Arduino Tutorial Common Syntax

Serial Monitor Output

Serial.begin (9600); : Connects the program to the serial monitor

Serial.print: Prints text in the dialogue box. Used for output readings from the Arduino board.

Serial.println(): Drops cursor down to the next line on the serial monitor

Serial.parseint() or Serial.parsefloat(): Allows user to read either integer or real numbers for input

Serial.flush (); : Waits for transmission of outgoing data to be complete

Serial.available (); : Waits until the input buffer has a character

Digital/Analog Commands

digitalWrite or analogWrite: Write code to the Arduino board (I.E digitalWrite (13, HIGH) = Turns LED light linked to Pin 13 On)

digitalRead or analogRead: Reads information from the Arduino Board. Typically used with a switch or sensor to determine if contact is being made (I.E digitalRead (5, Low) = Switch/Sensor linked to Pin 5 on the Arduino board is not making contact)

Serial.read: Reads and Output information from a specified pin on the Arduino Board or keyboard (NOTE Keyboard input will always read as character variable (even numbers)

Power Transfer/Wait Time

HIGH: Power transferred or switch will read in contact (On)

LOW: Power is not transferred or switch will read not in contact (Off)

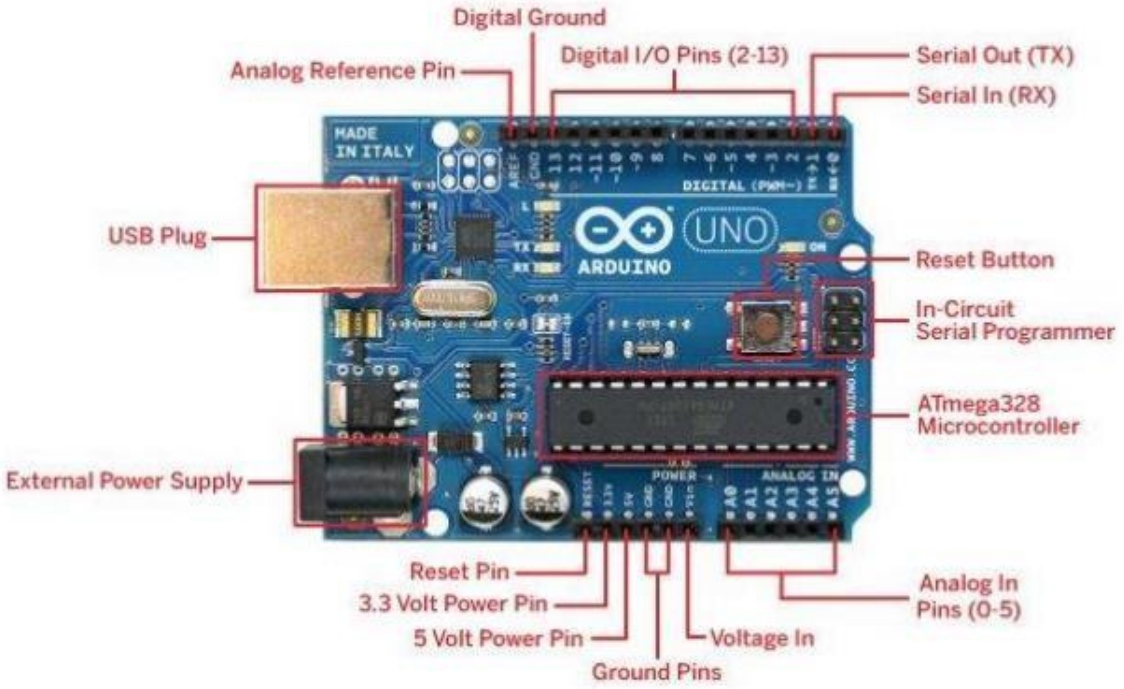
delay(): Will pause the program for x number of milliseconds (NOTE: 1000 milliseconds = 1 second)

