Introduction to Arduino Programming

2017 – Gordon Payne

Newmarket High School

Contents **Page**

Exploration and Inquiry time for remainder of session.

www.hausOfPayne.weebly.com/arduino

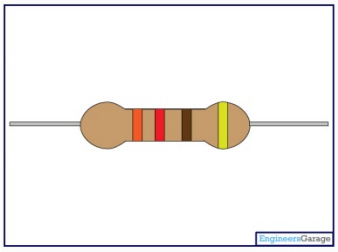
Introduction to Electronic Components

RULE NUMBER 1 – BE CAREFUL! If in doubt, check your circuit with your teacher before connecting it to the Arduino or power.

RULE NUMBER 2 – NEVER CONNECT A LIGHT EMITTING DIODE TO A 9V BATTERY. SERIOUS INJURY MAY RESULT!!!

RULE NUMBER 3 – IF IN DOUBT, RETURN TO RULES NUMBER 1 AND 2.

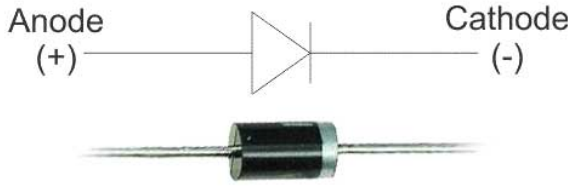
Resistors 'resist' the flow of current. They have 4 coloured bands on them that you decode to determine the value. Resistance is measured in Ohms. A 100R resistor is rated at 100 Ohms. A 100K resistor is rated at 100 THOUSAND Ohms**. Too much resistance means that the current will BACK UP in your circuit and probably cause burning and melting smells followed by death of the circuit. BE CAREFUL!!**



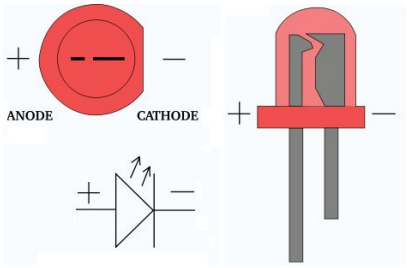
Resistors are NOT directional. Current can flow in both directions.

Diodes control the DIRECTION that current flows in a circuit. They are directional when inserted into a circuit.

The end with the 'band' on it, points the way the current will flow through the diode. If you have it backwards, the current won't flow and your circuit won't work (OR WORSE!!!) Be careful!



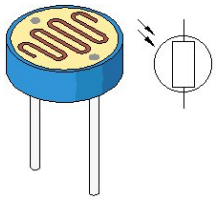
LEDs are Light Emitting Diodes. They glow when a current passes through them. They are DIRECTIONAL. The FLAT side of the LED must go towards the GROUND (-) side of the circuit. If it's backwards, it will not light up.



ALWAYS put a 220R, 470R or up to a 1K resistor in series with an LED to help prevent too much voltage flowing to the resistor and shortening its life.

LDRs are Light Dependent Resistors. As the light intensity hitting the LDR increases, the resistance inside the LDR goes down and current can flow through the LDR.

Typically used for Light Detection or Dark Detection circuits.

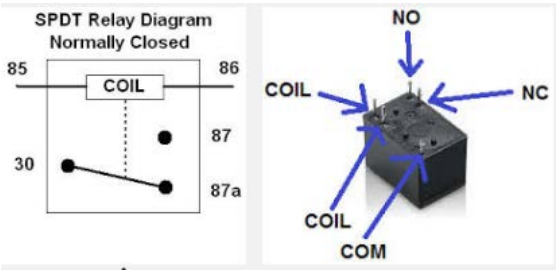


Relays are switches that can regulate LARGE current LOADS on the SUPPLY side of the circuit when a SMALL current flows through the COIL side of the relay**. YOU MUST PUT A DIODE** across the coil side to prevent fly-back damage to the circuit that may happen when the relay is activated and deactivated.

NO = Normal OPEN pole (LOAD SIDE)

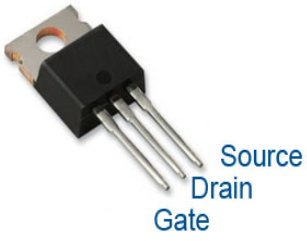
NC = Normal CLOSED pole (LOAD SIDE)
COM = COMMON pole (LOAD SIDE)

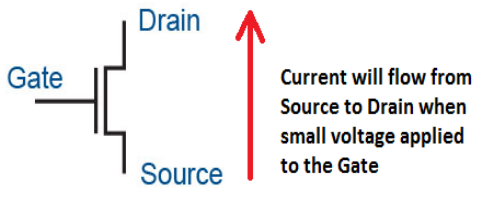
COIL = the in-circuit side of the relay (low current side)



DO NOT EXCEED THE VOLTAGE RATING OF THE RELAY

MOSFET – Metal Oxide Semiconductor Field Effect Transistor – This is a type of transistor (or switch) that can handle high voltage loads. **DO NOT EXCEED the voltage rating of the MOSFET.**





Activity 1 - Build the Circuit-du-Jour

We will use some or all of this circuit for all our other activities in this workshop.

