NUS National University of Singapore

ARDUINO Workshop

Acknowledgements:

- Dr Rajesh Panicker
- www.Arduino.cc
- www.sparkfun.com

(Some slides from Arduino introduction slides by Linz Craig, Nick Poole, Prashanta Aryal, Theo Simpson, Tai Johnson, and Eli Santistevan)

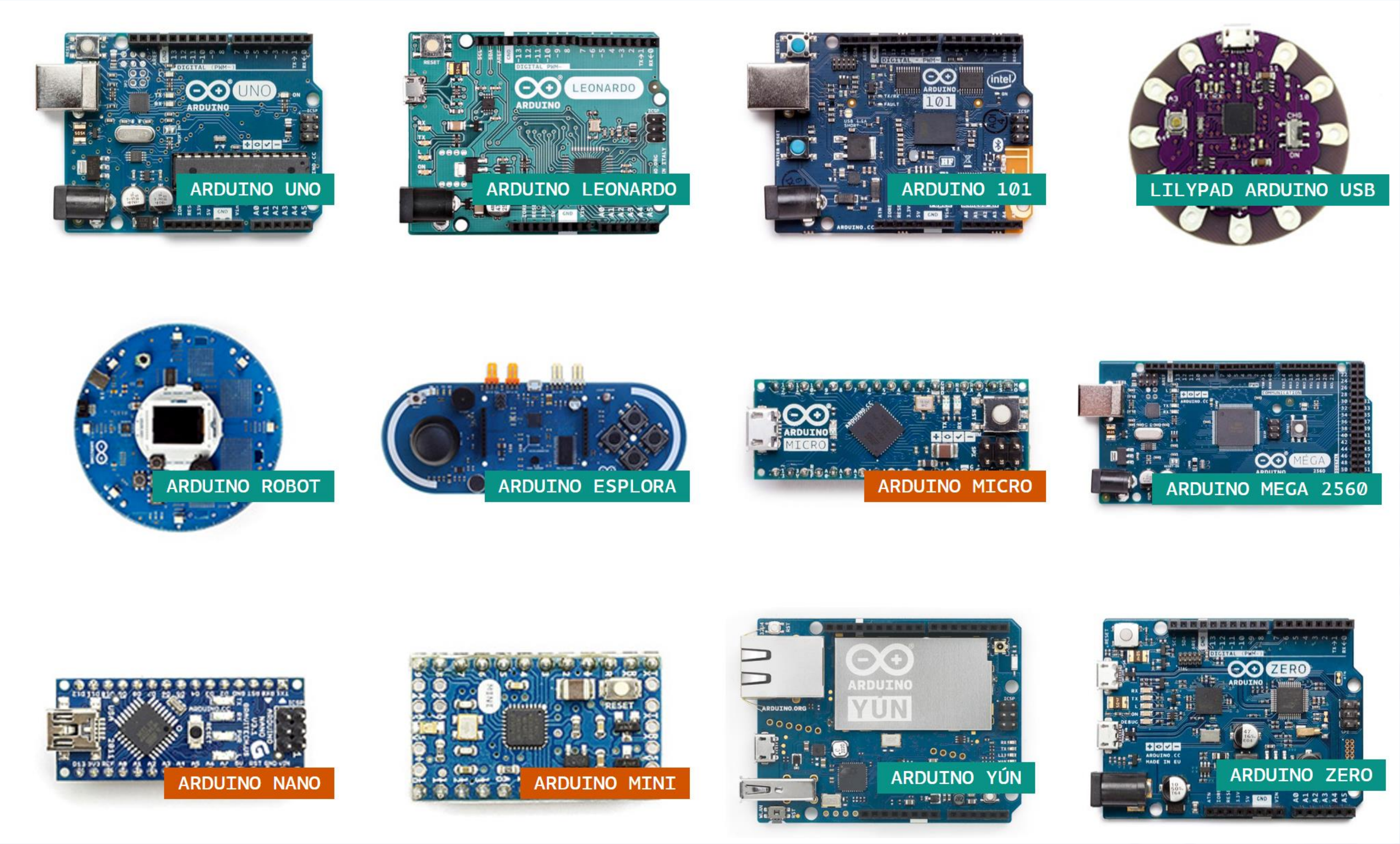
What is Arduino?

- Arduino is an open-source electronics platform based on easy-to-use hardware and software
- Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online
- You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (C++), and the Arduino Software (IDE)
- A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike

Why Arduino?

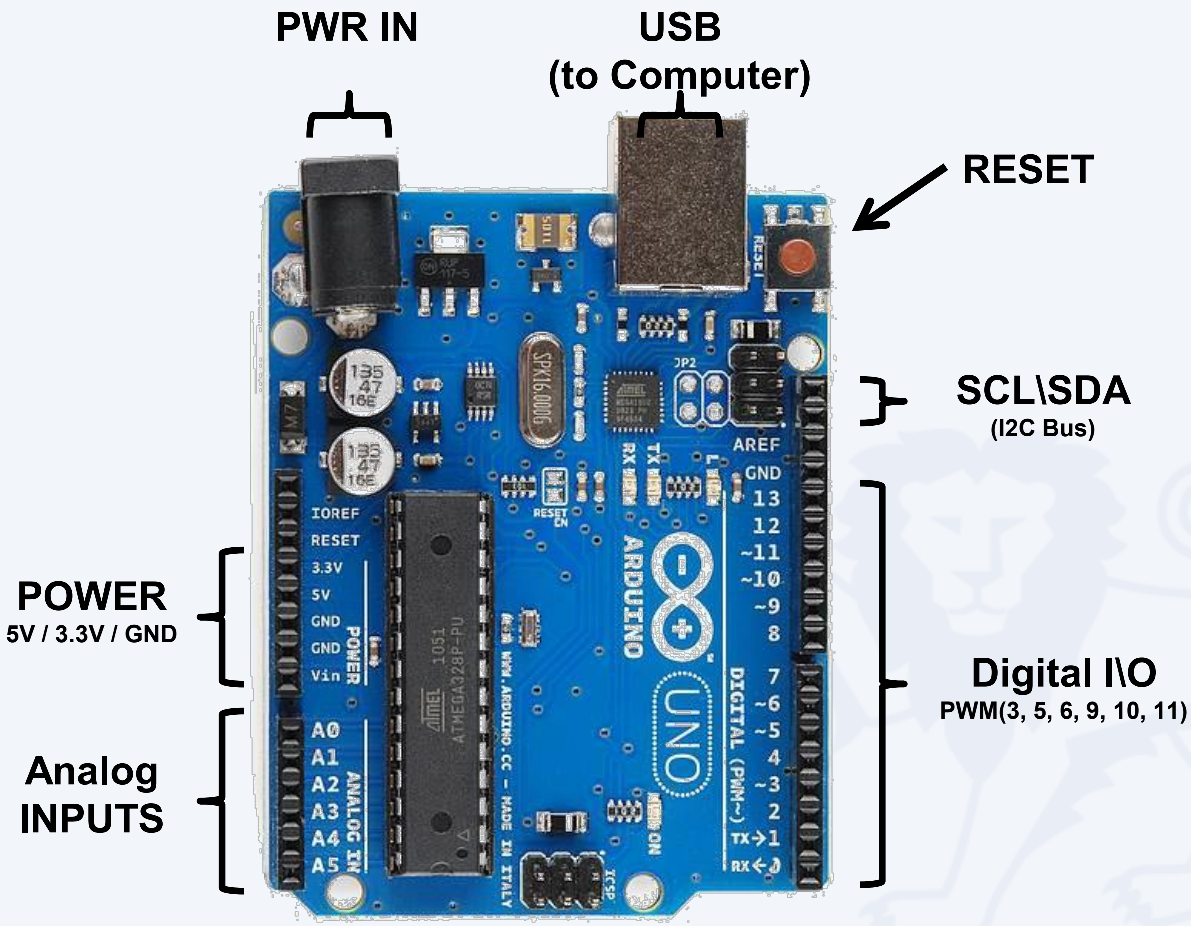
- Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments
- Inexpensive
- Cross-platform (IDE works on Windows, Mac and Linux, Raspberry Pi)
- Simple, clear programming environment
- Open-source hardware empowering users to build them independently and eventually adapt them to their particular needs
- Software growing through the contributions of users worldwide

