IME-100
Interdisciplinary Design and Manufacturing

Arduino Tutorial #2

Topics:

1. Handling Analog Signals in Arduino
    a. Analog input
    b. Analog output

2. Displaying Data on Serial Terminal

3. Displaying Message on LCDs

4. Displaying Message on 7-segment Displays

5. Interfacing Sensors
    a. Temperature sensor
    b. Force sensor
    c. Light sensor
    d. Ultrasonic sensor
    e. Pulse sensor

6. Color RGB LEDs
    a. RGB LED
    b. Neopixels

7. Bluetooth Communication

8. Homework Assignment

1. Handling Analog Signals in Arduino

a) Analog Input

Watch the following youtube videos to learn how the Arduino handles analog input signals:

http://www.youtube.com/watch?v=cgyeCRTPTYo

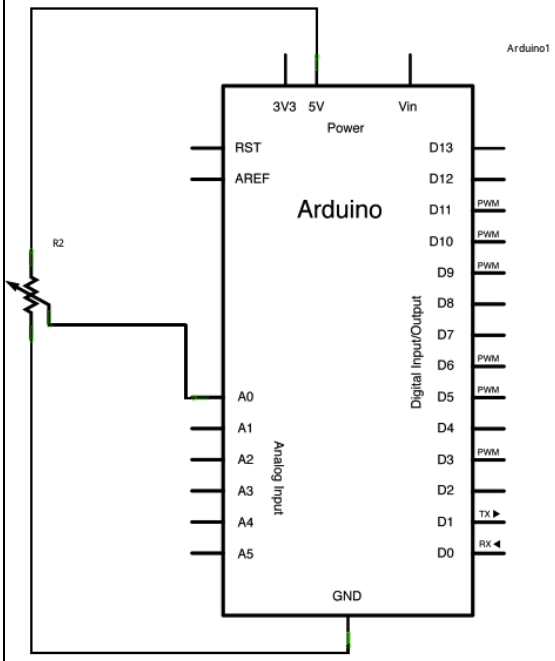
http://www.youtube.com/watch?v=WXXYFE8qLhQ

- Digital computer is naturally made to process data represented in digital format.
- ***Digital*** inputs such as those coming from switches (ON/OFF) could be directly handled by the Arduino without any conversion. Such digital signals are represented as HIGH and LOW, or 1 and 0.
- However, most of the things in nature are not in digital form. For example, if you want to build a weather station, your Arduino should be able to handle temperature, humidity, wind speed, rainfall, etc. The signals coming from these sensors are not restricted to HIGH and LOW (1 and 0, or ON/OFF), rather they typically assume a wide range of possible values. This type of signals is called ***analog***.
- Arduino has a built in hardware called ***analog-to-digital convertor (ADC)*** that can be used for converting analog signals to digital format so that they can be processed or stored in memory.
- The ***analogRead*(analogPin)** function uses the ADC module to convert a signal applied at one of the 6 analog pins of the Arduino Uno (A0, A1, … A5).
    - You do not need to call ***pinMode()*** to set the pin as an input before calling ***analogRead()***
    - The result of ***analogRead()*** is a 10-bit binary number in decimal range 0 to 1023.
    - For example, if 2V analog signal is applied to pin A5, the ***analogRead(A5)*** function returns (2V/5V) *1023 = 409 (assuming the ADC uses a reference voltage 5V)

Exercise: If the ***analogRead()*** function gave a reading of 512, what is the corresponding analog voltage applied to the analog input pin?

Example: A potentiometer is a device that provides a variable resistance, which you can read into the Arduino board as an analog value. In this example, we connect a potentiometer to one of the analog input A0 and use it to control the rate at which the built-in LED on pin 13 blinks.

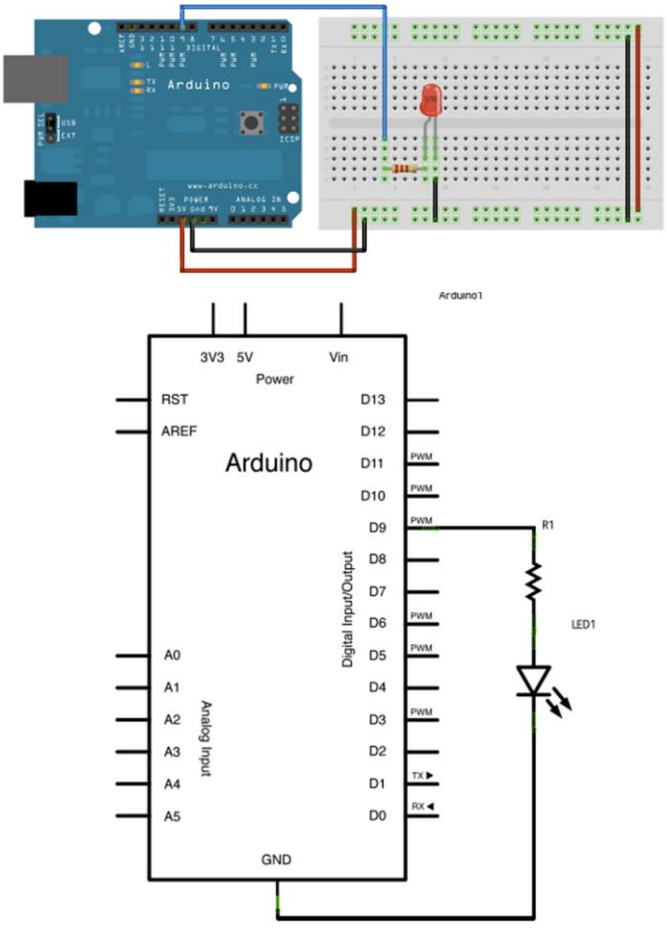
```
/*
 Analog Input
 Demonstrates analog input by reading an analog sensor on analog pin 0 and
 turning on and off a light emitting diode(LED)  connected to digital pin 13.
 The amount of time the LED will be on and off depends on
 the value obtained by analogRead().

 The circuit:
 * Potentiometer attached to analog input 0
 * center pin of the potentiometer to the analog pin
 * one side pin (either one) to ground
 * the other side pin to +5V
 * LED anode (long leg) attached to digital output 13
 * LED cathode (short leg) attached to ground

 Created by David Cuartielles
 modified 30 Aug 2011
 By Tom Igoe

 This example code is in the public domain.
 http://arduino.cc/en/Tutorial/AnalogInput
 */

int sensorPin = A0;    // select the input pin for the potentiometer
int ledPin = 13;      // select the pin for the LED
int sensorValue = 0;  // variable to store the value coming from the sensor

void setup() {
  // declare the ledPin as an OUTPUT:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
  // turn the ledPin on
  digitalWrite(ledPin, HIGH);
  // stop the program for <sensorValue> milliseconds:
  delay(sensorValue);
  // turn the ledPin off:
  digitalWrite(ledPin, LOW);
  // stop the program for for <sensorValue> milliseconds:
  delay(sensorValue);
}
```