Pin Declaration

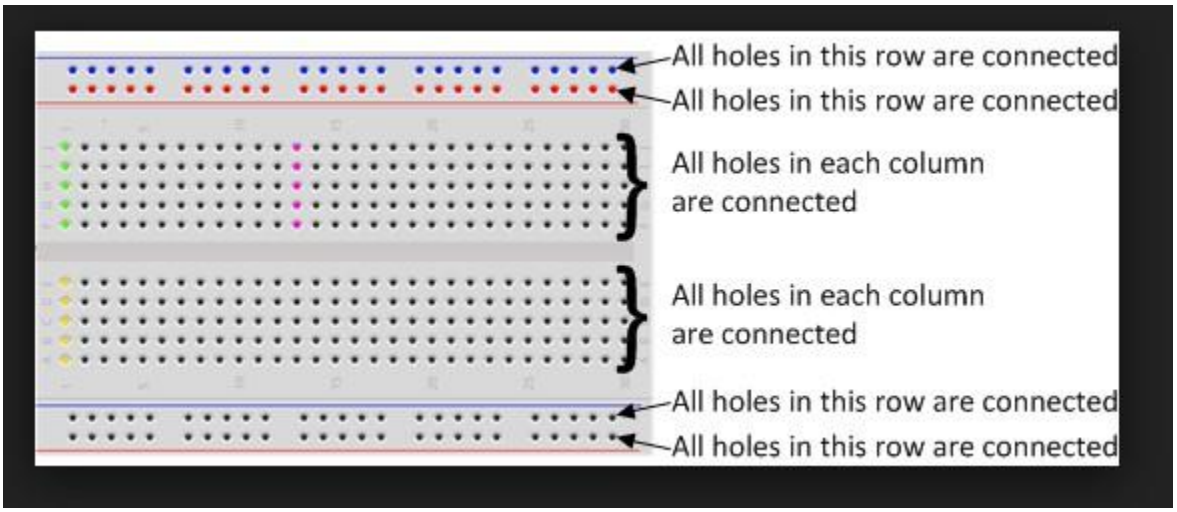
pinMode (): Declares the numbered pin to be either input or output (I.E pinMode (13, INPUT); or pinMode (13, OUTPUT)

NOTE: All Loops (for, while, do/while), if/then, if/else syntax is the same as learned in previous lessons

Basic Anatomy of the Arduino Board



Breadboard Setup

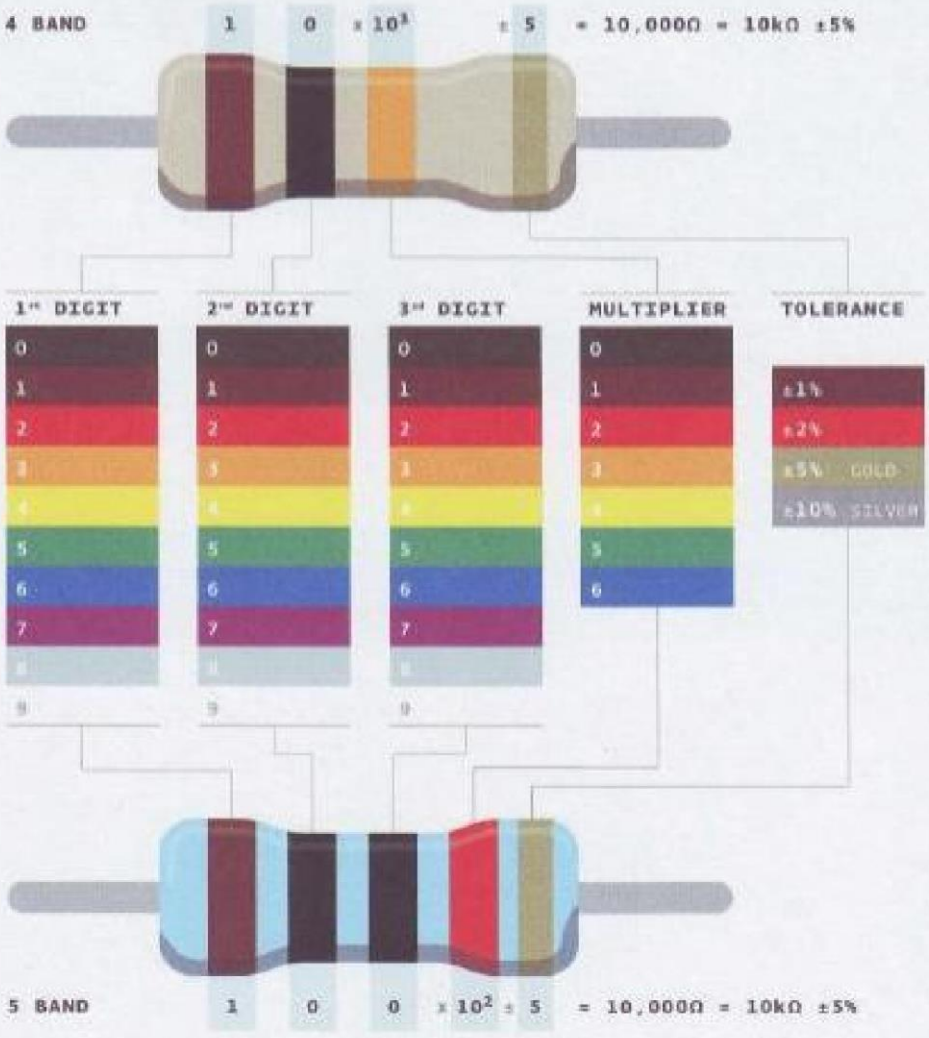


Reading a Resistor

HOW TO READ RESISTOR COLOR CODES

Resistor values are marked using colored bands, according to a code developed in the 1920s, when it was too difficult to write numbers on such tiny objects.