Arduino Boards (the "Brain")



https://www.arduino.cc/en/Main/Products

Arduino Uno (most popular)



PWR IN

USB
(to Computer)

RESET

SCL\SDA
(I2C Bus)

POWER
5V / 3.3V / GND

Analog
INPUTS

Digital I\O
PWM(3, 5, 6, 9, 10, 11)

Input vs. Output

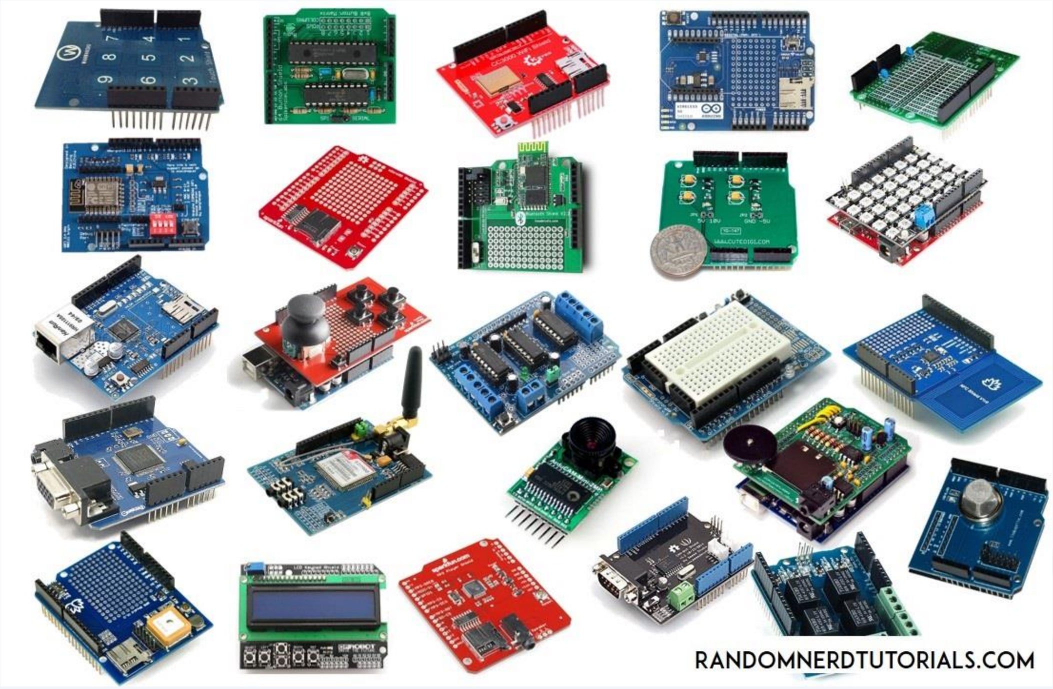
Referenced from the perspective of the Arduino Board

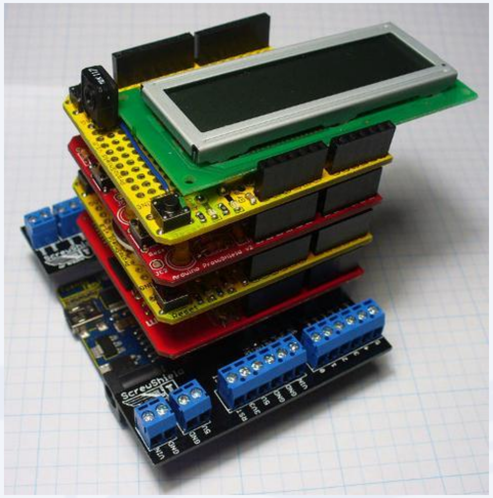
Inputs is a signal / information going into the board

Output is any signal exiting the board

- **Almost all systems that use physical computing will have some form of output**
- **A device which can provide input(s) is called an input device, usually referred to as sensors. Ex: Light sensors (LDRs), Accelerometers, Push buttons**
- **Output devices are usually referred to as actuators Ex: Motors, LEDs**

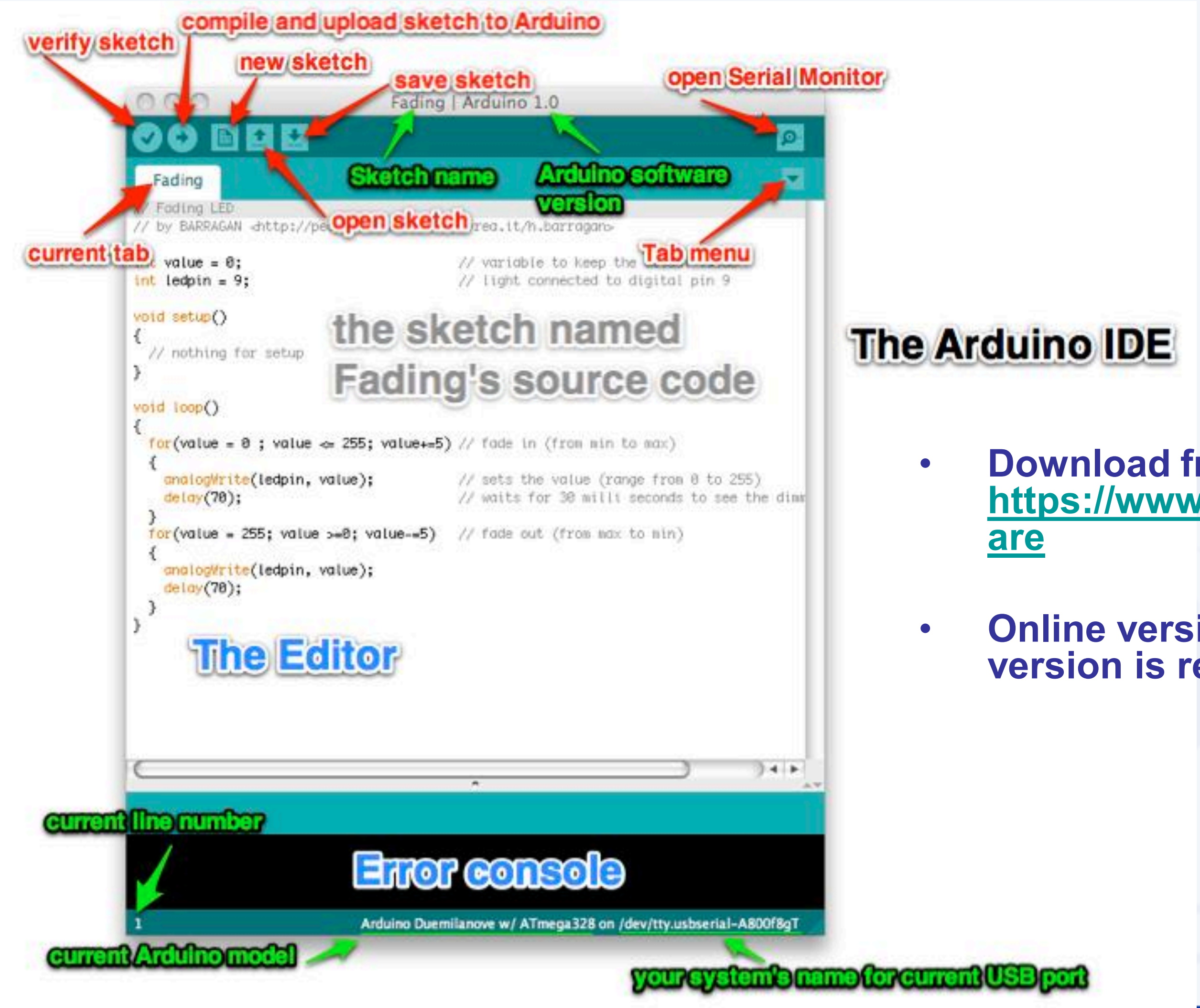
Shields (the "Body Parts")