The Breadboard

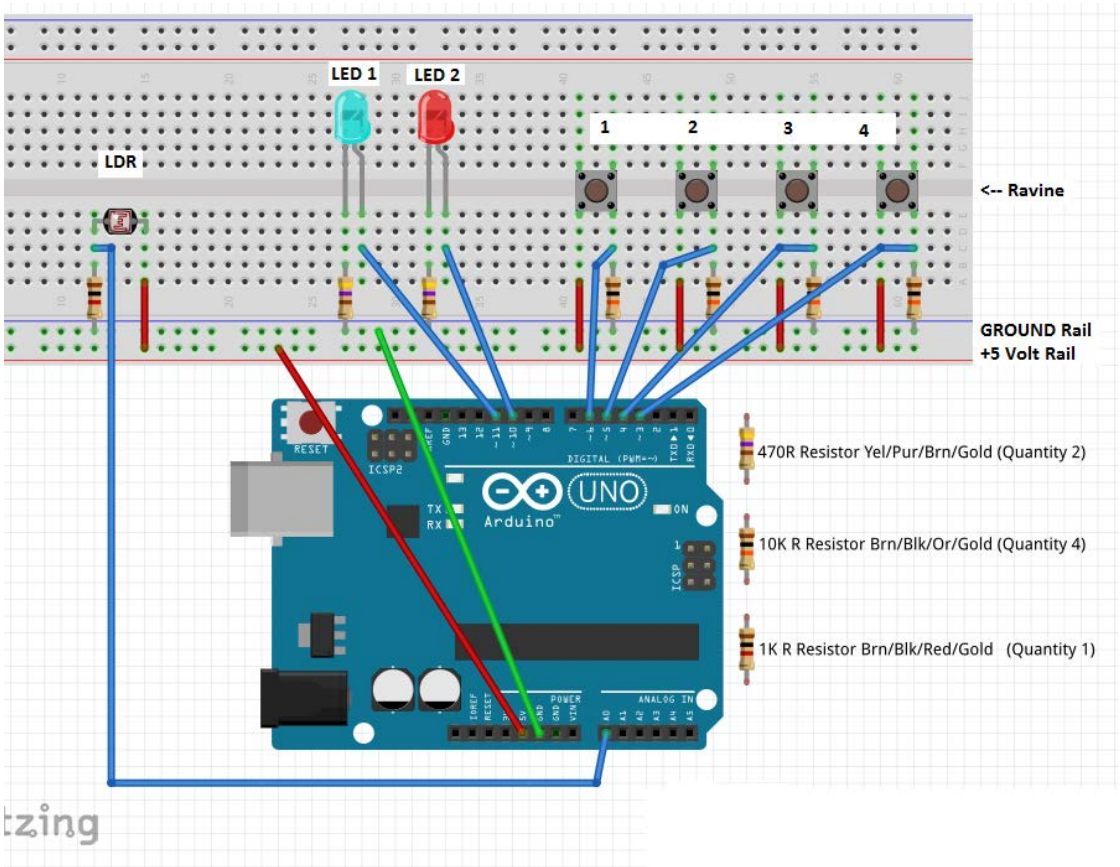
A bread board has distinct segments.

The BUSES are the HORIZONTAL red and blue indicated rows at the top and bottom of the board. All the pins in each row are connected electronically. If you need to ground(blue) or supply voltage(red) to a component, you can just connect that component to the appropriate row in the buses. Start by orienting the breadboard with the red 5 Volt rail at the bottom of the board as shown.

The Ground side is typically called **'GND'** and the positive voltage side is typically called **'+5V'**

The Connectors are the VERTICAL rows running across the board. If you wish to connect two components together, you just have them share a vertical row.

The RAVINE is a gutter that runs through the middle of the board. It is a PHYSICAL BARRIER between the two halves. You typically insert switches so they bridge this ravine. If you need to cross the ravine, you can just use small jumper wires.



Mount the Components

Insert the components into the board as shown on page 4.

