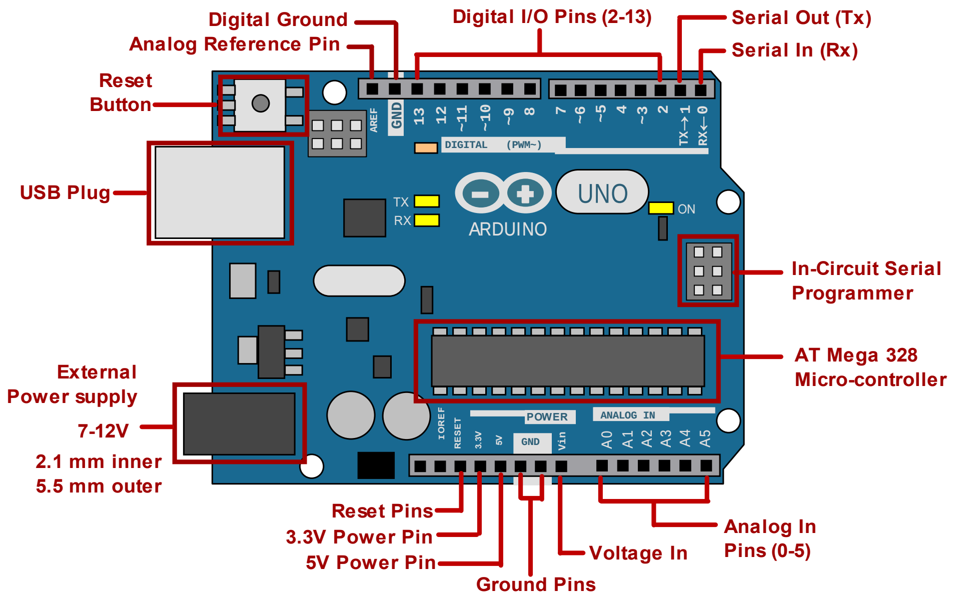
Electric Circuits Laboratory

ENGR 250L - LAB EXERCISES

LAB 10 - ARDUINO LED & PING (.5 lec + 1h)



Micro-controllers are computers that are designed to sense and control things in the physical world. They can be used to turn lights on/off, run motors, sense sound, light, acceleration, etc. The Arduino platform has become popular in the hobbyist and academic communities.

We will program Arduino to control the lighting of an LED. This exercise is a good introduction to controlling the input/output (I/O) pins on Arduino. Arduino - like any other computer - can be used to change the behavior of a machine through simple changes to the programming code. This is why so many things are controlled by computers now.

REFER to the ARDUINO GUIDE (Ch 1 - Intro, Ch 2 - Arduino IDE software, Ch 3 - I/O signals, pwm)
REFER to the C PROGRAMMING GUIDE

PARTS LIST

1. Arduino, cable, computer	4. 220-ohm resistor
2. Breadboard shield	5. wire
3. LED	6. Ultra-sonic "ping" sensor (HC-SR04)

10.1 BLINK SURFACE-MOUNTED LED

Here we get the "surface mounted" (built onto the board) LED (connected to pin 13) to blink on and off. This exercise only requires the Arduino board and cable.

LED blinking is often the first and simplest exercise with a micro-controller. This is done by changing digital pin 13 (HIGH or LOW). Although this is a simple exercise, the basic process of controlling the Arduino input/output is fundamental and forms the basis for more advanced tasks.

IMPORTANT
All Arduino programs must have 2 functions: setup and loop
"setup" is done first and once. "loop" is done after that and it repeats forever.
In digital electronics LOW means 0V and HIGH means 5V

CODE

```
// 1_Blink.ino ----------------
// Blink LED on Arduino (pin 13) repeatedly

int ledPin = 13;    // specify pin

void setup()   // done once
{
  pinMode(ledPin, OUTPUT);  // set pin as output
}

void loop()   // done repeatedly
{
  digitalWrite(ledPin, HIGH);         // LED on
  delay(300);                         // hold it for this many ms
  digitalWrite(ledPin, LOW);          // LED off
  delay(300);                         // hold it
}

// end -----------------------------------
```