RANDOMNERDTUTORIALS.COM


Stacked shields

- Shields provide an easy way to interface sensor and actuators with the Arduino – avoids having to wire them up manually
- Shields can be stacked (terms and conditions apply!)
- You can design your own shields

Arduino IDE



Fading | Arduino 1.0

Fading

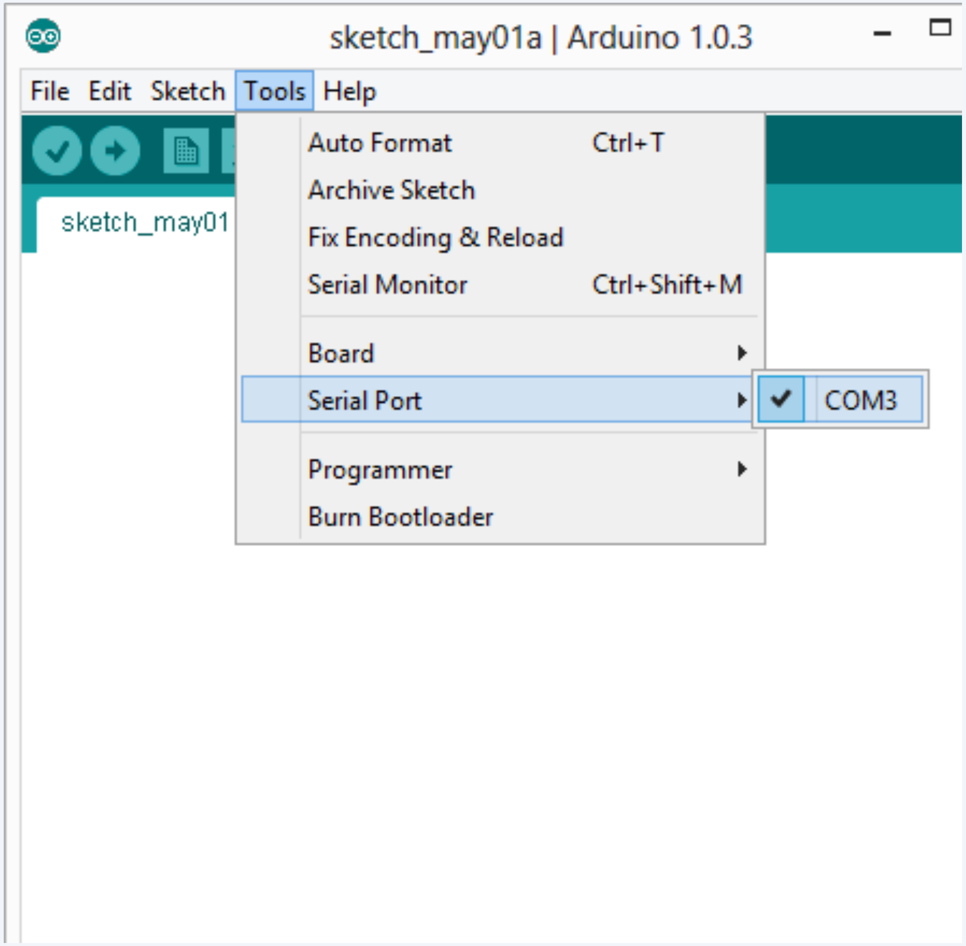
```
/ Fading LED
// by BARRAGAN <http://pe                rea.it/h.barragan>

  value = 0;                              // variable to keep the
int ledpin = 9;                           // light connected to digital pin 9

void setup()
{
  // nothing for setup
}

void loop()
{
  for(value = 0 ; value <= 255; value+=5) // fade in (from min to max)
  {
    analogWrite(ledpin, value);           // sets the value (range from 0 to 255)
    delay(70);                            // waits for 30 milli seconds to see the dimm
  }
  for(value = 255; value >=0; value-=5)   // fade out (from max to min)
  {
    analogWrite(ledpin, value);
    delay(70);
  }
}
```

1 Arduino Duemilanove w/ ATmega328 on /dev/tty.usbserial-A800f8gT

- Download from https://www.arduino.cc/en/Main/Software
- Online version available (but local version is recommended)

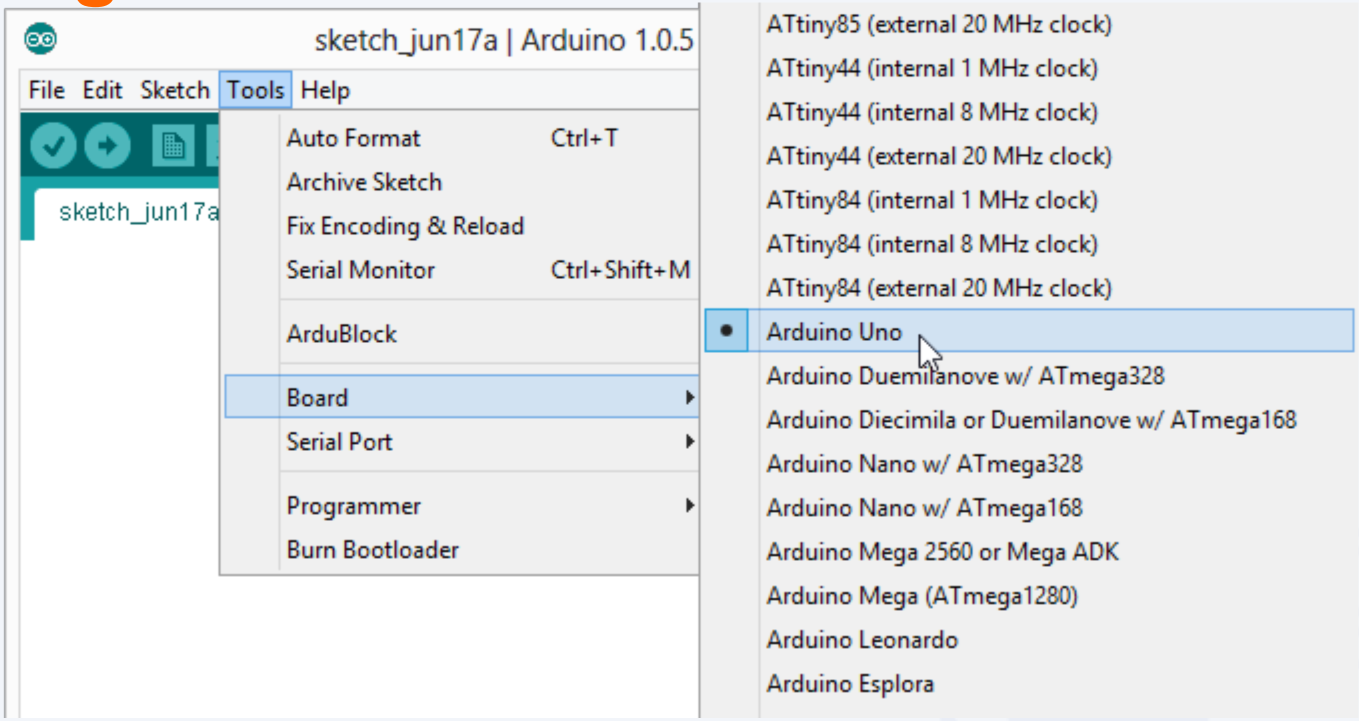
Image Courtesy : http://www.hacdc.org/summer-school-2013/

