Solution code is put in this doc so you can reference it on Google Drive easily.

### LAB OPERATION 1

/\*

Blink

Turns on an LED on for one second, then off for one second, repeatedly.

Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO

it is attached to digital pin 13, on MKR1000 on pin 6. LED\_BUILTIN takes care

of use the correct LED pin whatever is the board used.

If you want to know what pin the on-board LED is connected to on your Arduino model, check

the Technical Specs of your board at https://www.arduino.cc/en/Main/Products

This example code is in the public domain.

modified 8 May 2014

by Scott Fitzgerald

modified 2 Sep 2016

by Arturo Guadalupi

\*/

// the setup function runs once when you press reset or power the board

void setup() {

// initialize digital pin LED\_BUILTIN as an output.

pinMode(LED\_BUILTIN, OUTPUT);

}

// the loop function runs over and over again forever

void loop() {

digitalWrite(LED\_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)

delay(500); // wait for a second

digitalWrite(LED\_BUILTIN, LOW); // turn the LED off by making the voltage LOW

delay(500); // wait for a second

}

### LAB OPERATION 2

/\*

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.

created 2005

by DojoDave <http://www.0j0.org>

modified 30 Aug 2011

by Tom Igoe

This example code is in the public domain.

http://www.arduino.cc/en/Tutorial/Button

\*/

// 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);

}

}

### LAB OPERATION 3

NOTE: we will not accept just using the simple analog input example for this operation. Students MUST use the analogWrite function.

/\*

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

\* Note: because most Arduinos have a built-in LED attached

to pin 13 on the board, the LED is optional.

Created by David Cuartielles

modified 30 Aug 2011

By Tom Igoe

This example code is in the public domain.

http://www.arduino.cc/en/Tutorial/AnalogInput

\*/

int sensorPin = A0; // select the input pin for the potentiometer

int ledPin = 3; // select the pin for the LED

int sensorValue = 0; // variable to store the value coming from the sensor

int val = 0;

void setup() {

// declare the ledPin as an OUTPUT:

pinMode(ledPin, OUTPUT);

}

void loop() {

// read the value from the sensor:

sensorValue = analogRead(sensorPin);

//map value to analogWrite domain

val = map(val, 0, 1023, 0, 255);

//write mapped value to PWM pin

analogWrite(ledPin, val);

}