The Light Dependent Resistor has a 1K resistor (Brown/Black/Red/Gold) connecting it to GND rail.

The two LEDs each have a 470R resistor (Yellow/Purple/Brown/Gold) connecting the CATHODE (short wire side, the FLAT side of the LED) to the GND rail.

The four switches each have a 10K resistor (Brown/Black/Orange/Gold) connecting the right pin of the switch to the GND rail.

Connect the other sides of the LDR and the SWITCHES with a short length of wire to the +5V rail (red).

Connecting your Circuit to the Arduino

Use a length of wire to join LED 1 to pin 11 and LED 2 to pin 10.

Use a length of wire to join each switch to pins 3,4,5,6.

Use a length of wire to join the LDR to Analog Pin 0 – A0.

Use a length of wire to join the blue GND rail to a GND pin.

Use a length of wire to join the +5V rail to the +5V pin.

Connect your Arduino to the computer with the USB cable.

Double check your work and we're ready to proceed!

Activity 2 – Control Some LEDs

Light emitting diodes (LEDs) are illuminated when current flows from the anode(+) to the cathode(GND). We put a 470R resistor in series with the LED just to reduce the voltage going through the LED and thus, extending its life.

Launch the Arduino IDE and copy/paste the following script into the sketch screen.

```
int led1 = 11;  // LED 1 is attached to pin 11

void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(led1, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(led1, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for 1000 milliseconds
  digitalWrite(led1, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for 1000 milliseconds
}
```

Go to the Tools menu and mouse down to **'Board'**. If you have an UNO, select '**Arduino UNO'**. If you have a Leonardo, select '**Arduino Leonardo'**.

Go to the Tools menu and mouse down to **'Port'**. Select the port with the Arduino connected to it.
DO NOT PICK COM1 OR COM3. Those are OFTEN not Arduino ports. You may have to try all to see which one is active.

Click **the LEFT ARROW** at the top left of the screen. Arduino will check your code for errors, compile it and then upload it to the board. You'll see the little lights on the board blinking as it uploads your code.

When 'Done uploading' is shown at the bottom of the screen, your sketch is now on the board and it's running.

LED2 should be blinking off/on at 1 second intervals.

Explore:

Modify your sketch to make the other LED work.

Modify your sketch to make both LEDs blink in sequence

Activity 3 – Detect a Button Push

Buttons(switches) are commonly used to initiate a function. We will have buttons 1 and 2 turn LED 1 On/Off.

The switches are called 'SPST' or 'Momentary ON' switches. SPST means, "Single Pole, Single Throw". When you depress the button, the switch is ON. When you release it, the switch is OFF.

Enter the following code in a fresh Arduino sketch.

```
int btn1 = 6; // pins for buttons connected to Arduino
int btn2 = 5;
int bt1Val, bt2Val; // value is TRUE if button pressed, FALSE
                    // if no press
long btThresh = 100; // time between button presses is 1/10 of a second
long lastBt1 = millis();// current system time
long lastBt2 = millis();
int led1 = 11; // first LED


void setup() {
  pinMode(btn1, INPUT); // set the button pins as INPUTS
  pinMode(btn2, INPUT);
  pinMode(led1, OUTPUT);// set the LED pins as OUTPUTS
  digitalWrite(led1, LOW);// turn OFF the LEDs
}

void loop() {
  // scan the buttons for presses
  bt1Val = digitalRead(btn1);
  bt2Val = digitalRead(btn2);
  if (bt1Val) {  // if the button state is HIGH
    if (millis() - lastBt1 > btThresh) { // if the minimum of time has passed
                                         // since the last button 1 press
      digitalWrite(led1, HIGH); //turn LED1 ON
      lastBt1 = millis(); //reset for next button press
    }
  }
  if (bt2Val) { // if the button state is HIGH
    if (millis() - lastBt2 > btThresh) { // if the minimum of time has passed
                                         // since the last button 1 press
      digitalWrite(led1, LOW); //turn LED1 OFF
      lastBt2 = millis(); //reset for next button press
    }
  }
  delay(50); // slow down sketch a bit
}
```

Upload the sketch to the board and try pressing the buttons.

Explore:

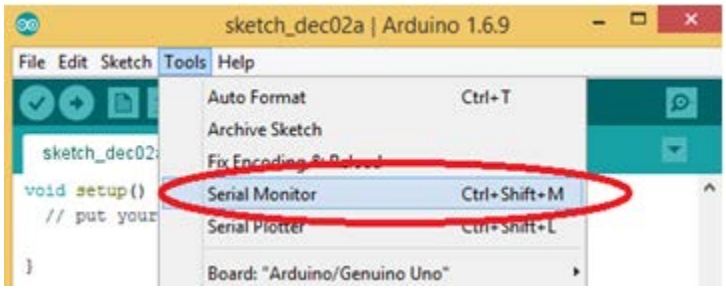
Modify your sketch so that LED 2 can be turned ON/OFF with buttons 3 and 4.