Settings: Tools → Serial Port



- Your computer communicates to the Arduino via a serial port → through a USB-Serial adapter
- Check to make sure that the drivers are properly installed
- The Serial Port wouldn't be 'COM1'.

Settings: Tools → Board



sketch_jun17a | Arduino 1.0.5

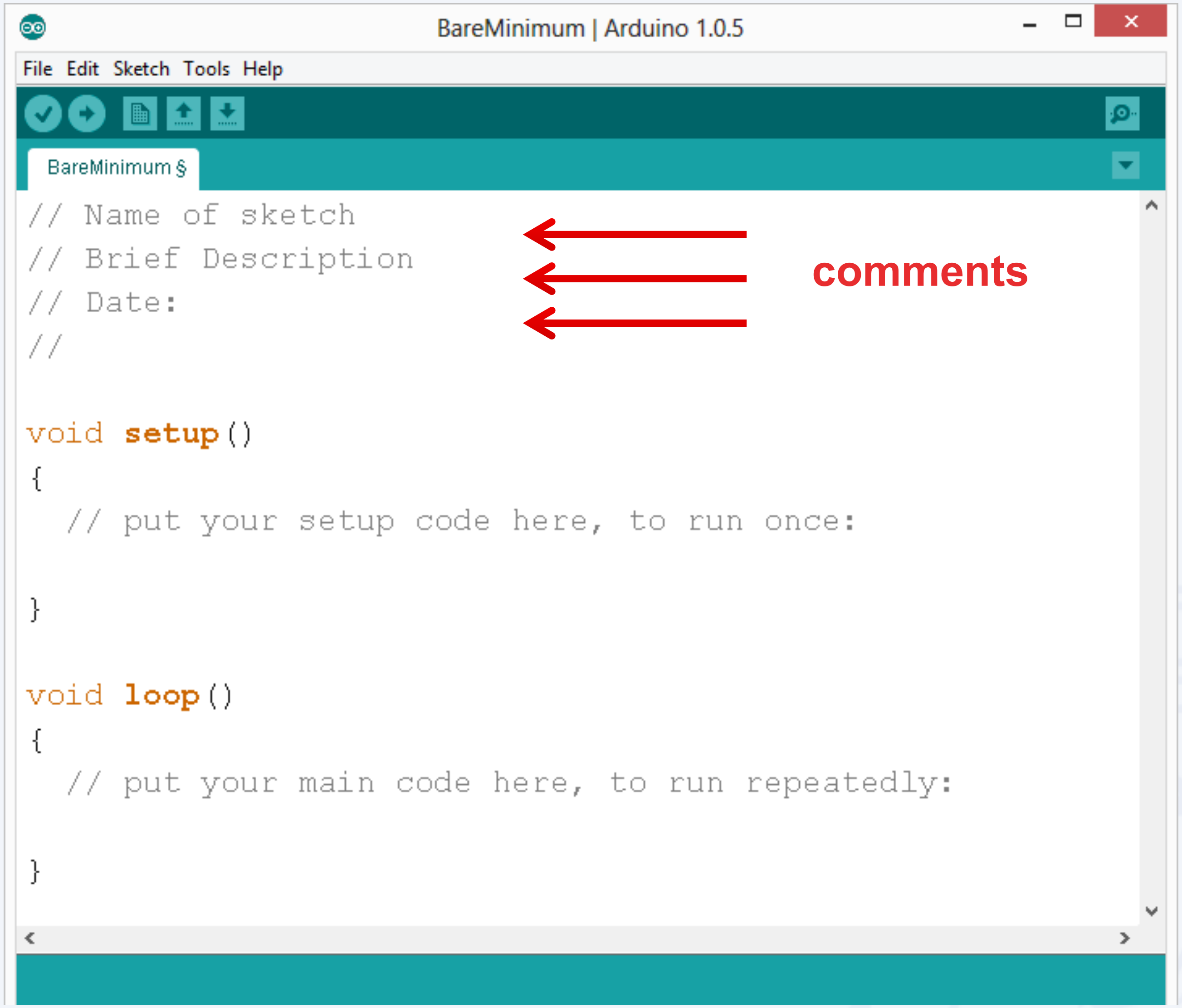
File
Edit
Sketch
Tools
Help
sketch_jun17a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
ArduBlock
Board
Serial Port
Programmer
Burn Bootloader
ATtiny85 (external 20 MHz clock)
ATtiny44 (internal 1 MHz clock)
ATtiny44 (internal 8 MHz clock)
ATtiny44 (external 20 MHz clock)
ATtiny84 (internal 1 MHz clock)
ATtiny84 (internal 8 MHz clock)
ATtiny84 (external 20 MHz clock)
● Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora

- **Next, double-check that the proper board is selected under the Tools→Board menu**

Comments

- Comments are for you – the programmer and your friends…or anyone else human/ Artificial Intelligence that might read your code
- Comments are *not run* on the Arduino board

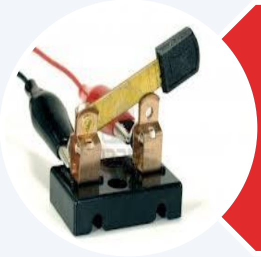
```
// this is for single line comments
// it’s good to put a description at the
// top and before anything ‘tricky’
/* this is for multi-line comments
   Like this…
   And this….
*/
```



```
// Name of sketch
// Brief Description
// Date:
//

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

}

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

}
```

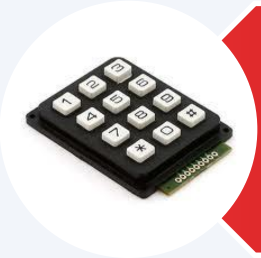
comments

BIG 6 CONCEPTS

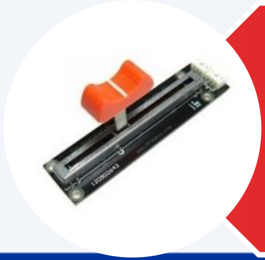


digitalWrite()

analogWrite()



digitalRead()



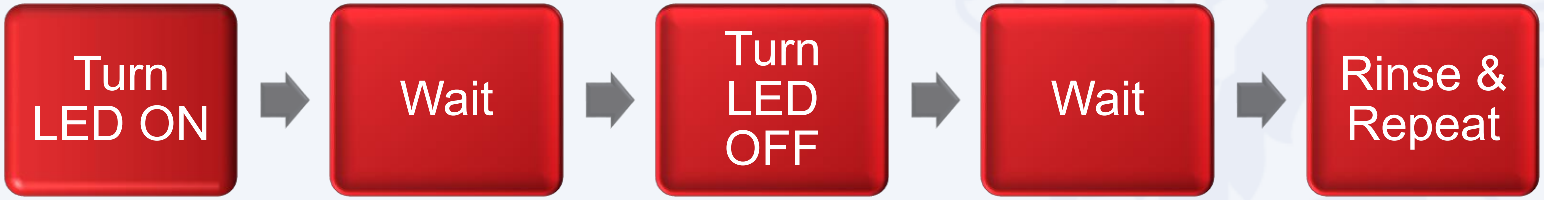
analogRead()

Let’s Get Started..