b) Analog output

- The Arduino Uno does not actually support a 'real' ***analog output*** signal. But it can emulate analog output using ***pulse width modulation (PWM)***.
- ***PWM*** operates by turning a digital pin on and off very quickly in a periodic fashion. ***Duty cycle*** is the percentage of the time the signal stays HIGH relative to the period.
- PWM based analog outputs are available on pins 3, 5, 6, 9, 10, and 11 of the Arduino Uno. Note that these are different from the analog input ports.
- ***analogRead(analogPin)*** function gets its input from the given analog pin.
- ***analogWrite(PWMpin,value)*** sends a PWM signal with a duty cycle in the range between 0 (always off) and 255 (always on), on the specified pin.

Example 2: Use of the *analogWrite()* function to control LED brightness and demonstrate a fading effect.




```
/*
 Fade -  This example shows how to fade an LED on pin 9
 using the analogWrite() function.
 This example code is in the public domain.
 */
int led = 9;          // the pin that the LED is attached to
int brightness = 0;   // how bright the LED is
int fadeAmount = 5;   // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup()  {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop()  {
  // set the brightness of pin 9:
  analogWrite(led, brightness);
  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;
  // reverse the direction of the fading at the ends of the fade:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

2. Displaying Data on Serial Terminal

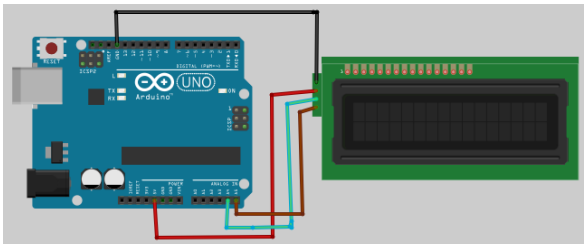
- Arduino provides Serial library to allow communication between the Arduino board and a computer or other devices.
- All Arduino boards have at least one serial port (also known as a UART or USART).
    - It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB.
    - Thus, if you use these functions, you cannot also use pins 0 and 1 for digital input or output.
- You can use the Arduino environment's built-in serial monitor to communicate with an Arduino board.
    - Click the serial monitor button in the toolbar and select the same baud rate used in the call to begin().

Example 3: Use of the serial monitor to display the data you receive from a potentiometer. Experiment and see how the data changes when you turn the potentiometer dial up and down.

```
/*
  AnalogReadSerial
  Reads an analog input on pin 0, prints the result to the serial monitor.
  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and ground.

 This example code is in the public domain.
 */

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```

3. Displaying Messages on LCDs

- The PC Serial Monitor is handy for displaying messages and testing your code while the Arduino is attached to your PC via the USB cable.
- However, if you want to display messages in stand-alone applications, portable and compact solutions will be more appropriate.
- In this section you will see how to use low-cost character LCD display modules for outputting messages from the Arduino. Such displays come in many forms, sizes and colors. For illustration purpose we have chosen the Arrela® I2C Serial LCD Module for Arduino.
  For further information refer to: http://arduino-info.wikispaces.com/LCD-Blue-I2C#v3
- The I2C Serial LCD module has 4 pins on its back that are used for interfacing with the Arduino. Connect Vcc and GND to 5V and GND pins on the Arduino board, respectively. Connect the SDA and SCL pins to analog pins A4 and A5, respectively.



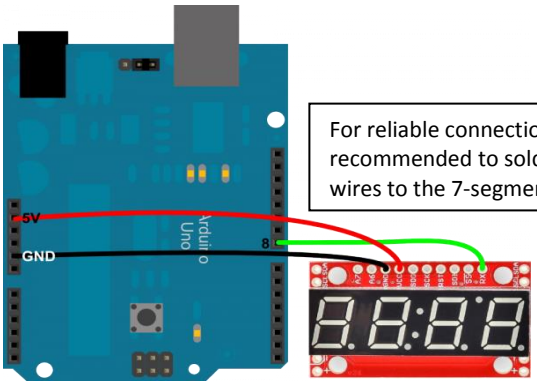
```
#include   <Wire.h>
#include  <LiquidCrystal_I2C.h>

LiquidCrystal_I2C   lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

void setup()
{
   lcd.begin(16,2);            // initialize the lcd
   lcd.home ();                // go home
   lcd.print("Hello, ARDUINO ");
   lcd.setCursor ( 0, 1 );       // go to the next line
   lcd.print ("INEN-101 ROCKS!!");
   delay ( 1000 );
}
void loop()
{
   lcd.home ();
   delay (200);
}
```

- The I2C Serial LCD library is available on Blackboard under Week-3 folder. Download the LiquidCrystal_I2C.zip file, unzip it and then copy the LiquidCrystal_I2C folder in to the Arduino library folder.
- The original library archive is also available at the following site: https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads
- The first parameter in the LCD initialization code is the I2C device address for the LCD. For the display we are using in this lab you need to set the address to the value **0x3F**, as shown in the example code.