Modify your sketch to the first LED so it is ON when button 1 is held down and OFF when it's released.

Modify your sketch to toggle LED1 ON/OFF when you just press button 1.

Activity 4 - Interact with the Serial Monitor

Overview:
The Serial Monitor is a tool to interact with the Arduino textually. You can have the Arduino send you messages about the values in variables or text messages about where the sketch is at a point in time.
You can also send text values (numbers, letters, words etc.) TO the Arduino and then have the sketch use those values to control the flow of the program.



We're going to make a sketch that can light up one or both LEDs based on values you type in the Status Monitor AND it will send a message TO the Monitor to indicate if you've pressed the button.

In the Arduino IDE, start a new sketch and copy/paste in the code below. Save it as 'SerialMonitorDemo'.

```
int btn1 = 6; // button attached to pin 3
int led1 = 11;// LED 1 on pin 11
int led2 = 10;// LED 2 on pin 10
char val; // value of character received from Serial Monitor
int bt1Val = 0; // start with button NOT pressed
long lastBt1 = millis();
long btThresh = 100;

void setup() {
  pinMode (btn1, INPUT);
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  Serial.begin(9600); // open communications between Arduino and Serial
                      // Monitor at 9600 bits-per-second
  digitalWrite(led1, LOW); // make sure both LEDs are OFF
  digitalWrite(led2, LOW);
}
```

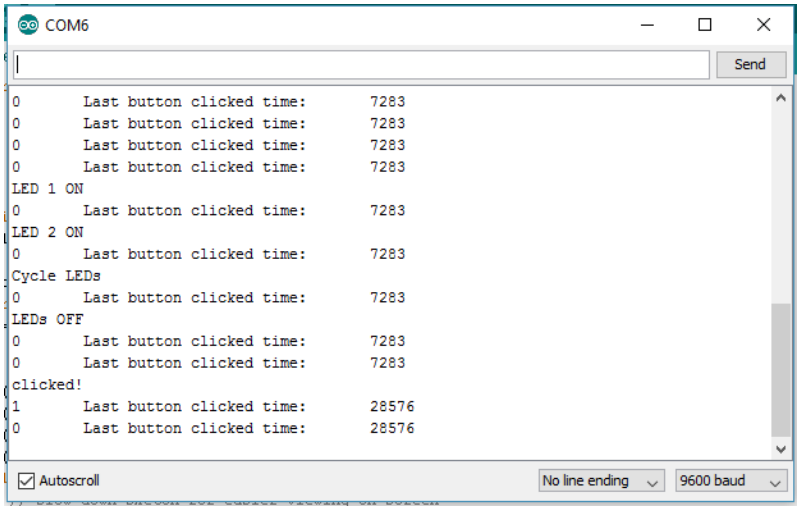
```
void loop() {
  if (Serial.available()) { // if there is data from the Serial Monitor
                      //available (ie. a character sent)
    val = Serial.read(); // read a byte of data

    switch (val) {  // do an action based on the character typed (val)
      case '1': // if a '1' received
        digitalWrite(led1, HIGH); // turn LED1 ON
        Serial.println("LED 1 ON");
        break;
      case '2': // if a '2' received
        digitalWrite(led2, HIGH); // turn LED2 ON
        Serial.println("LED 2 ON");
        break;
      case '3': // if a '3' received, cycle LEDs
        Serial.println("Cycle LEDs");
        for (int i = 1; i <= 4; i++) { // cycle each LED ON/OFF 4 times
          digitalWrite(led1, HIGH);
          delay(80);
          digitalWrite(led1, LOW);
          delay(80);
          digitalWrite(led2, HIGH);
          delay(80);
          digitalWrite(led2, LOW);
          delay(80);
        }
        break;
      case '4': // if a '4' received, turn BOTH LEDs ON
        Serial.println("BOTH ON");
         digitalWrite(led1, HIGH);
         digitalWrite(led2, HIGH);

        break;
      case '0': // if a '0' received, turn BOTH LEDs OFF
        digitalWrite(led1, LOW);
        digitalWrite(led2, LOW);
        Serial.println("LEDs OFF");
        break;
      default:  // disregard all other data sent from Serial Monitor
        break;
    }
  }
  bt1Val = digitalRead(btn1); // get the current state of the button pin
  if (millis() - lastBt1 > btThresh) { // if it's been at least 100 milliseconds since
                                //last button pin check
                                // filters out multiple button pin reads
                                // (like debouncing)
    if (bt1Val == HIGH) { //  if button is pressed
      Serial.println("clicked!");
```

```
    lastBt1 = millis(); // reset lastBt1 to the current time to be ready
    //for next click
   }
  }
  Serial.print(bt1Val);
  Serial.print("\t"); // '\t' is a TAB feed. Spaces out values across the line
  Serial.print("Last button clicked time:");
  Serial.print("\t");
  Serial.println(lastBt1);
  delay(150); // slow down sketch for easier viewing on screen
}
```

