IME-100

ECE

Lab 3

Electrical and Computer Engineering Department

Kettering University

Getting Started

1. Laboratory Computers
   1. Log-in with User Name: Kettering Student (no password required)
   2. IME-100 information (Lab presentation, files, etc.) in folder on desktop
   3. Arduino programming software on desktop
   4. At the end of lab, Logout of computer; arrange keyboard and mouse
2. Laboratory kit
   1. Arduino board, circuit prototyping breadboard, electronic components: LEDs, resistors, pushbutton switch, potentiometer
   2. At the end of lab, turn instrument power off, dismantle circuit, unplug Arduino USB from PC, neatly arrange leads on bench

IME-100, ECE Lab3
Arduino Microcontroller

In this laboratory exercise, you will do the following:

- Learn about microcontrollers
- Identify the main components of Arduino microcontroller
- Understand the Arduino programming process
- Write program to Arduino microcontroller
- Build simple LED and switch circuits on a breadboard and connect to Arduino
- Exercise: Build a traffic light circuit based on Arduino

What is a Microcontroller?

- Microcontroller is a single computer chip that incorporates a processor (CPU) and lots of other modules such as:
  - Memory
  - Timer functions
  - Serial communication interfaces
  - A/D, D/A converter
  - Parallel I/O interface
- CPU is the central processing unit that is in charge of executing program instructions and controlling the system.
- Embedded systems
  - A special purpose computer system designed to perform a dedicated function.
  - Very specific requirements.

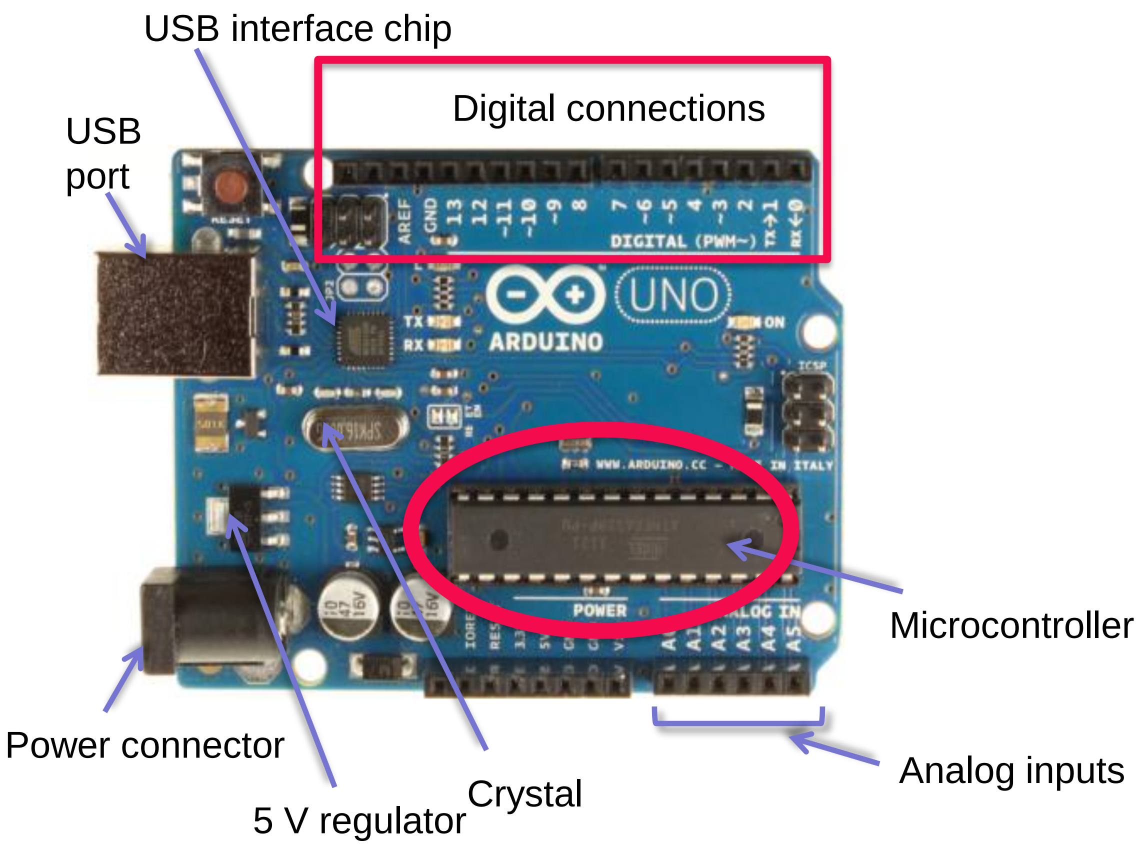
Embedded Microprocessors are ***Everywhere***