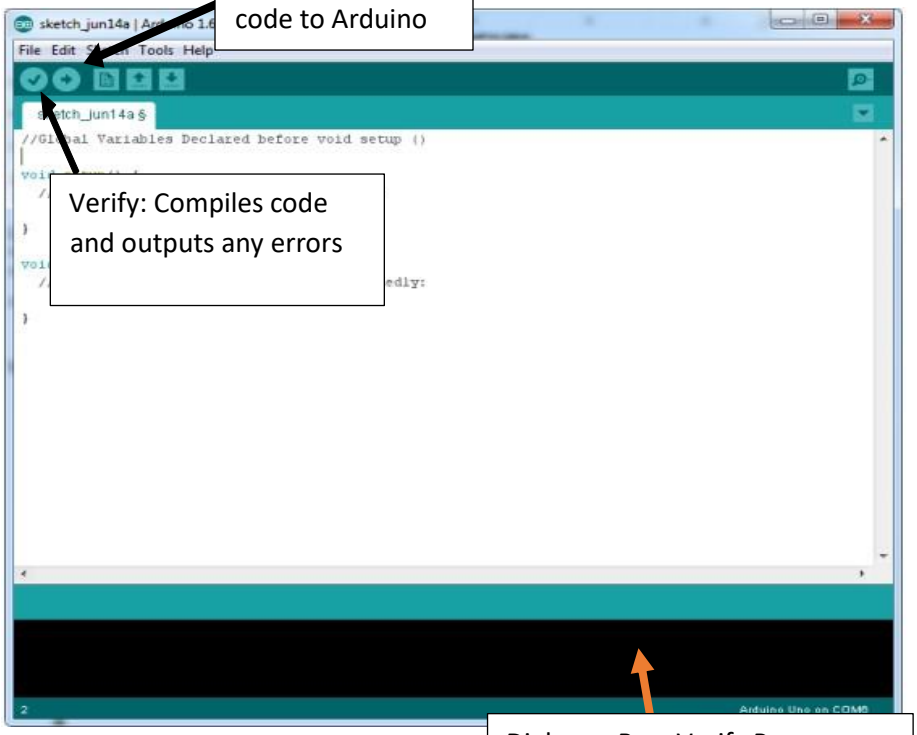
Each color corresponds to a number, like you see in the table below. Each resistor has either 4 or 5 bands. In the 4-band type, the first two bands indicate the first two digits of the value while the third one indicates the number of zeroes that follow (technically it reprents the power of ten). The last band specifies the tolerance: in the example below, gold indicates that the resistor value can be 10k ohm plus or minus 5%.



Workspace Setup

NOTE: It is not necessary to declare the #include <iostream> library or to int main() in your program.
Those codes are prebuilt into the structure of the Arduino program.

NOTE: Change Font Height of Text: Pull Down Menu File > Preferences > Font > Change to desired text height



Software provides 2 default functions:

void setup(){}: only runs the code between braces one time

void loop(){}: runs code between the braces infinitely. There is no way to end it. However program can be designed in a way to trap the program in an infinite loop that will not execute any commands within the loop

Connecting Arduino Uno to the Computer

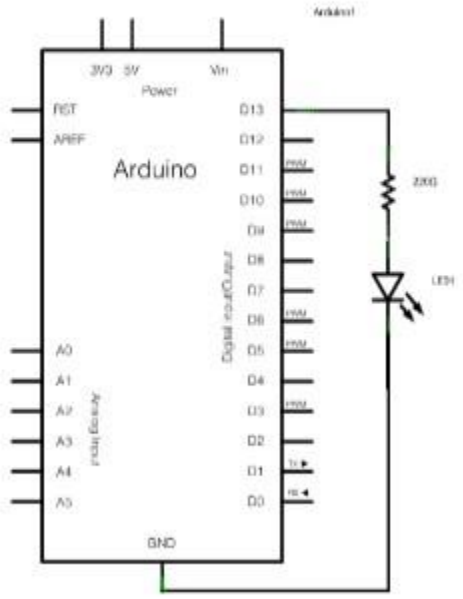
NOTE: These steps should been done each time Arduino Software is used.

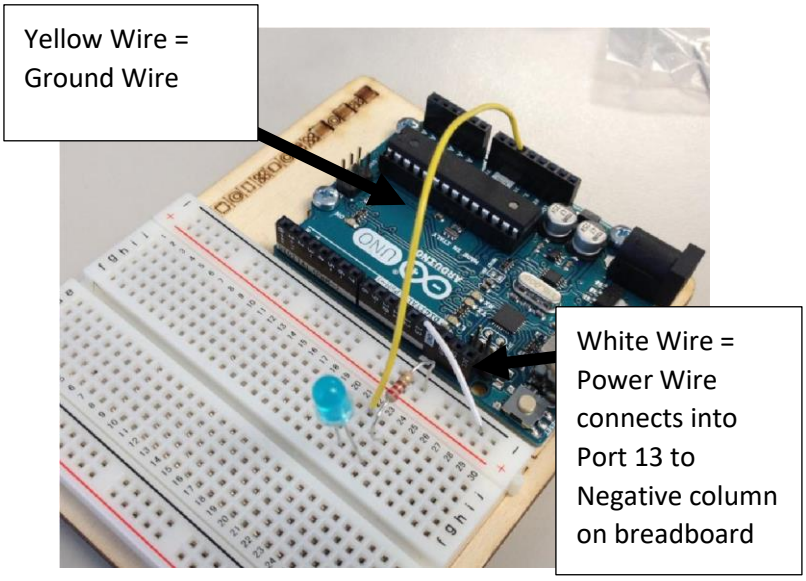
1. Using the USB-Fire Wire connector, connect the Arduino Board to the computer
2. Open Arduino Software from the desktop
3. Setting up Arudino Board Set the following
   a Tools > Board > Select Arduino Uno
   b Tools > Port > Select the Com# Port that has (Arduino Uno) in parentheses (NOTE: if Arduino Uno does not show up; look at the list of ports > unplug the wire see which port disappears (remember the number) > replug the wire in and choose the port that had disappeared from the list in the previous step)
   c Tools > Programmer > USBasp
   This will set the communication between the PC and the Arduino