4. Displaying Messages on 7-segment Display

- Seven segment displays offer alternative options for low-cost and low-power display solution for embedded system applications.
- To demonstrate how such devices can be interfaced with an Arduino type microcontroller, below we show an example of a serial 7-segment display that is wired using one of multiple communication options. As alternative to the I2C interface that was used for the LCD in the previous section, this time we use the UART serial interface (actually a software serial).

The wiring between the Arduino and the serial 7-segment display: 5V and GND from Arduino connected to Vcc (+) and GND(-), resp. on the display. Pin 8 is used as a software serial transmitter pin and connected to receive (Rx) pin on the display. More information available at: https://learn.sparkfun.com/tutorials/using-the-serial-7-segment-display/all#example-1-serial-uart



For reliable connections it is recommended to solder the wires to the 7-segment display.

```
/* Serial 7-Segment Display Example Code
   Serial Mode Stopwatch
  by: Jim Lindblom
    SparkFun Electronics
  date: November 27, 2012
  license: This code is public domain.

  This example code shows how you could use software serial
  Arduino library to interface with a Serial 7-Segment Display.

  There are example functions for setting the display's
  brightness, decimals and clearing the display.

  The print function is used with the SoftwareSerial library
  to send display data to the S7S.

  Circuit:
  Arduino ---------- Serial 7-Segment
  5V  ----------------- VCC
  GND ---------------- GND
  8  ------------------- RX
*/

#include <SoftwareSerial.h>
// These are the Arduino pins required to create a software seiral
//  instance. We'll actually only use the TX pin.
const int softwareTx = 8;
const int softwareRx = 7;
SoftwareSerial s7s(softwareRx, softwareTx);
unsigned int counter = 0;  // This variable will count up to 65k
char tempString[10];  // Will be used with sprintf to create strings
```

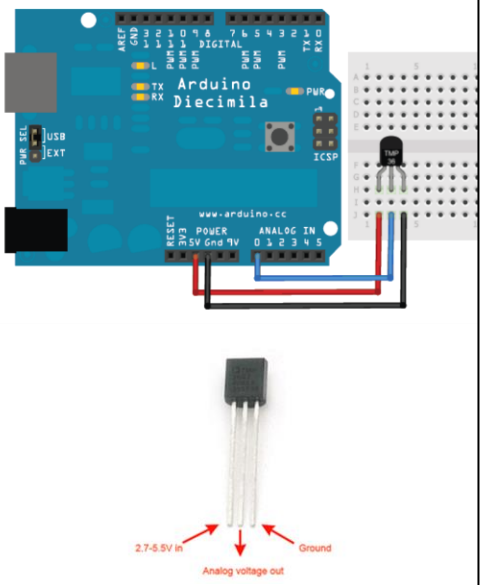
```
void setup()
{
  // Must begin s7s software serial at the correct baud rate.
  //  The default of the s7s is 9600.
  s7s.begin(9600);
  // Clear the display, and then turn on all segments and decimals
  clearDisplay();  // Clears display, resets cursor
  s7s.print("-HI-");  // Displays -HI- on all digits
  setDecimals(0b111111);  // Turn on all decimals, colon, apos
  // Flash brightness values at the beginning
  setBrightness(0);  // Lowest brightness
  delay(1500);
  setBrightness(127);  // Medium brightness
  delay(1500);
  setBrightness(255);  // High brightness
  delay(1500);
  // Clear the display before jumping into loop
  clearDisplay();
}

void loop()
{
  // Magical sprintf creates a string for us to send to the s7s.
  //  The %4d option creates a 4-digit integer.
  sprintf(tempString, "%4d", counter);
  // This will output the tempString to the S7S
  s7s.print(tempString);
  setDecimals(0b00000100);  // Sets digit 3 decimal on
  counter++;  // Increment the counter
  delay(100);  // This will make the display update at 10Hz.
}

// Send the clear display command (0x76)
//  This will clear the display and reset the cursor
void clearDisplay()
{
  s7s.write(0x76);  // Clear display command
}

// Set the displays brightness. Should receive byte with the value
//  to set the brightness to
//  dimmest------------->brightest
//     0--------127--------255
void setBrightness(byte value)
{
  s7s.write(0x7A);  // Set brightness command byte
  s7s.write(value);  // brightness data byte
}

// Turn on any, none, or all of the decimals.
//  The six lowest bits in the decimals parameter sets a decimal
//  (or colon, or apostrophe) on or off. A 1 indicates on, 0 off.
//  [MSB] (X)(X)(Apos)(Colon)(Digit 4)(Digit 3)(Digit2)(Digit1)
void setDecimals(byte decimals)
{
  s7s.write(0x77);
  s7s.write(decimals);
}
```

5. Interfacing Sensors

- Sensors in real-life application produce information, in electronic signal form, about the phenomenon they are supposed to sense, such as temperature, force, acceleration, light intensity, etc.
- Depending on the type of the sensor, the signal it generates could be analog or digital in nature. Arduino can be configured to accept analog or digital signals from sensors.

a) Temperature sensor

- To demonstrate how temperature sensors work, consider the TMP36 analog temperature sensor from Sparkfun (http://www.adafruit.com/products/165)
- It can be easily interfaced to the Arduino using one of its analog input pins. Refer to the following link for further information: https://learn.adafruit.com/tmp36-temperature-sensor/using-a-temp-sensor



To convert the voltage to temperature, simply use the basic formula:

Temp in °C = [(Vout in mV) - 500] / 10

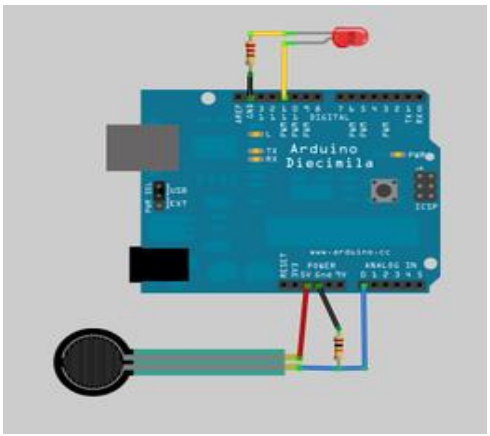
So for example, if the voltage out is 1V that means that the temperature is **((1000 mV - 500) / 10) = 50 °C**