Blinky

“Hello World” of Physical Computing

how do we implement this?



Turn LED ON → Wait → Turn LED OFF → Wait → Rinse & Repeat

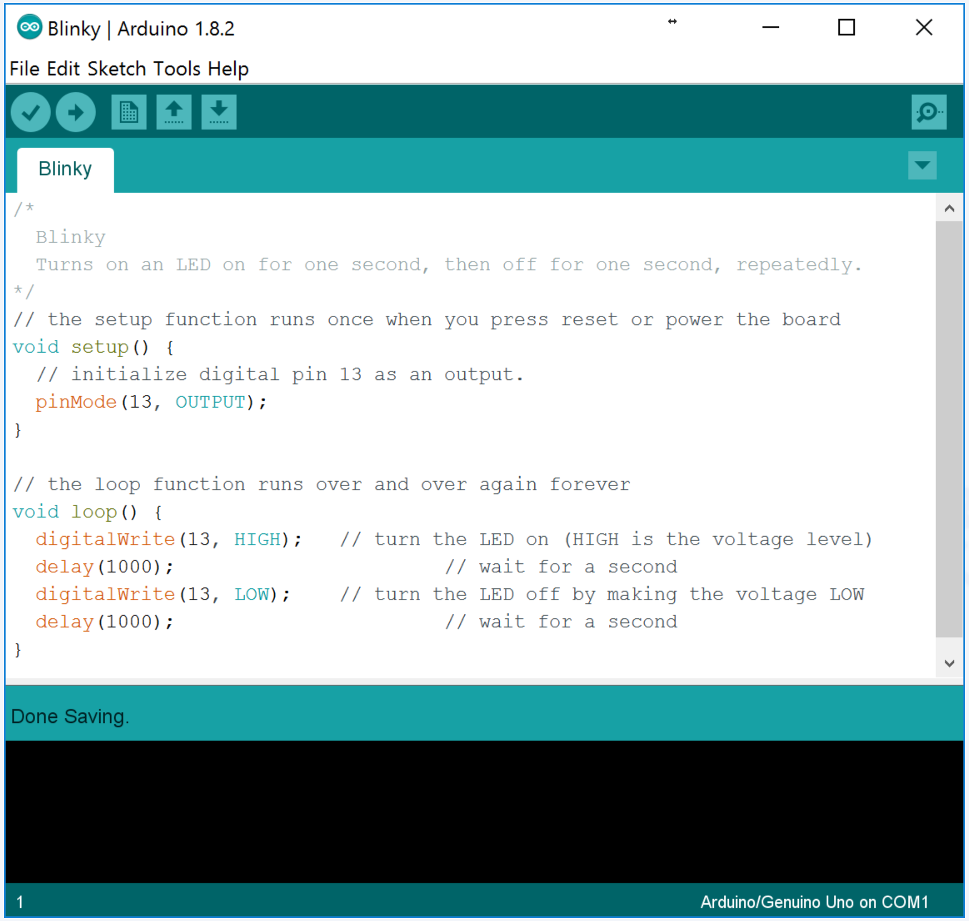
Digital Output

Three commands to know…

```
pinMode(pin, INPUT/OUTPUT);
ex: pinMode(13, OUTPUT);

digitalWrite(pin, HIGH/LOW);
ex: digitalWrite(13, HIGH);

delay(time_ms);
ex: delay(2500); // delay of 2.5 sec.

// NOTE: -> commands are CASE-sensitive
```