Program 1: Flashing Light and Hello World!

Part 1: Turn on a light for a specified duration

Schematic





Connections

Step	Component	From	To
1	Wire	Arduino 5V	Breadboard – (negative) Line 1
2	LED	Breadboard: Short Side of LED E8	Breadboard: Long Side of LED E9
3	Wire	Arduino: Pin 13	Breadboard A8
4	Resistor 220 Ohm (Red, Red, Brown, Gold)	Breadboard: - Row 10	Breadboard: C9

NOTE: Connection ports DO NOT need to be the same as long as the wires and electrical components match up properly.

NOTE: LED Anode (Long Leg) should always be linked to the resistor and Cathode (short Side) linked to the ground or the direction the circuit is going.

Long Leg = Electricity In, + (Positive)

Short Leg = Electricity Out, - (Negative)

Program Code

Type following code

```
void setup() {
  // put your setup code here, to run once:
pinMode (13, OUTPUT);          //Declares that port 13 is only setup to output
digitalWrite  (13,HIGH);       // Sends a signal to port 13 to open electricty to flow through the port turning the LED on
delay (1000);                  //Allows power to flow to through port 13 for 1000ms (or 1 second)
digitalWrite (13, LOW);        //Turns power off to port 13; turning the light off
}

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

}
```

Save Program as Blink

Connect the Arduino Board to the computer using the USB-Fire Wire Cable.
NOTE: If board is not connected through a COM port unplug the board and reconnect with the steps listed above (Page 4 Connecting Arduino Board to Computer)

Result: Light should turn on for 1 second and then turn off

Part 2: Make the light blink repeatedly

1. Copy the code from the void setup() and paste in the void loop() 2.
Add a delay statement after the digitalWrite (13, LOW) of 200.
3. This will make the light flash on and off repeatedly. Experiment in changing the delays to alter the flashing of the light.

Part 3: Adding Input/Output , Variables, If/Statement, and Loops **1.**

Delete the Blink Code; Type the following Code

```
//Program: Adding Two Numbers

float number1, number2, sum;

char junk = ' ';                          // Variable used to erase bad input
void setup()
{
  Serial.begin (9600);                    // Connects user to Serial Monitor for Output on Screen
                                          // 9600 represents the baud rate (characters/sec
  Serial.println ("Let's calculate sum"); // Serial.println: places Text on Serial Monitor
                                          // the ln of code moves cursor down to the next line after output

  Serial.flush();                         // Waits for the transmission of outgoing data to complete
}



void loop()
{
  // Input Process for Number 1
  Serial.println ("Enter Number 1: ");
  while (Serial.available() == 0);        // ; will wait until the input buffer has a character
  {
    number1 = Serial.parseFloat();        // Serial.parseFloat() returns the first valid number from the Serial Buffer
    Serial.print ("Number 1 = ");
    Serial.println (number1);             // Displays users input
    while (Serial.available() > 0)        // Removes non-numeric chacters from the buffer
    {
      junk = Serial.read();               // Clears the Keyboard Buffer for next input
    }
  }


  // Input Process for Number 2
  Serial.println ("Enter Number 2: ");
  while (Serial.available() == 0);
  {
    number1 = Serial.parseFloat();
    Serial.print ("Number 2 = ");
    Serial.println (number2);
    while (Serial.available() > 0)
    {
      junk = Serial.read();
    }
  }
  sum = number1 + number2;                // Calulates Sum
  Serial.print ("Sum = ");
  Serial.println (sum);                   // Outputs Sum
}
```

2. **Modify the code to do the following**
    a. Change the code to calculate the volume of a cone
    b. Flash the lights as follows
        i. Volume < 100
            - Flash ON = 2 sec
            - OFF = 2sec
            - 3 Times (USE For Loop or Do/While Loop)
        ii. 100 <= Volume <= 500
            - Flash ON = 1 sec
            - OFF = 3 sec
            - 3 Times (USE For Loop or Do/While Loop)
        iii. Volume > 500
            - Flash ON = .5 sec
            - OFF = 2sec
            - 3 Times (USE For Loop or Do/While Loop)