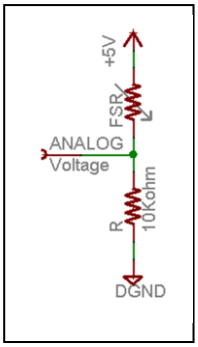
```
//TMP36 Pin Variables
int sensorPin = 0; //the analog pin of the TMP36's Vout (sense) pin is
                   // connected to the resolution is 10 mV / degree centigrade with
                   //a 500 mV offset to allow for negative temperatures
/*
 * setup() - this function runs once when you turn your Arduino on
 * We initialize the serial connection with the computer
 */
void setup()
{
  Serial.begin(9600);  //Start the serial connection with the computer
                       //to view the result open the serial monitor
}

void loop()                 // run over and over again
{
  int reading = analogRead(sensorPin);  //read voltage from temperature sensor
 // converting the analog reading to voltage, this is using 5V power for sensor
 float voltage = reading * 5.0;
 voltage = voltage/1024.0;
 Serial.print(voltage); Serial.println(" volts"); // print out the voltage
 float temperatureC = (voltage - 0.5) * 100 ;  //convert volt to degrees
                                               //((voltage - 500mV) times 100)
 Serial.print(temperatureC); Serial.println(" degrees C"); // print temperature
 // now convert to Fahrenheit
 float temperatureF = (temperatureC * 9.0 / 5.0) + 32.0;
 Serial.print(temperatureF); Serial.println(" degrees F");
 delay(1000);                          //waiting a second
}
```

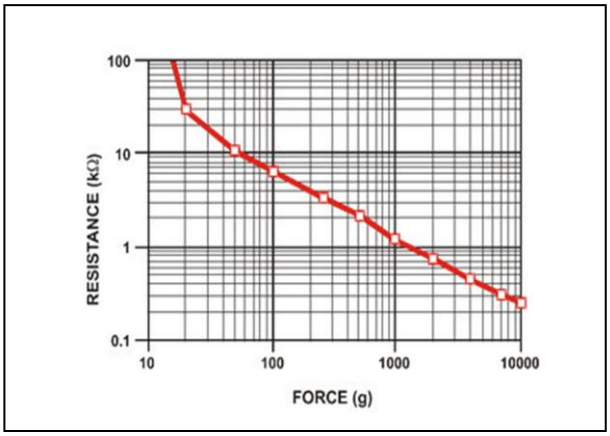
Exercise: Modify this temperature sensor example so that your measurements will be displayed on the serial 7-segment display unit.

b) Force sensor

- Force sensitive resistor (FSR) allows you to detect physical pressure, squeezing, and weight applied to the element.
- FSR is basically a resistor that changes its resistance (in ohms, Ω) depending on how much it is pressed. These sensors are fairly low cost, and easy to use but they are rarely accurate. They also vary some from sensor to sensor perhaps by up to 10%. So basically when you use FSRs you should only expect to get *ranges* of response. While they can detect weight, they're a bad choice for measuring exactly how many pounds of weight are on them.
- The following example circuit connects the FSR to the Arduino using a voltage division circuit. Analog voltage from this circuit is connected to the Arduino analog input pin A0. An LED is connected to PWM pin 11. The program reads the voltage from the FSR and controls the brightness of the LED based on the applied pressure. For further information refer to: https://learn.adafruit.com/force-sensitive-resistor-fsr/using-an-fsr



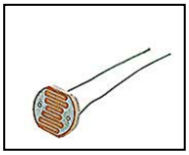




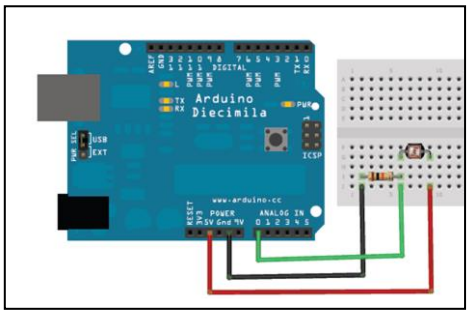

```
/* FSR testing sketch.

Connect one end of FSR to 5V, the other end to Analog 0.
Then connect one end of a 10K resistor from Analog 0 to ground
Connect LED from pin 11 through a resistor to ground

For more information see www.ladyada.net/learn/sensors/fsr.html
*/

int fsrAnalogPin = 0; // FSR is connected to analog 0
int LEDpin = 11;      // connect Red LED to pin 11 (PWM pin)
int fsrReading;      // the analog reading from the FSR resistor
                     // voltage divider
int LEDbrightness;

void setup(void) {
  Serial.begin(9600);   // for debugging via the Serial monitor
  pinMode(LEDpin, OUTPUT);
}
```

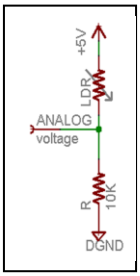
```
void loop(void) {
  fsrReading = analogRead(fsrAnalogPin);
  Serial.print("Analog reading = ");
  Serial.println(fsrReading);

  // we'll need to change the range from the analog reading (0-1023)
  // down to the range used by analogWrite (0-255) with map!
  LEDbrightness = map(fsrReading, 0, 1023, 0, 255);
  // LED gets brighter the harder you press
  analogWrite(LEDpin, LEDbrightness);

  delay(100);
}
```

c) Light sensor



- Photoresistor is a device that changes its electrical resistance property according to the intensity of light illuminated to it. It is made up of cadmium sulfite. It responds to visible light similar to human eye.
- This device can easily be used as a light sensor and has many applications, including for automatic switching of street lights, for making robotic line followers, etc.
- Think of some ideas you could explore to use light sensor for wearable application…for example automatic control of safety LED lights that construction workers, bikers, runners, hikers, etc. could wear.
- Photoresistor resistance could range between 200 KΩ in the dark and 500 Ω in a brightly lit room.
- As in the case of the FSR in the previous section, changes in the resistance of the photoresistor can be measured using a voltage divider circuit connected to one of the analog input pins of the Arduino. For further information refer to: https://learn.adafruit.com/photocells/using-a-photocell