When you open the Serial Monitor from the Tools menu and enter 1,2,3,4 or 0, or press the button, you should get output like below:



Activity 5 - Read a Light Dependent Resistor

A light dependent resistor(LDR) allows more current to flow as more light falls upon it.

We can use this property to detect levels of light and then operate other components based on that light level. Applications of LDRs include night-lights, intruder-detection or turning on a light on the front of your house when the sun goes down.

The resistance level of the LDR will increase as the LDR is covered. This value will be read by analog port A0. The range of values is 0 to 1023 but we will reduce it to a range of 0 to 255 (the maximum value that can be sent to a PWM port).

The scaled down value from the LDR will then be sent to Pin 11 as a PWM (Pulse Wave Modulation) signal to give a variable voltage to the LED and make it glow based on the value from the LDR.

Activity Part 1:
Enter the code below and upload it to the Arduino.

```
int led1 = 11;// LED on pin 11 for feedback from LDR value
int ldr = A0; // LDR connected to port A0 (analog port 0)
int outVal = 0; // will be used for scaling the photocallInputValue to give more variation
// in LED brightness
int ldrVal; // the value of light hitting the LDR

void setup()  {
  pinMode(led1, OUTPUT);
  Serial.begin(9600); // open channel to the Serial Monitor
}
void loop()  {
  ldrVal = analogRead(ldr);// read the level of light hitting the LDR
  outVal = map(ldrVal, 0, 1023, 0 , 255); // scales the input 0-1023 to 0-255
  analogWrite(led1, outVal); // set the LED pin brightness
  Serial.println(outVal);
  // The delay can be changed to get the desired dimming effect
  delay(20);
}
```

Now select the Serial Monitor from the Tools menu so you can see how the light level varies as you cover/expose the LDR.

Notice that as you cover the LDR, the brightness of the LED decreases. Unfortunately, it doesn't decrease very much.

Reason? The maximum brightness of the LED will match a value of 255 and the minimum brightness will match 0. However, we're only getting around 100 down to 30 in our actual values.

We can use some mathematics to further scale the available values to give the LED's supply voltage a much greater span so that when you darken the LDR, you get closer to 0 brightness and when you give the LDR full light, you'll get closer to maximum brightness of 255 on the LED.

Exponential functions are a base to a power like $y = 2^x$

$2^8 = 256$ and $2^0 = 1$.

If we subtract 1 from each of those expressions, we get 255 and 0, the full range of the LED brightness.

So look at our actual values of about 100 (full light) to 30 (covered). To get from 100 to 8, we could divide 100 by 12 and that would give us a bit more than 8. 30 divided by 12 is less than 3.

$2^3-1 = 7$ and $2^8-1 = 255$. Now we have a much nicer range in brightness values.

Activity Part 2:

Change your code to look like this:

```
void loop()  {
  ldrVal = analogRead(ldr);// read the level of light hitting the LDR
  outVal = map(ldrVal, 0, 1023, 0 , 255); // scales the input 0-1023 to 0-255
  int scaledVal = pow(2, outVal / 11) - 1; // raising 2 to the power of outVal/11
  analogWrite(led1, scaledVal); // set the LED pin brightness
  Serial.println(scaledVal);
  // The delay can be changed to get the desired dimming effect
  delay(20);
}
```

Upload your sketch to the Arduino and get back to the Status Monitor. Try your LDR. You should see it glowing very bright when uncovered and almost off when covered. Yeehaa!

NOTE: If you're getting scaledVal readings bigger than 255, just reduce the 255 in the 'map' statement so you don't get scaledVal being bigger than 255.

But we can do better!

If we get a low value of say 7 that's very dim, but we'd like it OFF. We could just say, "anything below 8, set to 0" (ie. Shut off the LED).

