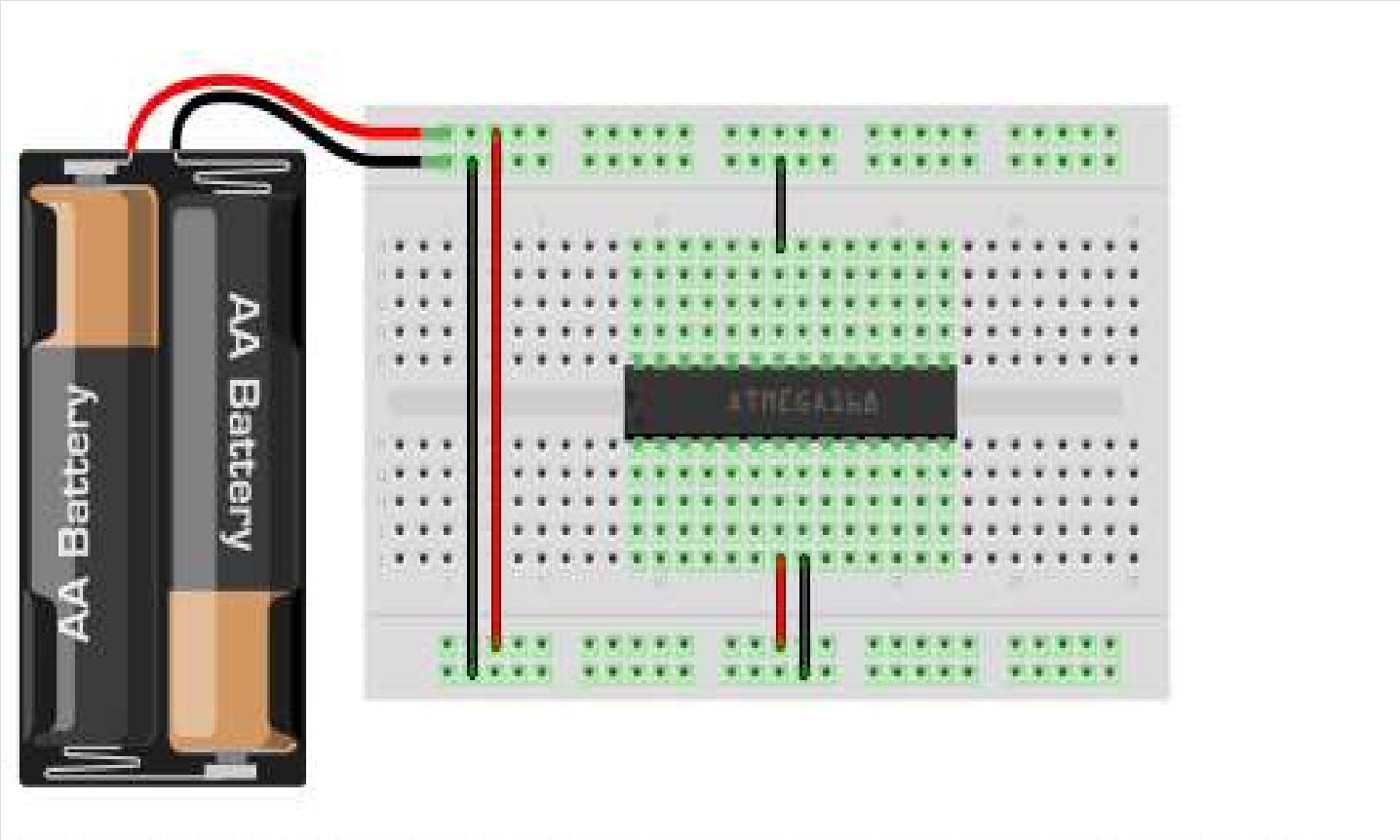
Basic Setup

- Supply



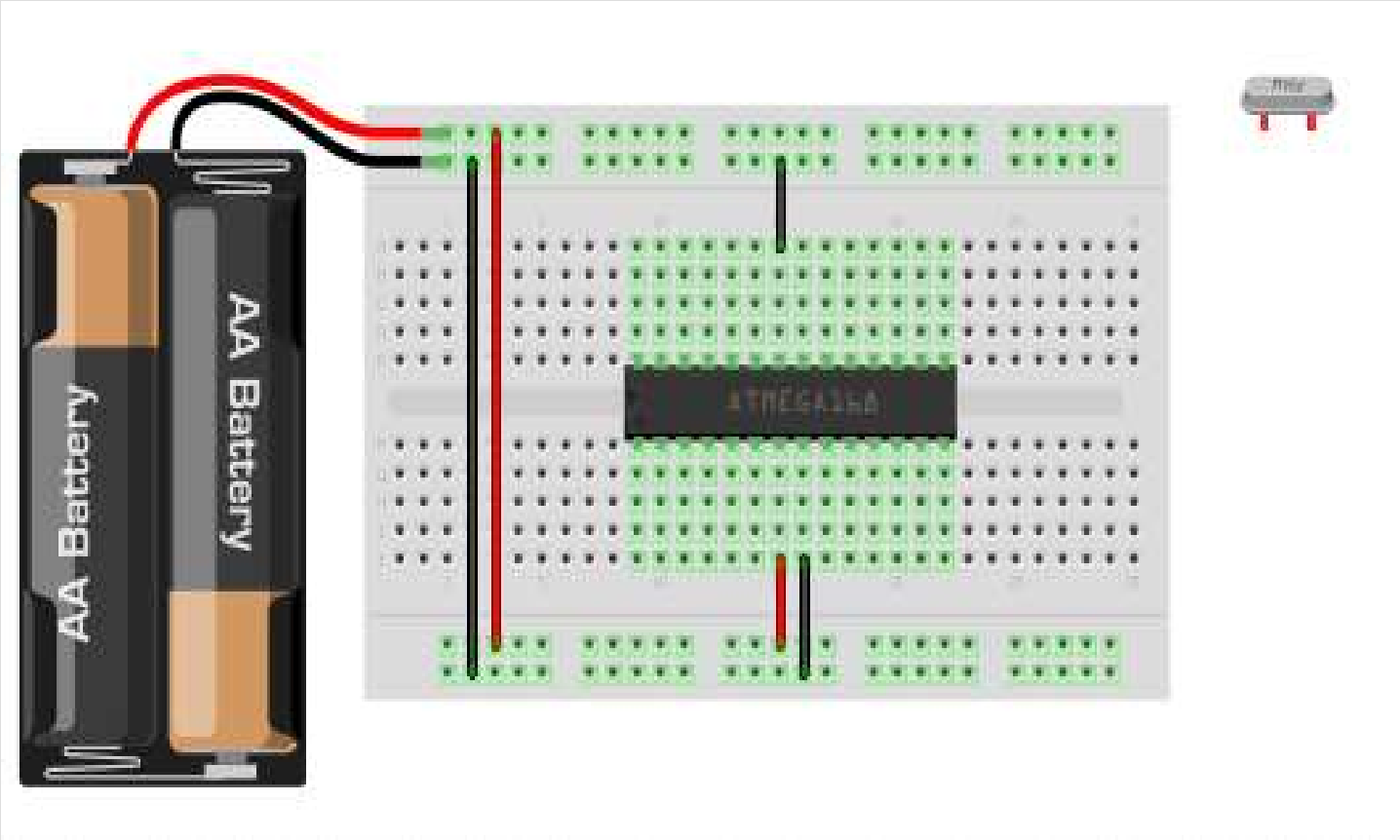
Basic Setup

- Supply



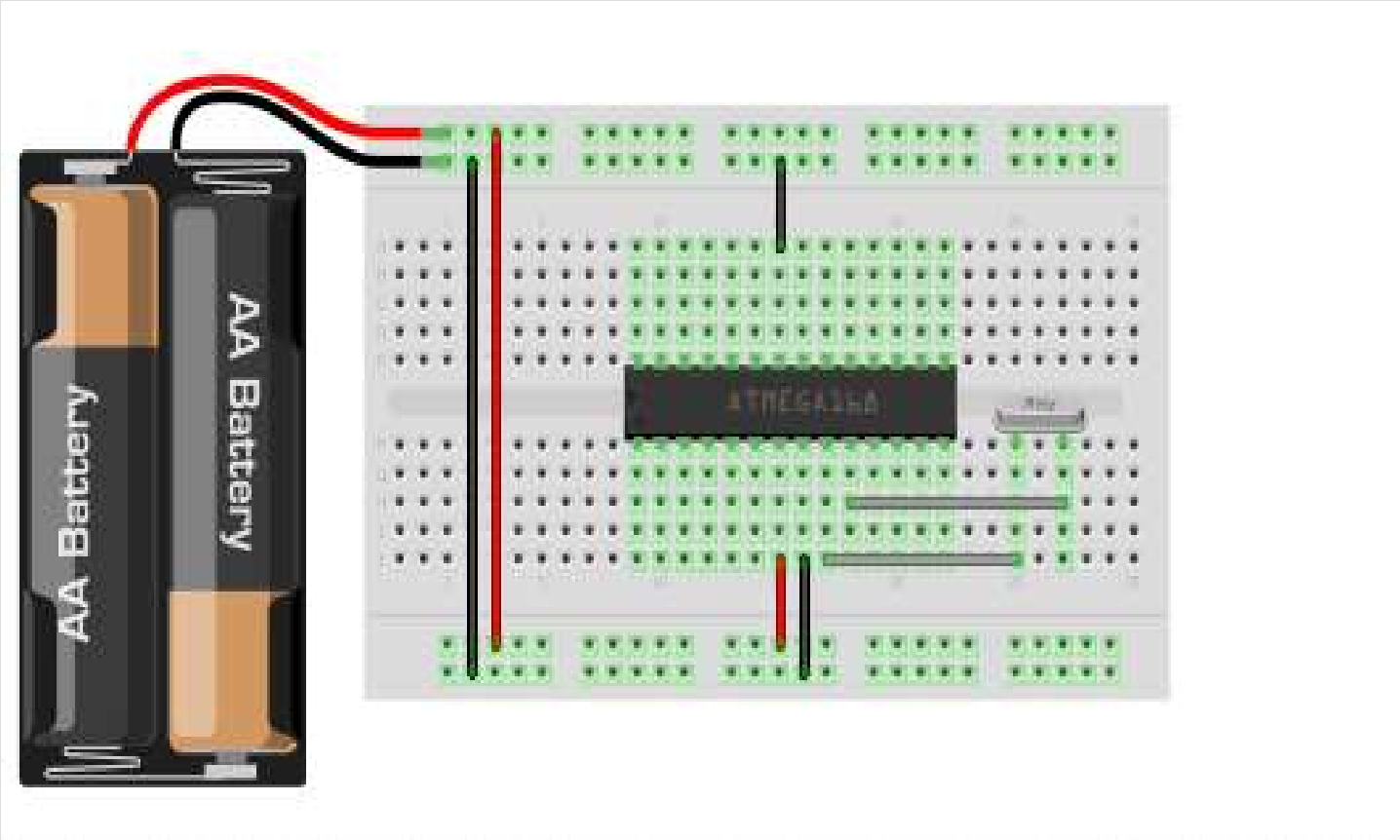
Basic Setup

- Crystal Oscillator



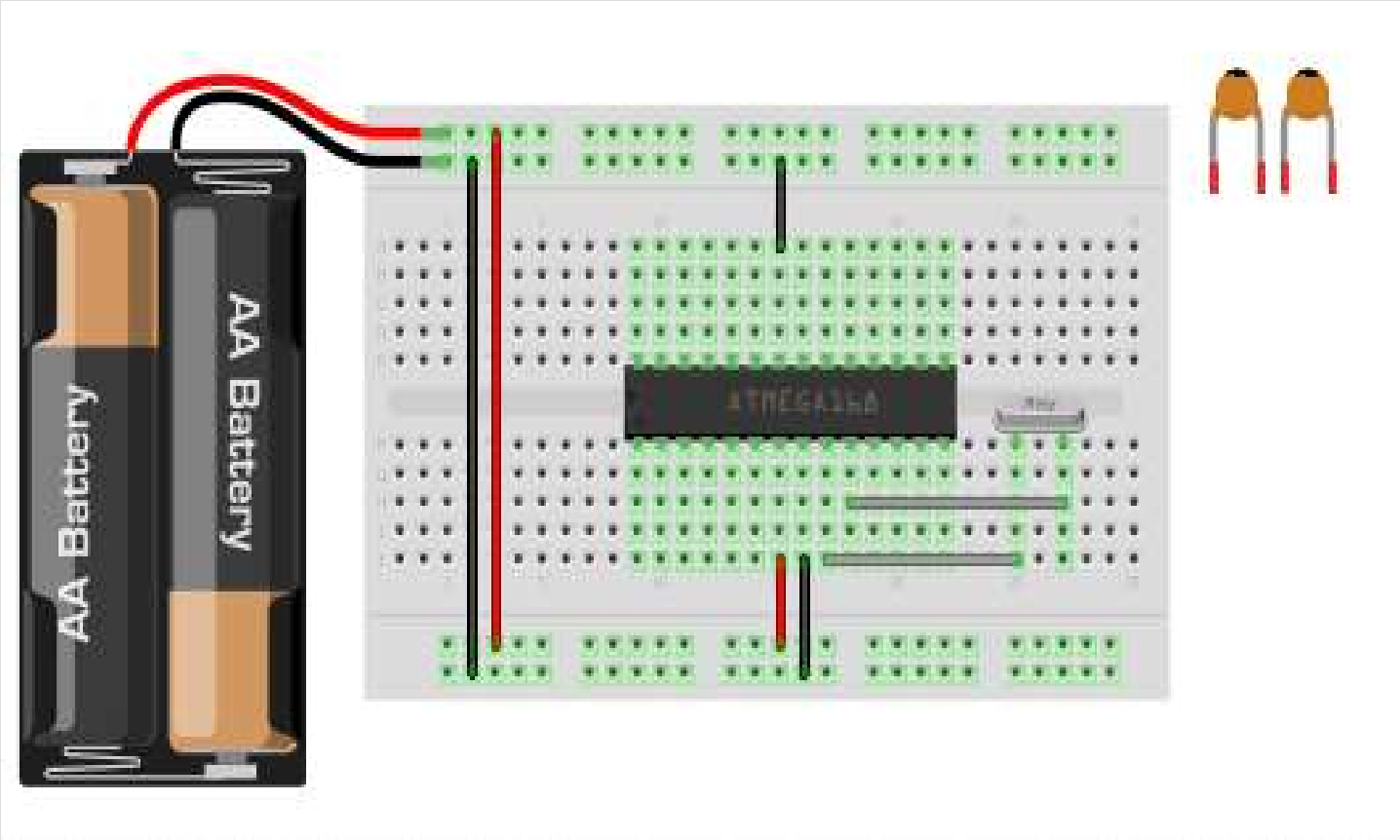
Basic Setup

- Crystal Oscillator



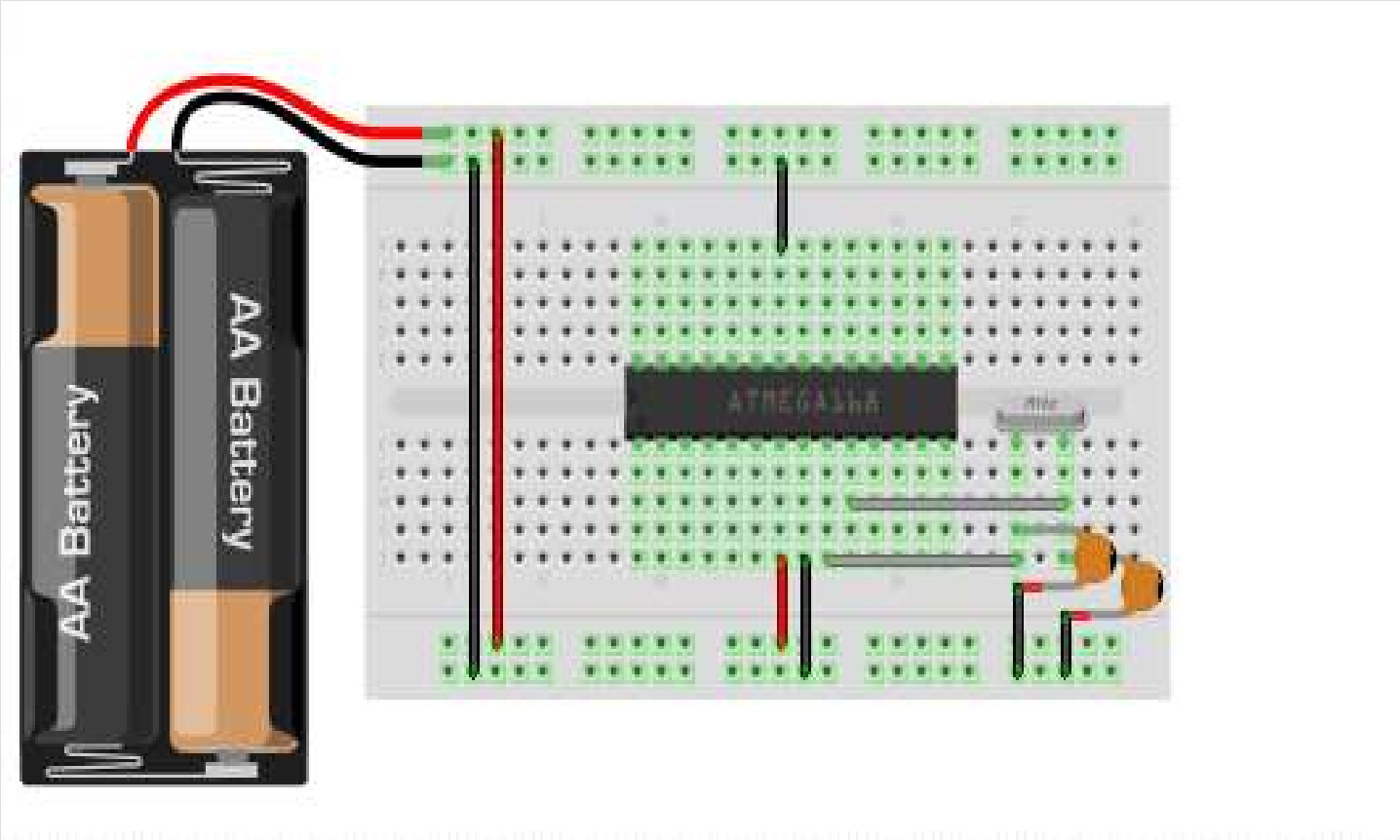
Basic Setup

- Resonator Bypass



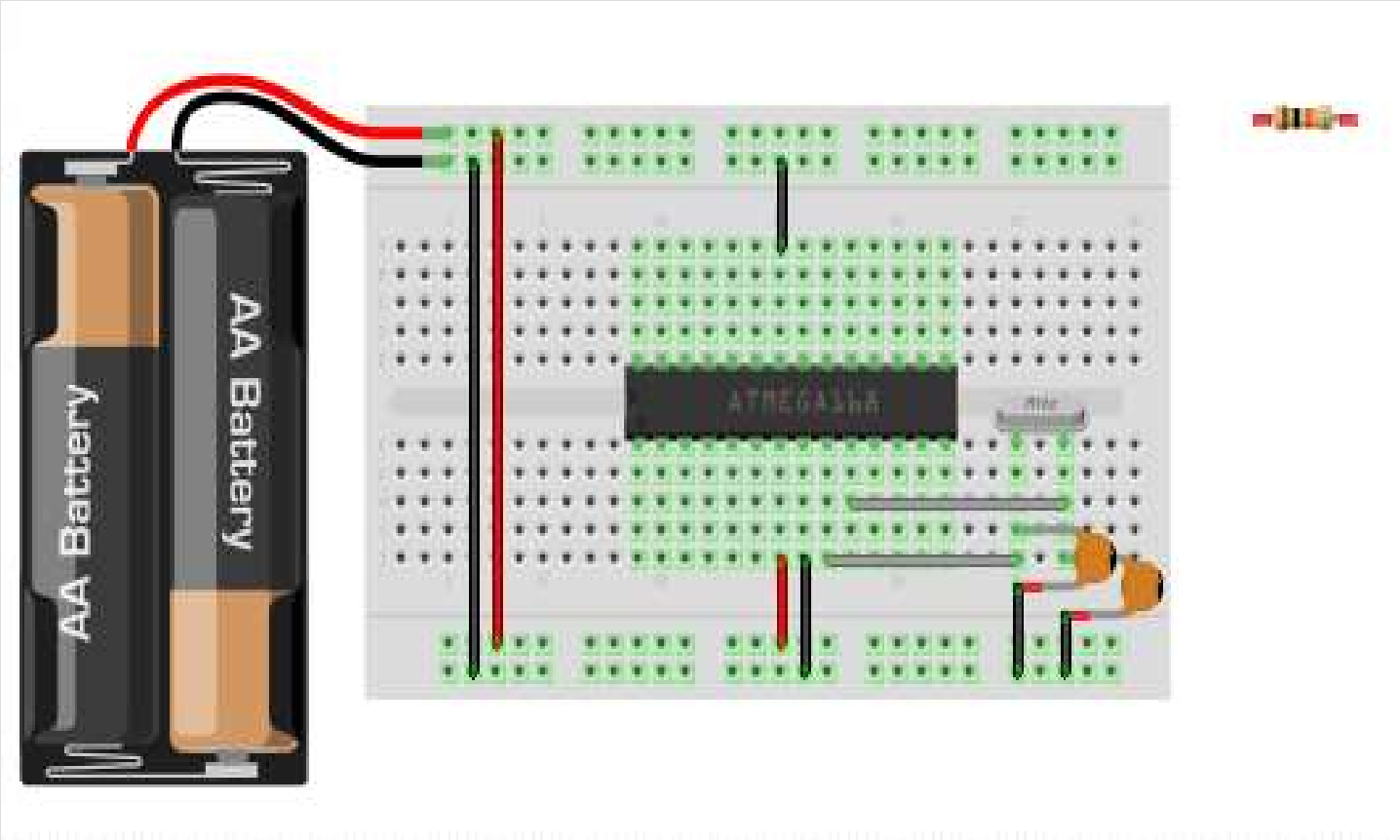
Basic Setup