- ***In your home:*** Alarm Clock, TV, microwave, thermostat
- ***In the office:*** copy machine, telephone, computer monitor
- ***In the hospital:*** MRI machine, heart monitor
- ***Even in your pocket:*** cell phone, car key remote

Arduino

What is Arduino?

- A low-cost, open-source physical computing platform based on a simple microcontroller board
- Development environment for writing software for the board.
- Programming language: Simplified version of C/C++. (Easy to learn)



Refer to the Arduino website for more information:
https://www.arduino.cc/en/Guide/Introduction
https://www.arduino.cc/en/Tutorial/Foundations

Programming the Arduino

- The Arduino Programming environment is accessible from your PC desktop.
- The Arduino software programming environment is free and can be downloaded from the following website:

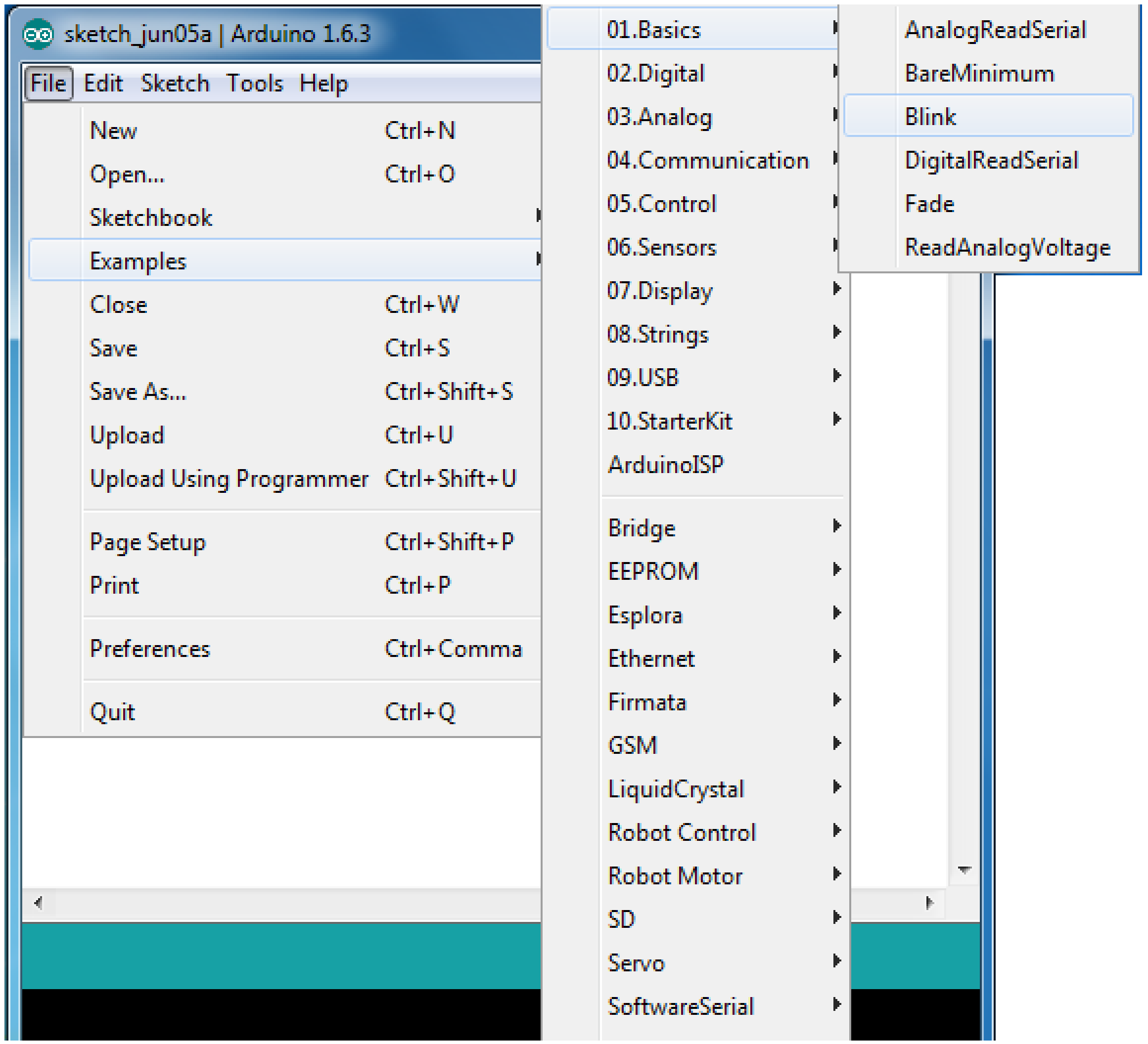
  http://www.arduino.cc/en/Main/Software
- Start the program by double clicking its icon on your PC desktop:

Programming the Arduino

- A good starting point to learn programming is to go over example code:
- Open the Blink example code:

  File -> Examples ->

  01.Basics -> Blink



Programming the Arduino

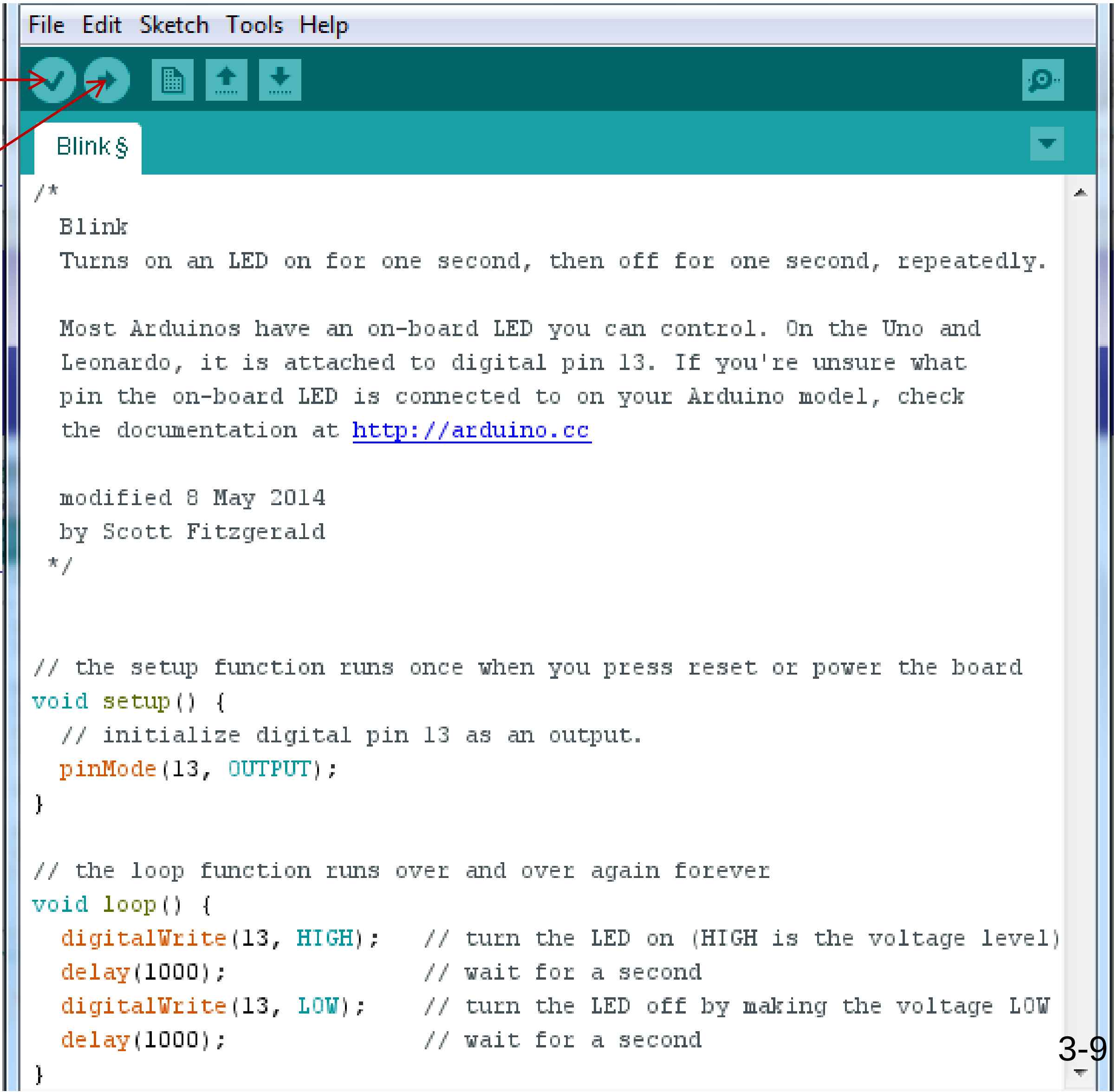
Click to “Verify” syntax correctness of program