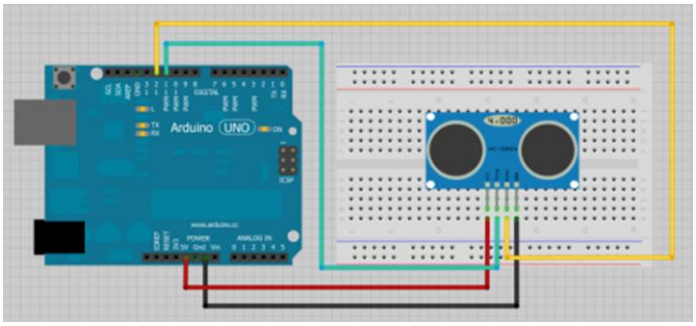
When testing this sample program of the light sensor circuit, the message output will be displayed on the computer's serial monitor. Open the serial monitor from your Arduino IDE by going to the Tools menu or clicking the corresponding button at the top right of the IDE. Use your hands or a piece of cloth to cover the sensor and observe its measurements when it is open and covered.

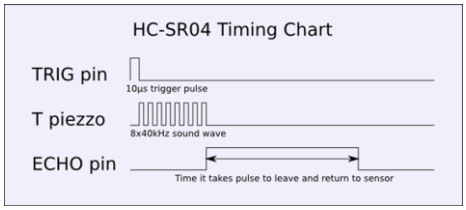
Exercise: Modify this example so that you display the light sensor values on the serial LCD module.

```
/* Photocell simple testing sketch.

 Connect one end of the photocell to 5V, the other end to Analog 0.
Then connect one end of a 10K resistor from Analog 0 to ground

 For more information see http://learn.adafruit.com/photocells */

 int photocellPin = 1;     // the cell and 10K pulldown are connected to A1
 int photocellReading;   // the analog reading from the resistor divider

 void setup(void) {
  // We'll send debugging information via the Serial monitor
  Serial.begin(9600);
}

void loop(void) {
  photocellReading = analogRead(photocellPin);

  Serial.print("Analog reading = ");
  Serial.print(photocellReading);     // the raw analog reading

   // We'll have a few threshholds, qualitatively determined
  if (photocellReading < 10) {
    Serial.println(" - Dark");

    Serial.println(" - Dim");
  } else if (photocellReading < 500) {
    Serial.println(" - Light");
  } else if (photocellReading < 800) {
    Serial.println(" - Bright");
  } else {
    Serial.println(" - Very bright");
  }
  delay(1000);
}
```

d) Ultrasonic sensor

- An ultrasound senor is used to detect targets and measure their distance from the sensor. The transmitter part of the sensor emits high frequency sound ping, which bounces off objects and is read back by the receiver of the sensor. The time it takes for the signal to return back is used to estimate the distance to the target. This sensor can also be used for motion sensing, by monitoring any changes to the measurements of the sensor.
- To demonstrate how ultrasonic sensor is interfaced to the Arduino, we consider an example using the HC-SR04 module. It has 4 pins (Vcc, GND, Trig, Echo). Connect Vcc and GND to 5V and GND on the Arduino. Wire Trig and Echo on the ultrasonic module to pin 11 and pin 12, respectively.
- Source: http://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/





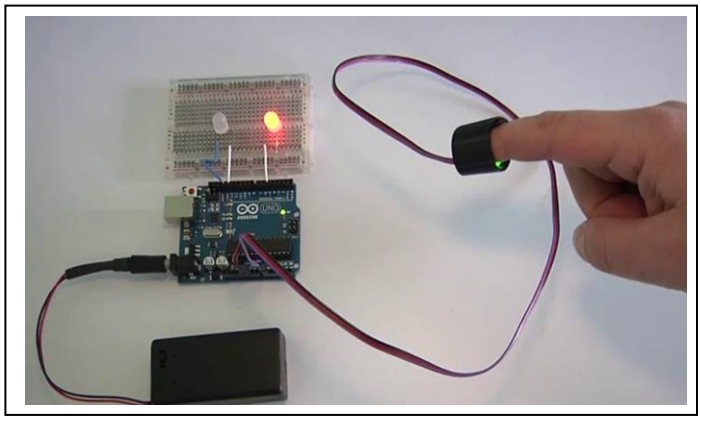
Time width of Echo pulse relates to distance:
Distance (cm) = Time / 58
Distance (in) = Time / 148

```
#define echoPin     12          // Echo Pin
#define trigPin     11          // Trigger Pin

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);
}
```

```
void loop() {
      /* The trigPin/echoPin cycle is used to determine the distance
of the nearest object */
       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;
      // Send "Out of range" message
      if (distance >= maximumRange || distance <= minimumRange)
      {
          Serial.println( "Out of range");
      }
      else
      { // Send the distance to the display
            Serial.print( "D = ");
            Serial.println(distance);
      }
      //Delay 50ms before next reading.
      delay(50);
  }
```

e) Pulse Sensor

- Pulse Sensor is a well-designed heart-rate sensor for Arduino. It can be used by anyone who wants to easily incorporate live heart-rate data into their projects.
- The sensor clips into a fingertip or earlobe and plugs right into Arduino. It also includes an open-source monitoring application that graphs your pulse rate in real-time.
- Information about the Pulse Sensor and how you can interface it to the Arduino is given at the site: http://pulsesensor.com/pages/code-and-guide



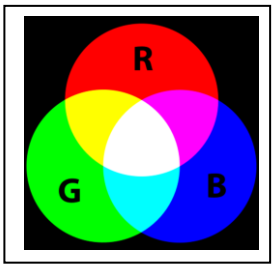
- Code for Arduino can be downloaded from the same site. For graphical visualization of the heart beat waveform a PC based Processing program is provided. If you want to test this visualizer you need to download and install the open-source and free Processing program, which is based on the Processing Programming language and development environment. This language is developed to be easy to learn, and powerful for creating visualizations, arts, animations, etc. http://hello.processing.org/
- This sensor could provide you with some ideas that you can further investigate for real-time health monitoring or activity tracking type of applications.