- Resonator Bypass



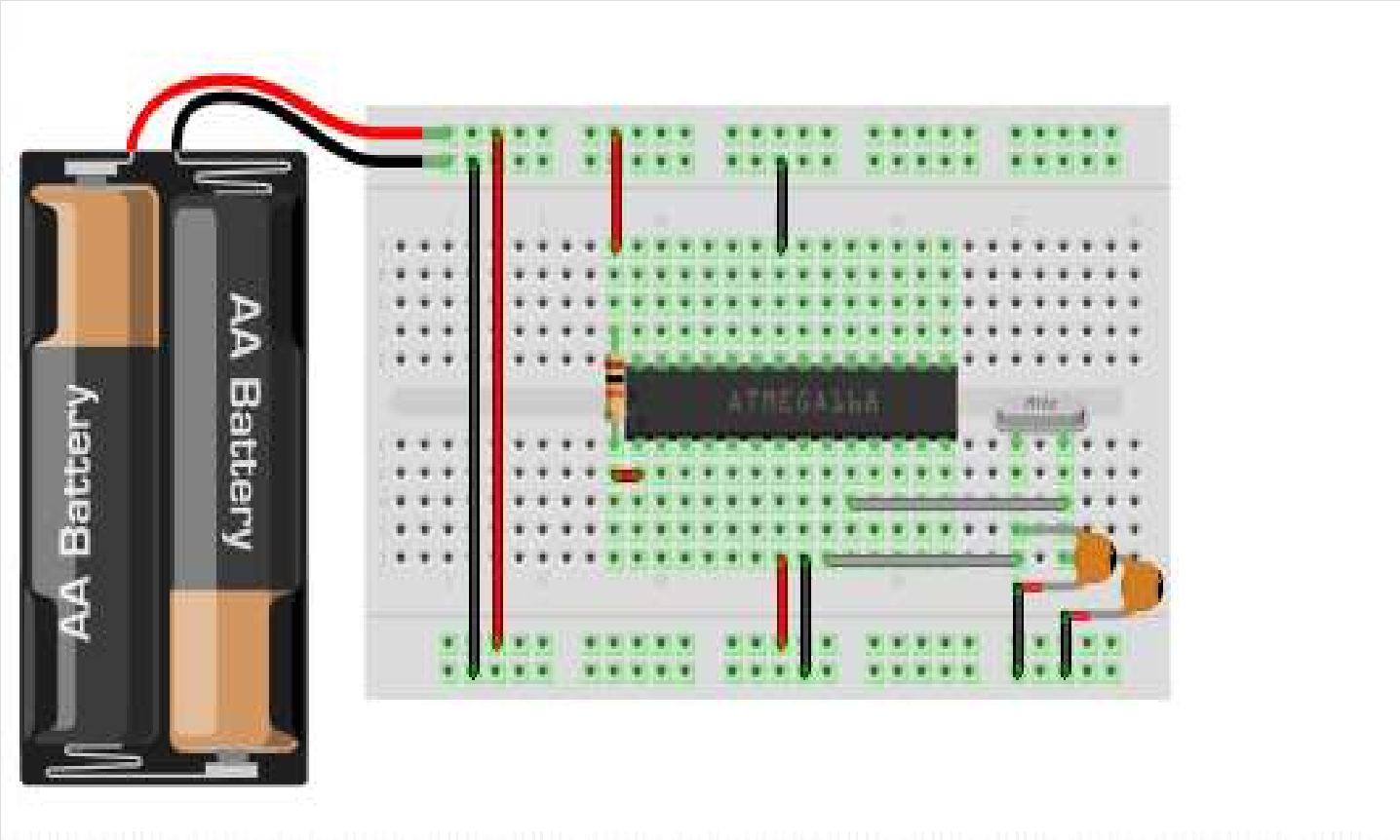
Basic Setup

- Reset Pullup



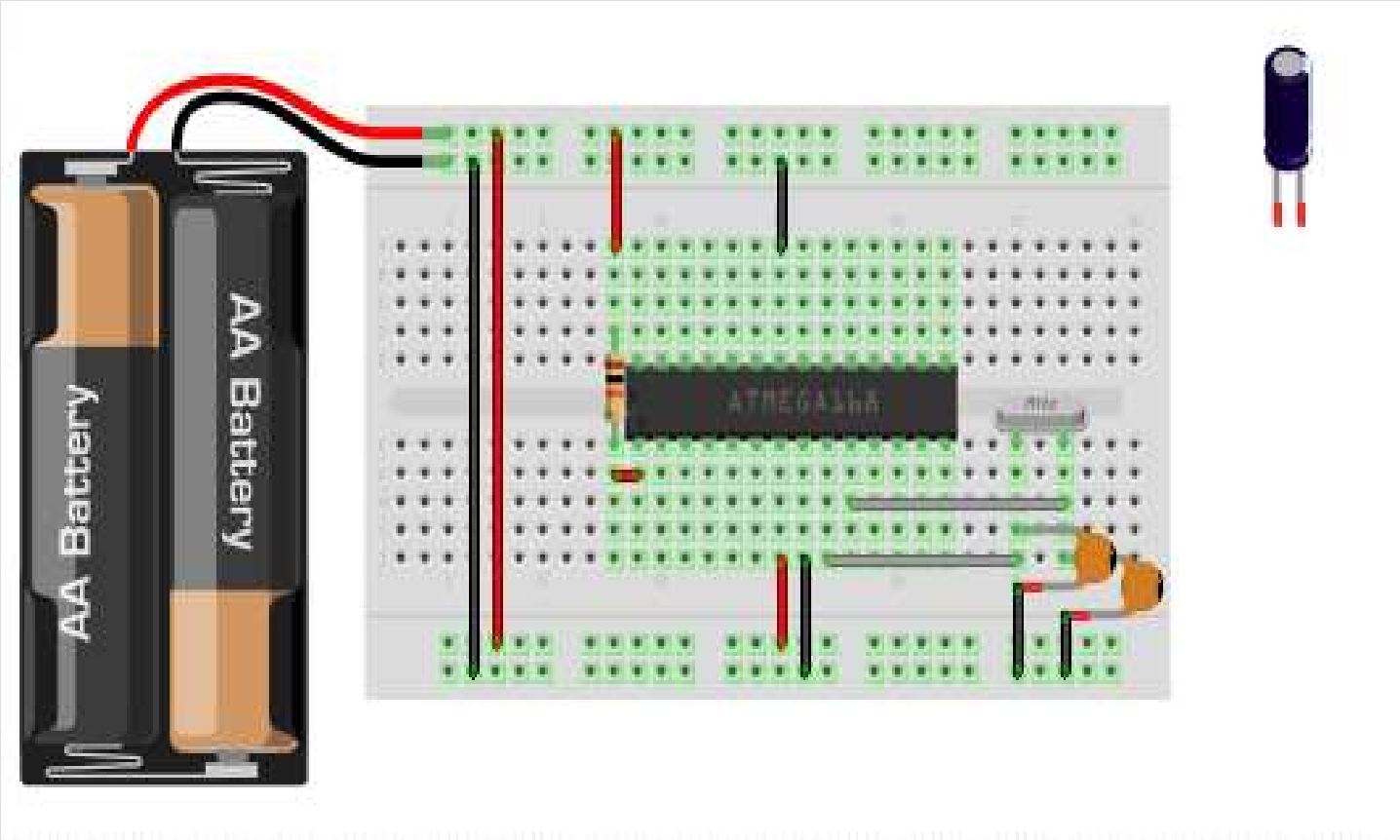
Basic Setup

- Reset Pullup



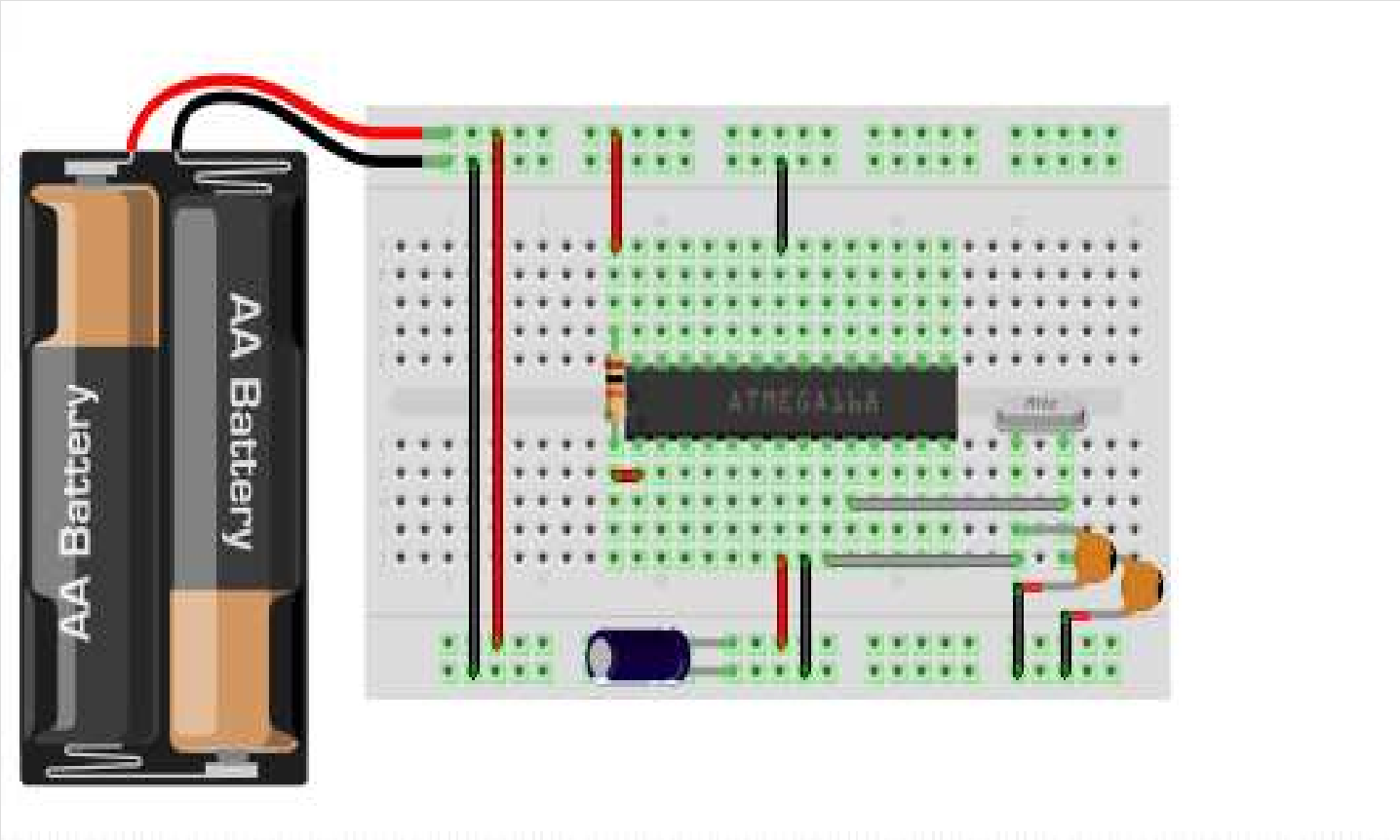
Basic Setup

- Filter Capacitor



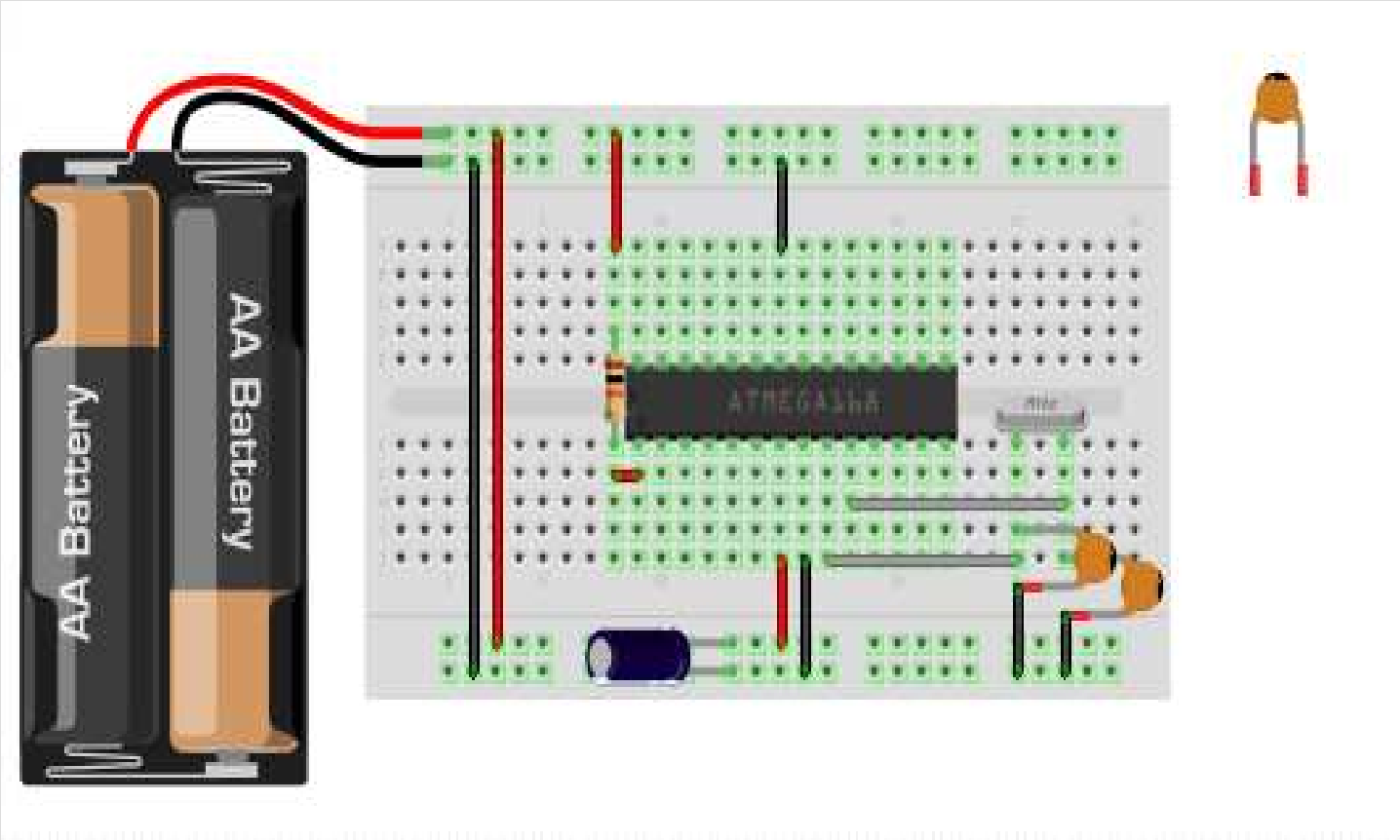
Basic Setup

- Filter Capacitor



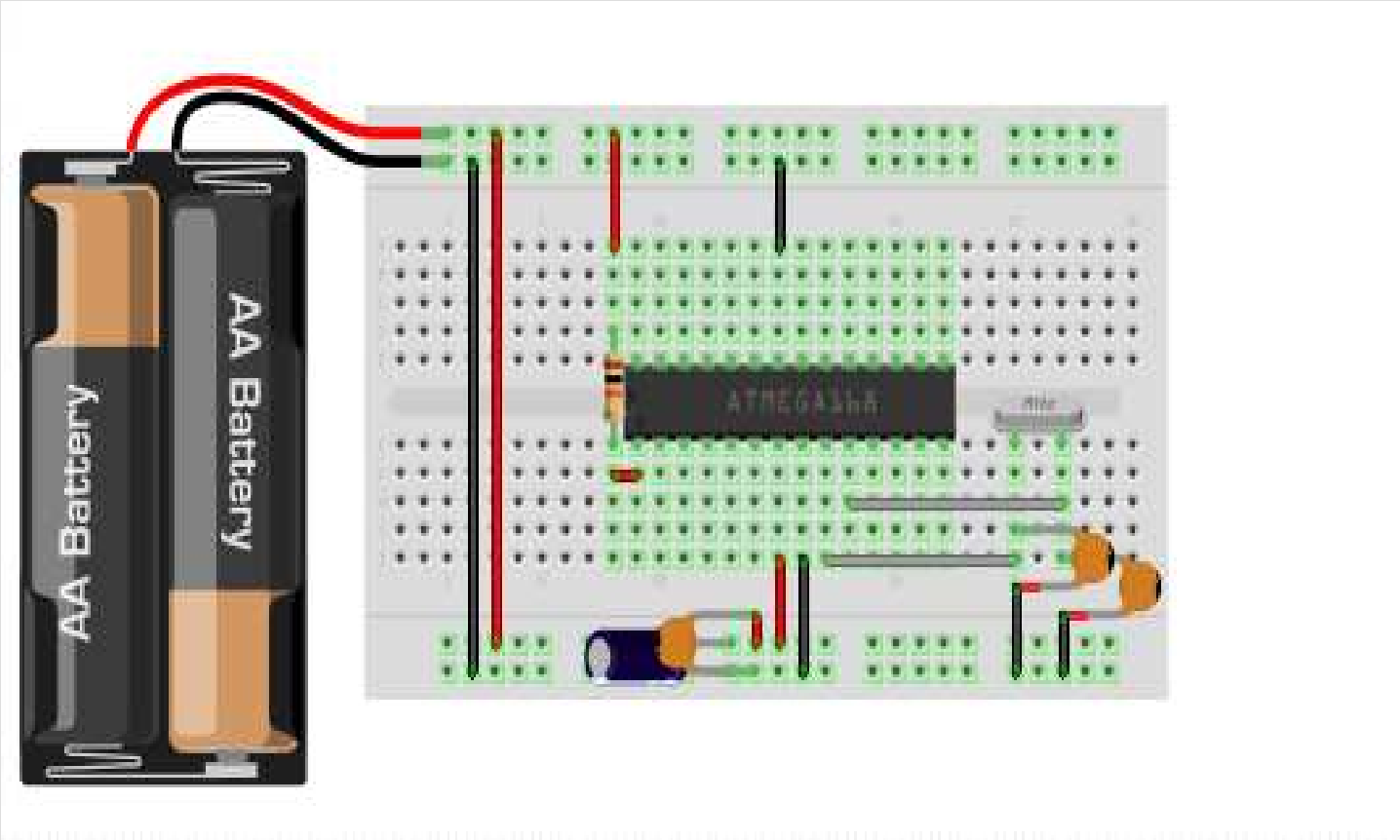
Basic Setup

- Bypass Capacitor



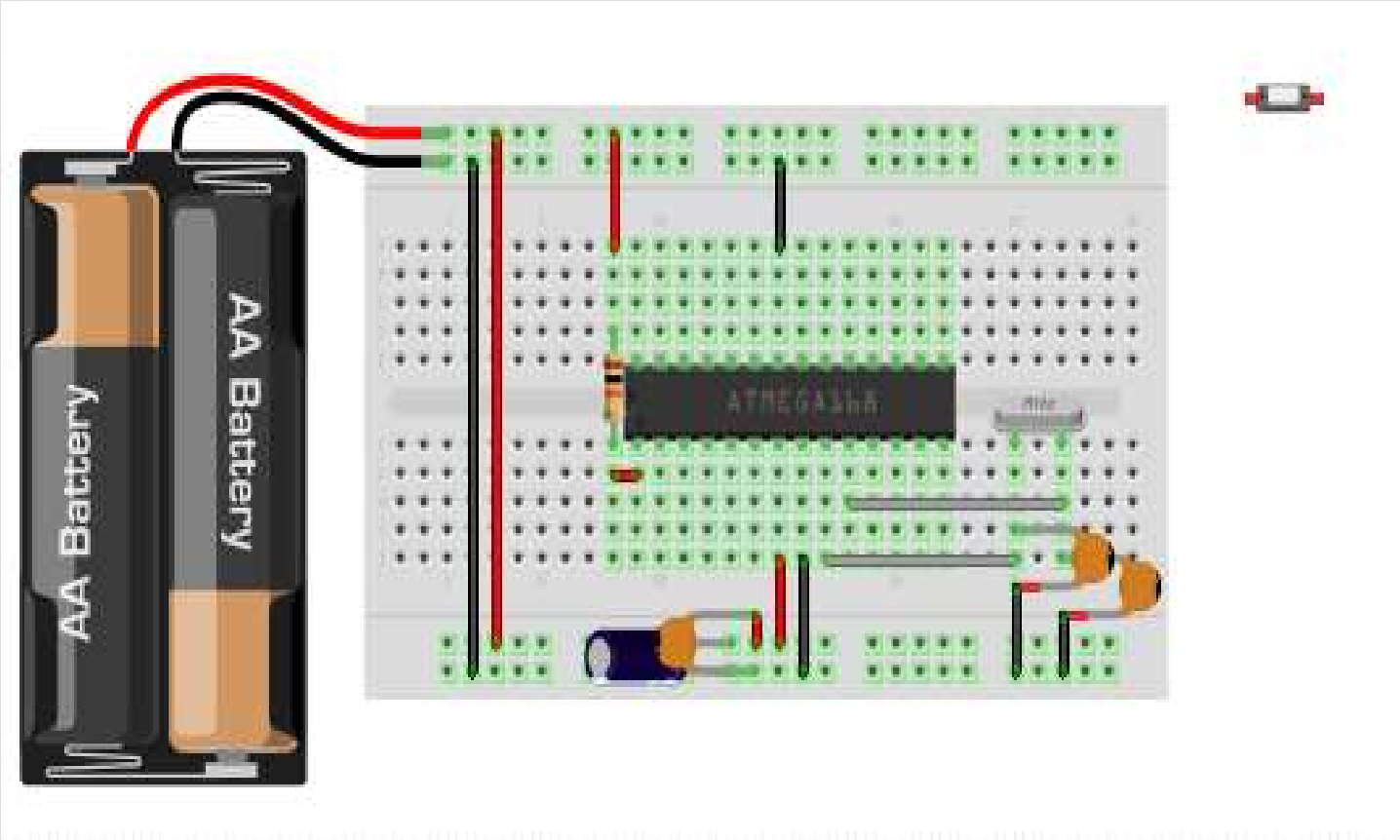