https://www.arduino.cc/en/Reference

Blinky



```
/*
  Blinky
  Turns on an LED on for one second, then off for one second, repeatedly.
*/
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Done Saving.

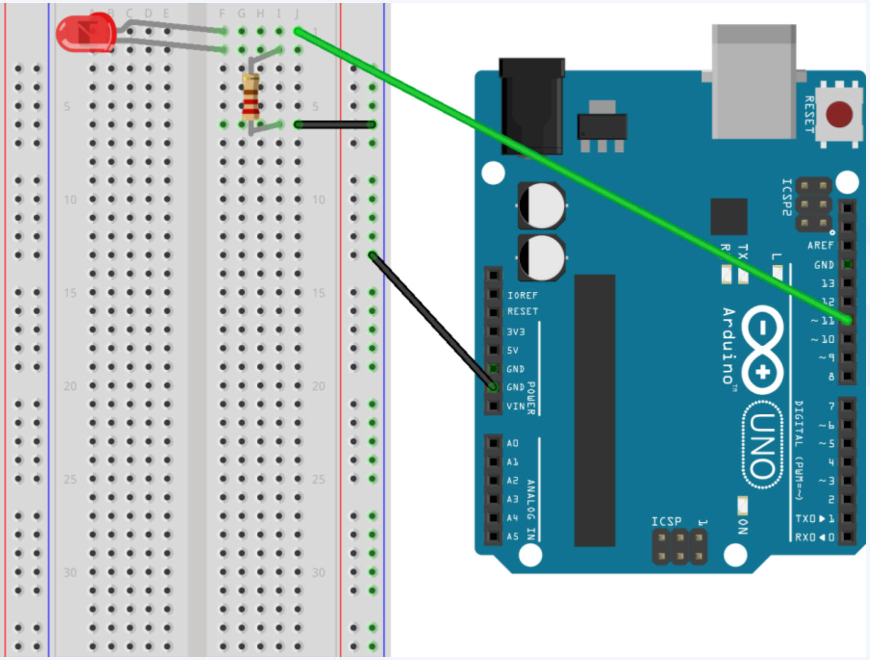
Type this code, click "Upload" and observe the LED close to pin 13

You have just completed your first Arduino program!

Blinking the LED

Move the green wire from the power pin to pin 13 on the Arduino board without changing the program

Try changing the connection from pin 13 to pin 11 (as shown in the image). How should your program be modified?



Defining pins as variables!!

- **In the previous example you changed the LED output from pin 13 to pin 11**
  - You had to change it in every instance of pin 13 appearing in your code
- **It is better to define the pins as variables at the start of the program so that you need to change the value only once**
  - `int ledPin = 11; // select the pin for the LED`
- **Once we define it at the top of the program we need to modify the variable to select a different pin**

Let’s go further!

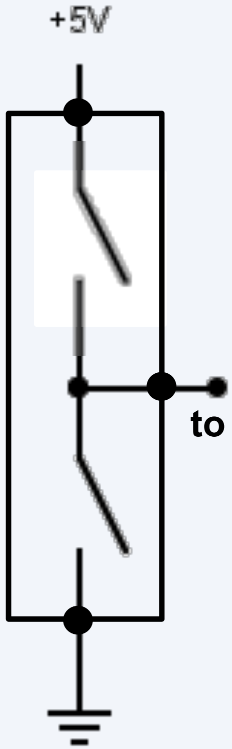
- How will the program need to be modified for the following cases?
  - Make LED blink faster
  - Make LED blink slower
  - Turn ON LED for twice the duration of OFF time
  - Any cool “codes” you can produce using an LED?

Digital Input

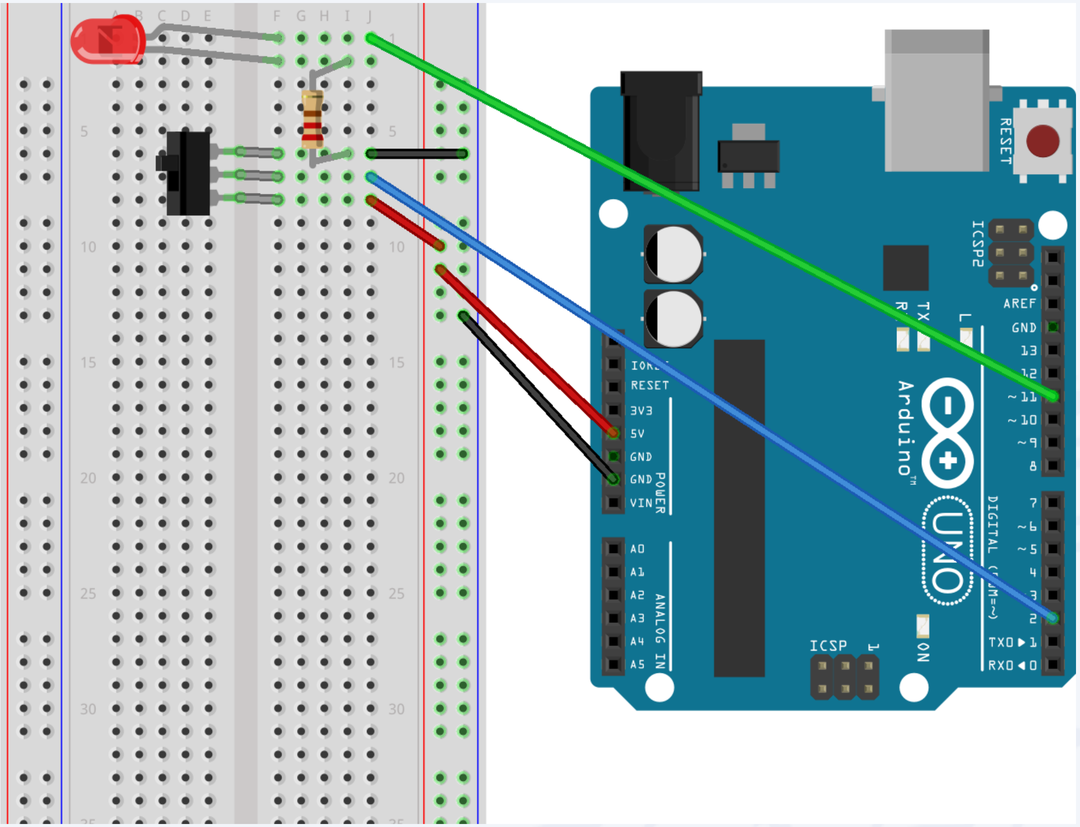
Digital Sensors

- Digital sensors are (more) straight forward (than Analog)
- No matter what the sensor there are only two settings: On and Off
- Signal is always either HIGH (On) or LOW (Off)
- Voltage signal for HIGH will be 5V (more or less) on Arduino Uno. Other Arduinos could use different voltages!
- Voltage signal for LOW will be 0V on most systems

Digital Input – Switch



to Digital Pin 2



Digital Input

- Connect digital input to your Arduino using Pins # 0 – 13 (Avoid pins # 0 & 1 though as they are used for *Serial* later, and pin #11 and 13 as we are already using it)

- Digital Input needs a `pinMode` command:

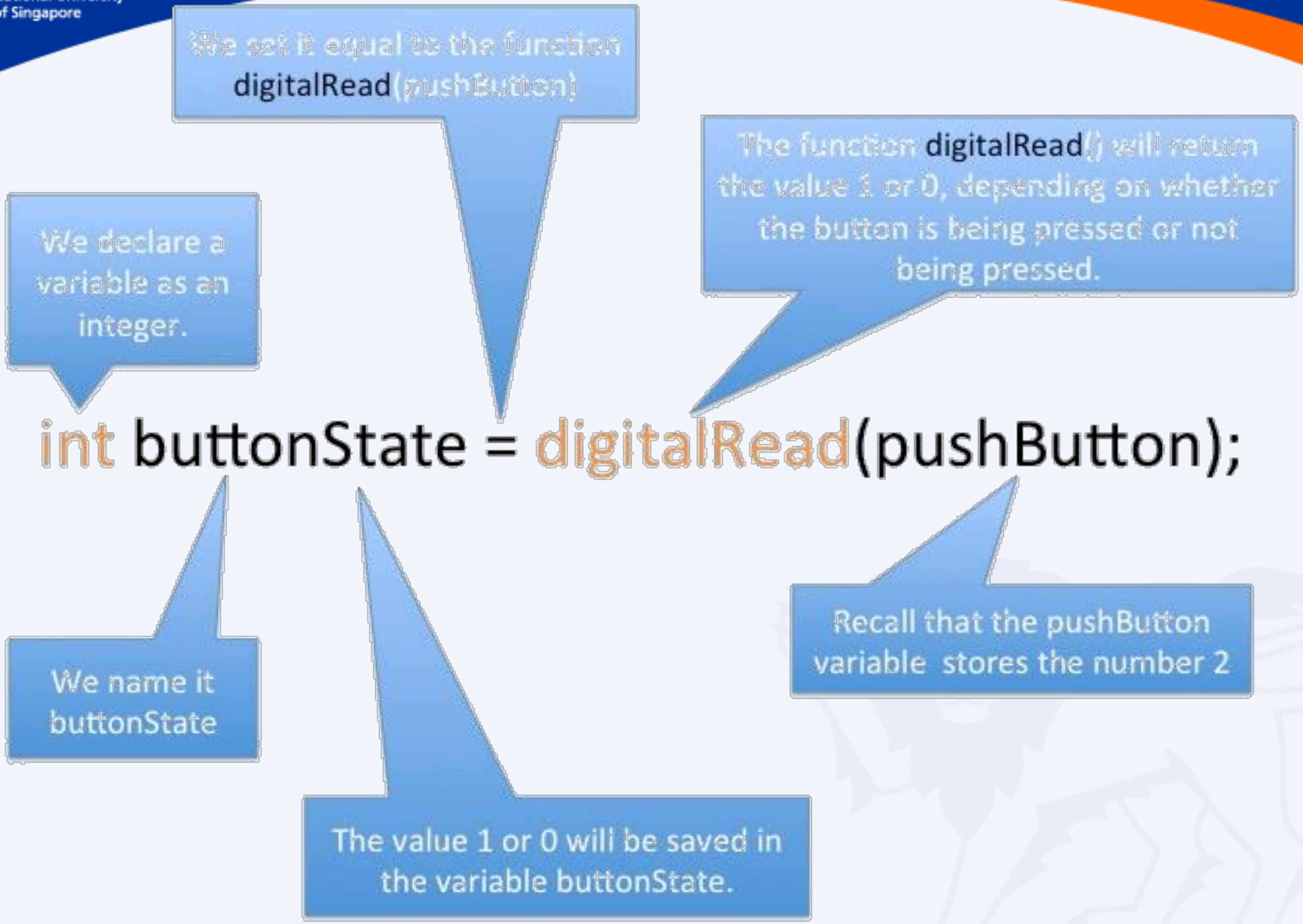
```
    pinMode (pinNumber, INPUT);
```

Make sure to use ALL CAPS for INPUT

- To get a digital reading:

```
int buttonState = digitalRead (pinNumber);
```

- Digital Input values are only HIGH (On) or LOW (Off)



We set it equal to the function digitalRead(pushButton)

We declare a variable as an integer.

The function digitalRead() will return the value 1 or 0, depending on whether the button is being pressed or not being pressed.