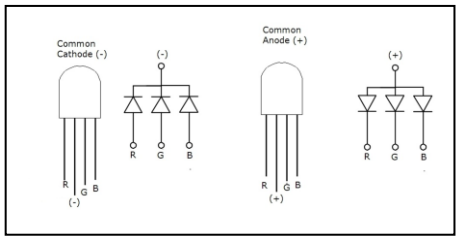
6. Color RGB LEDs

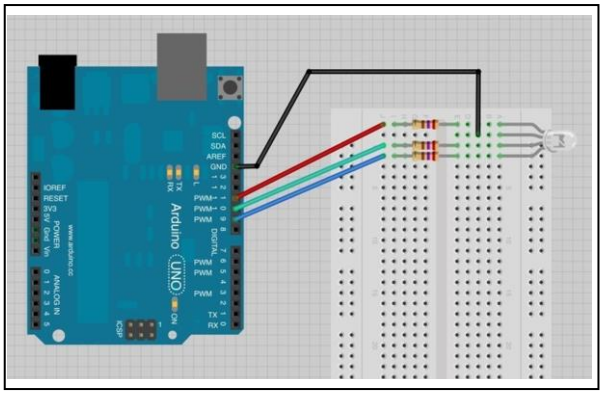
- You are already familiar with basic LEDs that may come with a variety of colors (Red, Green, Yellow, Blue, White are common).

a) RGB LEDs

- The RGB LEDs contain a single package LED that can generate a wide spectrum of possible colors by mixing any combination of the basic colors (Red, Green, Blue).







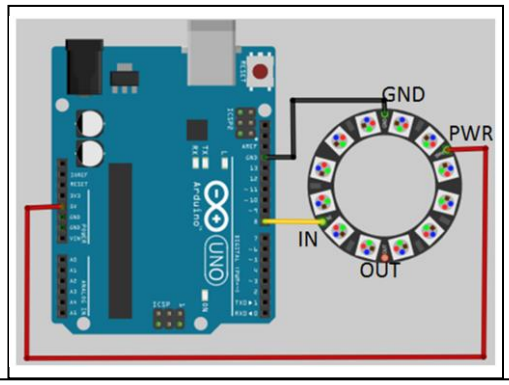
- For example the RGB LED from Adafruit (http://www.adafruit.com/products/159) or Sparkfun (https://www.sparkfun.com/products/105) are common cathode design with 4 pins. One pin is ground and the rest correspond to inputs for the three colors.
- The actual net color on the LED is controlled using PWM signals generated by using Arduino's ***analogWrite(…)*** function. Wiring diagram and programming example are available at: https://learn.adafruit.com/adafruit-arduino-lesson-3-rgb-leds/overview

```
/*
Adafruit Arduino - Lesson 3. RGB LED
*/

int redPin = 11;
int greenPin = 10;
int bluePin = 9;

//uncomment this line if using a Common Anode LED
//#define COMMON_ANODE

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

```
void loop()
{
  setColor(255, 0, 0);  // red
  delay(1000);
  setColor(0, 255, 0);  // green
  delay(1000);
  setColor(0, 0, 255);  // blue
  delay(1000);
  setColor(255, 255, 0);  // yellow
  delay(1000);
  setColor(80, 0, 80);  // purple
  delay(1000);
  setColor(0, 255, 255);  // aqua
  delay(1000);
}

void setColor(int red, int green, int blue)
{
  #ifdef COMMON_ANODE
    red = 255 - red;
    green = 255 - green;
    blue = 255 - blue;
  #endif
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}
```

b) NeoPixels

- These are ultra-bright RGB LEDs with integrated drivers, designed and manufactured by Adafruit. They come in many different shapes and sizes. Refer to http://www.adafruit.com/category/275 for more information.
- For the purpose of demonstration of how these NeoPixel LEDs are controlled we will consider the NeoPixel Ring with 12 individually addressable RGB LEDs (http://www.adafruit.com/products/1643)
- The driving mechanism needs only a single data line with a very timing specific protocol. The rings are 'chainable' - connect the output pin of one to the input pin of another. Use only one microcontroller pin to control as many as you can chain together! Each LED is addressable as the driver chip is inside the LED.



- The Arduino library for the NeoPixel is available from: https://github.com/adafruit/Adafruit_NeoPixel
  The download command is on the right side of the page. After downloading the zip file, unzip it first, rename it to Adafruit_NeoPixel, and then copy it in your Arduino Library folder. To test the NeoPixel, wire the circuit as shown here and use the example code below.

```
// NeoPixel Ring simple sketch (c) 2013 Shae Erisson
// released under the GPLv3 license to match the rest of the
AdaFruit NeoPixel library

#include <Adafruit_NeoPixel.h>
#include <avr/power.h>

// Which pin on the Arduino is connected to the NeoPixels?
// On a Trinket or Gemma we suggest changing this to 1
#define PIN        8

// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS      12

// When we setup the NeoPixel library, we tell it how many
// pixels, and which pin to use to send signals.
// Note that for older NeoPixel strips you might need to
// change the third parameter--see the strandtest
// example for more information on possible values.