Basic Setup

- Bypass Capacitor



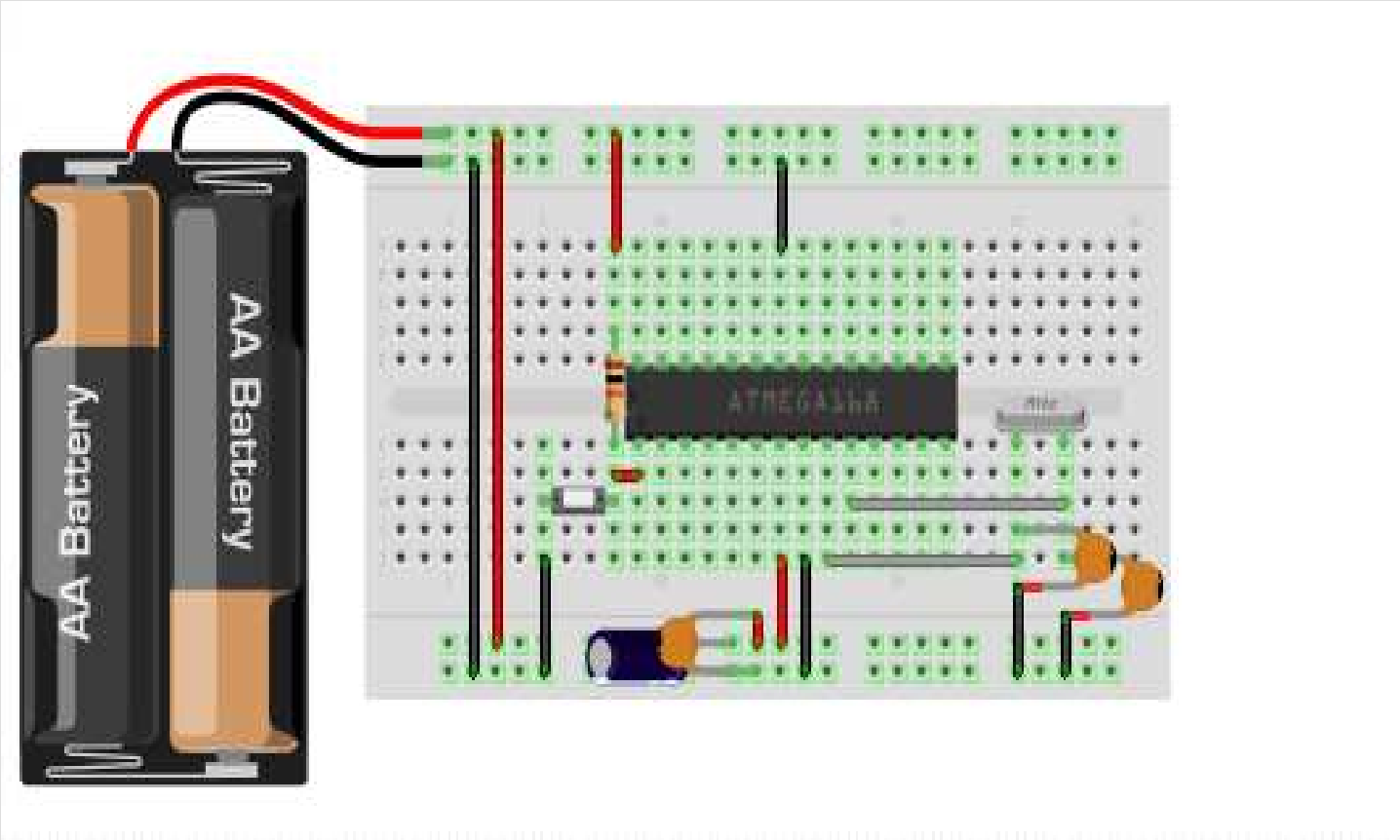
Basic Setup

- Reset Switch



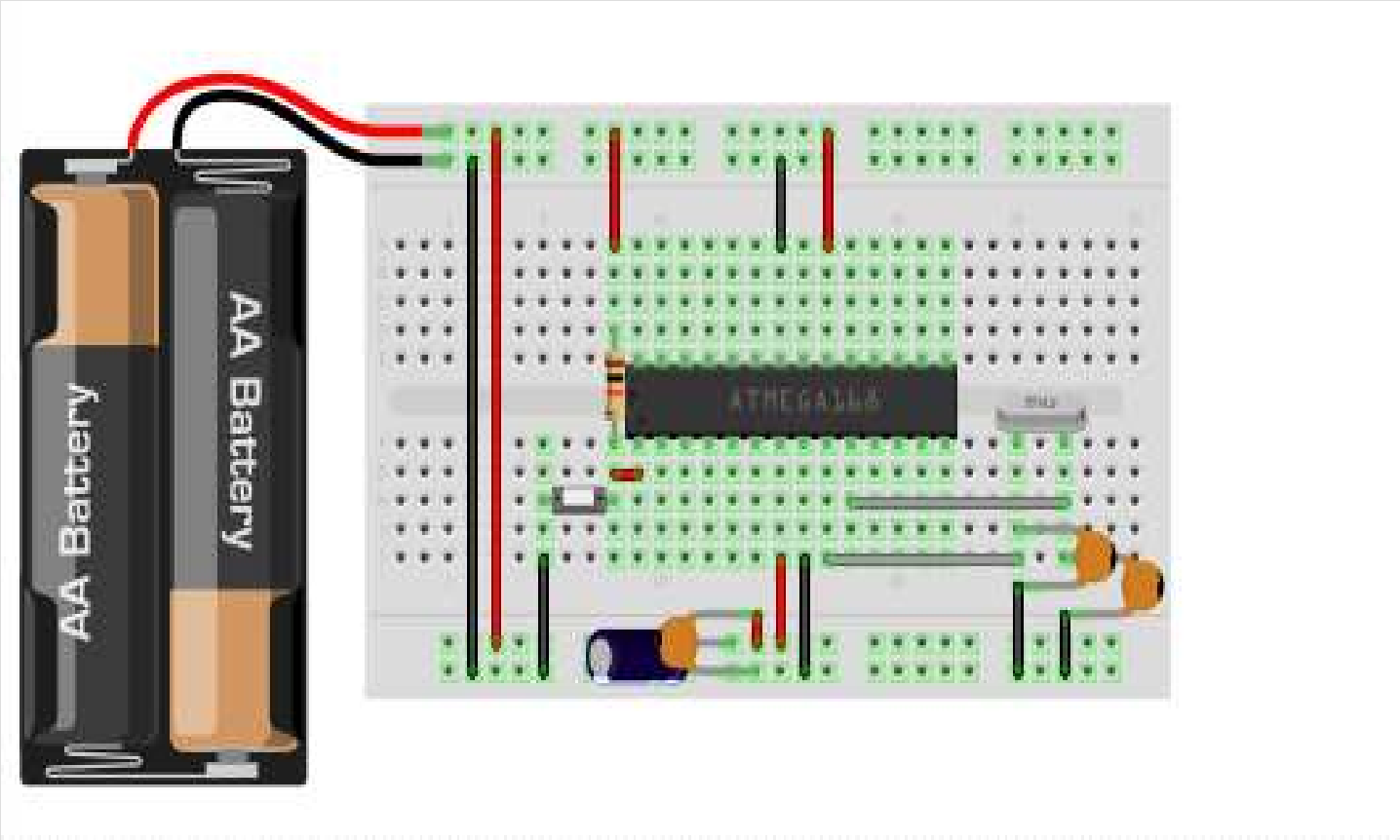
Basic Setup

- Reset Switch



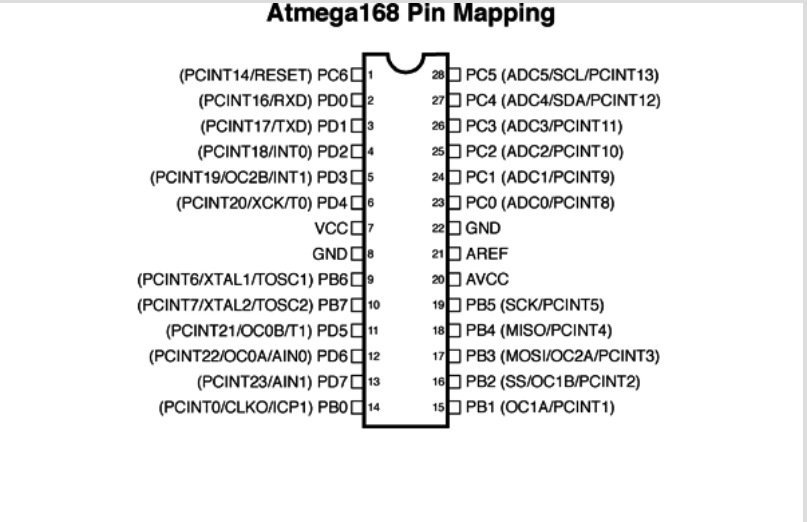
Basic Setup

- ADC Supply

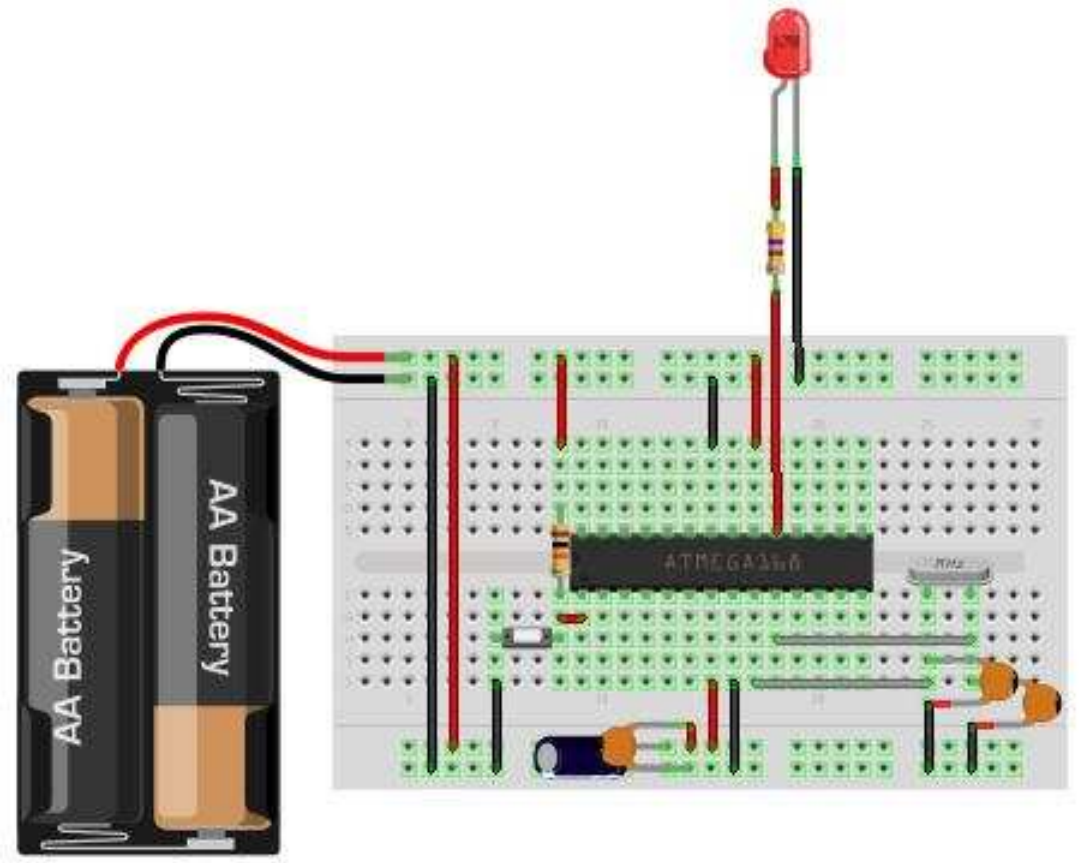


Atmega168

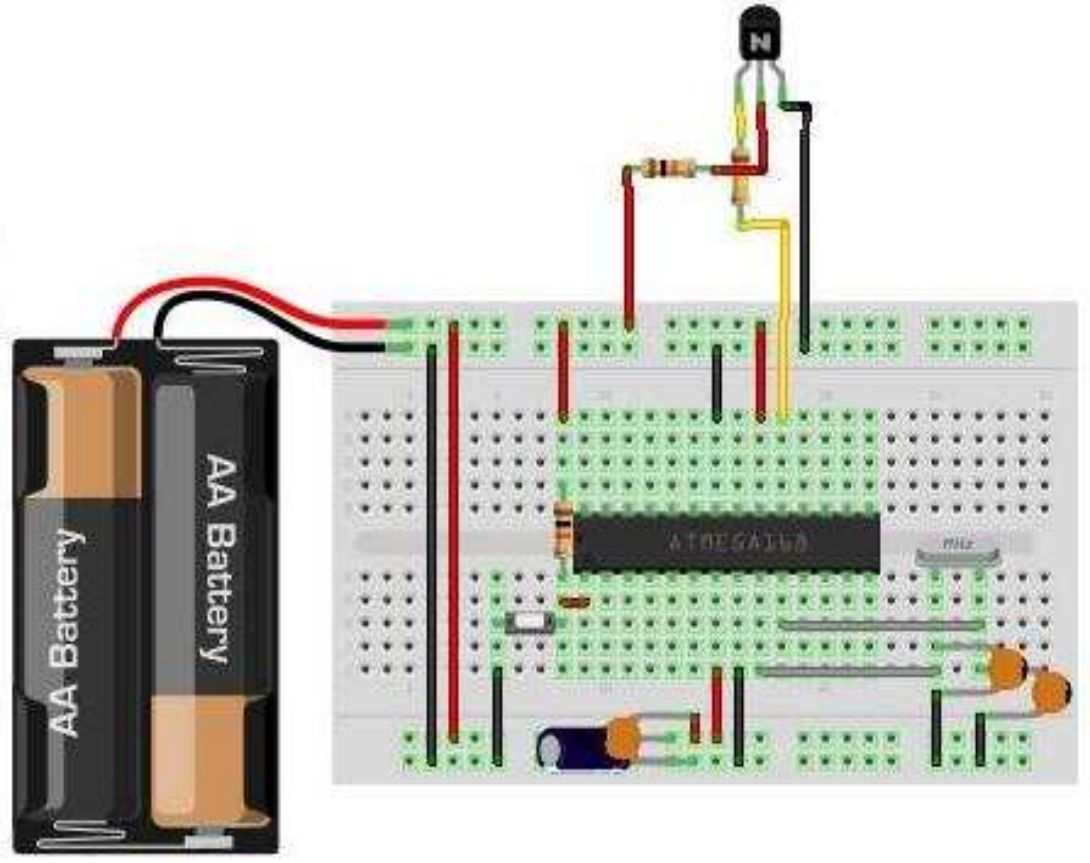
- Specifications
  - Pins
    - 23 General Purpose IOs