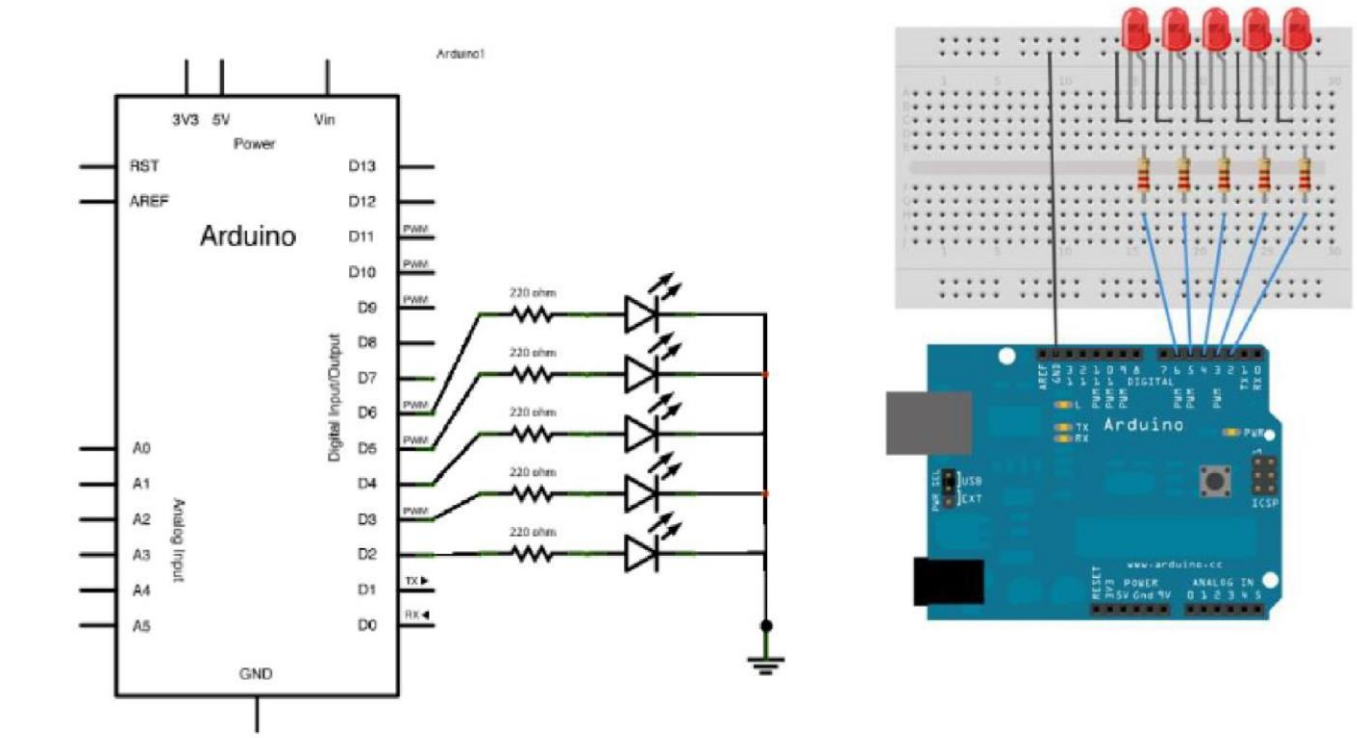
Part 4: Limiting the times through the loop

NOTE: There is no way to exit the Void Loop() in Arduino. However a delay statement or moving into a loop that does nothing can solve this problem. (I.E delay (100000000); or Create a Loop: while (TRUE) {} (while TRUE will always make the loop true so it will run infinitely.) Modify the program to do the following

1. After the first time through the program ask the user of they want to find the volume of another cone. Use 0 or Yes and 1 for No
    a. If Yes repeat the program
    b. If No end the program and say “Goodbye!”

Program 2: Switch Statement

Circuit Design



NOTE: LED Anode (Long Leg) should always be linked to the resistor and Cathode (short Side) linked to the ground or the direction the circuit is going.

Long Leg = Electricty In, + (Positive)

Short Leg = Electricity Out, - (Negative)

Program Code

Switch statement is like an If/Else/If/Else Statement. It only allows the user to compare discrete values (one thing at a time). It is not possible to compare ranges of objects. (For that an If statement is needed).