So just before the analogWrite line in your sketch add:

```
    if(scaledVal<8) scaledVal = 0;
```

Upload your sketch and try it again. Now your LED should go completely off when you cover the LDR.

Explore – Instead of a Light Detector, change your circuit to a DARK detector. It's really simple. Less than a line of code. In fact you can just change one statement and it should work!

Explore 2 – Make an Intruder Detector. When the light falls below a desired level, print out "Intruder! Intruder!"

Activity 6 - Communications between Arduino and a Computer running Processing

The ability to communicate between an Arduino and a computer via the USB serial port is really helpful when trying to design more complex projects.

Some examples:

An Arduino soda pop machine

Keeping slot cars from colliding or counting laps/timing on a slot car track

Control a Radio-control car with Arduino and Processing

Building an Arduino-based weather reporting station

Creating an Arduino-based scientific instrument monitoring system

Building an Arduino-based joystick.

We will use the 2 LEDs and the 4 buttons on the circuit to demonstrate two-way communications between an Arduino and a computer running the Processing language.

Create a new Arduino sketch and copy/paste this code into the sketch.

```
char val; // data received from Processing app
int btn1 = 6; // pins for buttons connected to Arduino
int btn2 = 5;
int btn3 = 4;
int btn4 = 3;
int bt1Val, bt2Val, bt3Val, bt4Val; // value is TRUE if button pressed, FALSE
                                    // if no press
long btThresh = 100; // time between button presses is 1/10 of a second
long lastBt1 = millis();// current system time
long lastBt2 = millis();
long lastBt3 = millis();
long lastBt4 = millis();
int led1 = 11; // first LED
int led2 = 10;// second LED

void setup() {
  pinMode(btn1, INPUT); // set the button pins as INPUTS
  pinMode(btn2, INPUT);
  pinMode(btn3, INPUT);
  pinMode(btn4, INPUT);
  pinMode(led1, OUTPUT);// set the LED pins as OUTPUTS
  pinMode(led2, OUTPUT);
  digitalWrite(led1, LOW);// turn OFF the LEDs
  digitalWrite(led2, LOW);
  Serial.begin(9600);  // open a serial communications channel between Arduino
                       // and the computer
}
```

```
void loop() {
  if (Serial.available()) { // If data is available to read,
    val = Serial.read(); // read it and store it in val
    if (val == 'L') {  // if 'L' sent from the Processing application
      for (int i = 1; i <= 5; i++) { // flash the LEDs 5 times
        digitalWrite(led1, HIGH); // Turn on LED 1
        delay(100);
        digitalWrite(led2, HIGH); // Turn on LED 2
        delay(100);
        digitalWrite(led1, LOW); // Turn off LED 1
        delay(50);
        digitalWrite(led2, LOW); // Turn off LED 2
      }
    }
  }
  // scan the buttons for presses
  bt1Val = digitalRead(btn1);
  if (bt1Val) {
    if (millis() - lastBt1 > btThresh) { // if the minimum of time has passed
                                         // since the last button 1 press
      Serial.write('1'); // send a '1' to Processing app
      lastBt1 = millis(); //reset for next button press
    }
  }
  bt2Val = digitalRead(btn2);
  if (bt2Val) {
    if (millis() - lastBt2 > btThresh) {
      Serial.write('2');
      lastBt2 = millis();
    }
  }
  bt3Val = digitalRead(btn3);
  if (bt3Val) {
    if (millis() - lastBt3 > btThresh) {
      Serial.write('3');
      lastBt3 = millis();
    }
  }
  bt4Val = digitalRead(btn4);
  if (bt4Val) {
    if (millis() - lastBt4 > btThresh) {
            for(int i = 1;i<=3;i++){ // send 3 repetitions of codes to
                                     // Processing
              Serial.write('1');
              delay(50);
              Serial.write('2');
```

```
            delay(50);
            Serial.write('3');
            delay(50);
    }
      lastBt4 = millis();
    }
  }
  delay(50); // slow down sketch a bit
}
```

Save the sketch and upload it to the Arduino.

You can test this sketch to see if it works by opening the Status Monitor and pressing the buttons. You should see '1', '2','3' if you press the first three buttons. If you press the 4th button (left-most), it should send 3 cycles of '1' '2' '3' to the status monitor.

To test the LEDs, type an 'L' in the Status Monitor and press the 'Send' button. The LEDs should flash.

Once your Arduino sketch is working we will now create the Processing program that will communicate with the Arduino.

If the Processing sketch RECEIVES a '1', it will draw a red circle on the screen. For a '2', it will draw a green circle and for a '3', it will draw a blue circle.