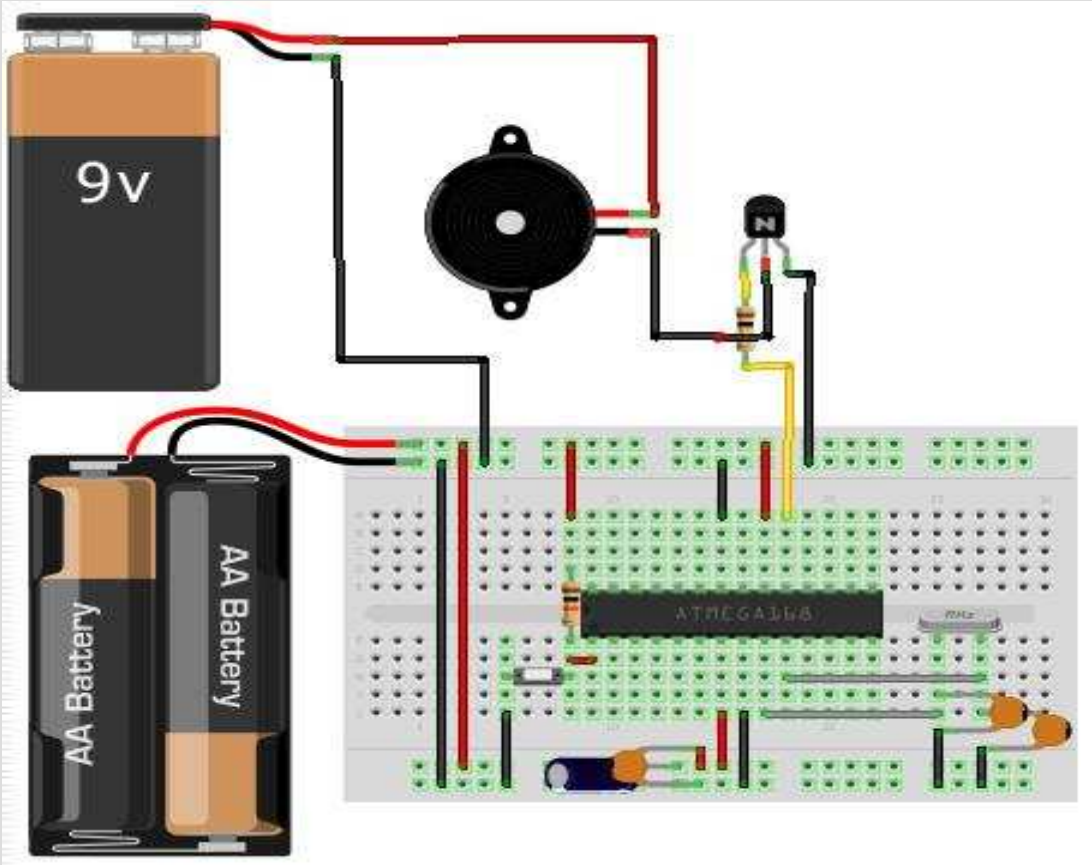
Outputs



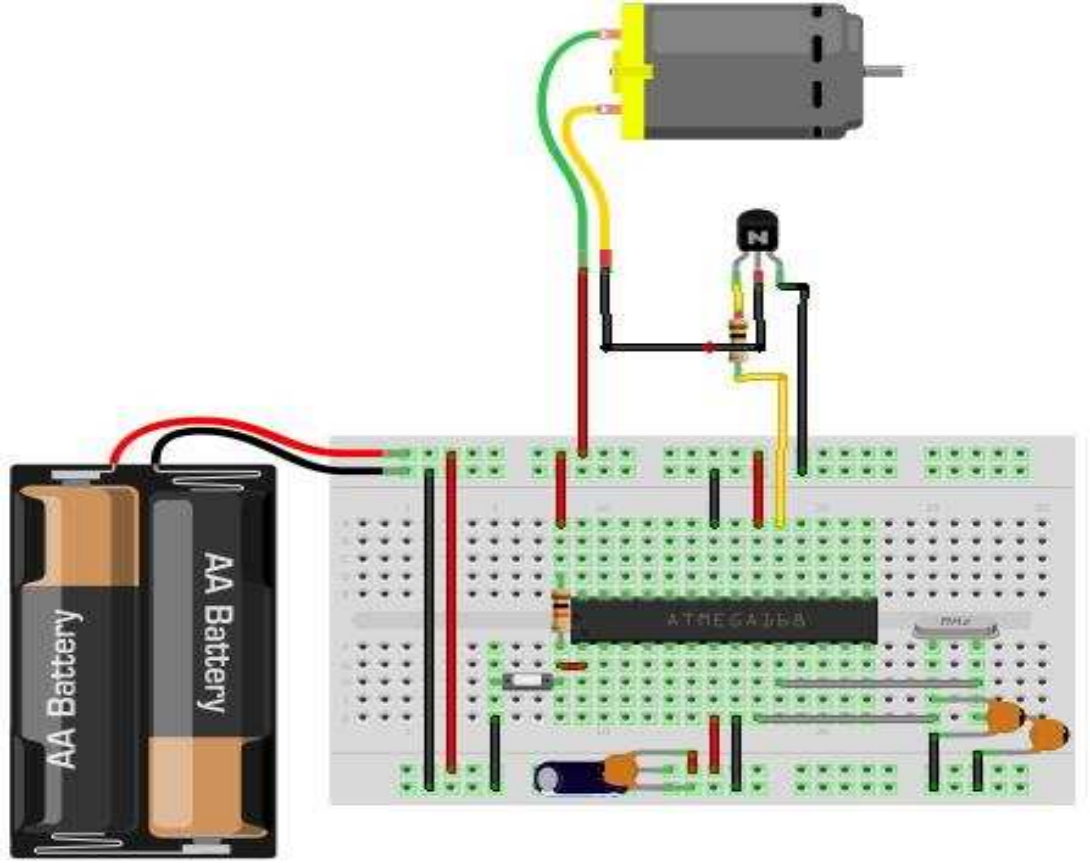
Outputs



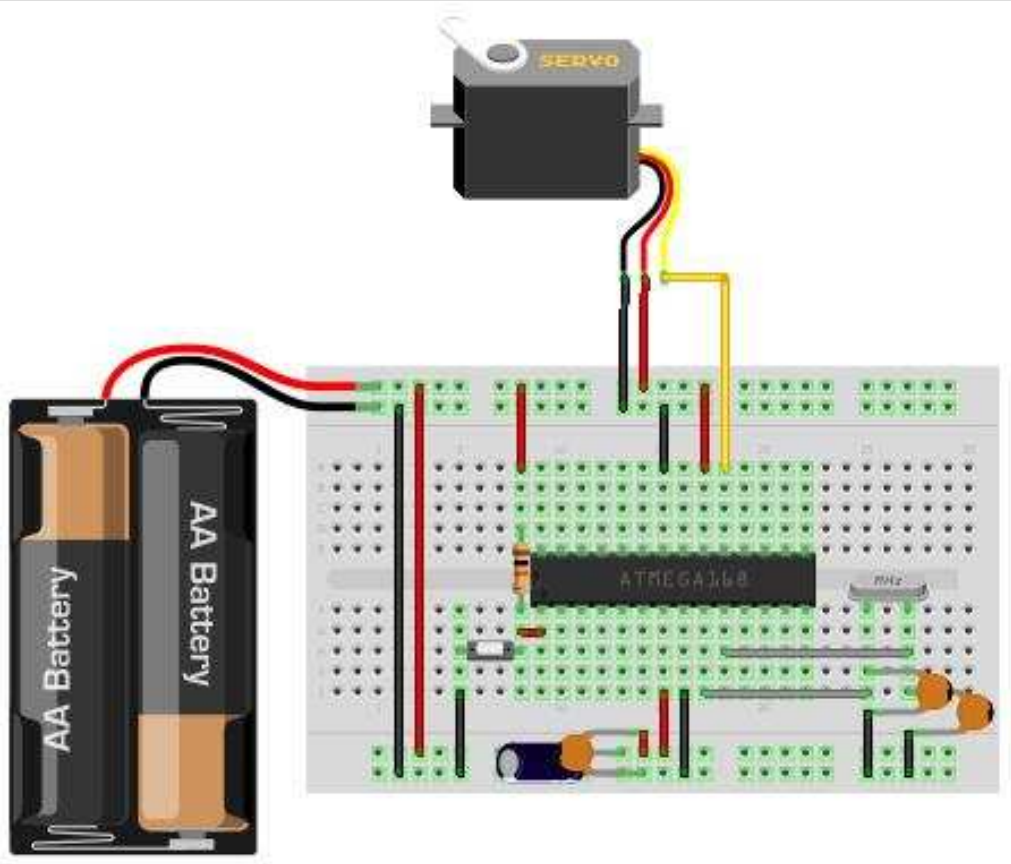
Outputs



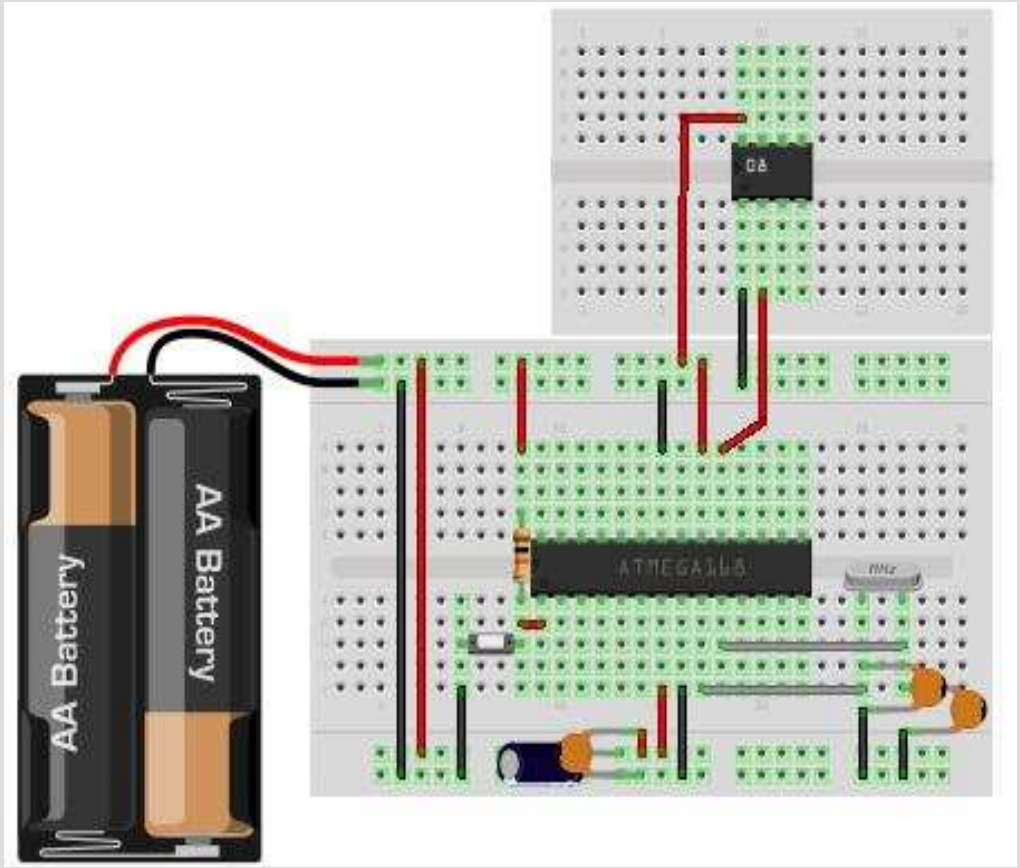
Outputs



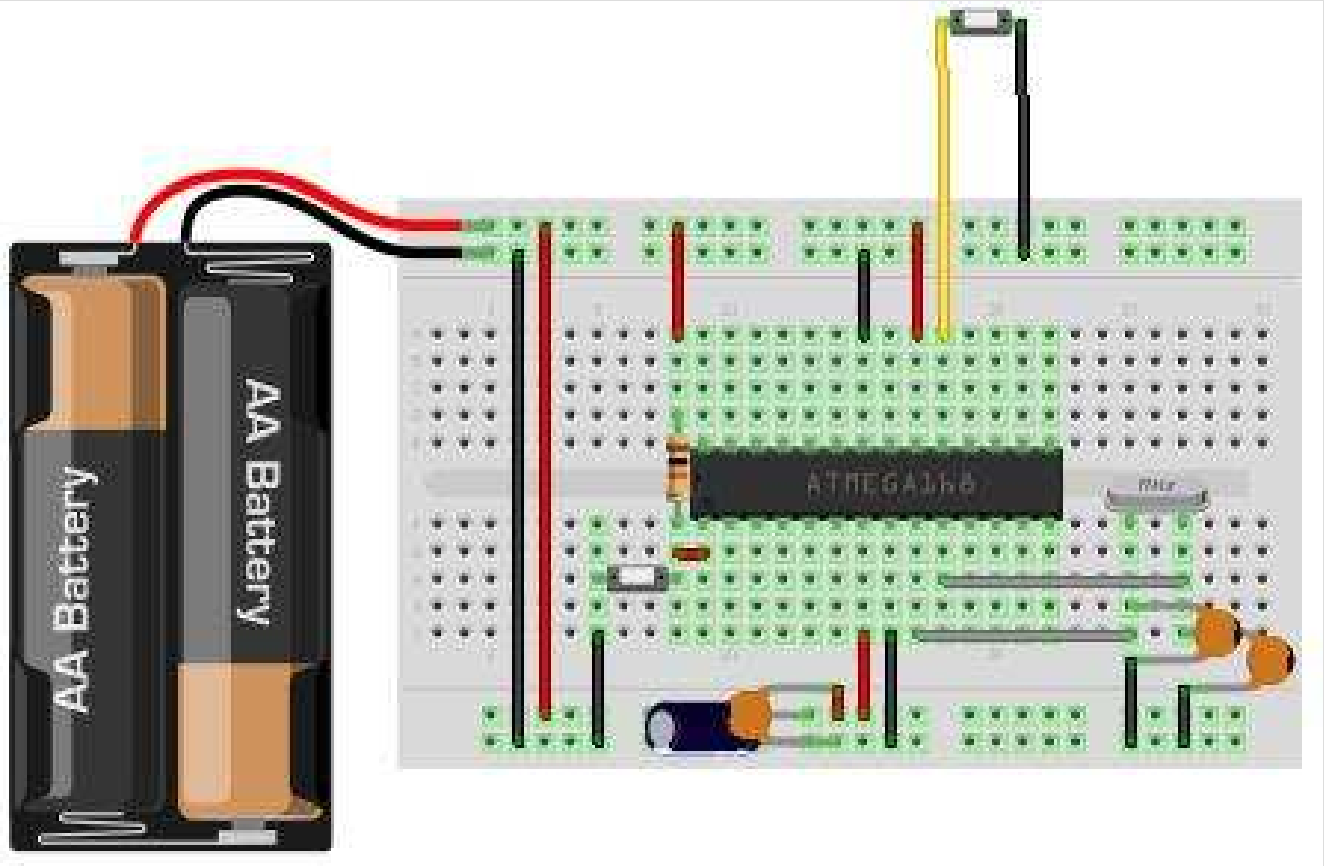
Outputs



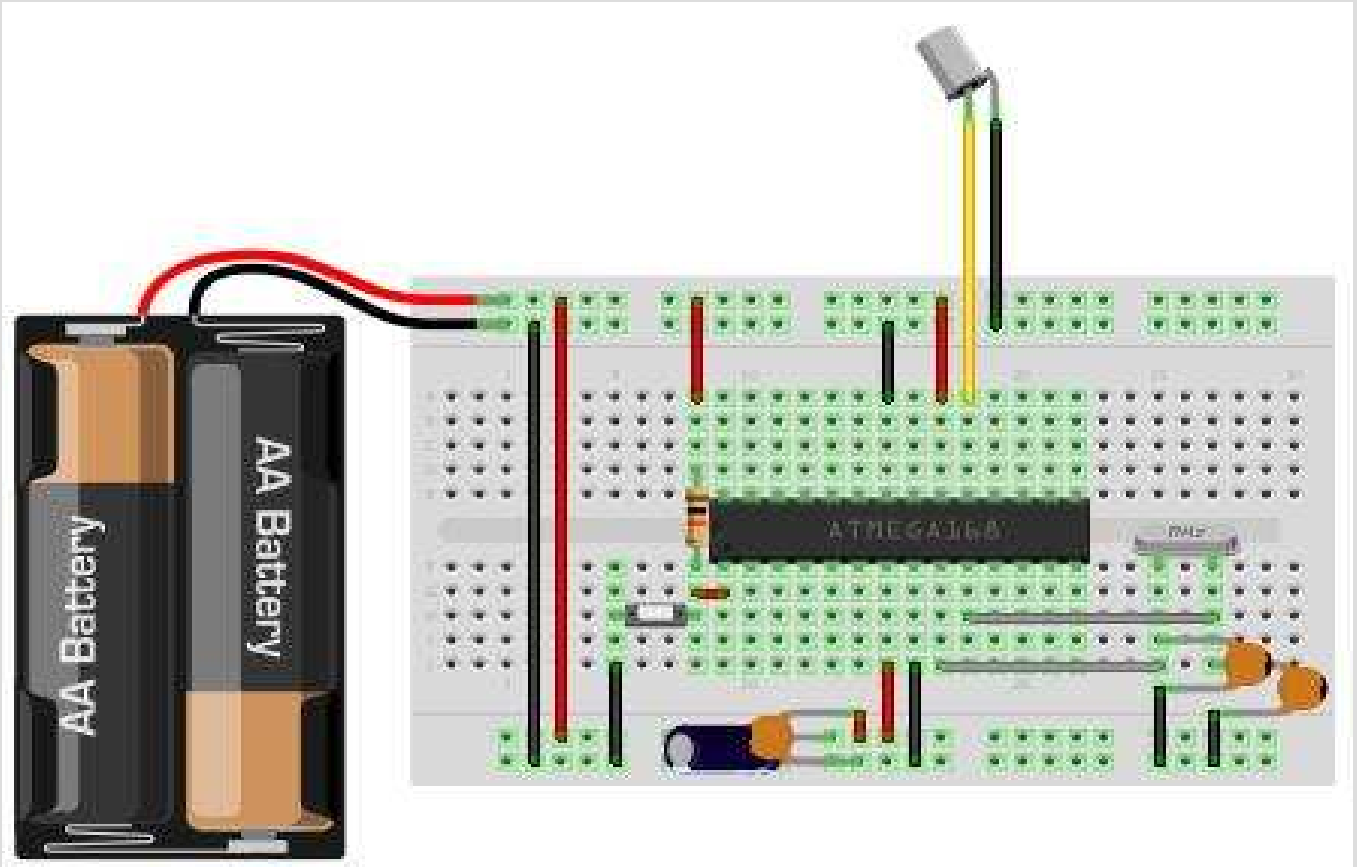
Outputs



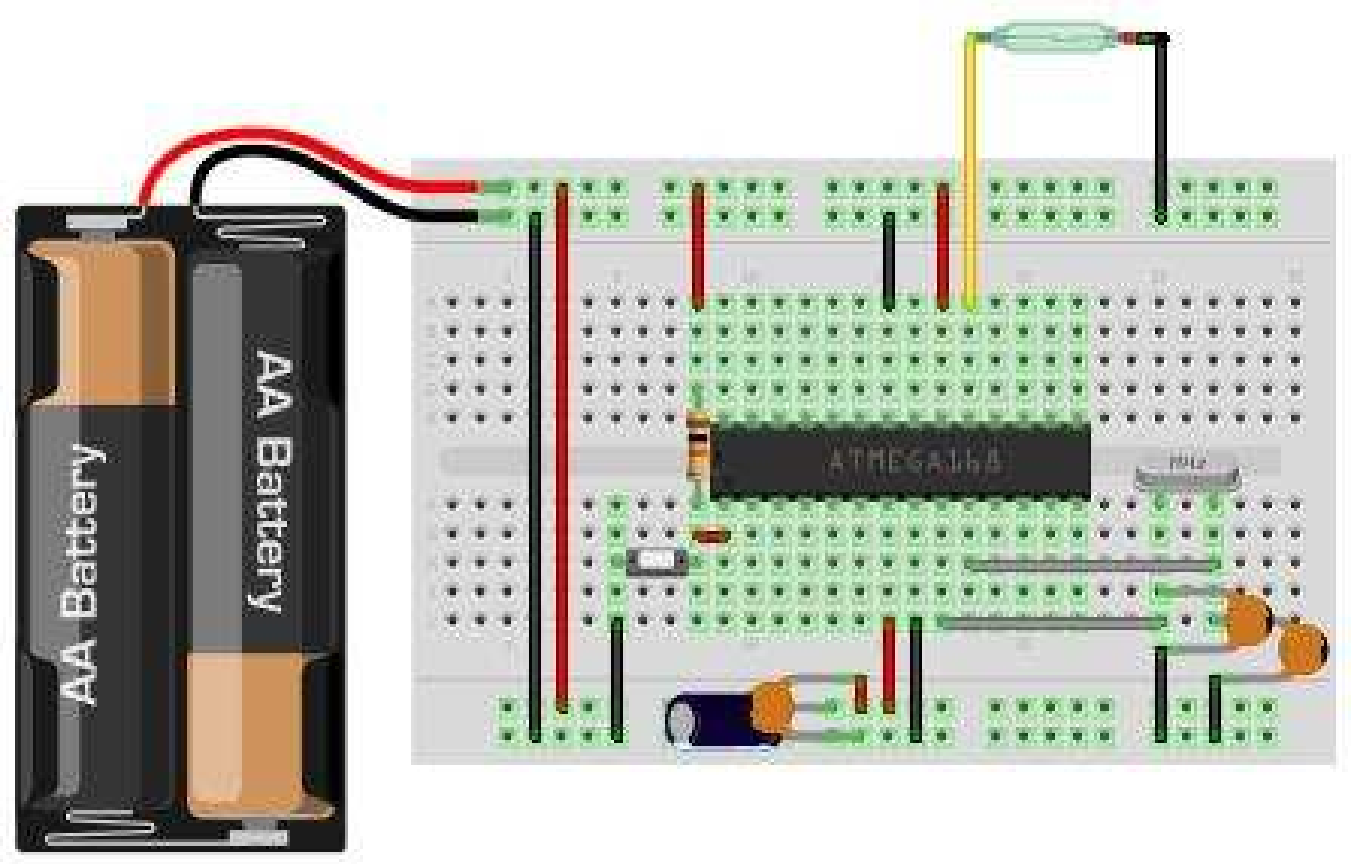
Inputs



Inputs

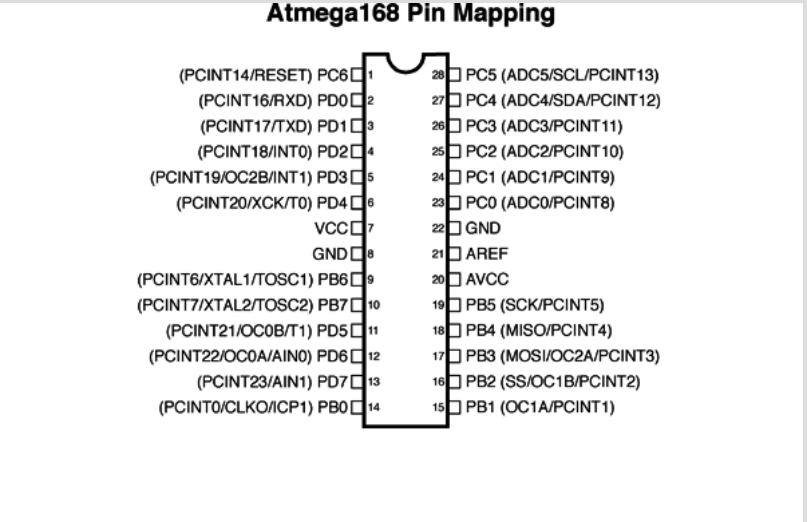


Inputs

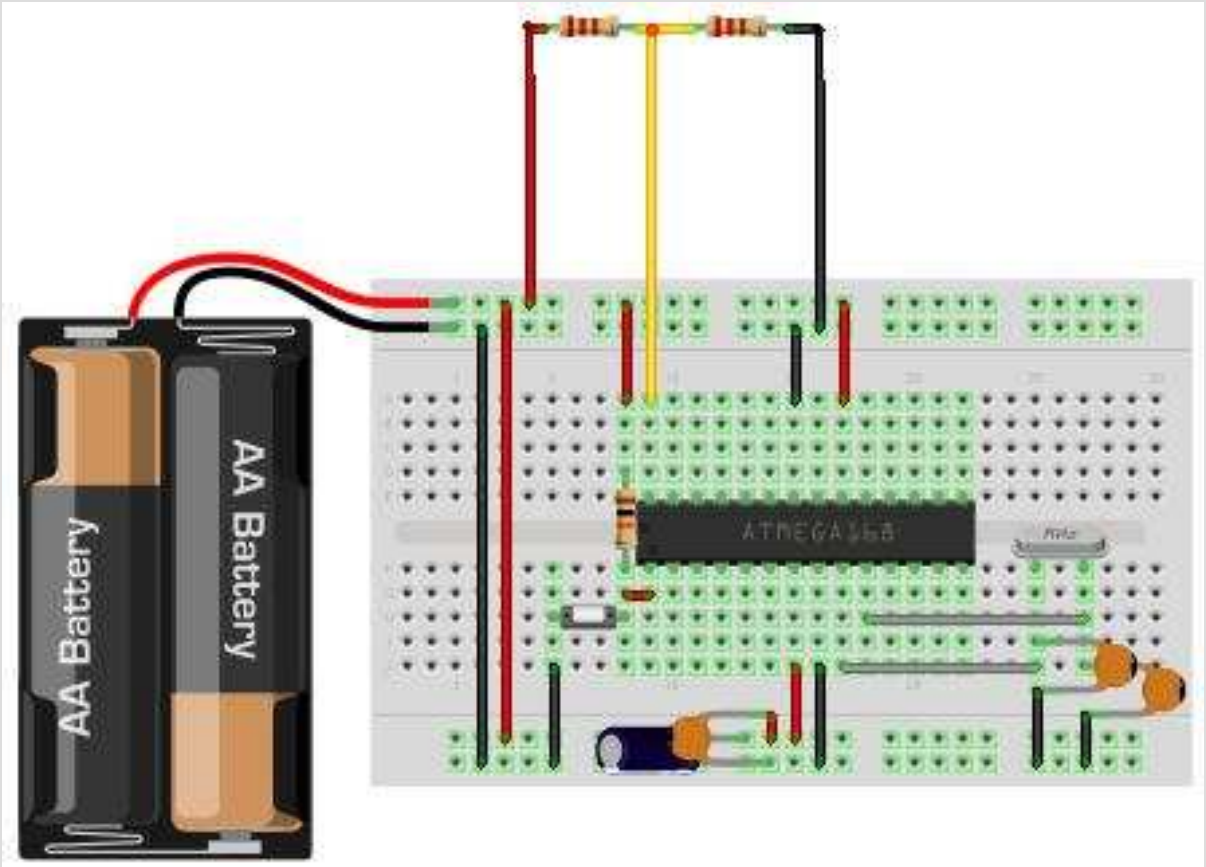


Atmega168

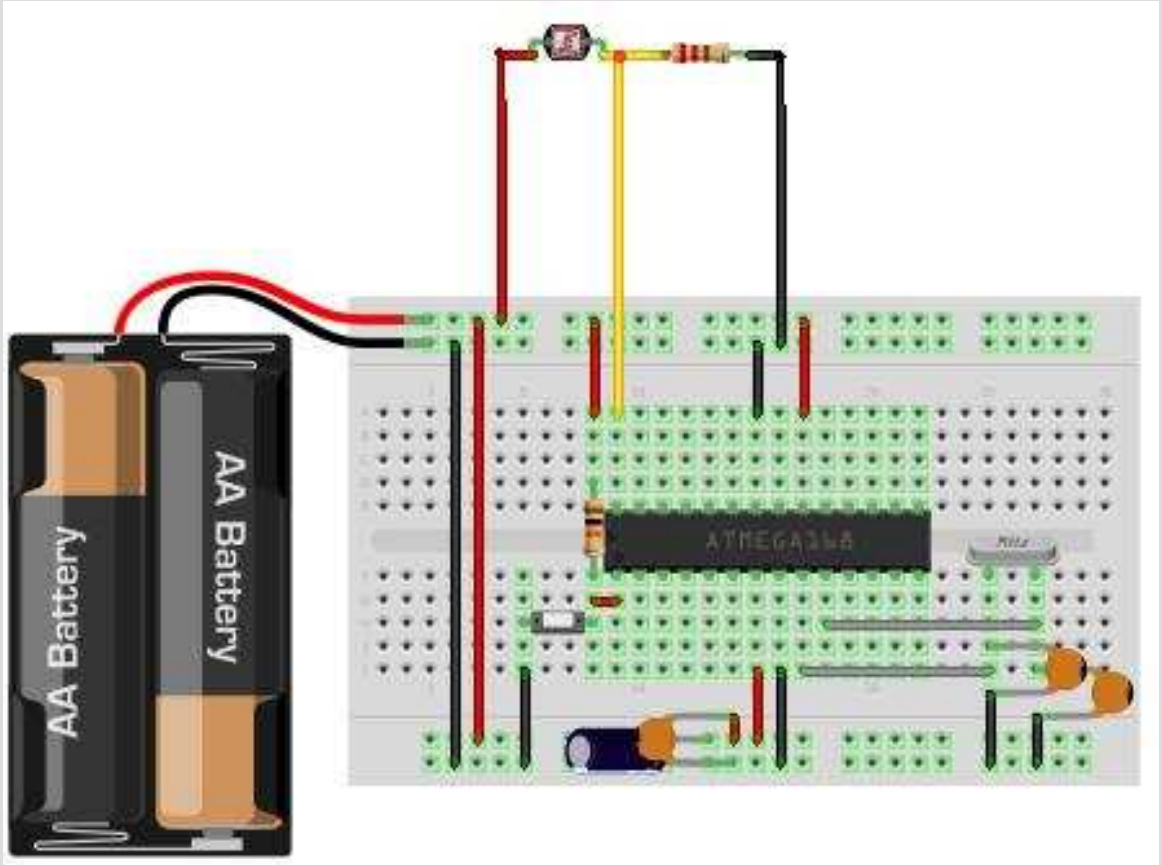
- Specifications
  - Pins
    - 23 General Purpose IOs
    - 6 Channel 10bit ADC