```
int buttonState = digitalRead(pushButton);
```

We name it buttonState

The value 1 or 0 will be saved in the variable buttonState.

Recall that the pushButton variable stores the number 2

http://opensourcehardwarejunkies.com/tutorial-03-digitalread-and-serial-port-communication/

Programming: Conditional Statements if()

```
void loop()
{
        int buttonState = digitalRead(2);
        if(buttonState == HIGH)
        {     // do something
        }
        else
        {  // do something else
        }
}
```

Exercise 1

Modify your blinky program such that it blinks only when the switch is turned ON

Hint :

In `setup()`, `pinMode(2, INPUT);` should be inserted

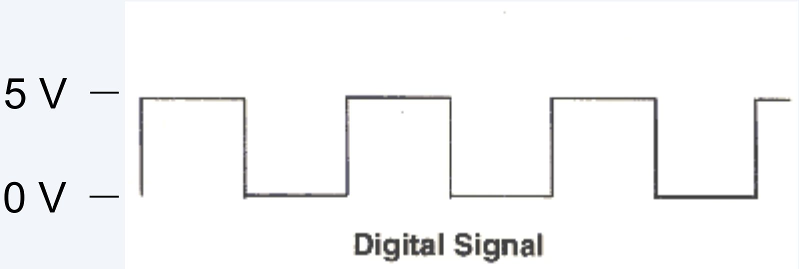
Place the code for blinking the LED below `// do something` in the previous slide

Analog Output

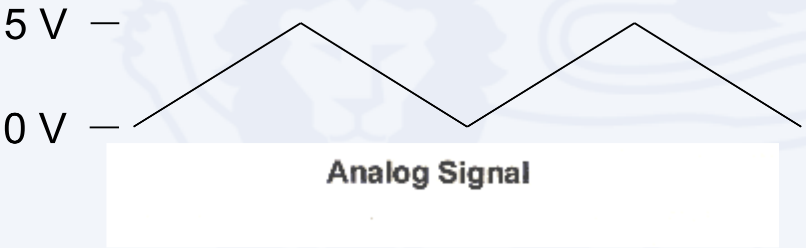
Analog vs. Digital

- Arduinos are digital devices – ON or OFF. Also called discrete

- Analog signals are anything that can be a full range of values. Examples?



Digital Signal



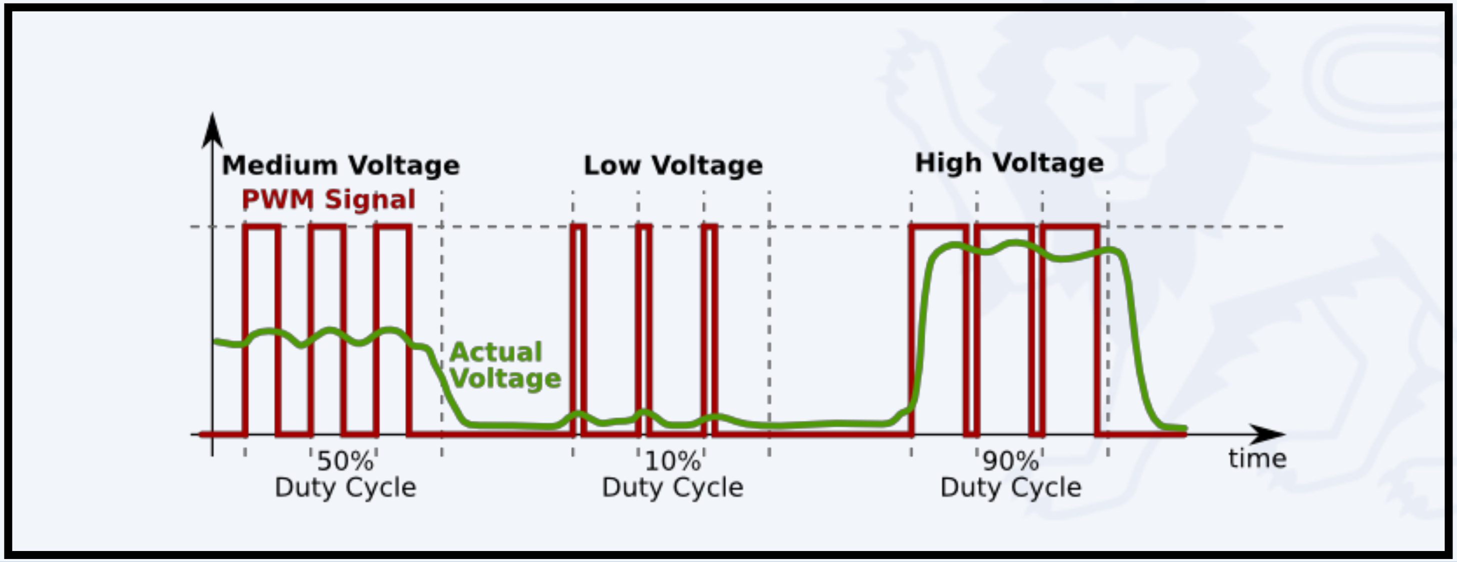
Analog Signal

- How do we generate the effect of *analog* using *digital*?

Analog vs. Digital

- To create (mimic) an analog signal, the Arduino uses a technique called Pulse Width Modulation (PWM). By varying the duty cycle, we can mimic an "average" analog voltage

Pulse Width Modulation (PWM)



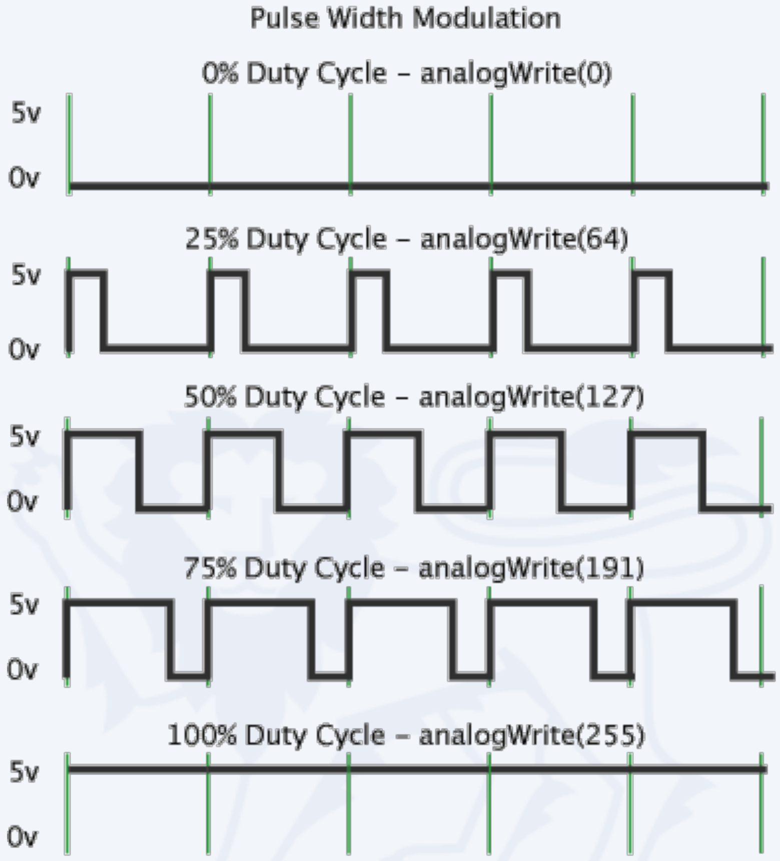
analogWrite()