If you click the mouse on the screen, it will SEND an 'L' to the Arduino and the Arduino will flash the LEDs.

If you press button 4 on the Arduino, you will see the coloured circles cycle on/off several times.

Launch Processing (disregard the warnings or requests to update).
Copy the following code and paste it into the new Processing Sketch. Save the sketch.

```
// Demo of communicating between Arduino and Processing
import processing.serial.*;  // include the serial management library

Serial myPort;  // Create object from Serial class to connect to the Arduino

PFont f; // define a font object for text to be displayed on the screen
char val = ' '; // this variable will store the data sent from the Arduino

void setup() {
  size (600, 600);// set to the width and height of your window
  String portName = Serial.list()[0];// get the port Arduino connected to
  myPort = new Serial(this, portName, 9600); //assign this port to variable

  println(portName);// just confirm you got a valid port
  f = createFont("Arial", 24); //assign the font you'll use for the screen text
```

```
  textFont(f); // set the current font
}

void draw() {
  delay(100); // slow down the sketch a bit
  background(0); // redraw black background
  fill(255); // set colour to white
  text("Click mouse to send 'blink' to Arduino", 100, 300); //screen text

  // read data from the Arduino board
  if (myPort.available()>0) { // if something came from Arduino
    val = myPort.readChar(); // read the character
    println(val); // print out the data you got from the Arduino
  }
  switch(val) { // based on the value received from Arduino
  case '1': // if '1' sent from Arduino
    fill(200, 0, 0); // draw a red circle
    ellipse(100, 200, 50, 50);
    break;
  case '2'://if '2' sent from Arduino
    fill(0, 200, 0); // draw a green circle
    ellipse(300, 200, 50, 50);
    break;
  case '3'://if '3' sent from Arduino
    fill(0, 0, 200);// draw a blue circle
    ellipse(500, 200, 50, 50);
    break;
  default:
    break;
  }
}

void mousePressed() { // if the mouse is clicked
  myPort.write('L'); // send signal to Arduino to flash LEDs
}
```

Save this sketch.

QUIT the Arduino App before you run your Processing sketch. Arduino and Processing cannot share the Arduino board at the same time, so just quit Arduino.

Now with the Arduino connected to your computer, run the Processing sketch and try pressing the buttons on the Arduino. You should see different coloured circles displayed based on the buttons you pressed. For the 4th (right most button), you should see a cycle of the red,blue,green circles flashing on the screen.

Click your mouse on the window. You should see the two LEDs on the Arduino circuit flash several times.

Explore – Try to send the LDR value from Arduino (you could send it based on a button 4 press) and have that value used to change the shade of colour when you press buttons 1 to 3. This is challenging.

Online Resources

hausOfPayne.weebly.com/arduino – Mr. Payne's webpage of classroom activities and projects.

www.arduino.cc The home website for all things Arduino. The Reference tab is very helpful

www.processing.org The home website for all things Processing. The Reference tab is very helpful.

www.openprocessing.org Fantastic site for exploring the power of Processing.

www.sparkfun.com Fantastic US website for lessons on Arduino, electronics and all things robotic and computer engineering.

Youtube has lots of great Arduino project demos.

In the Arduino program, the File-Examples menu option has hundreds of demonstration sketches for doing different tasks.

In the Processing program, the File-Examples menu option has hundreds of demonstration sketches for doing different tasks.

Buying Arduinos and Components

Sayal Electronics
You can buy GENUINE Arduinos here.
Southwest corner of Woodbine Ave and 14th Ave in Markham (just south of 407).
FANTASTIC store for all things electronic. Convenient but can be expensive.
Their main location is in North York on Victoria Park, just south of Gordon Baker Rd. (past the Sunoco station on the left side)
Also a location in Vaughan.

Creatron Inc.
Southwest corner of Sheppard Ave and Pharmacy Ave in Scarborough
Arduinos, components and lots of maker stuff. 3D printers, stepper motors etc.