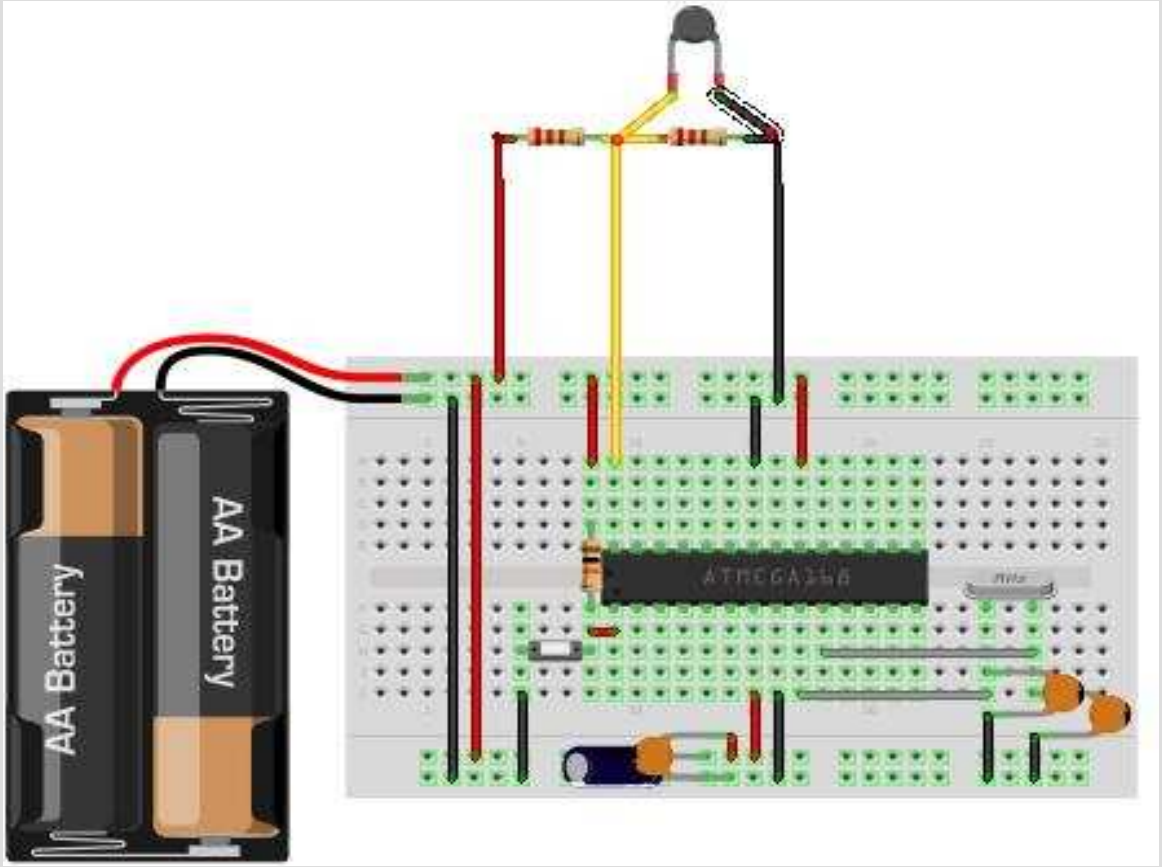
ADC



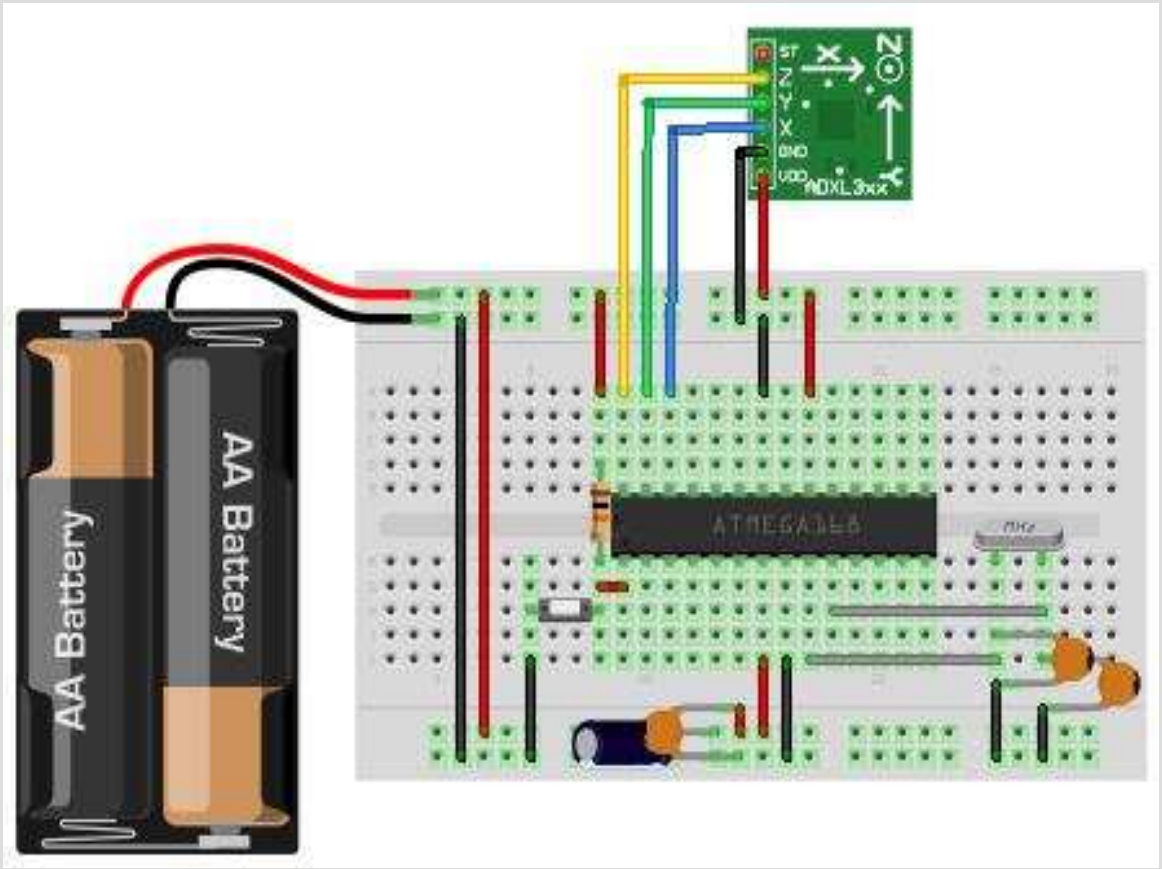
ADC



ADC

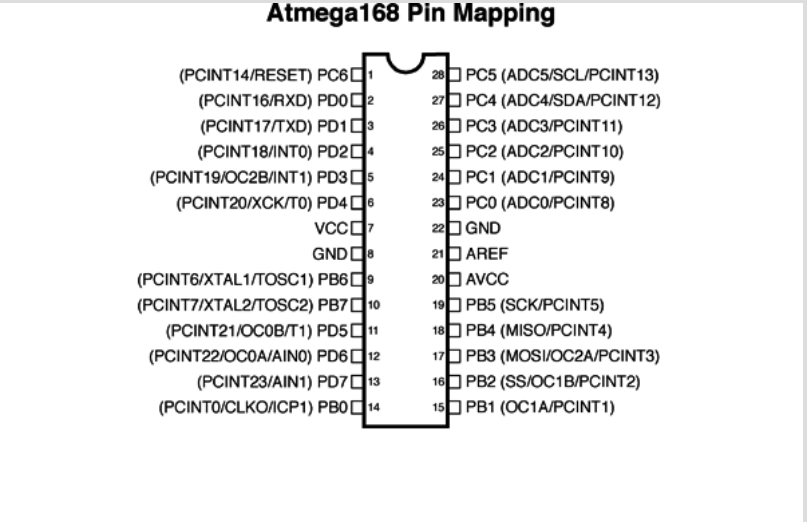


ADC

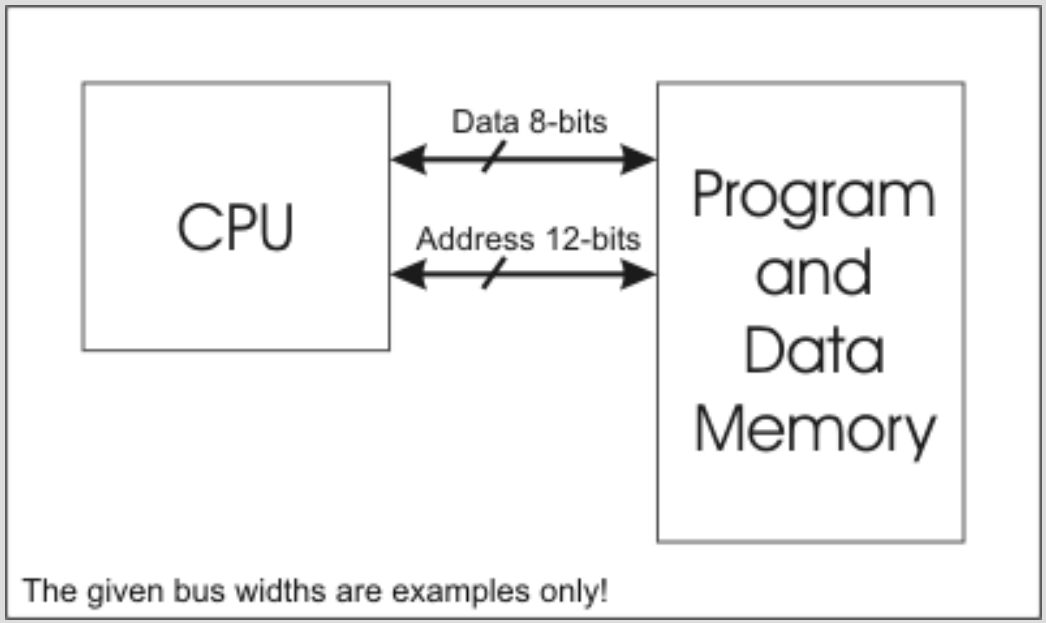


Atmega168

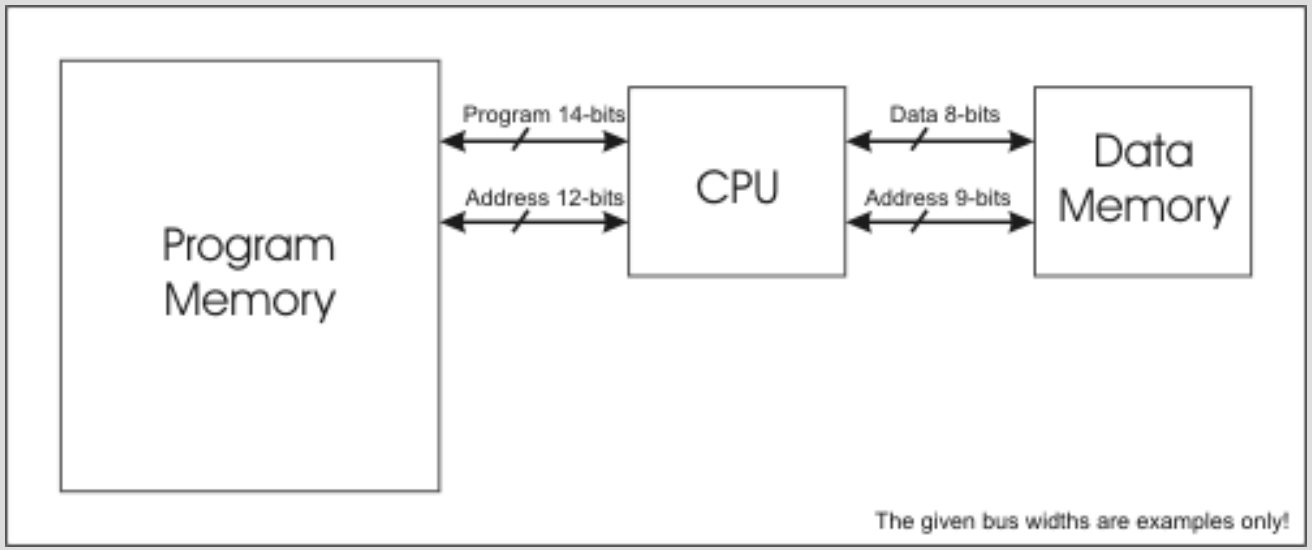
- Specifications
  - Pins
    - 23 General Purpose IOs
    - 6 Channel 10bit ADC
    - 6 PWM Channels
  - With Built-in Memory
    - 16 KBytes ROM
    - 1 KBytes RAM
    - 512 Bytes EEPROM



Von Neuman


CPU
Data 8-bits
Address 12-bits
Program and Data Memory
The given bus widths are examples only!

Hardvard


Program Memory
Program 14-bits
Address 12-bits
CPU
Data 8-bits
Address 9-bits
Data Memory
The given bus widths are examples only!

Programming Basic

- Challenge I
  - Instruct the robot to draw a ***10cm x 10cm yellow*** square starting ***5cm from top and 5cm*** from the left side of the paper.

Programming Basic

- Challenge II
  - Instruct the robot to draw that same square mentioned on Challenge I ***10 times more***.

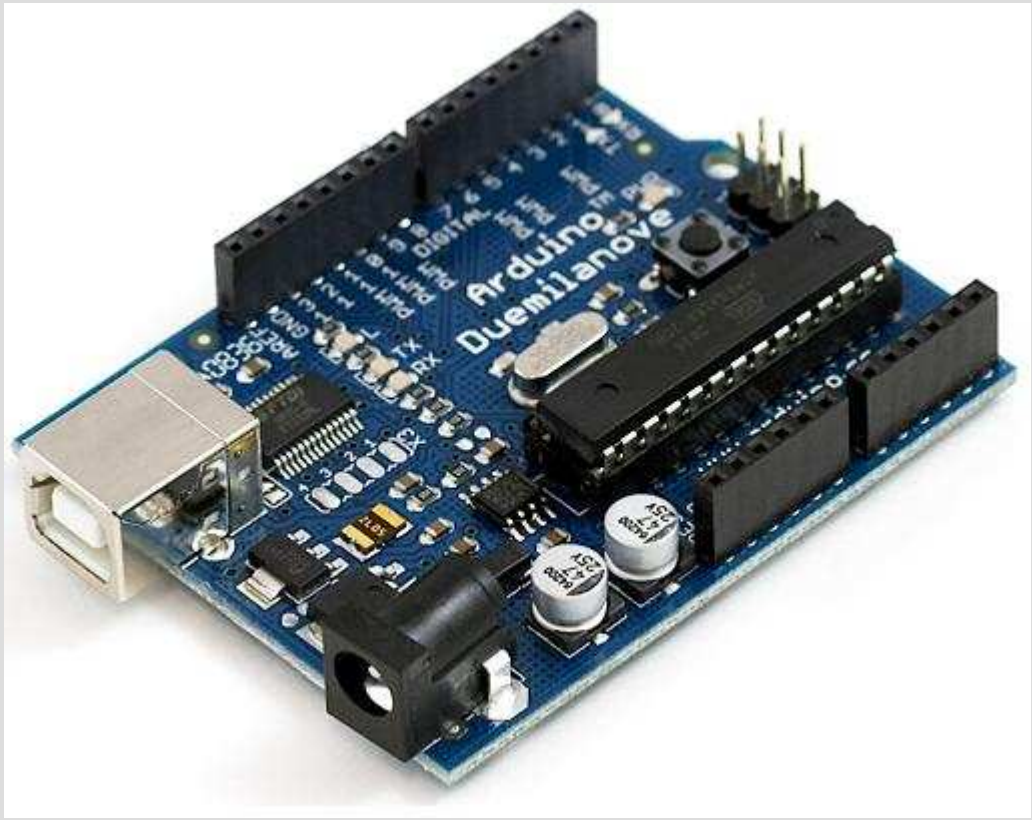
Programming Basic

- Challenge III
  - Instruct the robot to draw that same square for sets 10 pieces of paper ***forever***…

How MCU Execute Instructions

- Pipeline

Who is Arduino?



What is Arduino

- A project which began in ***Ivrea*** (Italy) in 2005
  - ***Massimo Banzi***
  - ***David Cuartielles***
- device for controlling ***student-built interaction design***
  - less expensive
  - more modern
  - than what is available
  - Basic Stamp

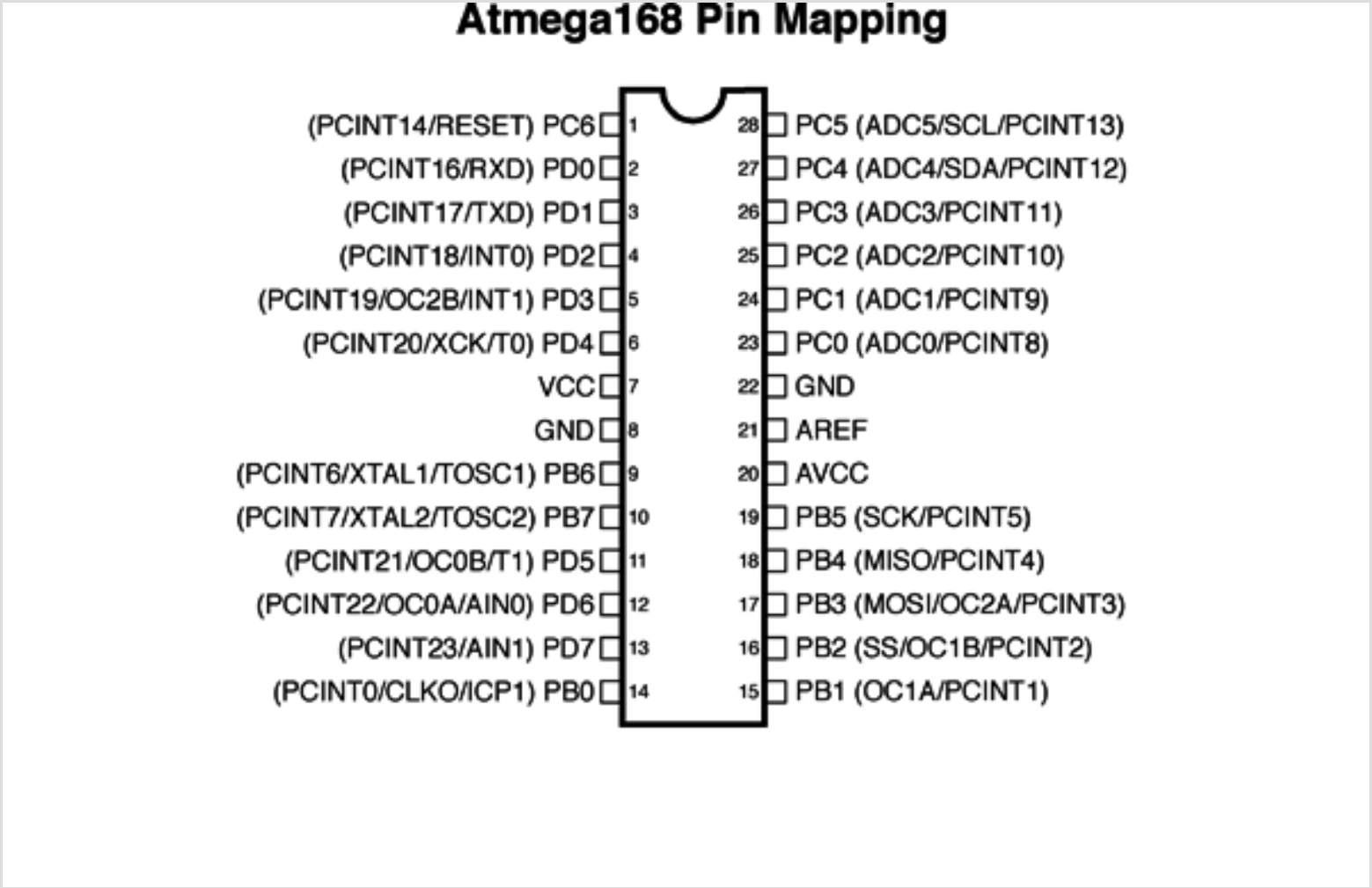
What is Arduino

- Open Source Hardware
  - Based on ***Wiring***
- Open Source Software
  - Based on ***Processing***
- Multi-patform
- ***Programmed via USB***

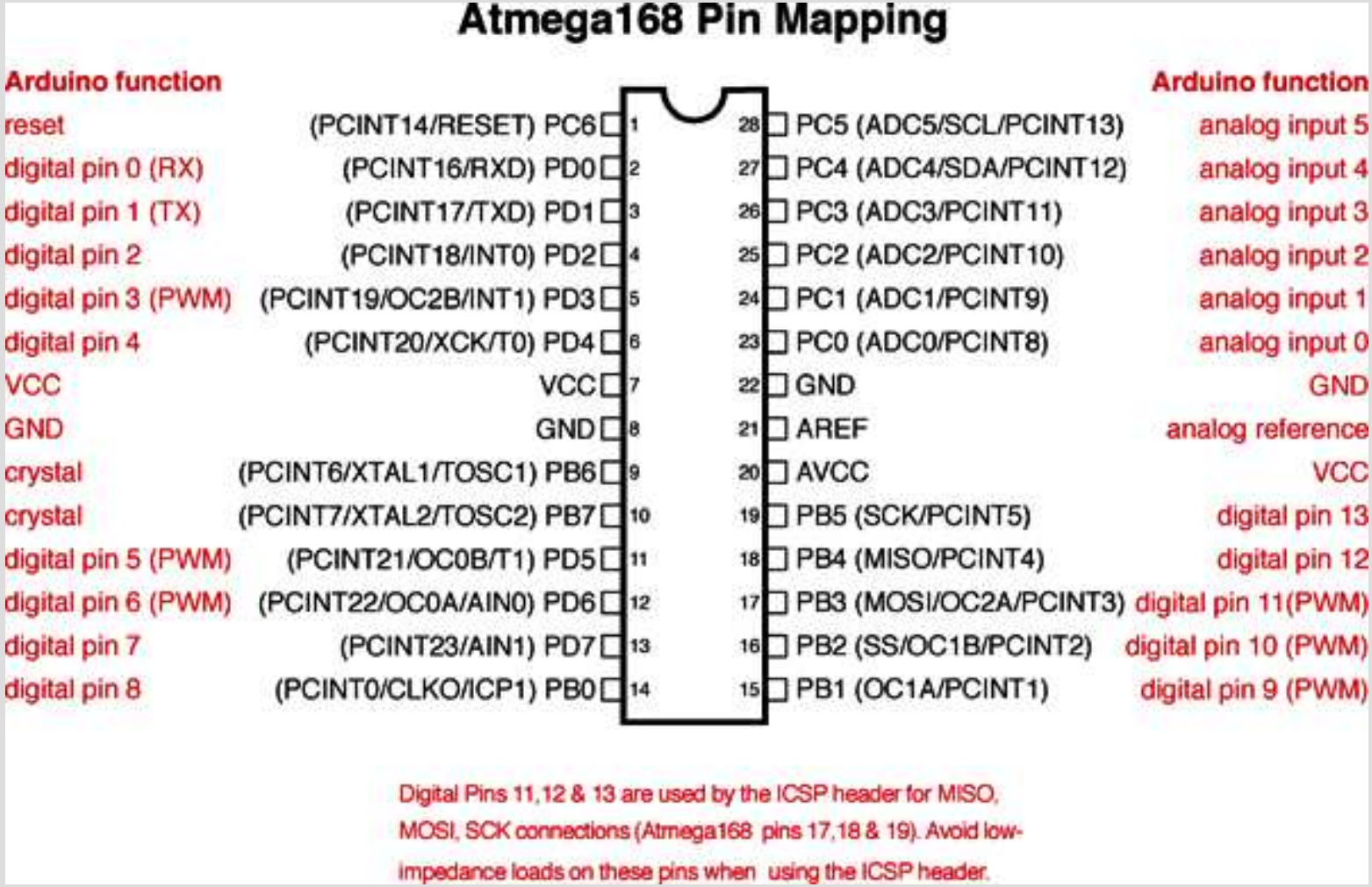
What is Arduino

- named the project after a ***local bar***
  - italian masculine first name
  - "***strong friend***"
  - English equivalent is "Hardwin"