Click to “upload” code to the Arduino board

Comment
(for humans only,
ignored by the Arduono)

setup
(section of the code
run only once at beginning)

loop
(section of the code
That runs “forever”)



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

  Most Arduinos have an on-board LED you can control. On the Uno and
  Leonardo, it is attached to digital pin 13. If you're unsure what
  pin the on-board LED is connected to on your Arduino model, check
  the documentation at http://arduino.cc

  modified 8 May 2014
  by Scott Fitzgerald
 */


// 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
}
```

Programming the Arduino Basic Programming Concepts

- Variable
  - A place for storing a piece of data.
  - It has a Name, a Type, a Value.
    - Example: int ledPin = 13;
      - ➢ ledPin: Name
      - ➢ int: Type
      - ➢ 13: Value
  - Whenever you use the variable name in the code, the value will be retrieved.
  - The value can be changed while your program is running.
    - This is why we call it a *variable*.
- Data type. Arduino supports the following data types for variables

boolean : true/false	
char : -128 ~ 127	**byte** : 0 ~ 255
int : -32,768 ~ 32,767	**unsigned int** : 0 ~ 65,535
long	**unsigned long** :
float	**double** : (= **float** in Arduino).

Programming the Arduino Basic Programming Concepts

- Function
  - A named piece of code that can be used form elsewhere in your program.
  - Example:

```
void setup() // return type, name of the function, parameters
{
  pinMode(ledPin, OUTPUT); //call pinMode function with two parameters
}
```

- Two special functions that you should include in Arduino.
  - *setup()*
    - This is called once when your program starts.
    - Good place to initialize anything.
  - *loop()*
    - This is called over and over.

Programming the Arduino
Basic Programming Concepts

- Comments
  - Multiline comments

```
/* comments start with the slash and * sign
       it can extend on multiple lines
        should end with * and the slash just like the line below
*/
```

    - Everything between /* and */ will be ignored.
  - Single line comments