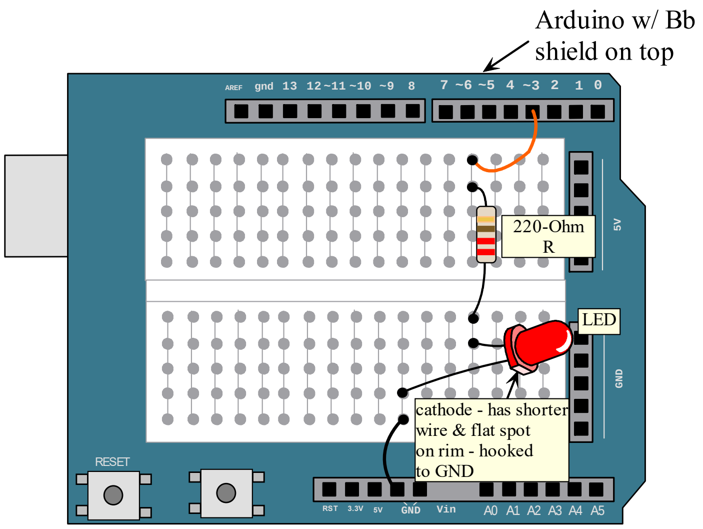
FOLLOW UP

Alter the delay times (change the number in the delay ( ) function) and observe the changes in the blinking LED. Make the LED blink faster and then record the result. This demonstrates how software can easily alter the behavior of hardware.

OUTPUT

The surface mount LED should blink repeatedly. The blinking rate should change as you make changes to the delay time.

10.2 BLINK EXTERNAL LED


Arduino w/ Bb shield on top
AREF gnd 13 12 ~11 ~10 ~9 8
7 ~6 ~5 4 ~3 2 1 0
220-Ohm R
LED
5V
GND
cathode - has shorter wire & flat spot on rim - hooked to GND
RESET
RST 3.3V 5V GND Vin
A0 A1 A2 A3 A4 A5

We've blinked the surface-mounted LED. Now we'll blink an external LED so we'll need a breadboard onto which to build the circuit. We'll use a "BREADBOARD SHIELD". Shields are circuit boards that mount on top of the Arduino and add to its capability. A breadboard shield simply provides a breadboard while pulling up the Arduino pins for easy access.

NOTE

Always build and modify circuits with power disconnected.
NEVER create short circuits (connecting different voltage values directly. eg - 5V and GND).
GND - is usually the (-) side of power supply.

CIRCUIT

The figure shows the Arduino with a BREADBOARD SHIELD that gets assembled on top of the Arduino.
A Bb shield simply provides a breadboard on which to build a circuit, and it pulls up the I/O connections for easy accessibility.
The programming code MUST match the physical circuit (e.g., the specified pins).
The LED must be inserted in the correct direction (cathode closer to GND).
The 220-ohm resistor (red-red-brown) in series with the LED limits current in LED to about 20 mA.

CODE

```
// 2_externalLED.ino ---------------------------
// blink an external LED

int ledPin = 3;    // led pin - must match physical connections

// duplicate the code from first exercise

// end ---------------------------------------------
```

OUTPUT

The external LED should blink repeatedly just like the surface-mounted LED.

10.3 BRIGHTNESS

Now let's get the LED to step through a series of brightness changes, and when done, cycle back again. You could change LED brightness by changing the voltage applied, but Arduino does not have analog output capability. Instead, Arduino has so-called PULSE-WIDTH MODULATION (or PWM). In reality the LED is blinking, but it's blinking so fast our eyes and brain interpret this as a dimmed LED.

CIRCUIT

There is NO CHANGE from previous exercise.

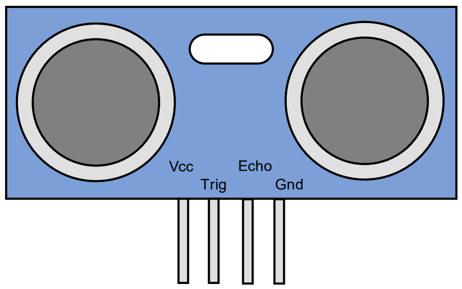
CODE

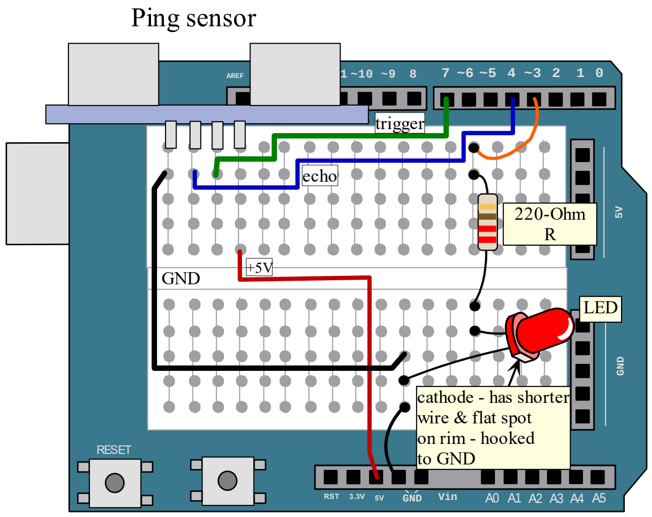
```
// 3_Brightness.ino -------------------------------------------
// Make LED change brightness

int ledPin = 3;           // pwm pin
int ledLevel = 0;
int duration = 500;       // time at ea level (alter this)
int noLevels = 5;         // pwm 0-255, (alter this)

void setup()
{
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  long tnow;        //get current time
  ledLevel = 0;

  for (int i = 1; i < noLevels; i++)
  {
    analogWrite(ledPin, ledLevel);
    ledLevel += (255/noLevels);
    delay(duration);
  }
}


// end -----------------------------------------
```

OUTPUT

The external LED should increase in brightness in steps.
After achieving max brightness, it should turn off and then repeat.
Try changing the duration and the number of brightness levels.