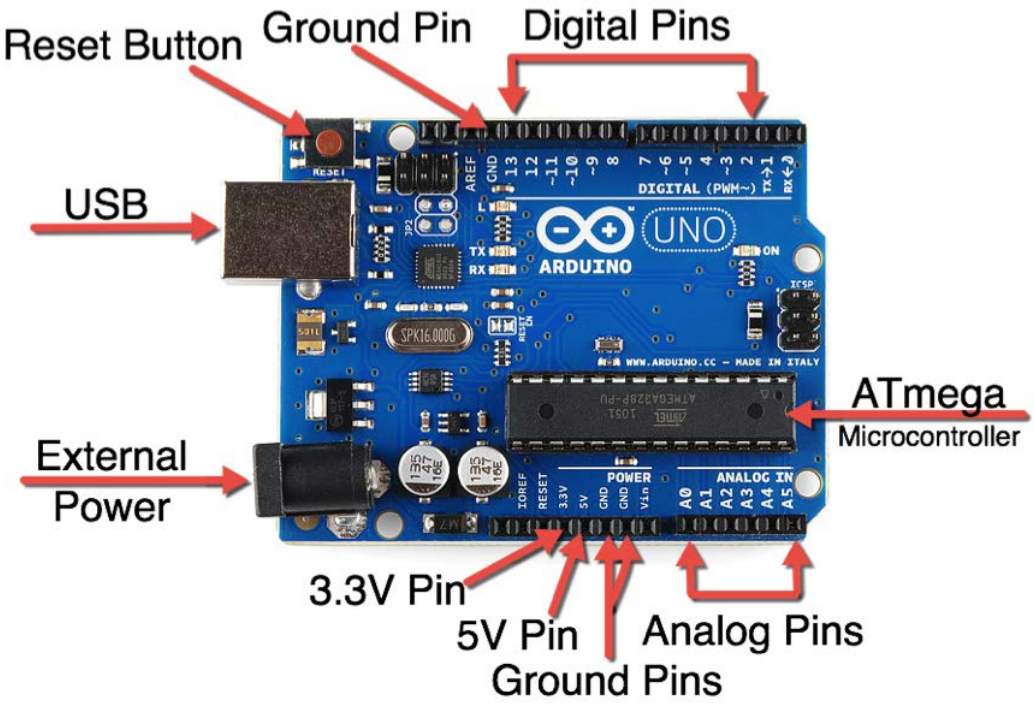
eBay
The least expensive place to order Arduinos and Components. Many times more affordable than retail. Just have to wait for delivery which is usually free. Almost ALL the Arduinos are clones/knockoffs. Lots of great starter kits with loads of components and devices.

HINT: If you are buying an Arduino Uno R3 clone, try to get the ones with the long Atmega328p chip. It's the long chip on the board. It's most likely to work with stock Arduino drivers. Otherwise, you may need to get the CH340 USB driver installer.

Sparkfun.com
US Superstore for Arduinos and components. Fantastic online lessons and videos.
They make RedBoard licensed versions of Arduino. Shipping can be expensive.

Amazon.ca	Genuine and knockoff Arduinos as well as great starter kits with lots of sensors and components.
Aliexpress.com	Chinese super-website (like Amazon). Usually free shipping.

Final Thoughts



Pressing the Reset button restarts your sketch from the top.

Arduinos are incredibly robust and mistake-tolerant devices. Don't be afraid you'll break them!

If you find the board or the chip are getting hot, just disconnect and let it cool down. Fix your circuit and reconnect and try again.

The 'Vin' pin can be used to supply more than 5 Volts to the Arduino. DO NOT supply more than 12 volts to an Arduino.

Do not try to activate any device requiring more than 5V from the Arduino pins directly. Use a MOSFET or a MOSFET/Relay combination and provide external power to the relay or MOSFET. See examples on my webpage.

The External Power jack can be used to power an Arduino without needing it to be connected to a computer USB port. (ex. Cars, Bluetooth devices etc). For example, a servo may require more than the 200 milliamps that an

Arduino can supply so it's helpful to use an **external 5V 1 Amp power supply** that you can plug into this jack. Plugging a 9V battery into the jack using an adapter is a handy way to provide external power.

If you ever find you have a 'runaway sketch', you can replace it with a 'BareMinimum' empty sketch from File-Examples-Basics menu. Once you have it on the screen, HOLD DOWN the RESET button on the Arduino and then click the Upload button on the screen. It will overwrite the runaway sketch with an empty sketch.

If you're getting an error when you try to upload a sketch, check the Tools-Board and make sure you have the correct type of board selected (Uno, Leonardo, Mega etc.). Also check the Tools-Port menu and make sure you're actually connected to the Arduino.

You cannot have Arduino OPEN when you are running a Processing sketch that talks to the Arduino. So just make sure the Arduino App is closed when you're running your Processing sketch.

The best way to learn Arduino is to experiment. Look online for something similar to what you want to do and start with it. Then build your own specific functionality onto the sketch/circuits.

Questions?

Feel free to contact me at gordon.payne@yrdsb.ca

NOW GO OUT AND MAKE SOMETHING WONDERFUL!!!