```
// two slashes are an indicator of starting comments.
```

Programming the Arduino
Basic Programming Concepts

- Digital: Only two values are allowed.
  - 1 or 0
  - True or False
  - HIGH or LOW
- Analog: Continuous range of values
  - 0~1023
- The pins on the Arduino
  - They can be configured as either inputs or outputs.
- Sneak peek of basic functions to know
  - **pinMode**: Configure the specified pin to either an input or output.
  - **digitalWrite**: Write a HIGH or LOW to a digital PIN.
  - **digitalRead**: Read a value from the specified digital pin. Returns HIGH or LOW.
  - **analogRead**: Read an analog value from the specified analog pin. Returns 0~1023.
  - **analogWrite**: Write an analog value to the specified pin.
  - **delay**: Pauses the program for the amount of the specified time (in milliseconds).

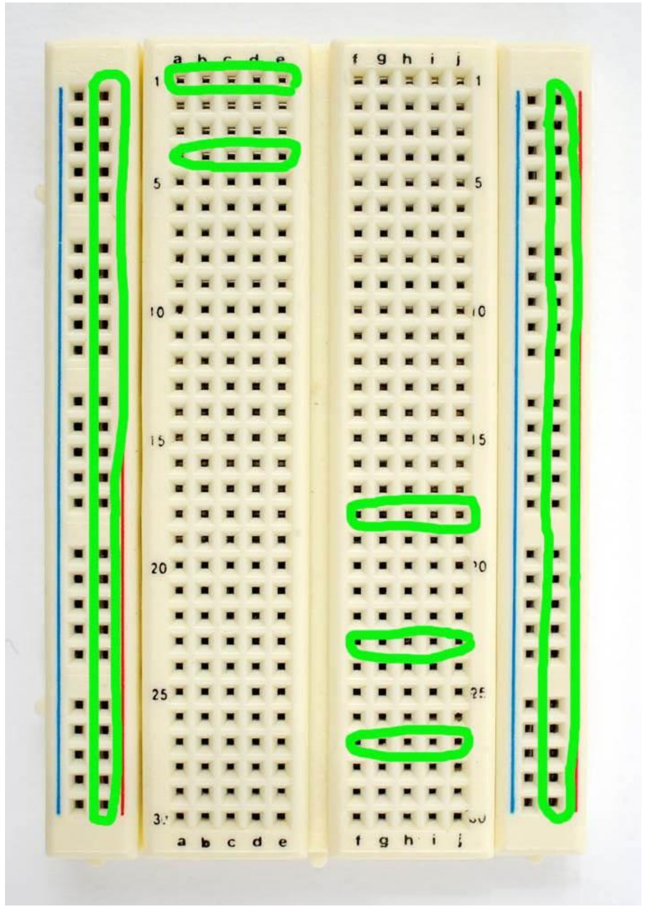
Refer to the Arduino website for more information:
https://www.arduino.cc/en/Reference/HomePage

Programming the Arduino Basic Programming Concepts

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

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int ledPin = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(ledPin, OUTPUT);
}

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

Programming the Arduino Breadboard

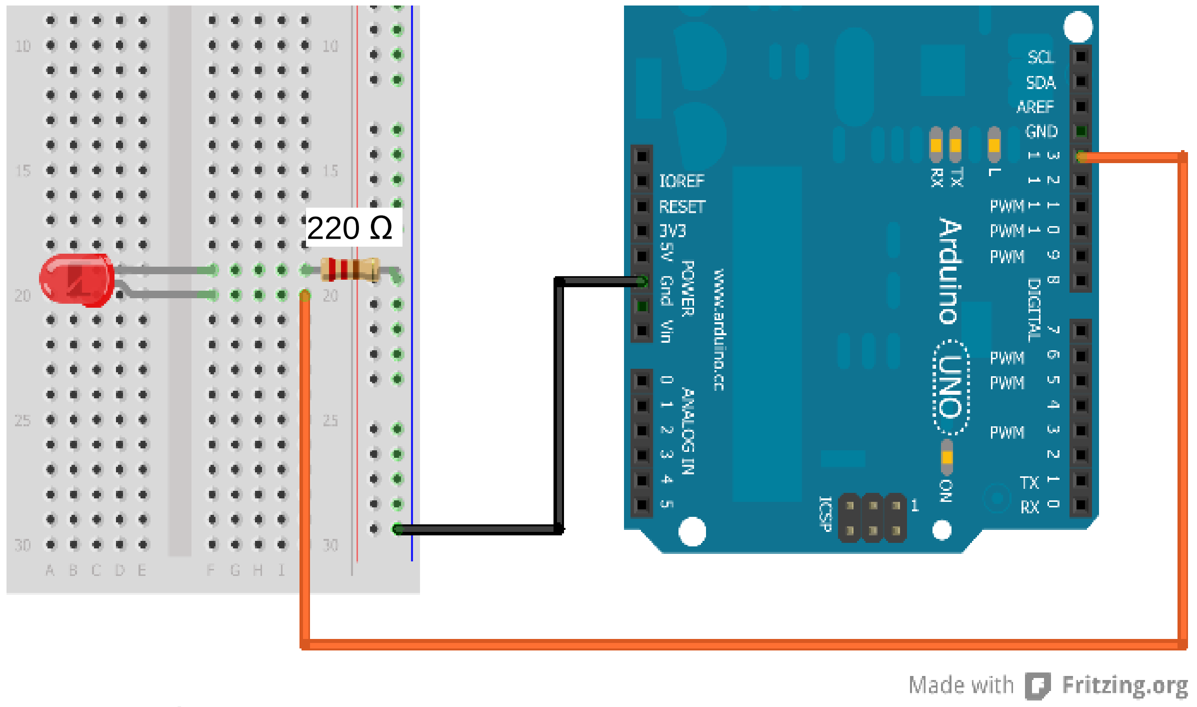
- Also known as Proto Board