10.4 PING DISTANCE





Ping sensor
trigger
echo
220-Ohm R
+5V
GND
LED
cathode - has shorter wire & flat spot on rim - hooked to GND
RESET
5V
GND

In this exercise we will obtain distance data from an ultrasonic "PING" sensor. Later we will use the distance data to drive the brightness of an LED and the movement of a small servo motor. The use of sensor data to drive actuators is the basis for robotics and control systems.

BACKGROUND

ULTRA-SONIC "PING" SENSORS sense distance by sending out an ultrasonic pulse (at 40 kHz, which you can't hear) of sound and measuring the time it takes for that sound pulse to be received back. Distance can be inferred from the time it takes for the pulse to be received. This sensor is commonly included in many Arduino kits. While the two cylindrical parts look like eyes or speakers, they are not. One is an ultrasound transmitter, and the other a receiver. There are 2 popular sensors on the market. One is by Parallax called the "PING" sensor ($30). The other is the HC-SR04 ($4). The PING sensor is more expensive but it works better with smaller targets and only uses 1 I/O pin. The HC-SR04 has more erroneous signals with smaller targets and uses 2 I/O pins. I will refer to theses sensors as "PING SENSORS" from now on, even though you may be using the HC-SR04.

Vcc	5 VDC
Frequency	40 kHz
Range	2 cm to 4 meters
Accuracy	+/- 3 mm
Target size	> 0.5 m2.
Echo pulse	150 us - 25 ms (38 ms if no object is detected)

The ping sensor will generate an 8-cycle ultra-sonic (at 40 kHz) burst (or pulse) in response to a 10 us (microsecond, which is one millionth of a second) pulse on the trigger pin. The receiver will wait to receive the reflected sonic burst and the sensor will return a pulse on the echo pin that is equal to the time elapsed between transmit and receive (in micro-seconds, us).

Watch this informative video:
https://www.youtube.com/watch?v=ZejQOX69K5M

The statement below

```
distance = duration * 0.0343/2;
```

computes distance traveled by sound which travels .0343 at cm/us. You divide by 2 because sound is traveling twice the distance you want to display (pulse goes out, reflects off a surface, and returns to the sensor). The code will return distance in cm.
In this exercise the ping sensor will sense distance to an obstacle (your hand or a piece of cardboard) placed in front of it, and Arduino will display the result (in cm) on the serial monitor (or plotter).

The ping sensor will sometimes output erroneous data (distance displays an unreasonably large value). This is often happens with small targets. If we are using the distance data to control, for instance, a motor, then we need to do something about this error. Otherwise the motor will not move as desired. One approach is save the data point from the last time through the loop. If the data point gathered in the current loop seems erroneous (too big), then use the last good data point.

Also we will create the function "getDistance()". We will place the code for reading the ultrasonic sensor in this function. This is how you should organize computer code.

CIRCUIT

Do not remove the LED circuit. Just add the ping sensor circuit.
The ping sensor has 4 connections: power (5v), ground, trigger, and echo. Trigger will be hooked to an I/O pin that sets off the measurement. Echo is hooked to an I/O pin that records the time measured.

CODE

```
// 4_pingDistance.ino ----------------
// display distance (in cm) on SM using ping sensor, correct for erroneous data

const int trigPin = 7;
const int echoPin = 4;

long duration;
int distance;
int distanceLast = 0;
int distanceCap = 400;

void setup()
{
  pinMode (trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
}

void loop()
{
  distance = getDistance(trigPin, echoPin);
  Serial.print(distance);
  Serial.println(" cm");
}

int getDistance(int trigPin, int echoPin) // ---------------
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);      // send out 10 us pulse

  duration = pulseIn(echoPin, HIGH);   // get echo signal
```

```
    distance = duration * 0.0343 / 2;    // compute distance

    if (distance > distanceCap)    // if get bad data pt
    {
      distance = distanceLast;     // use last good data pt
    }
    else                            // good data pts
    {
      distance = constrain(distance, 2, 40);  // constrain distance: 2 - 40 cm
      distanceLast = distance;       // update last good value
    }

    return distance;
  }

  // end -------------------------------------------------------
```

OUTPUT

As an obstacle is moved closer and farther from the ping sensor, the distance should be displayed on the serial monitor. Also try using the serial plotter which should plot the distance in real time.