```
analogWrite(pin, val);
```

`pin` – refers to the OUTPUT pin (limited to pins 3, 5, 6, 9, 10, 11.) – denoted by a ~ symbol

`val` – 8 bit value (0 – 255).

0 => 0V | 255 => 5V


Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
5v
0v
25% Duty Cycle – analogWrite(64)
5v
0v
50% Duty Cycle – analogWrite(127)
5v
0v
75% Duty Cycle – analogWrite(191)
5v
0v
100% Duty Cycle – analogWrite(255)
5v
0v

Exercise 2

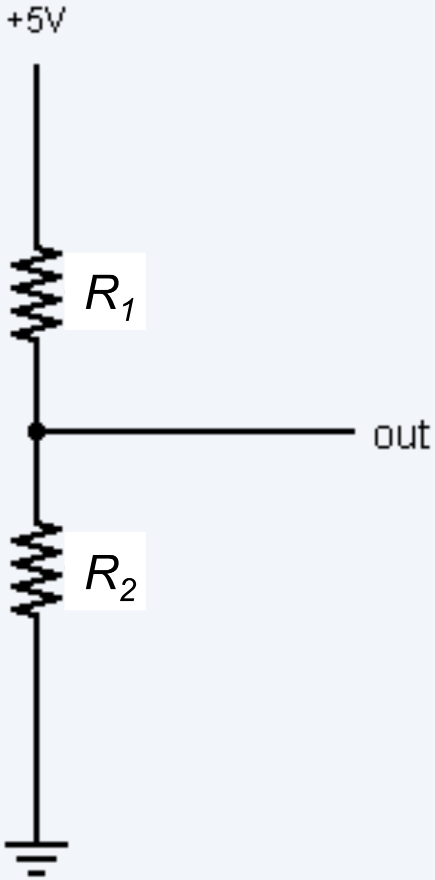
- **Create a program such that the LED brightness gradually increases from 0 to 255, and then goes abruptly to 0**

Hints :

- Use pin 11. If you are already having the LED connected to pin 11, you need not change any connection. Why can't you use pin 13?
- You will have to use a **for** loop. Lookup **for** in https://www.arduino.cc/en/Reference
- The delay should be around 5-10 milliseconds
- Can you modify your program to decrease the brightness gradually from 255 to 0 instead of an abrupt change?

Analog Input

Voltage Divider



$$V_{R1} = V_{CC} \cdot \left(\frac{R_1}{\mathrm{R}_{Total}}\right)$$

$$V_{R2} = V_{CC} \cdot \left(\frac{R_2}{\mathrm{R}_{Total}}\right)$$

$$R_{Total} = R_1 + R_2$$

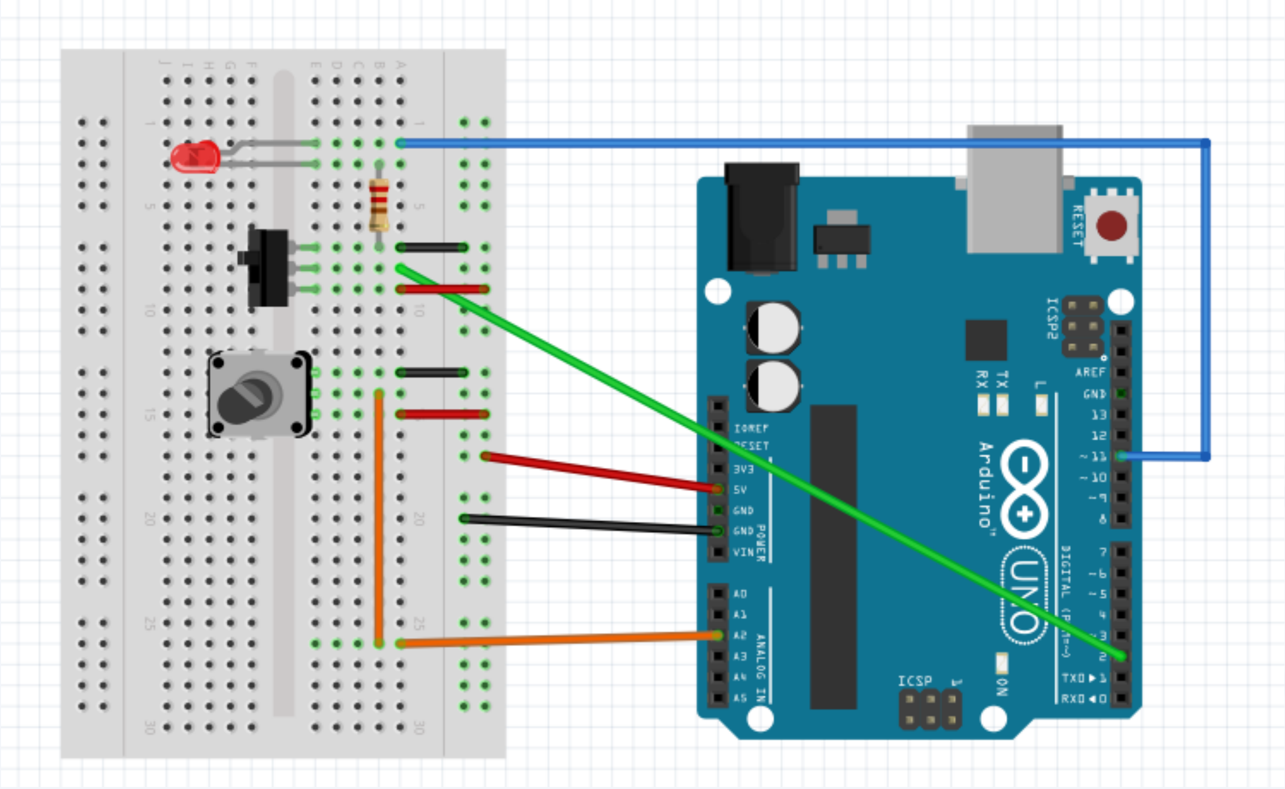
analogRead()

- **Arduino uses a 10-bit A/D Converter:**
  - What is the maximum 10-bit input value?
- **The input values from 0 to Max are mapped to 0V to 5V**
- **Command you need to know**

```
int sensorValue = analogRead(A0);
```

Exercise 3

- Modify your program to control blinky LED’s delay based on the resistance value in a potentiometer



Thank You!!