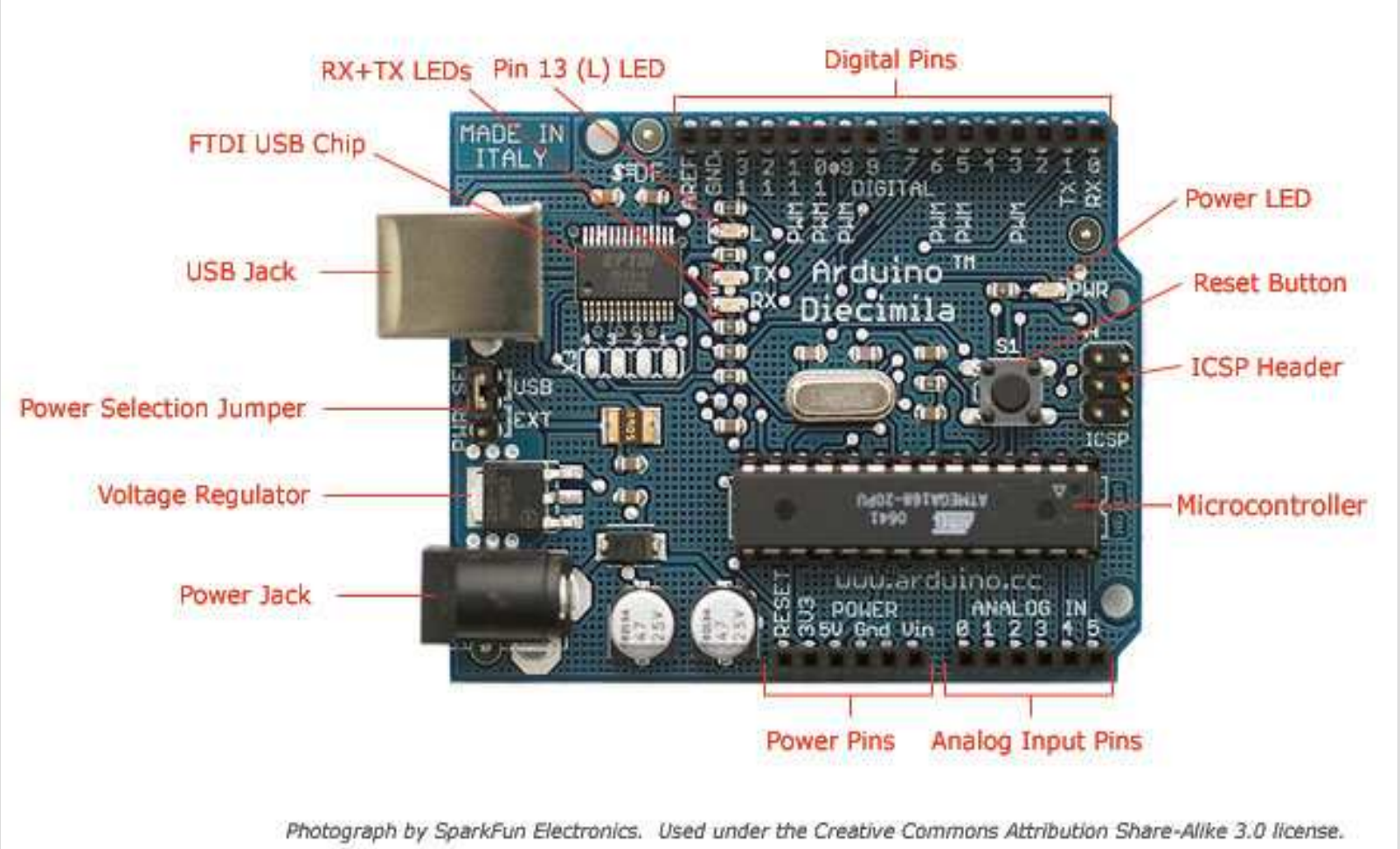
What is Arduino



What is Arduino



Arduino

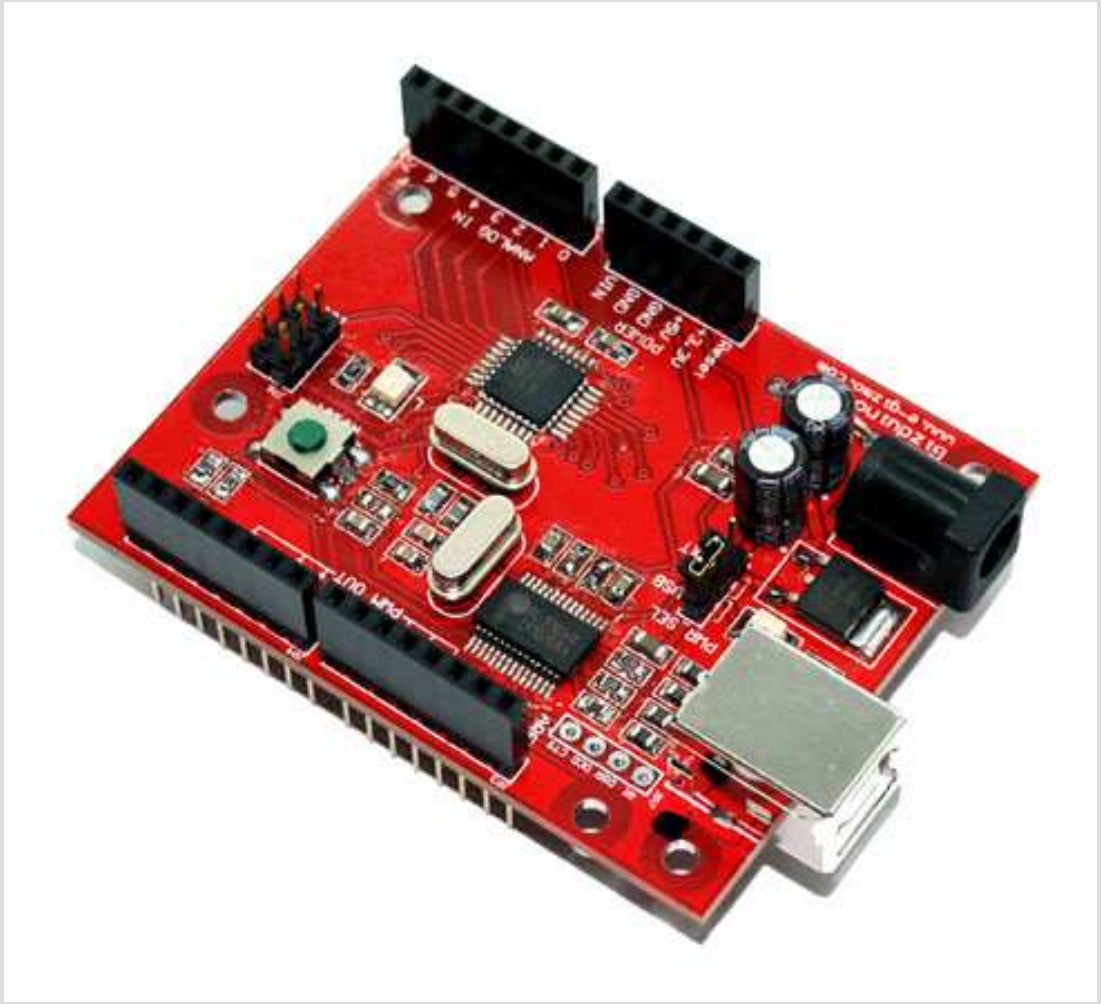


Photograph by SparkFun Electronics. Used under the Creative Commons Attribution Share-Alike 3.0 license.

Arduino


RobotShop
.com
Arduino
Duemilanove
ATMEGA 328
Arduino MEGA
GND
V+

Gizduino

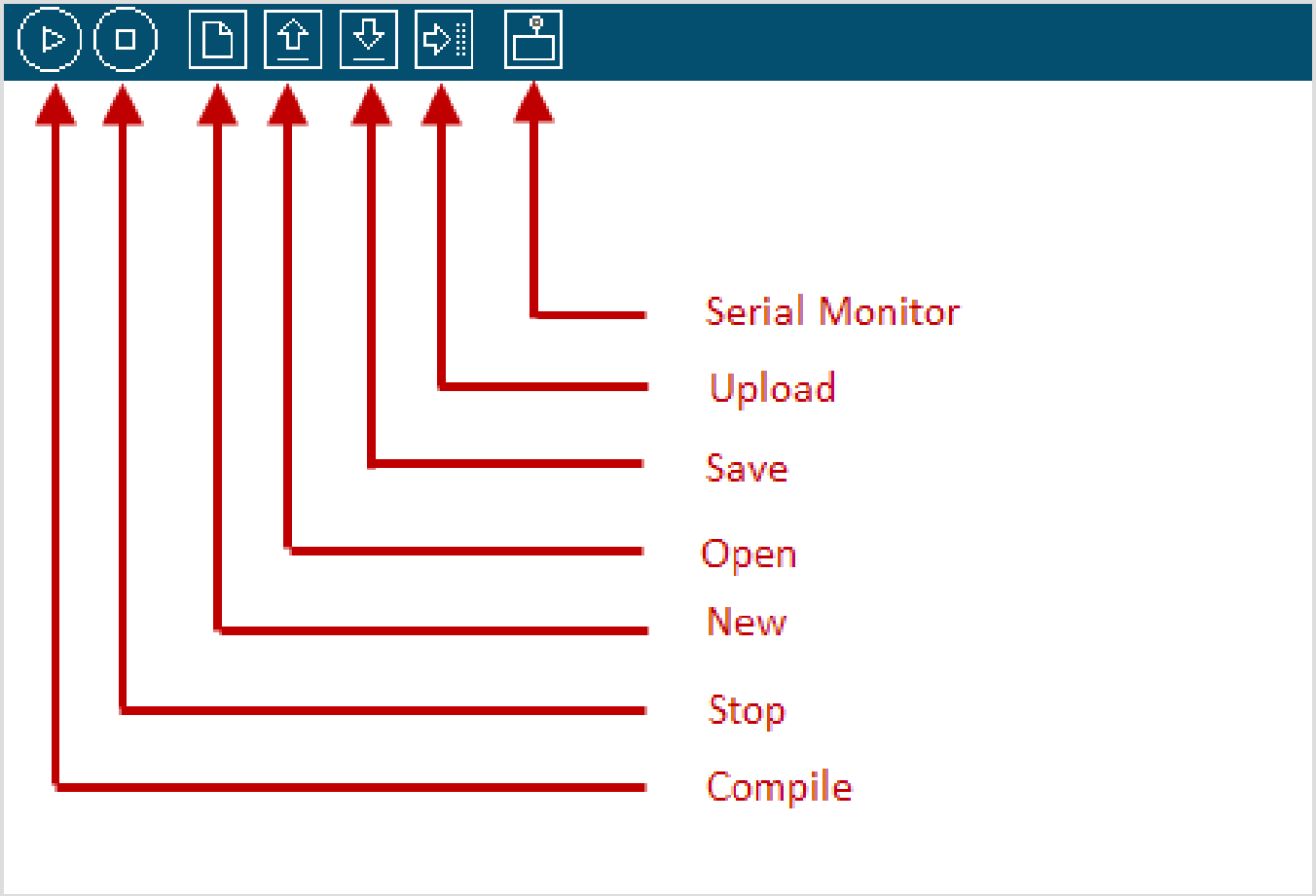


Getting Started with Arduino

- Arduino IDE

Getting Started with Arduino

- Arduino IDE



Getting Started with Arduino

- Arduino IDE
  - Homepage: http://arduino.cc/
  - Download: http://arduino.cc/en/Main/Software
  - arduino-00xx.zip

- Device Driver
  - Gizduino uses prolific chip

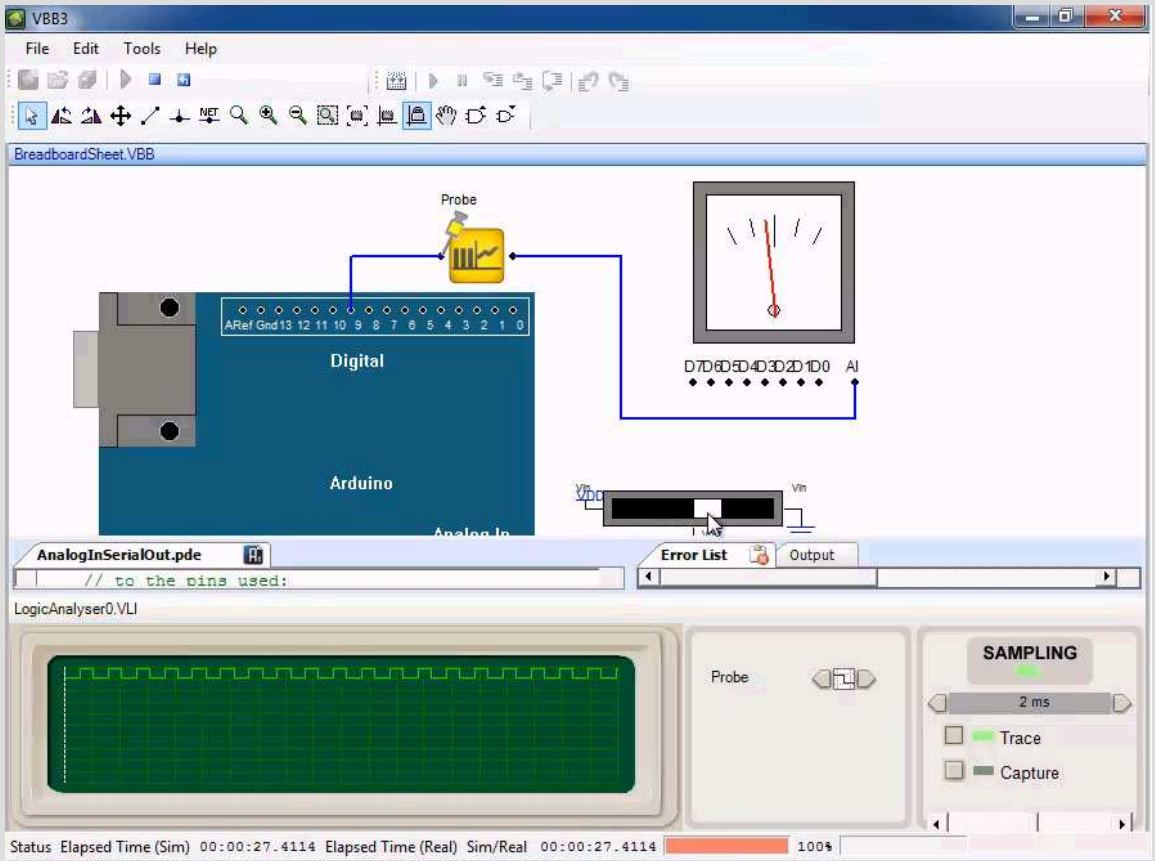
http://www.prolific.com.tw/eng/downloads.asp?id=31

Getting Started with Arduino

- Testing the Arduino
  - Run Clip 2...

Getting Started with Arduino

- Virtual Breadboard



Getting Started with Arduino

- Virtual Breadboard
  - Run clip 3...

Getting Started with Arduino

- Virtual Breadboard
  - Homepage: http://www.virtualbreadboard.net/
  - Download: http://www.virtualbreadboard.net/Download/tabid/150/Default.aspx

Getting Started with Arduino

- Virtual Breadboard
  - Dependencies
    - Latest version of DirectX
    - .Net 2.0 Redistributable
    - J# 2.0 Redistributable
    - Java JRE 1.6 (version 6)

Getting Started with Arduino

- Source Code
- Application
- Programmer
- Sketch
- IDE (Integrated Design Environment)

Arduino Programming

- Setup and Loop

```
void setup()
{

}

void loop()
{

}
```

Arduino Programming

- Setup()
  - ***called when a sketch starts running***
  - use it to initialize
    - variables,
    - pin modes,
    - start using libraries
  - will only ***run once***, after each powerup or reset of the Arduino board

Arduino Programming

- Setup()

```
int LEDPin;

void setup()
{
  LEDPin = 13;
  pinMode(LEDPin, OUTPUT);
  digitalWrite(LEDPin, LOW);
}

void loop()
{

}
```

Arduino Programming

- Setup()

```
int LEDPin;

void setup()
{
  LEDPin = 13;
  pinMode(LEDPin, OUTPUT);
  digitalWrite(LEDPin, LOW);
}

void loop()
{

}
```

Arduino Programming

- Loop()
  - ***Executed repeatedly***
  - Allows the program to change and respond.

Arduino Programming

- Loop()

```
int LEDPin;

// setup initializes the LED pin as output
// and initially turned off
void setup()
{
  LEDPin = 13;
  pinMode(LEDPin, OUTPUT);
  digitalWrite(LEDPin, LOW);
}

// loop checks the button pin each time,
void loop()
{
  delay(500);
  digitalWrite(LEDPin, HIGH);
  delay(500);
  digitalWrite(LEDPin, LOW);
}
```

Arduino Programming

- Loop()

```
int LEDPin;

// setup initializes the LED pin as output
// and initially turned off
void setup()
{
  LEDPin = 13;
  pinMode(LEDPin, OUTPUT);
  digitalWrite(LEDPin, LOW);
}

// loop checks the button pin each time,
void loop()
{
  delay(500);
  digitalWrite(LEDPin, HIGH);
  delay(500);
  digitalWrite(LEDPin, LOW);
}
```

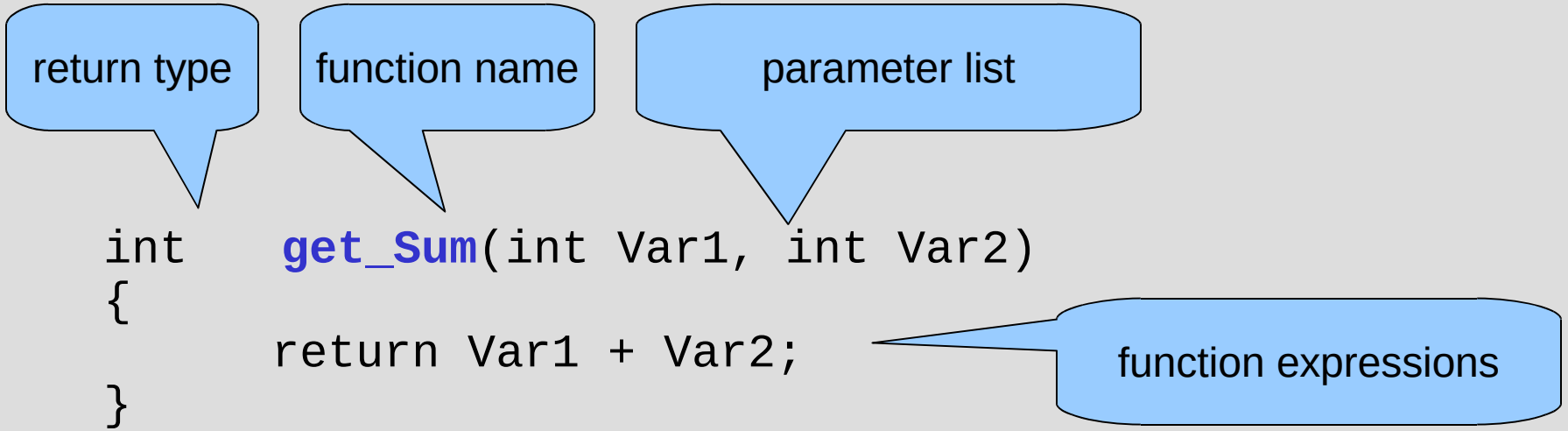
Arduino Programming

- Functions
  - a small ***set / group of instructions*** designed to operate on its given input

Read more: http://wiki.answers.com/Q/What_is_function_in_C#ixzz1AhkkJ9RB

Arduino Programming

- Functions



Arduino Programming

- Function name
  - ***Unique identifier*** for a function
- Parameter / Argument List
  - ***Inputs*** to a function
- Function Return
  - **Output** of a function
- Expressions
  - ***Describes*** what ***a function*** do

Arduino Programming

- Calling a Functions

```
void loop()
{
      result = get_Sum(2,3);
}

int get_Sum(int Var1, int Var2)
{
       return Var1 + Var2;
}
```

will give us a result of 5

functions must be defined before it can be used

Arduino Programming

- Arduino ***Built-in Functions***
  - Predefined functions to easily configure Arduino
    - Pin Operation
    - Peripheral Operation
  - shortcuts

Arduino Programming

- **pinMode(pin, mode)**
  - Configures a digital pin to behave either as an input or an output
  - ***pin:*** the pin number whose mode you wish to set
  - ***mode:*** either INPUT or OUTPUT
- Example

```
pinMode(13, OUTPUT);
```