Program uses the Serial Monitor. Type in the characters a, b, c, d, e will turn on the LEDS they are connected to.

```
void setup() {
  // initialize serial communication:
  Serial.begin(9600);
  // initialize the LED pins:
  for (int thisPin = 2; thisPin < 7; thisPin++) {
    pinMode(thisPin, OUTPUT);
  }
}

void loop() {
  // read the sensor:
  if (Serial.available() > 0) {
    int inByte = Serial.read();
    // do something different depending on the character received.
    // The switch statement expects single number values for each case;
    // in this exmaple, though, you're using single quotes to tell
    // the controller to get the ASCII value for the character.  For
    // example 'a' = 97, 'b' = 98, and so forth:

    switch (inByte) {
      case 'a':
        digitalWrite(2, HIGH);
        break;
      case 'b':
        digitalWrite(3, HIGH);
        break;
      case 'c':
        digitalWrite(4, HIGH);
        break;
      case 'd':
        digitalWrite(5, HIGH);
        break;
      case 'e':
        digitalWrite(6, HIGH);
        break;
    }
  }
}
```

Using a Switch/Case statement is the equivalent of writing multiple If/Else Statements

For Example:

Switch (inByte) { case ‘a’: }

is equivalent to writing

if (inByte == ‘a’)

{then do this line(s) of code}

Upload the program to test it.

Notice how when one key is pressed then another the light of the previous key does not turn off.

Assignment: LED Lights Modify the program to do the following

1. Only have the inputted LED Lamp Turned ON (All other LED's turn off with each new input) I.E When 'a' is pressed 'a' LED light is on, then when 'b' is pressed the 'b' LED light is on and the 'a' LED is turned off.
2. Output if Input is NOT a, b, c, d, or e
    a. Blink all of the LED Lamps for a duration of 2 seconds
    b. Output in the Serial Monitor "ERROR, INPUT INVALID!"
    (HINT: if inbyte < 97 || inbyte >101)

NOTE: Serial.read converts all input to ASCCII code form. See below of values

DEC Value	Character
32	space
33	!
34	"
35	#
36	$
37	%
38	&
39	'
40	(
41	)
42	*
43	+
44	,
45	-
46	.
47	/
48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<
61	=
62	>
63	?

DEC Value	Character
64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
90	Z
91	[
92	\
93	]
94	^
95	_

DEC Value	Character
96	`
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z
123	{
124	|
125	}
126	~
127	

Program 3: Push Button

Source Code

```
/*
 Button
Turns on and off a light emitting diode(LED) connected to digital
pin 13, when pressing a pushbutton attached to pin 2.
The circuit:
* LED attached from pin 13 to ground
* pushbutton attached to pin 2 from +5V
* 10K resistor attached to pin 2 from ground
* Note: on most Arduinos there is already an LED on the board
attached to pin 13.
*/

// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;    // the number of the pushbutton pin
const int ledPin =  13;     // the number of the LED pin

// 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);
 }
}
```

Wiring Diagram Option 1

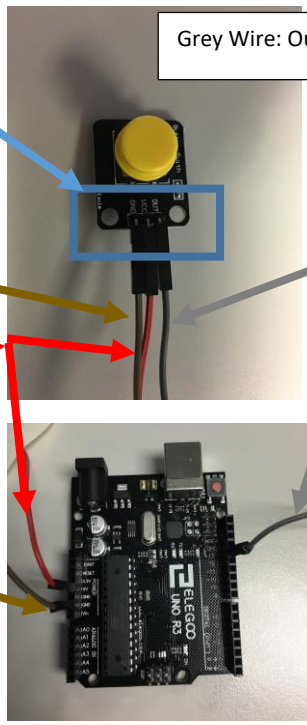
Direct Wiring to Arduino

Defines each pin

VCC = Volt

GND = Ground

Out = Digital Signal Port



Grey Wire: Out goes to Port 13

Brown Wire: GND goes to GND

Red Wire: VCC to 5V

Wiring Diagram Option 2

Direct Wiring to Arduino

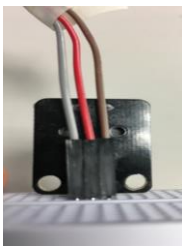
Brown Wire: Ground (GND) to Arduino Ground (GND)