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN,
NEO_GRB + NEO_KHZ800);

int delayval = 500; // delay for half a second
```

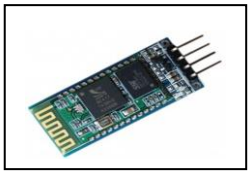
```
void setup() {
// This is for Trinket 5V 16MHz, you can remove these three lines if
// you are not using a Trinket
#if defined (__AVR_ATtiny85__)
  if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
#endif                    // End of trinket special code

   pixels.begin(); // This initializes the NeoPixel library.
}

void loop() {
// For a set of NeoPixels the first NeoPixel is 0, second is 1, all the
// way up to the count of pixels minus one.

  for(int i=0;i<NUMPIXELS;i++){

    // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
    pixels.setPixelColor(i, pixels.Color(0,50,0)); // Moderately bright
                                                   // green color.

    pixels.show(); // This sends the updated pixel color to the hardware.

    delay(delayval); // Delay for a period of time (in milliseconds).
  }
}
```

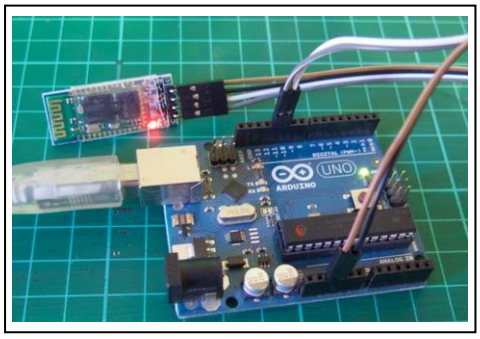
Exercise: Modify this program so that in the first cycle it turns all the LEDs sequentially using red color, then turns them off sequentially. In the next cycle it turns them on sequentially using green, and then turns them off sequentially. Finally, in the third cycle it turns them on sequentially using blue, and then turns them off sequentially. Repeat indefinitely.

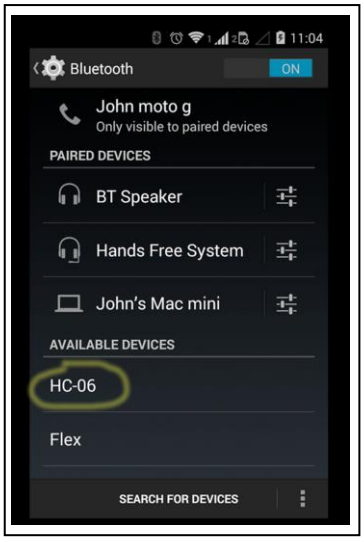
7) Bluetooth (BT) Communication

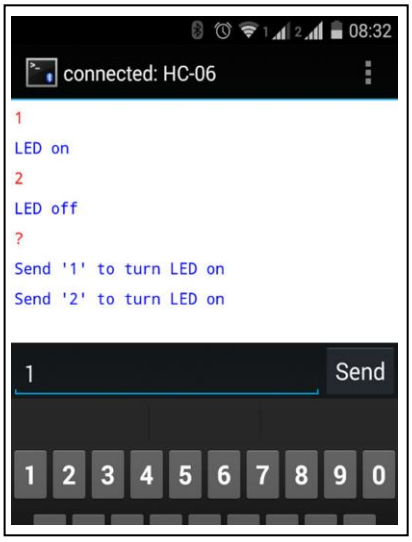
- Now let's get to the fun part of this tutorial. Do you want to control your Arduino wirelessly from your Smartphone? Thanks to a low-cost Bluetooth module that is available for Arduino, we can easily set up a communication between an Android phone and the Arduino. This lets you control devices remotely or access information from the Arduino as needed.
- Refer to the online reference at the following site for further information: http://tronixlabs.com/news/tutorial-using-hc06-bluetooth-to-serial-wireless-uart-adaptors-with-arduino/
- The hardware interface between the Arduino and the BT module requires 4 wires. Vcc and GND from the BT connect to 5V and GND on the Arduino. The actual data communication between the BT and the Arduino takes place through serial communication. Connect the Tx and Rx pins from the BT to digital pins 10 and 11, respectively, on the Arduino. Since these are regular digital pins on the Arduino board, you need to use Software Serial library to enable the Arduino handle serial communication over regular digital pins. Note, only digital pins 0 and 1 on the Arduino UNO support Hardware Serial interface.
- For the Android Smartphone we will use ***Bluetooth Terminal*** by *qwerty* (https://play.google.com/store/apps/details?id=Qwerty.BluetoothTerminal&hl=en), a free terminal emulation app that will allow us to send and receive characters. Download this app from the Play Store and install it on your Android phone.
- Once is installed, you will need to pair your BT module to the Smartphone. To do this, enter the Bluetooth menu inside Settings, and then search for new devices. After a moment the device "HC06" or "HC05" or something similar, corresponding to your BT module will appear.
- Tap the "HC06" or "HC05" from the list corresponding to your device, and then enter a PIN – it is typically 1234. Finally, open your terminal app on the smartphone, and select "Connect a device" from the app menu. Select your BT device and then wait a moment. When the connection is established the blinking pattern of the red LED on the BT changes (it becomes steady or slower blinking depending on the device type) and the app shows a connected message.

- The figures below show the BT module interfacing and screenshots of its pairing and connection to the Android Smartphone and BT terminal app.









- Once you have successfully established the BT connection, you can now send commands, using characters, from your phone to the Arduino.
- The sample program we are using for demonstration expects you to send numbers, such as 1 and 2, to control the on-board LED attached to pin 13 on the Arduino. Sending a 1 turns the LED on, while sending a 2 turns it off.