Arduino Programming

- **digitalWrite(pin, value)**
  - Write a HIGH or a LOW value to a digital pin
  - ***pin:*** the pin number whose value you wish to set
  - ***value:*** either HIGH or LOW
- Example

```
digitalWrite(13, HIGH);
```

Arduino Programming

- **digitalRead(pin)**
  - Reads the value from a specified digital pin
  - ***pin:*** the digital pin number you want to read
  - returns either HIGH or LOW
- Example

```
result = digitalRead(13);
```

Arduino Programming

- **analogRead(pin)**
  - Reads the value from a specified analog pin
  - ***pin:*** the analog pin number you want to read
  - returns an integer (0 to 1023)corresponding to pin voltage
- Example

```
result = analogRead(13);
```

result = pinV * 1023/Vref

Arduino Programming

- **analogWrite(pin, value)**
  - Writes an analog value (PWM wave) to a pin
  - pin will generate a steady square wave of the specified duty cycle
  - 490 Hz

Arduino Programming

- **analogWrite(pin, value)**
  - Can be used to vary
    - Brightness of LED
    - Speed of the motor
  - Output a sound
    - Low Pass Filter needed

Arduino Programming

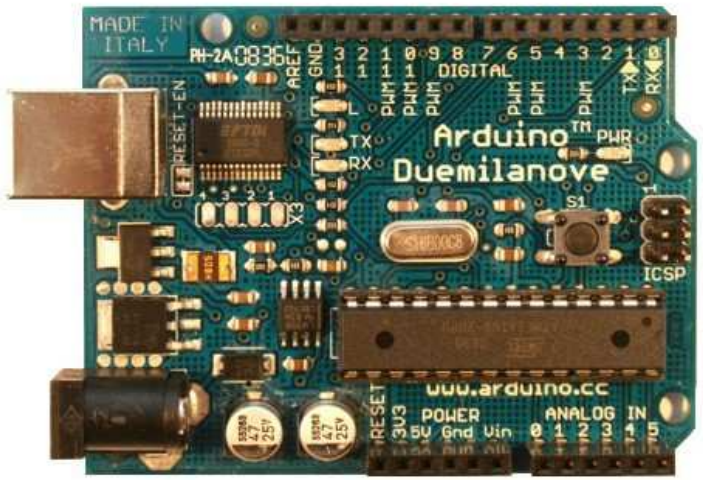
- **analogWrite(pin, value)**
  - ***pin:*** the pin to write to
  - ***value:*** the duty cycle: between 0 (always off) and 255 (always on)
- Example

```
analogWrite(9, 127);
```

Duty_Cycle = value / 255 * 100%
~50% = 100 / 255 * 100%

Arduino Programming

- **analogWrite(pin, value)**



Arduino Programming

- **delay(value)**
  - Pauses the program for the amount of time
  - **value:** the number of milliseconds to pause
- Example

```
delay(1000);
```

will pause for 1000ms, equivalent to 1sec pause

Arduino Programming

- Blinking LED Example

```
int LEDPin;

// setup initializes the LED pin as output
// and initially turned off
void setup()
{
  LEDPin = 13;
  pinMode(LEDPin, OUTPUT);
  digitalWrite(LEDPin, LOW);
}

// loop checks the button pin each time,
void loop()
{
  delay(500);
  digitalWrite(LEDPin, HIGH);
  delay(500);
  digitalWrite(LEDPin, LOW);
}
```

Arduino Programming

- Comments

```
int LEDPin;

// setup initializes the LED pin as output
// and initially turned off
void setup()
{
  LEDPin = 13;
  pinMode(LEDPin, OUTPUT);
  digitalWrite(LEDPin, LOW);
}

// loop checks the button pin each time,
void loop()
{
  delay(500);
  digitalWrite(LEDPin, HIGH);
  delay(500);
  digitalWrite(LEDPin, LOW);
}
```

Arduino Programming

- Comments

```
int LEDPin;

// setup initializes the LED pin as output
// and initially turned off
void setup()
{
  LEDPin = 13;
  pinMode(LEDPin, OUTPUT);
  digitalWrite(LEDPin, LOW);
}

// loop checks the button pin each time,
void loop()
{
  delay(500);
  digitalWrite(LEDPin, HIGH);
  delay(500);
  digitalWrite(LEDPin, LOW);
}
```

Arduino Programming

- Comments
  - Starts with characters '***//***'
  - Text ***after the start marker are ignored*** by the compiler
  - This is for the programmer (person)

Arduino Programming

```
int LEDPin;

// setup initializes the LED pin as output
// and initially turned off
void setup()
{
  LEDPin = 13;
  pinMode(LEDPin, OUTPUT);
  digitalWrite(LEDPin, LOW);
}

// loop checks the button pin each time,
void loop()
{
  delay(500);
  digitalWrite(LEDPin, HIGH);
  delay(500);
  digitalWrite(LEDPin, LOW);
}
```

Arduino Programming

- Variables

```
int LEDPin;

// setup initializes the LED pin as output
// and initially turned off
void setup()
{
  LEDPin = 13;
  pinMode(LEDPin, OUTPUT);
  digitalWrite(LEDPin, LOW);
}

// loop checks the button pin each time,
void loop()
{
  delay(500);
  digitalWrite(LEDPin, HIGH);
  delay(500);
  digitalWrite(LEDPin, LOW);
}
```

Arduino Programming

- Variables

```
int LEDPin;

// setup initializes the LED pin as output
// and initially turned off
void setup()
{
  LEDPin = 13;
  pinMode(LEDPin, OUTPUT);
  digitalWrite(LEDPin, LOW);
}

// loop checks the button pin each time,
void loop()
{
  delay(500);
  digitalWrite(LEDPin, HIGH);
  delay(500);
  digitalWrite(LEDPin, LOW);
}
```

Arduino Programming

- Variables

```
int LEDPin;

// setup initializes the LED pin as output
// and initially turned off
void setup()
{
  LEDPin = 13;
  pinMode(LEDPin, OUTPUT);
  digitalWrite(LEDPin, LOW);
}

// loop checks the button pin each time,
void loop()
{
  delay(500);
  digitalWrite(LEDPin, HIGH);
  delay(500);
  digitalWrite(LEDPin, LOW);
}
```

Arduino Programming

- Variables
  - memory location in ***RAM*** that carries a value
- Variables ***must be declared***
  - name is chosen to uniquely identify a variable
  - the type defines
    - how much memory is used
    - and how it is stored

Arduino Programming

- Data Types

Type	Description	Sample
boolean	boolean	true, false
char	character	A-Z, a-z, CR, LF
unsigned char	same as byte	0 -255
byte	occupies 1 byte / 8bits	0 - 255
int	integer, occupies 2 bytes / 16bit	-32,768 to 32,767
unsigned int	unsigned integer, occupies 2 bytes / 16bit	0 to 65,535
word	same as unsigned int	0 to 65,535
long	long integer, 4bytes / 32bits	-2,147,483,648 to 2,147,483,647
unsigned long	unsigned long integer, 4bytes / 32bits	0 to 4,294,967,295
float	floating point, decimal point, 4bytes / 32bits	6-7 decimal digits

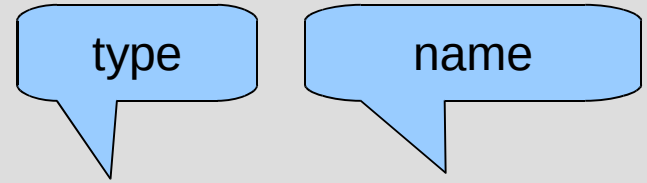
Arduino Programming

- Declaring a Variables

```
int       VariableName1;
```

Arduino Programming

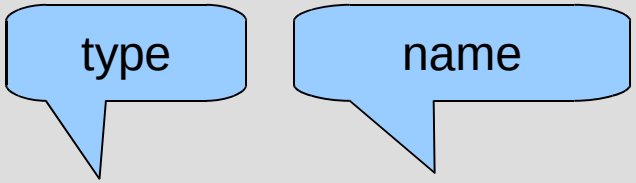
- Declaring a Variables




```
int         VariableName1;
```

Arduino Programming

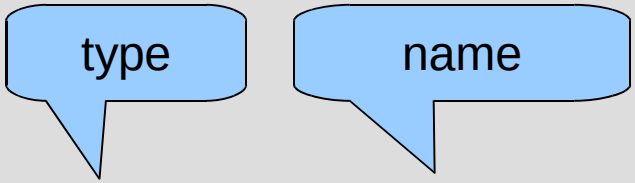
- Declaring a Variables




```
int         VariableName1;
byte        VariableName2
```

Arduino Programming

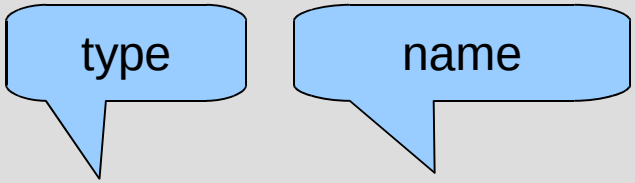
- Declaring a Variables




```
int       VariableName1;
byte      VariableName2    // not terminated by semicolon
```

Arduino Programming

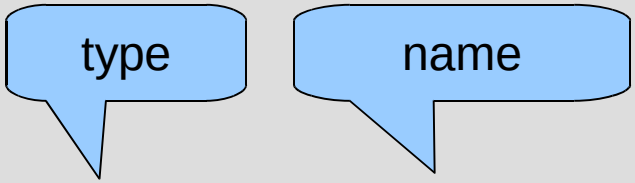
- Declaring a Variables




```
int         VariableName1;
byte        VariableName2    // not terminated by semicolon
int         VariableName3, VariableName4, VariableName5;
```

Arduino Programming

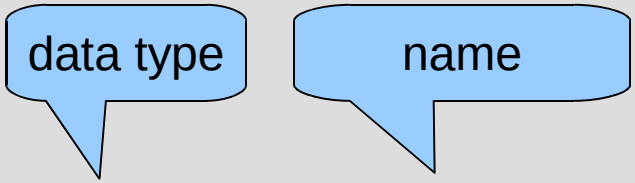
- Declaring a Variables




```
int         VariableName1;
byte        VariableName2    // not terminated by semicolon
int         VariableName3, VariableName4, VariableName5;
```

Arduino Programming

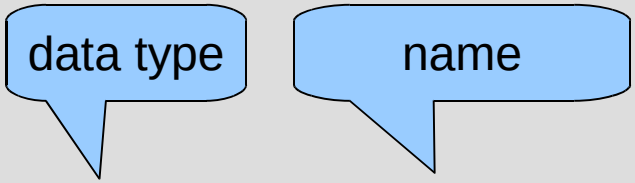
- Declaring a Variables




```
int         VariableName1;
byte        VariableName2    // not terminated by semicolon
int         VariableName3, VariableName4, VariableName5;
void        Variavle6;
```

Arduino Programming

- Declaring a Variables




```
int       VariableName1;
byte      VariableName2    // not terminated by semicolon
int       VariableName3, VariableName4, VariableName5;
void      Variavle6;       // there is no void type
```

Arduino Programming

- Naming a Variable
  - ***Starts with an alphabet character***
  - ***No spaces, no special characters***

```
int       1Delta;
```

Arduino Programming

- Naming a Variable
  - Starts with an alphabet character
  - No spaces, no special characters

```
int       1Delta;           // variable names cannot start
                            // with numbers
```

Arduino Programming

- Naming a Variable
  - Starts with an alphabet character
  - No spaces, no special characters

```
int       1Delta;          // variable names cannot start
                           // with numbers
byte      Delta1;
```

Arduino Programming

- Naming a Variable
  - Starts with an alphabet character
  - No spaces, no special characters

```
int       1Delta;            // variable names cannot start
                             // with numbers
byte      Delta1;
int       Serial Number;
```

Arduino Programming

- Naming a Variable
  - Starts with an alphabet character
  - No spaces, no special characters

```
int       1Delta;            // variable names cannot start
                             // with numbers
byte      Delta1;
int       Serial Number;     // must contain no spaces
                             // SerialNumber
                             // Serial_Number
```

Arduino Programming

- Naming a Variable
  - Starts with an alphabet character
  - No spaces, no special characters

```
int       1Delta;            // variable names cannot start
                             // with numbers
byte      Delta1;
int       Serial Number;     // must contain no spaces
                             // SerialNumber
                             // Serial_Number
int       Serial,Number;
```

Arduino Programming

- Naming a Variable
  - Starts with an alphabet character
  - No spaces, no special characters

```
int       1Delta;            // variable names cannot start
                             // with numbers
byte      Delta1;
int       Serial Number;     // must contain no spaces
                             // SerialNumber
                             // Serial_Number
int       Serial,Number;     // These are 2 different variables
                             // having an integer type
```

Arduino Programming

- Assigning Values

```
int LEDPin;

// setup initializes the LED pin as output
// and initially turned off
void setup()
{
  LEDPin = 13;
  pinMode(LEDPin, OUTPUT);
  digitalWrite(LEDPin, LOW);
}

// loop checks the button pin each time,
void loop()
{
  delay(500);
  digitalWrite(LEDPin, HIGH);
  delay(500);
  digitalWrite(LEDPin, LOW);
}
```

Arduino Programming

- Assigning Values

```
int LEDPin;

// setup initializes the LED pin as output
// and initially turned off
void setup()
{
  LEDPin = 13;
  pinMode(LEDPin, OUTPUT);
  digitalWrite(LEDPin, LOW);
}

// loop checks the button pin each time,
void loop()
{
  delay(500);
  digitalWrite(LEDPin, HIGH);
  delay(500);
  digitalWrite(LEDPin, LOW);
}
```

Arduino Programming

- Assigning Values

```
int LEDPin = 13;

// setup initializes the LED pin as output
// and initially turned off
void setup()
{

   pinMode(LEDPin, OUTPUT);
   digitalWrite(LEDPin, LOW);
}

// loop checks the button pin each time,
void loop()
{
   delay(500);
   digitalWrite(LEDPin, HIGH);
   delay(500);
   digitalWrite(LEDPin, LOW);
}
```

Arduino Programming

- Decision Making
  - “if” statement
  - used to compare
    - variable to a variable

```
if(variable1 > variable2)
{
	function();
}
```

    - variable to a constant

```
if(variable1 > 2)
{
	function();
}
```

Arduino Programming

- Decision Making
  - execute the statement enclosed between { } if condition is satisfied

```
if(variable1 > variable2)
{
  function1();
}

function2();
```

Arduino Programming

- Decision Making
  - Comparison operator
    - Less than “<”
    - Greater than “>”
    - Equal “==”
    - Not equal “!=”
    - Less than or equal “<=”
    - Greater than or equal “>=”

Arduino Programming

- Decision Making
  - “if/else” statement
  - execute the statement enclosed between { } if condition is satisfied
  - execute the statement enclosed between { } after “else” when not

```
if(variable1 > variable2)
{
    function1();
}
else
{
    function2();
}
```

Arduino Programming

- Decision Making
  - Nested “if/else” statement

```
if(variable1 > variable2)
{
    function1();
}
else if(variable1 < variable2)
{
    function2();
}
```

Arduino Programming

- Decision Making
  - Nested “if/else” statement

```
if(variable1 > variable2)
{
    function1();
}
else if(variable1 < variable2)
{
    function2();
}
else
{
    function3();
}
```

Arduino Programming

- Blinking LED Example

```
int LEDPin = 13;

void setup()
{
  pinMode(LEDPin, OUTPUT);        // set pin 13 output
  digitalWrite(LEDPin, LOW);      // Set initially to off
}

void loop()
{
  delay(500);
  digitalWrite(LEDPin, HIGH);     // Turn LED on
  delay(500);
  digitalWrite(LEDPin, LOW);      // Turn LED off
}
```

Arduino Programming

- Blinking LED Example

```
int LEDPin = 13, SWITCHPin = 12, PinState;

void setup()
{
  pinMode(LEDPin, OUTPUT);        // set pin 13 output
  digitalWrite(LEDPin, LOW);      // Set initially to off
  pinMode(SWITCHPin, INPUT);      // set pin 12 input
}

void loop()
{
  delay(500);
  digitalWrite(LEDPin, HIGH);     // Turn LED on
  delay(500);
  digitalWrite(LEDPin, LOW);      // Turn LED off

  PinState = digitalRead(SWITCHPin);
}
```

Arduino Programming

- Better Blinking LED Example

```
int LEDPin = 13, SWITCHPin = 12, PinState, ledState;
unsigned long previousMillis;

void setup()
{
   pinMode(LEDPin, OUTPUT);          // set pin 13 output
   digitalWrite(LEDPin, LOW);        // Set initially to off
   pinMode(SWITCHPin, INPUT);        // set pin 12 input
}

void loop()
{
  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis >= 500)       // check if >= 500mS
  {
    previousMillis = currentMillis;               // initialize prev with current

    // if the LED is off turn it on and vice-versa:
    if (ledState == LOW)
    {
      ledState = HIGH;
    }
    else
    {
      ledState = LOW;
    }

    digitalWrite(LEDPin, ledState);               // Turn LED on/off
  }

  PinState = digitalRead(SWITCHPin);
}
```

Arduino Programming

- Better Blinking LED Example

```
int LEDPin = 13, SWITCHPin = 12, PinState, LEDState;
unsigned long previousMillis;

void setup()
{
   pinMode(LEDPin, OUTPUT);          // set pin 13 output
   digitalWrite(LEDPin, LOW);        // Set initially to off
   pinMode(SWITCHPin, INPUT);        // set pin 12 input
}

void loop()
{
  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis >= 500)       // check if >= 500mS
  {
    previousMillis = currentMillis;               // initialize prev with current

    // if the LED is off turn it on and vice-versa:
    if (ledState == LOW)
    {
      ledState = HIGH;
    }
    else
    {
      ledState = LOW;
    }

    digitalWrite(LEDPin, ledState);               // Turn LED on/off
  }

  PinState = digitalRead(SWITCHPin);
}
```

Arduino Programming

- Better Blinking LED Example

```
int LEDPin = 13, SWITCHPin = 12, PinState, LEDState;
unsigned long previousMillis;

void setup()
{
   pinMode(LEDPin, OUTPUT);          // set pin 13 output
   digitalWrite(LEDPin, LOW);        // Set initially to off
   pinMode(SWITCHPin, INPUT);        // set pin 12 input
}

void loop()
{
  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis >= 500)       // check if >= 500mS
  {
    previousMillis = currentMillis;               // initialize prev with current

    // if the LED is off turn it on and vice-versa:
    if (ledState == LOW)
    {
      ledState = HIGH;
    }
    else
    {
      ledState = LOW;
    }

     digitalWrite(LEDPin, ledState);              // Turn LED on/off
  }

  PinState = digitalRead(SWITCHPin);
}
```

Arduino Programming

- Better Blinking LED Example

```
int LEDPin = 13, SWITCHPin = 12, PinState, LEDState;
unsigned long previousMillis;

void setup()
{
   pinMode(LEDPin, OUTPUT);          // set pin 13 output
   digitalWrite(LEDPin, LOW);        // Set initially to off
   pinMode(SWITCHPin, INPUT);        // set pin 12 input
}

void loop()
{
  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis >= 500)       // check if >= 500mS
  {
    previousMillis = currentMillis;               // initialize prev with current

    // if the LED is off turn it on and vice-versa:
    if (ledState == LOW)
    {
      ledState = HIGH;
    }
    else
    {
      ledState = LOW;
    }

     digitalWrite(LEDPin, ledState);              // Turn LED on/off
  }

  PinState = digitalRead(SWITCHPin);
}
```

Arduino Programming

- Decision Making
  - Logical / boolean operator
    - Logic AND &&
    - Logic OR ||
    - Logic Not or Inversion !