Red Wire: Voltage (VCC) to Arduino 5V

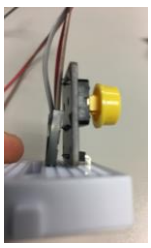
Grey Wire: Out to Arduino Port 13

Align each wire in the same row as the pin connection

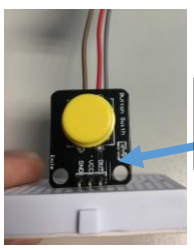
Top



Back



Left Side



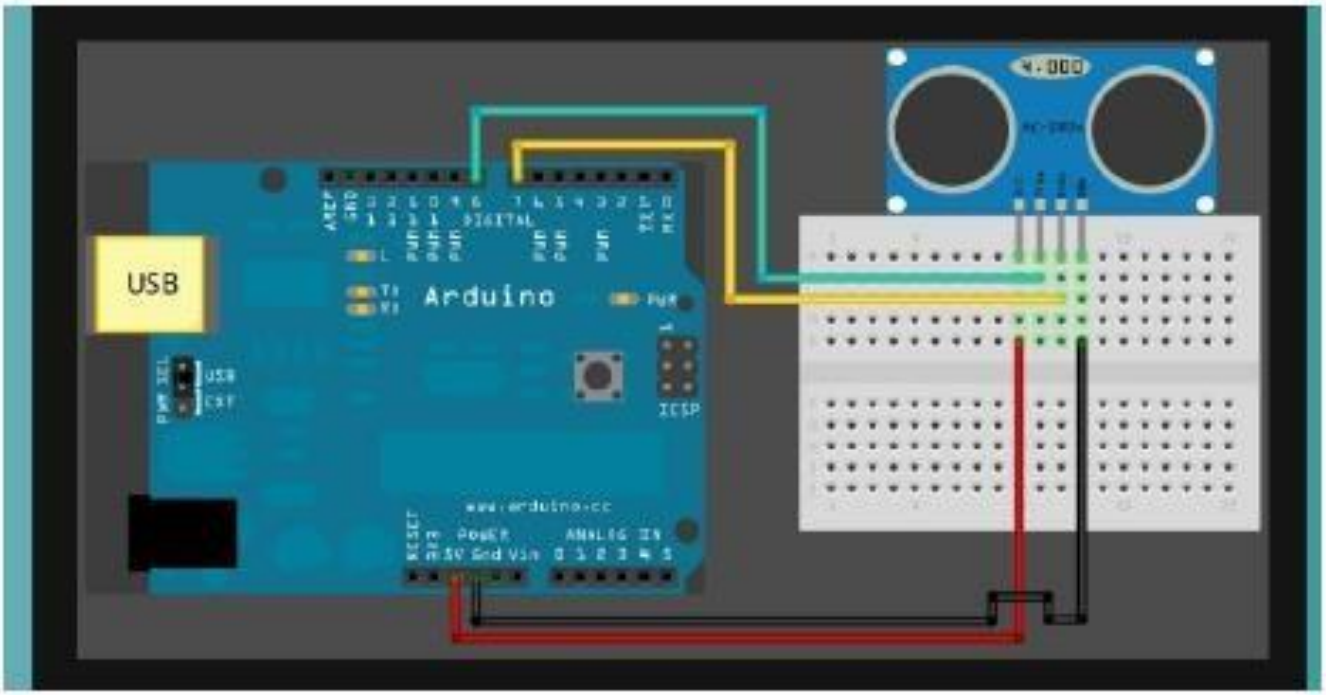
Shows what each pin represents

FRONT

Assignment: Push Button

1. Modify the program so the serial monitor screen shows
    a. Text On (Button Pressed) and Off (Button Released)
    b. Counter showing how many times the button has been released
2. Limit of On/Off
    a. Maximum presses = 5
    b. Once Limit is met do the following
        i. Stop the light from turning on and off when button is pressed
        ii. Serial Port Print “ALL DONE!”
        iii. Only print “ALL DONE!” twice on separate lines (Serial.println)

Program 4: Ultrasonic Sensor



Source Code

```
/*
 HC-SR04 Ping distance sensor:
 VCC to arduino 5v
 GND to arduino GND
 Echo to Arduino pin 7
 Trig to Arduino pin 8

 This sketch originates from Virtualmix: http://goo.gl/kJ8Gl
 Has been modified by Winkle ink here:
http://winkleink.blogspot.com.au/2012/05/arduino-hc-sr04-ultrasonic-distance.html
  And modified further by ScottC here:
http://arduinobasics.blogspot.com.au/2012/11/arduinobasics-hc-sr04-ultrasonic-
sensor.html
  on 10 Nov 2012.
  */
#define echoPin 7 // Echo Pin
#define trigPin 8 // Trigger Pin
#define LEDPin 13 // Onboard LED

int maximumRange = 200; // Maximum range needed
int minimumRange = 0; // Minimum range needed
long duration, distance; // Duration used to calculate distance

void setup() {
 Serial.begin (9600);
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
 pinMode(LEDPin, OUTPUT); // Use LED indicator (if required)
}

void loop() {
/* The following trigPin/echoPin cycle is used to determine the
 distance of the nearest object by bouncing soundwaves off of it. */
 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);

 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);

 digitalWrite(trigPin, LOW);
 duration = pulseIn(echoPin, HIGH);

 //Calculate the distance (in cm) based on the speed of sound.
 distance = duration/58.2;

 if (distance >= maximumRange || distance <= minimumRange){
 /* Send a negative number to computer and Turn LED ON
 to indicate "out of range" */
 Serial.println("-1");
 digitalWrite(LEDPin, HIGH);
 }


 else {
 /* Send the distance to the computer using Serial protocol, and
 turn LED OFF to indicate successful reading. */
 Serial.println(distance);
 digitalWrite(LEDPin, LOW);
 }

 //Delay 50ms before next reading.
 delay(50);
}
```

Upload Program and Test it. NOTE the program will print the distance in the serial port