```
#include <SoftwareSerial.h>

SoftwareSerial BT(10, 11);

// creates a "virtual" serial port/UART
// connect BT module TX to D10
// connect BT module RX to D11
// connect BT Vcc to 5V, GND to GND

void setup()
{
  // set digital pin to control as an output
  pinMode(13, OUTPUT);
  // set the data rate for the SoftwareSerial port
  BT.begin(9600);
  // Send test message to other device
  BT.println("Hello from Arduino");
}

char a; // stores incoming character from other device
```

For those who are interested to develop their own Android app from scratch, you can refer to the *MIT App Inventor*, a free and easy to learn, blocks-based programming tool: http://appinventor.mit.edu/explore/

```
void loop()
{
  if (BT.available())
  // if text arrived in from BT serial...
  {
    a=(BT.read());
    if (a=='1')
    {
      digitalWrite(13, HIGH);
      BT.println("LED on");
    }
    if (a=='2')
    {
      digitalWrite(13, LOW);
      BT.println("LED off");
    }
    if (a=='?')
    {
      BT.println("Send '1' to turn LED on");
      BT.println("Send '2' to turn LED on");
    }
    // you can add more "if" statements with other
    // characters to add more commands
  }
}
```

Exercise : Controlling the colors of NeoPixel LEDs from Android device

Wire the NeoPixel ring with the 12 RGB LEDs to your Arduino as discussed in the tutorial 6.b on page 14. Also wire your Bluetooth module to the Arduino as in tutorial 7 on page 15-16. Verify that the wiring is done correctly.

The next step is to write a program for your Arduino that will allow you to control the color of the NeoPixel LEDs with your command from your Android device. Your program should be supporting the following commands from the user:

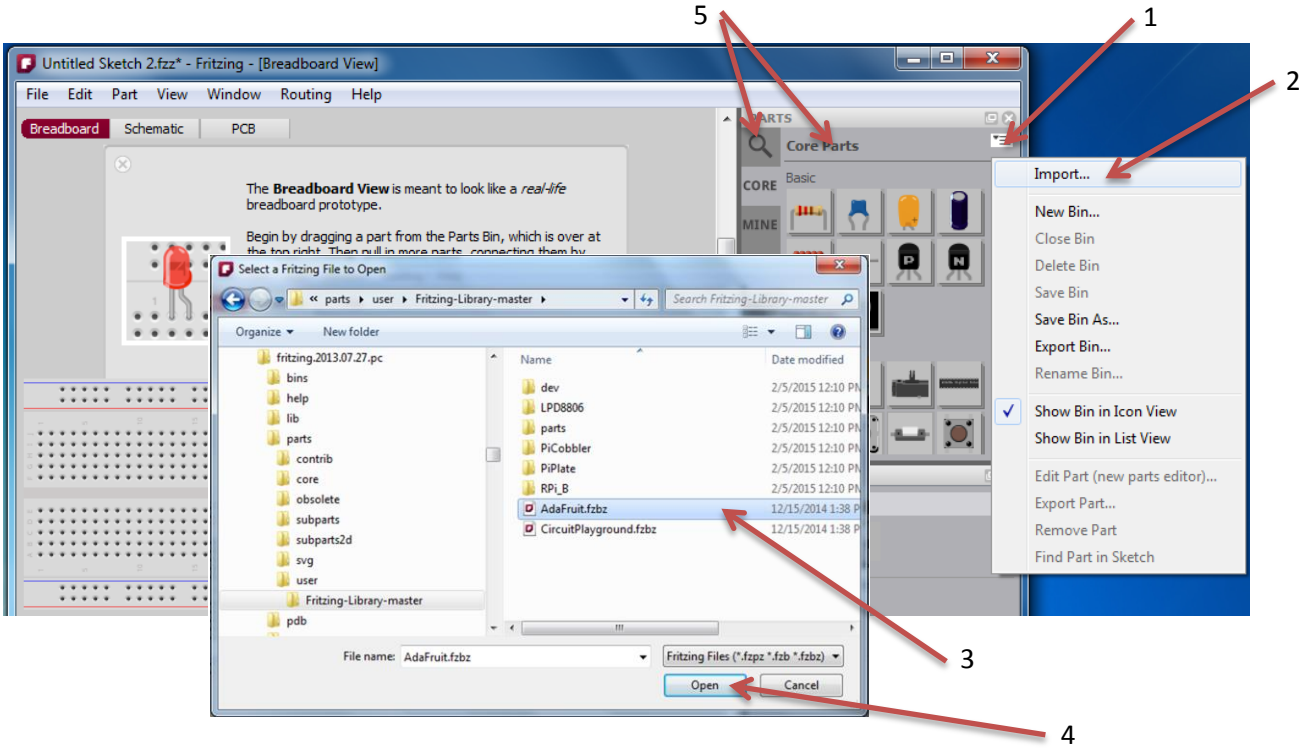
Character sent (from Android device)	**Effect on NeoPixel LEDs ring**
R or r	RED sequence on NeoPixel LEDs ring
G or g	GREEN sequence on NeoPixel LEDs ring
B or b	BLUE sequence on NeoPixel LEDs ring
O or o	OFF sequence on NeoPixel LEDs ring

Note: when you do each sequence, start by turning on the LEDs with the selected color, one LED at a time with a 100 ms interval between them. Then after all LEDs in the ring are on, turn them off in the same sequential manner. And repeat the operations until a new color command is requested.

Fritzing help:

When you want to draw your circuit design in Fritzing you may find out that the symbol for the part you want to draw may not be available in the parts library. If that is the case, and if the part's manufacturer has already created a Fritzing library, here are the steps you can follow to import the library so you can access it. This illustration shows how you can import the Adafruit Fritzing library that will give you access to the neopixel symbols as well as other Adafruit parts.

1. Download the Adafruit Fritzing library from the following site: https://github.com/adafruit/Fritzing-Library
   Note that the download link is on the right hand side of the page with a button labelled "Download ZIP". Click this button to download.
2. Unzip the downloaded zip file into Fritzing -> parts -> user
3. Start the Fritzing application
4. Import the Adafruit parts library you downloaded, by clicking the small down arrow symbol under the "PARTS" window at the far right of the screen, next to the search line. Follow the 1, 2, 3, and 4 steps below.



5. Once the import is completed, then you can search for the part you are looking for (the neopixel ring) to find its diagram and bring it to your Breadboard view for use in your drawing. Note that the search keywords can be entered at location number 5 in the above diagram.