Arduino Programming

- Better Blinking LED Example

```
int LEDPin = 13, SWITCHPin = 12, PinState, LEDState;
unsigned long previousMillis;

void setup()
{
   pinMode(LEDPin, OUTPUT);          // set pin 13 output
   digitalWrite(LEDPin, LOW);        // Set initially to off
   pinMode(SWITCHPin, INPUT);        // set pin 12 input
}

void loop()
{
  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis >= 500)       // check if >= 500mS
  {
    previousMillis = currentMillis;               // initialize prev with current

    // if the LED is off turn it on and vice-versa:
    if (ledState == LOW)
    {
      ledState = HIGH;
    }
    else
    {
      ledState = LOW;
    }

     digitalWrite(LEDPin, ledState);              // Turn LED on/off
  }

  PinState = digitalRead(SWITCHPin);
}
```

Arduino Programming

- Better Blinking LED Example

```
int LEDPin = 13, SWITCHPin = 12, PinState;
unsigned long previousMillis;
boolean LEDOn;

void setup()
{
   pinMode(LEDPin, OUTPUT);          // set pin 13 output
   digitalWrite(LEDPin, LOW);        // Set initially to off
   pinMode(SWITCHPin, INPUT);        // set pin 12 input
}

void loop()
{
  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis >= 500)       // check if >= 500mS
  {
    previousMillis = currentMillis;               // initialize prev with current

    // if the LED is off turn it on and vice-versa:
    if (LEDOn)
    {
      LEDOn = false;
    }
    else
    {
      LEDOn = true;
    }

    digitalWrite(LEDPin, LEDOn);                  // Turn LED on/off
  }

  PinState = digitalRead(SWITCHPin);
}
```

Arduino Programming

- Better Blinking LED Example

```
int LEDPin = 13, SWITCHPin = 12, PinState;
unsigned long previousMillis;
boolean LEDOn, BlinkEnabled = TRUE;

void setup()
{
   pinMode(LEDPin, OUTPUT);          // set pin 13 output
   digitalWrite(LEDPin, LOW);        // Set initially to off
   pinMode(SWITCHPin, INPUT);        // set pin 12 input
}

void loop()
{
  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis >= 500)       // check if >= 500mS
  {
    previousMillis = currentMillis;               // initialize prev with current

    // if the LED is off turn it on and vice-versa:
    if (LEDOn && BlinkEnabled)
    {
      LEDOn = false;
    }
    else
    {
      LEDOn = true;
    }

    digitalWrite(LEDPin, LEDOn);                  // Turn LED on/off
  }

  PinState = digitalRead(SWITCHPin);
}
```

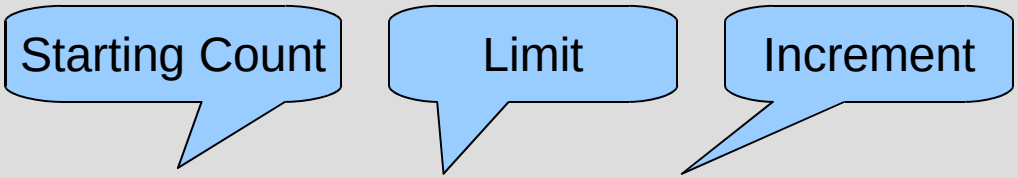
Arduino Programming

- Better Blinking LED Example

```
int LEDPin = 13, SWITCHPin = 12;
unsigned long previousMillis;
boolean LEDOn, BlinkEnabled;

void setup()
{
   pinMode(LEDPin, OUTPUT);          // set pin 13 output
   digitalWrite(LEDPin, LOW);        // Set initially to off
   pinMode(SWITCHPin, INPUT);        // set pin 12 input
}

void loop()
{
  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis >= 500)       // check if >= 500mS
  {
    previousMillis = currentMillis;               // initialize prev with current

    // if the LED is off turn it on and vice-versa:
    if (LEDOn && BlinkEnabled)
    {
      LEDOn = false;
    }
    else
    {
      LEDOn = true;
    }

    digitalWrite(LEDPin, LEDOn);                  // Turn LED on/off
  }

  BlinkEnabled = !digitalRead(SWITCHPin);
}
```

Arduino Programming

- Looping
  - “for” loop
  - execute expressions enclosed between {}
  - continuously as long as a condition is satisfied




```
for(i = 1; i <= 10; i++)
{
  function();
}
```

Arduino Programming

- Libraries

```
#include <Servo.h>

Servo myservo;    // create servo object to control a servo
                  // a maximum of eight servo objects can be created

int pos = 0;      // variable to store the servo position

void setup()
{
  myservo.attach(9);   // attaches the servo on pin 9 to the servo object
}

void loop()
{
  for(pos = 0; pos < 180; pos += 1)  // goes from 0 degrees to 180 degrees
  {                                  // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable
   'pos'
    delay(15);                       // waits 15ms for the servo to reach the
   position
  }
  for(pos = 180; pos>=1; pos-=1)     // goes from 180 degrees to 0 degrees
  {
    myservo.write(pos);              // tell servo to go to position in variable
   'pos'
    delay(15);                       // waits 15ms for the servo to reach the
   position
```

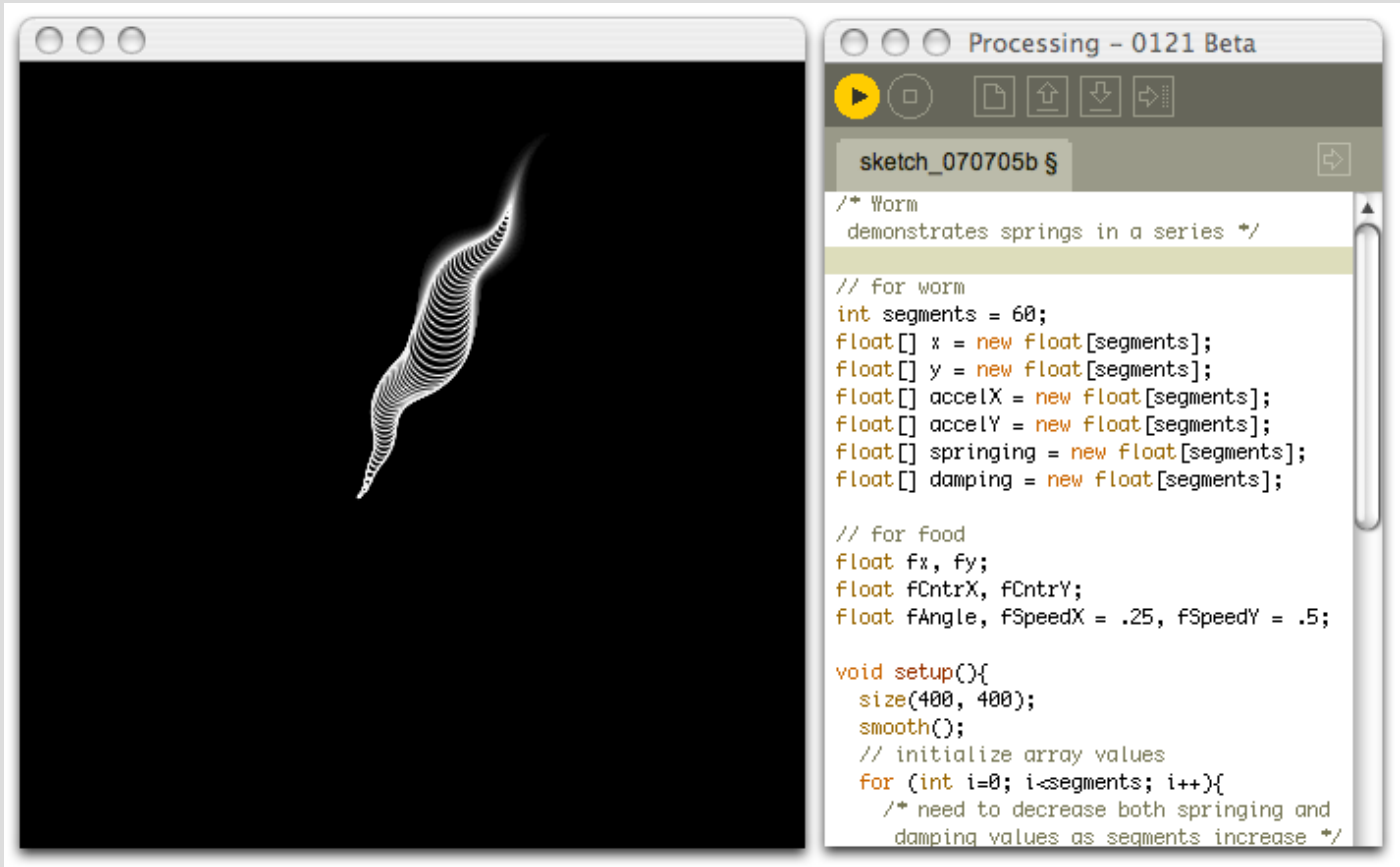
Reference

- www.arduin.cc
- Arduino Programming Notebook
- Getting Started with Arduino 3rd Edition

Beyond Arduino

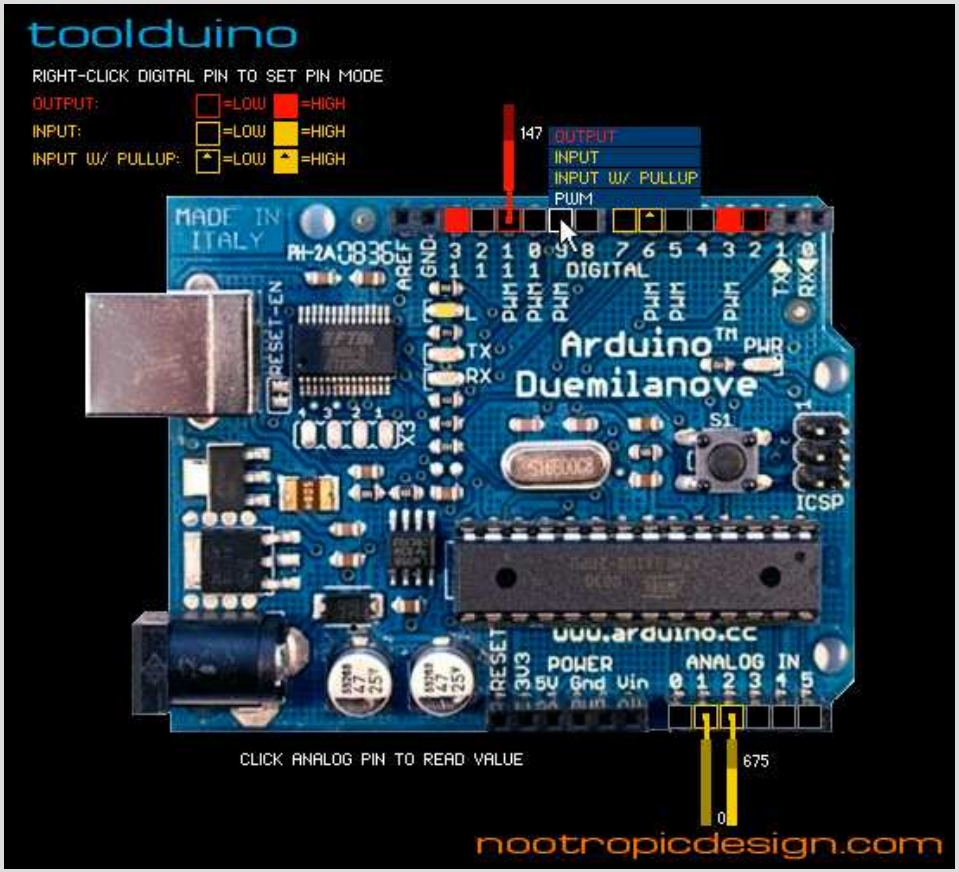
Beyond Arduino

- Processing



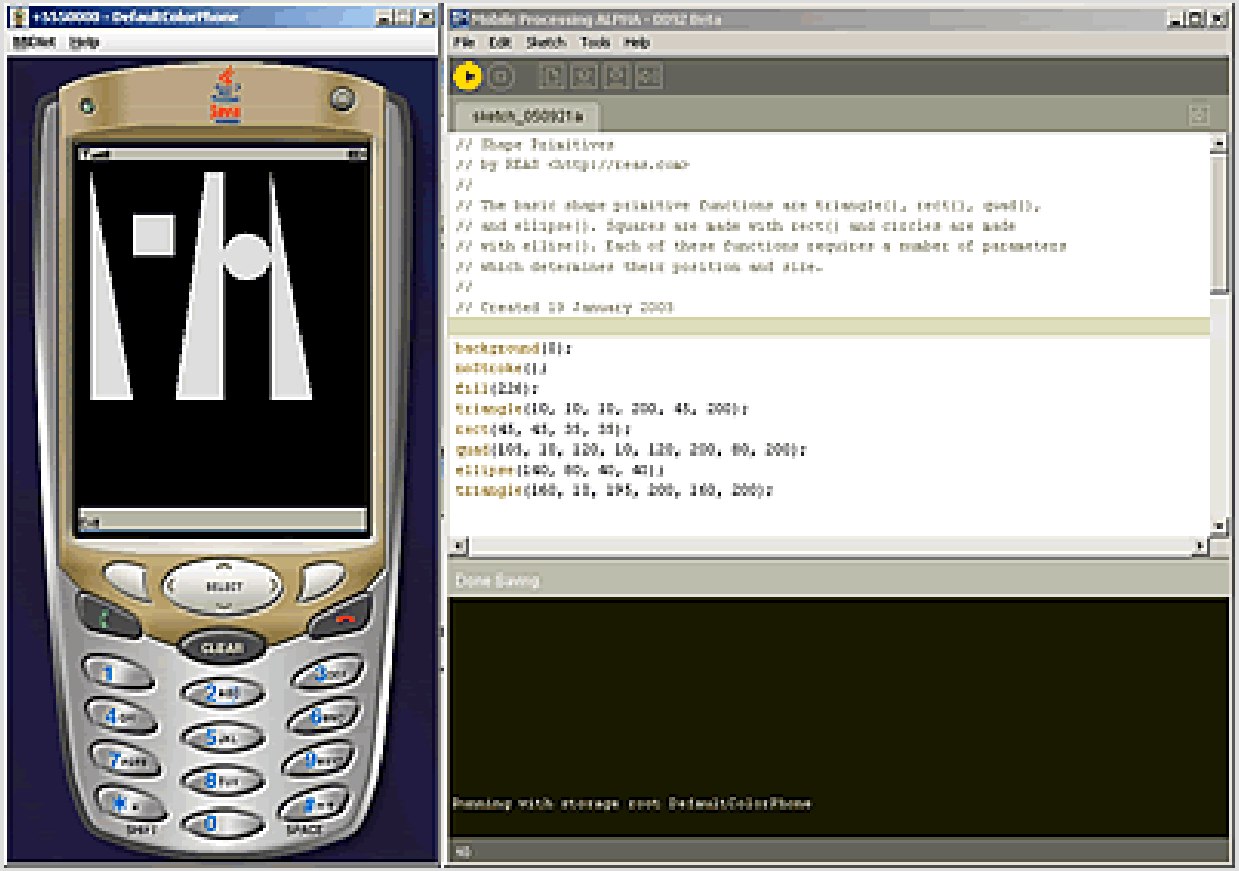
Beyond Arduino

- Toolduino



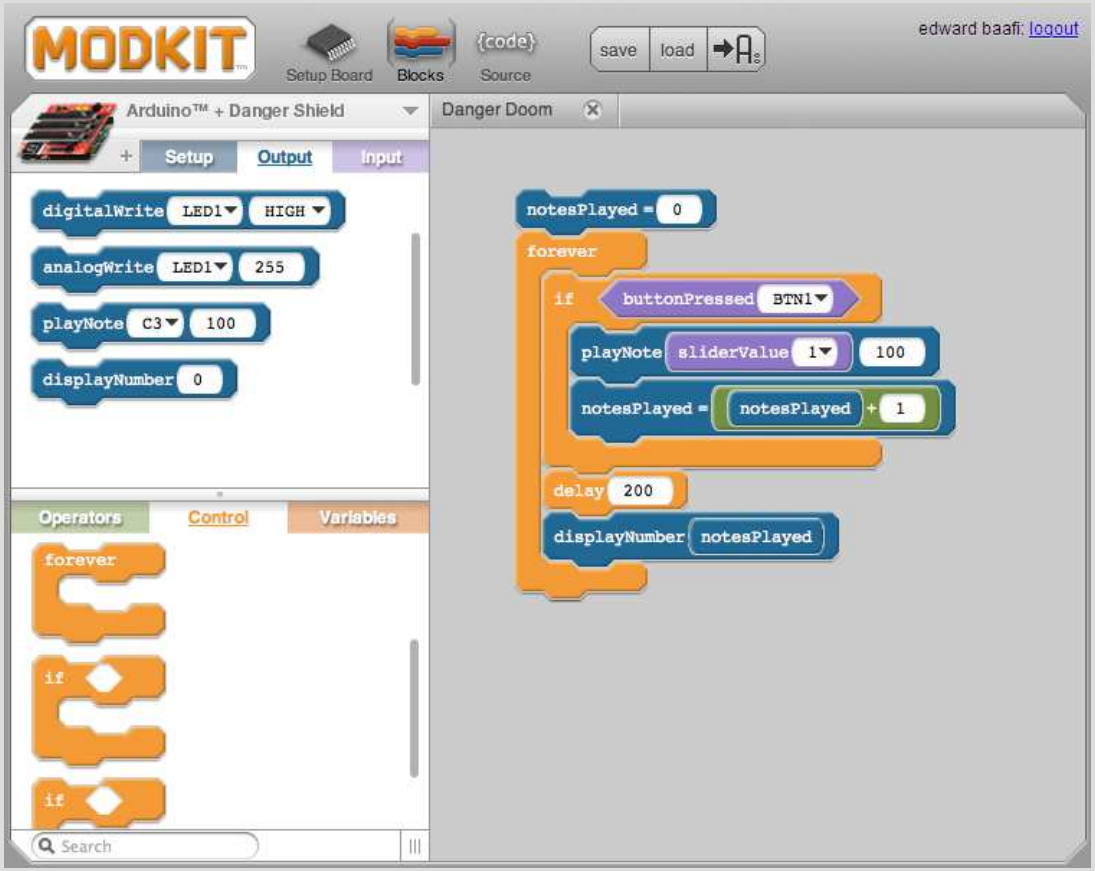
Beyond Arduino

- Mobile Processing



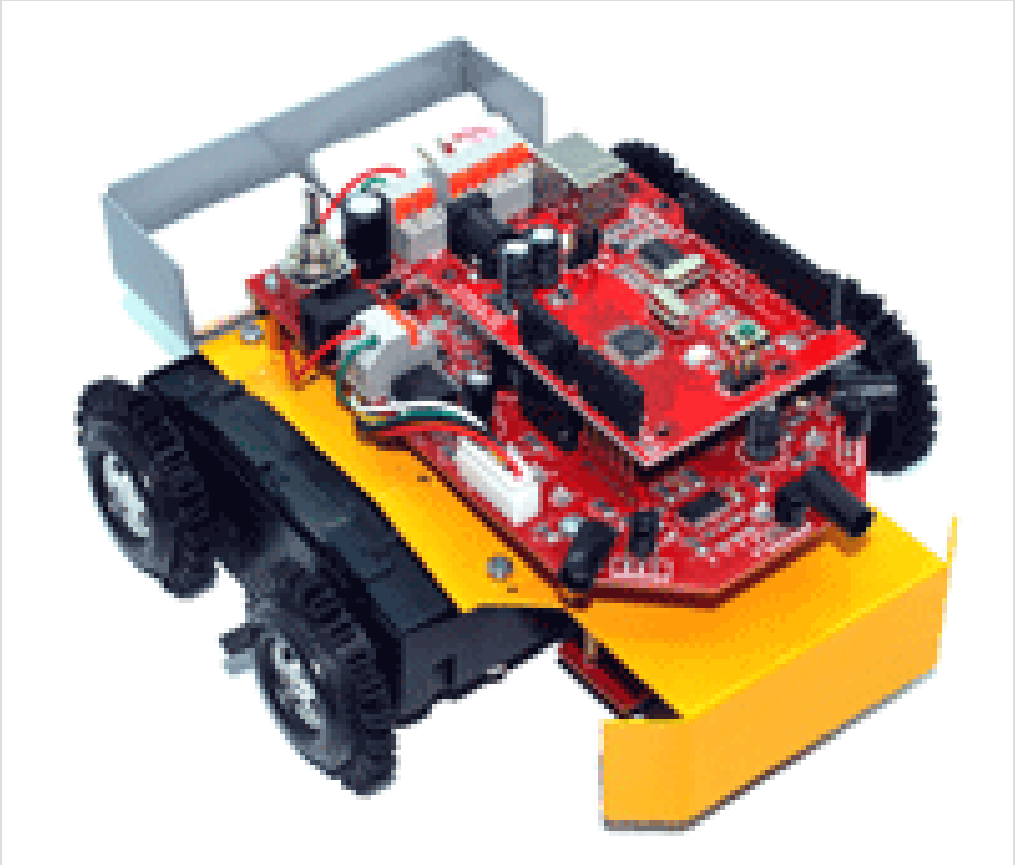
Beyond Arduino

- S4A (Scratch for Arduino)



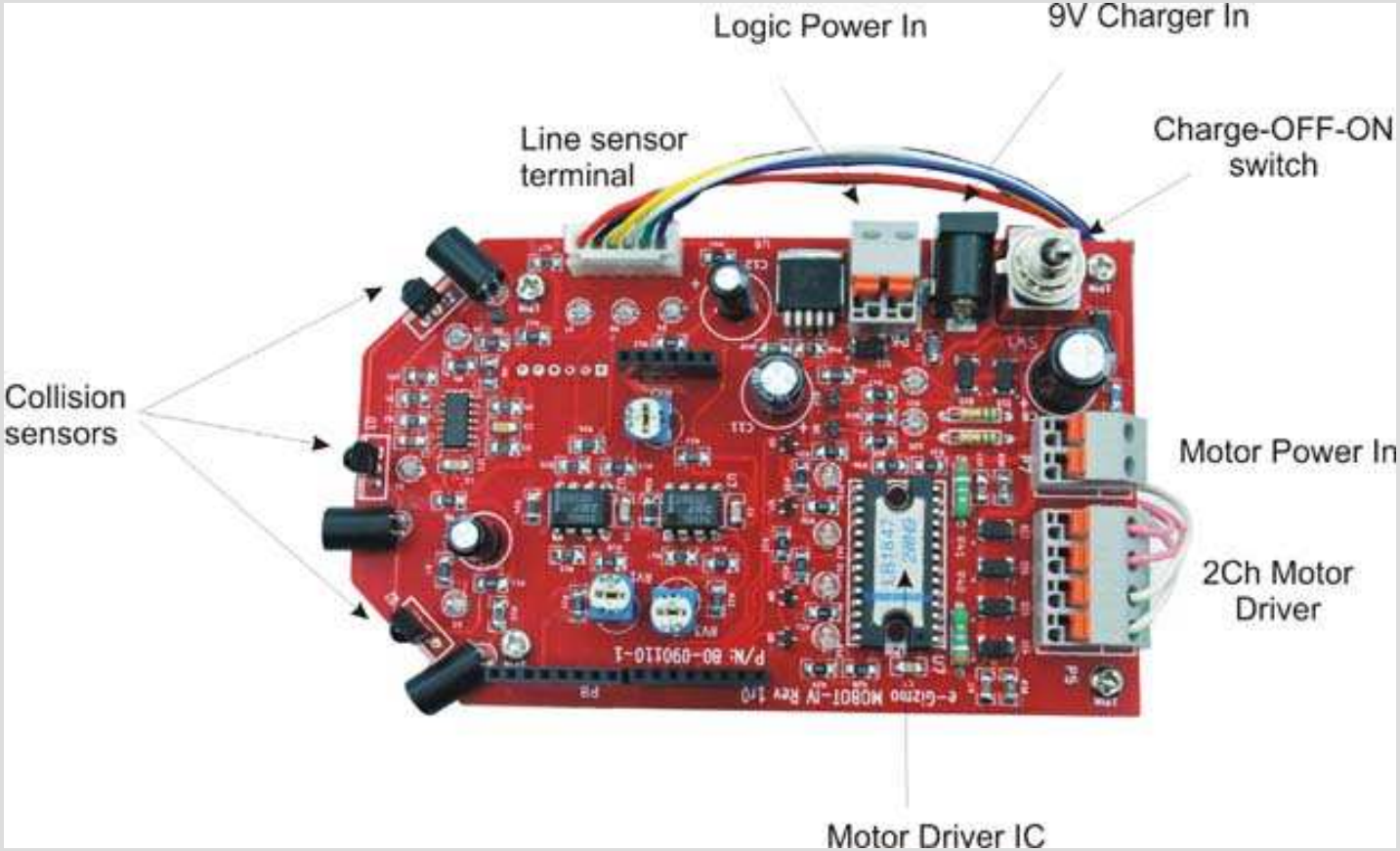
Beyond Arduino

- P-Bot



Beyond Arduino

- P-Bot



Questions, Comments, Suggestions

- glutnix.neo on gmail
- glutnix_neo → electronicslab.ph/forum

Arduino

End of Presentation
01/22/11