- A construction base for prototyping electronics.
- Soldering is not required.
- Sockets are already connected as shown
  - Outside (both ends): vertically
  - Inside: horizontally.

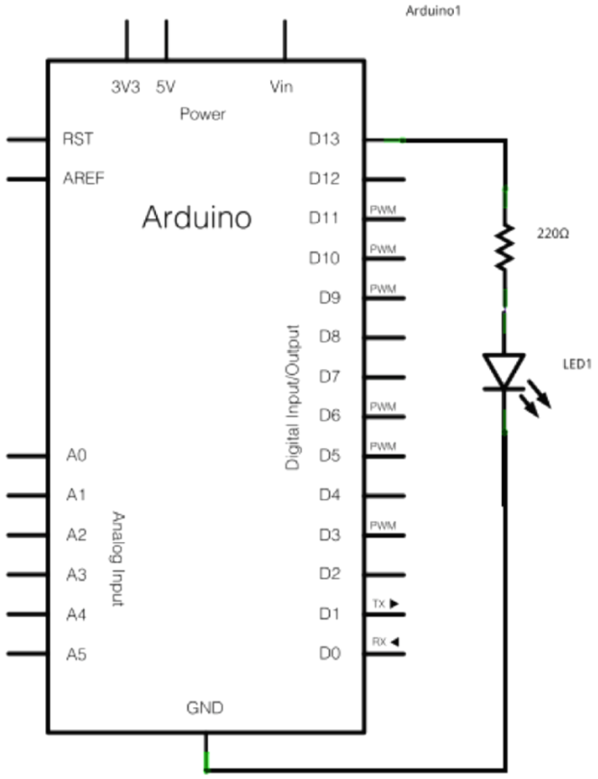


Programming the Arduino Breadboard

From Schematic diagram to actual wiring:

- Build an LED circuit on the Breadboard and test your Arduino code with it

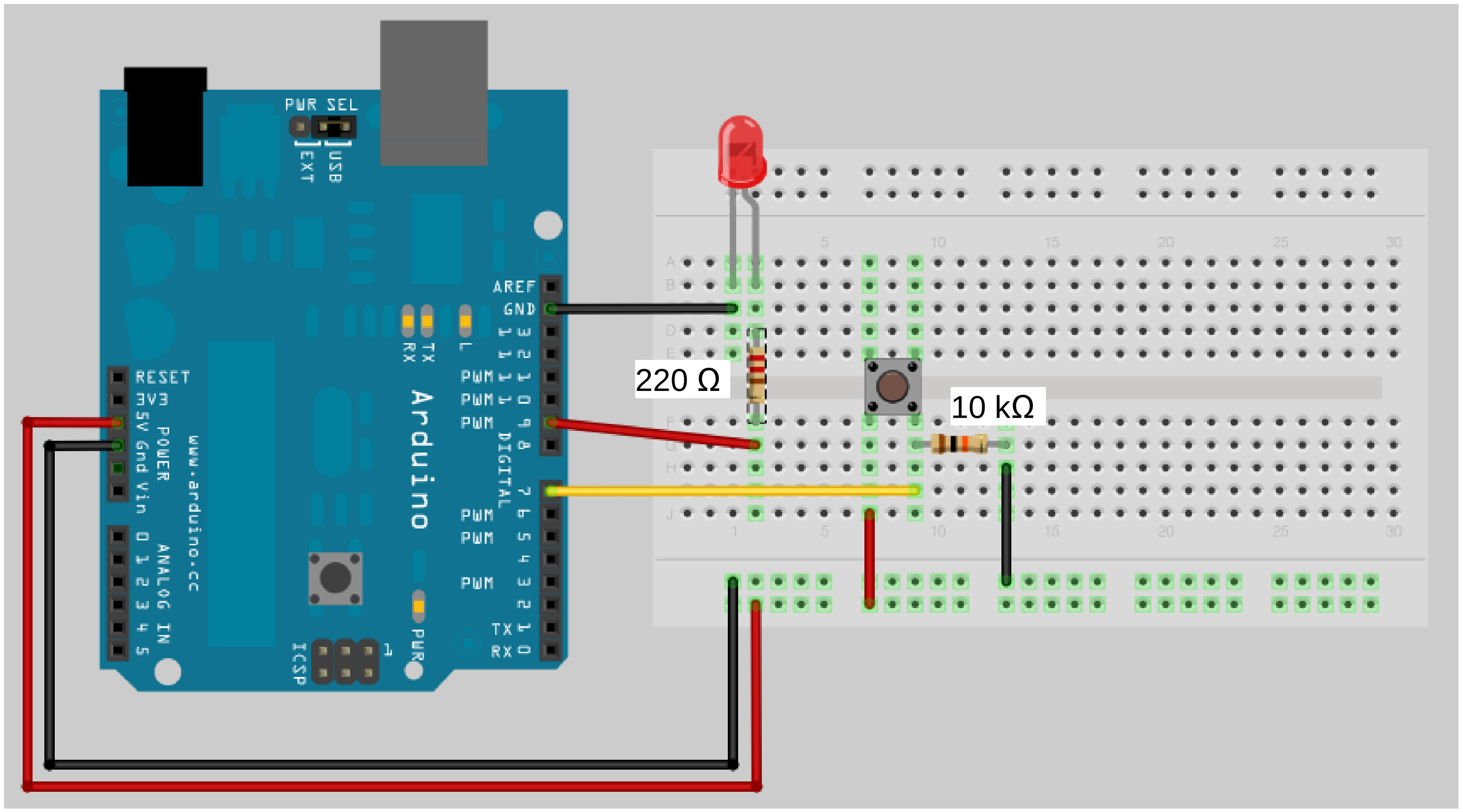




Controlling LED with a button

Add a pushbutton switch to control the LED:

- Read the status of the button and turn on the LED if the button is pressed, and turn it off otherwise.



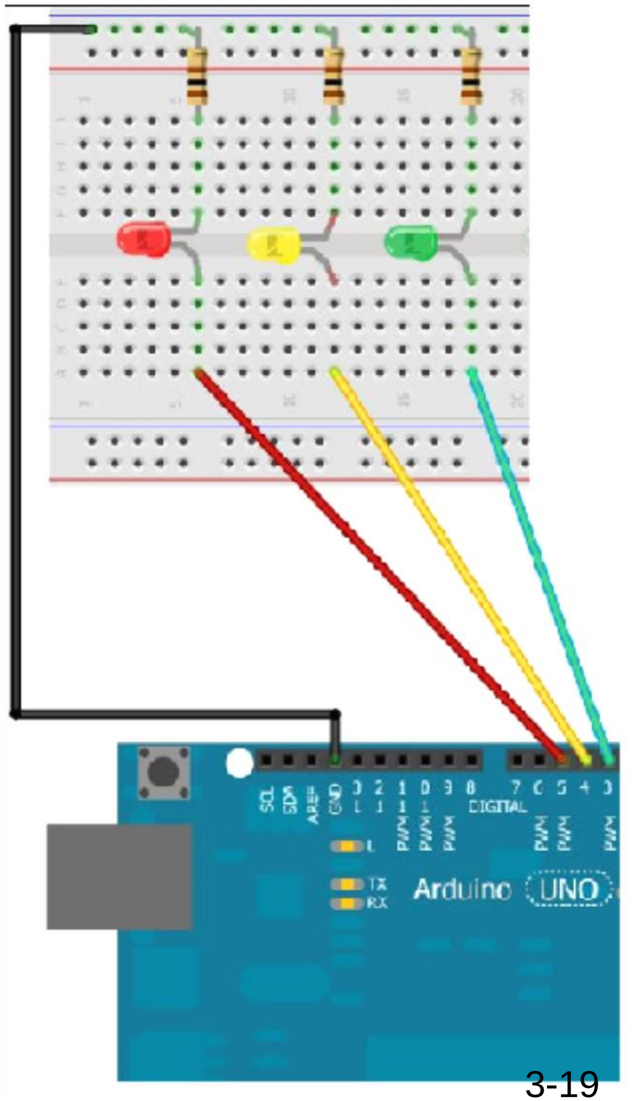
Controlling LED with a button