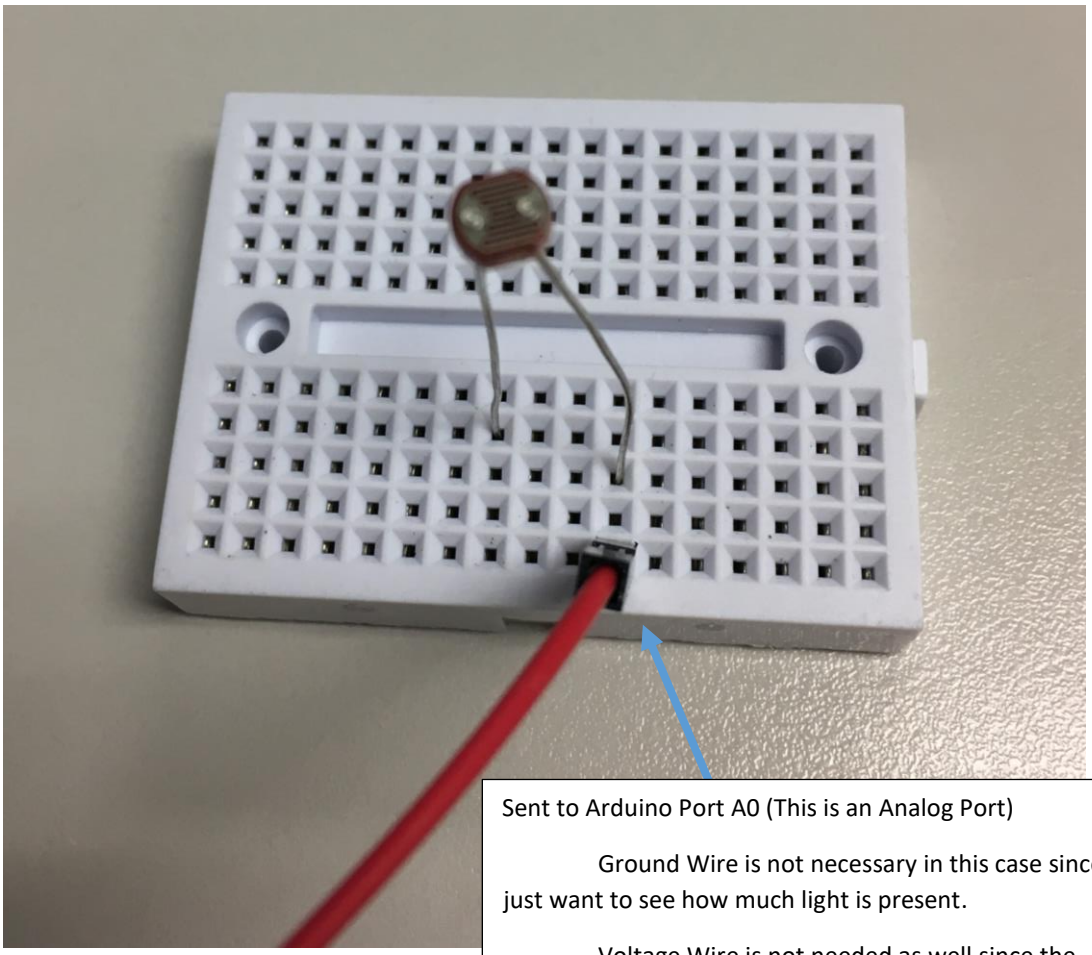
Assignment: Ultrasonic Sensor

1. Add a 220 Ohm resistor and LED on the circuit
2. Turn the light on when the an object gets with 20cm of ultrasonic sensor and turns off when it is out of range

Program 5: Photoresistor Sensor

Allows the user to implement a variable resistor that uses light to determine the resistances on the circuit based on the intensity of the light. Can also be used to sense the amount to light intensity around the photoresistor and then can activate other circuits given a defined set of constraints.

Wiring Diagram



Sent to Arduino Port A0 (This is an Analog Port)

Ground Wire is not necessary in this case since we just want to see how much light is present.

Voltage Wire is not needed as well since the surrounding light could supply the power if needed and adjust based on the intensity of the light

Legs of the photoresistor are the same. Does not matter which leg the wire lines up to

Write the following program

```
int sensorPin = A0;  //sets the photoresistor to Port A0 on the Arduino Board
int value = 0;

void setup()
{
Serial.begin(9600);
}

void loop()
{
value = analogRead (sensorPin);  //Reads the amount of light on the photoresistor
Serial.println(value,DEC);       //Converts the sensor reading to a decimal value
delay(50);                       //Provides Time between Readings
}
```

Testing:

Upload the program

a. Open the Serial Monitor
b. Place your hand over the Photoresistor > Notice the value change. Analyze the data
c. Close the Serial Monitor > Click on Tools Drop down menu > Select Serial Plotter > now the data is graphed. Analyze the data

Assignment: Simulate Morning, Day and Night Without Looking out the Window

Modify the code that can do the following.

Wire 3 different color LEDs to Digital Ports 1,2,3 (May consider using a Switch/Case for comparison)

Declare 3 different ranges (I.E 0-100 = Night)

1. Represents Dawn/Dusk = Middle Values
2. Represents Day = Highest Values (Maximum amount of light)
3. Represents Night: Lowest Values (Least amount of light)