```
const int buttonPin = 7;
const int ledPin =  9;

// variables will change:
int buttonState = 0; // variable for reading the
                     //pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}
```

```
void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

Traffic Light Controller using Arduino

- Wire three LEDs (red, yellow, green) on your breadboad to the Arduino pins 5, 4, and 3, and connect 100 Ω series resistors to ground for each LED as shown in the diagram.
- Modify your previous Arduino program so that it blinks the three LEDs in the following manner:
  - First, only the Green LED is ON for four second
  - Next, only the Yellow LED is ON for one second
  - Then, only the Red LED is ON for four seconds
- The program will repeat the above sequence indefinitely until the power is disconnected.



Traffic Light Controller using Arduino

```
int redPin = 5;
int yellowPin = 4;
int greenPin = 3;

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
}
```

```
void loop() {
  digitalWrite(greenPin, HIGH);
  digitalWrite(yellowPin, LOW);
  digitalWrite(redPin, LOW);
  delay(4000); //wait 4 seconds
  digitalWrite(greenPin, LOW);
  digitalWrite(yellowPin, HIGH);
  delay(1000); //wait 1 second
  digitalWrite(yellowPin, LOW);
  digitalWrite(redPin, HIGH);
  delay(4000); //wait 4 second
}
```

Exercise 1: Traffic Light with override feature

Add a pushbutton switch to control the operating mode of the traffic light:

- First, wire a pushbutton switch as shown in slide 17, using a 10 kΩ resistor to one of the pins on your Arduino.
- Next, write an Arduino program that controls your traffic light to operate in one of two operating modes at any time based on the status of the pushbutton switch.
  - If the button is not pressed the traffic light operates in a normal sequence (Green – Yellow – Red) with the same timing as before.
  - Otherwise, when an officer needs to override the normal operating mode he/she can press the button, in which case only the red light flashes on and off once per second.
- Demonstrate your working traffic light circuit with the button control option to your instructor

Exercise 2: Traffic Light with Pedestrian Crossing Timer

The task in this exercise is to provide pedestrians with control for activating a “walk” signal to cross a street. Assume the traffic signal stays green until pedestrians presses the button when they want to cross the street. After the button is pressed wait for 5 seconds and switch the signal to yellow, which will then turn to red after 1 second.

Use “white” LED as a “walk” signal for pedestrians. Turn on the “walk” signal (“white” LED) for 5 seconds to notify pedestrians that it is safe to cross the street. After the 5 seconds pass, flash the “white” LED for 3 seconds as a warning for the pedestrians before switching off the “white” LED. After turning off the “white” LED for one second then turn the traffic signal back to green.

Homework: Making Connections and For the Curious You …

Due: Beginning of 4th Week Lab

Making Connections

You are now able to design and implement a traffic light circuit using two different solutions:

a) Using a hardware approach

b) Using a microcontroller, programming, and basic circuit elements

Discuss the ***benefits and drawbacks*** of each approach based on different criteria, such as ***cost***, ***complexity***, ***time to market***, and ***design flexibility***.

For the Curious You

Research and brainstorm/painstorm with your lab partners and present ***innovative ways of improving specific existing products/services by using a microcontroller*** such as the Arduino. For extra credit opportunity propose new a product/service with good analysis on its market potential.

Turn in a 2-3 pages report for your answers.

Homework: Online tutorial

From the online tutorial and additional resources come prepared to the next week class with background information on the following topics:

- Displaying data to serial terminal
- Analog output
- Interfacing sensors
- Bluetooth communication

Next week:

- Use a motor control circuit to drive the robot
- Basic robot control
- Robotic sensors
- Use ultrasonic sensor to avoid crashes
- Light sensor and line following algorithm

Finishing Up

(and to get full-credit in the lab)

1. Clean-up at bench – Leave it better than you found it!
   i. Pick-up any spare parts, wire-trimmings, etc
   ii. Detangle and coil wire leads
   iii. Soldering stations and tools neatly arranged
   iv. Turn off instrument power, arrange neatly
   v. Logout of computer; arrange keyboard and mouse
   vi. Neatly arrange the chairs
2. Check-out with the instructor
   i. Leave the check-out sheet with your group names at your station

Lab 3 Check-Out Sheet

(to be left on the bench at the end of lab)

Group Members (please print name clearly):

____

____

____

Instructor (check all that apply):

☐ Traffic Light basic operation demonstrated

☐ Traffic Light with override feature demonstrated

☐ Traffic Light with potentiometer for timing control and with override feature demonstrated

☐ Computer Logout

☐ Bench clean-up

Wires, detangled and coiled,
Disposal of wire clipping, etc.
Arduino circuit dismantled
Arduino unplugged from PC USB
Instrument power off and arranged
Keyboard and Mouse arranged
Chairs arranged

Additional Comments: